# Fully Automatic Cross-Associations

Deepayan Chakrabarti Spiros Papadimitriou
Dharmendra S. Modha[1] Christos Faloutsos[2]

August 2004
CMU-CALD-04-107

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1] Author's affiliation: IBM Almaden Research Center, San Jose, CA, USA.

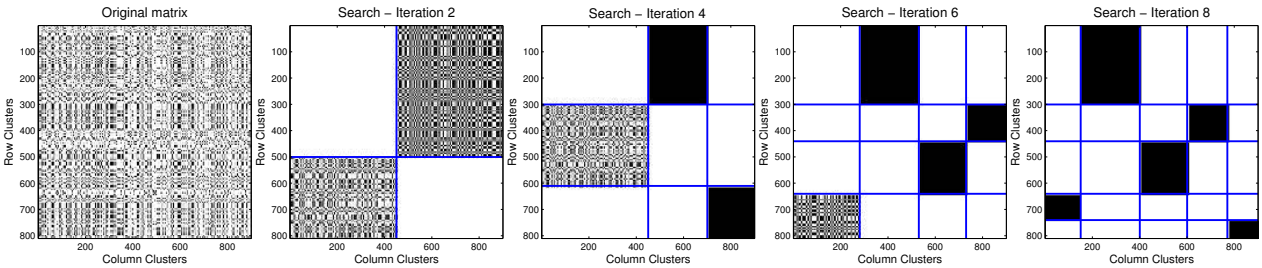
[2] This material is based upon work supported by the National Science Foundation under Grants No. IIS-9817496, IIS-9988876, IIS-0083148, IIS-0113089, IIS-0209107 IIS-0205224 INT-0318547 SENSOR-0329549 EF-0331657IIS-0326322 by the Pennsylvania Infrastructure Technology Alliance (PITA) Grant No. 22-901-0001, and by the Defense Advanced Research Projects Agency under Contract No. N66001-00-1-8936. Additional funding was provided by donations from Intel, and by a gift from Northrop-Grumman Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

**Keywords:** Information Theory, MDL, Cross-Association

**Abstract**

Large, sparse binary matrices arise in numerous data mining applications, such as the analysis of market baskets, web graphs, social networks, co-citations, as well as information retrieval, collaborative filtering, sparse matrix reordering, etc. Virtually all popular methods for the analysis of such matrices—e.g., $k$-means clustering, METIS graph partitioning, SVD/PCA and frequent itemset mining—require the user to specify various parameters, such as the number of clusters, number of principal components, number of partitions, and "support." Choosing suitable values for such parameters is a challenging problem.

*Cross-association* is a joint decomposition of a binary matrix into disjoint row and column groups such that the rectangular intersections of groups are homogeneous. Starting from first principles, we furnish a clear, information-theoretic criterion to choose a good cross-association as well as its parameters, namely, the number of row and column groups. We provide scalable algorithms to approach the optimal. Our algorithm is *parameter-free*, and requires no user intervention. In practice it scales linearly with the problem size, and is thus applicable to very large matrices. Finally, we present experiments on multiple synthetic and real-life datasets, where our method gives high-quality, intuitive results.




(a) Original matrix (b) Iteration pair 1 (c) Iteration pair 2 (d) Iteration pair 3 (e) Iteration pair 4

Figure 1: *Searching for cross-associations*: Starting with the original matrix (plot (a)), our algorithm successively increases the number of groups. At each stage, starting with the current arrangement into groups, rows and columns are rearranged to improve the code cost.

# 1 Introduction - Motivation

Large, sparse binary matrices arise in many applications, under several guises. Consequently, because of its importance and prevalence, the problem of discovering structure in binary matrices has been widely studied in several domains:*(1) Market basket analysis and frequent itemsets:* The rows of the matrix represent customers (or transactions) and the columns represent products. Entry $(i, j)$ of the matrix is 1 if customer $i$ purchased product $j$ and 0 otherwise. *(2) Information retrieval:* Rows correspond to documents, columns to words and an entry in the matrix represent whether a certain word is present in a document or not. *(3) Graph partitioning and community detection:* Rows and columns correspond to source and target objects and matrix entries represent links from a source to a destination. *(4) Collaborative filtering, microarray analysis,* and numerous other applications—in fact, any setting that has a *many-to-many* relationship (in database terminology) in which we need to find patterns.

We ideally want a method that discovers structure in such datasets and has the following main properties:

**(P1)** It is fully automatic; in particular, we want a principled and intuitive problem formulation, such that the user does not need to set *any* parameters.

**(P2)** It simultaneously discovers both row and column groups.

**(P3)** It scales up for large matrices.

**Cross-association and Our Contributions** The fundamental question in mining large, sparse binary matrices is whether there is any underlying structure. In these cases, the labels (or, equivalently, the ordering) of the rows and columns is immaterial. The binary matrix contains information about *associations* between objects, irrespective of their labeling. Intuitively, we seek row and column groupings (equivalently, labellings) that reveal the underlying structure. We can group rows, based on some notion of "similarity" and we could do the same for columns. Better yet, we would like to *simultaneously* find row and column groups, which divide the matrix into rectangular regions as "similar" or "homogeneous" as possible. These intersections of row and column groups, or *cross-associations*, succinctly summarize the underlying structure of object associations. The corresponding rectangular regions of varying density can be used to quickly navigate through the structure of the matrix.

In short, we would like a method that will take as input a matrix like in Figure 1(a), and will quickly and automatically (i) determine a good number of row groups $k$ and column groups $l$ and (ii) re-order the rows and columns, to reveal the hidden structure of the matrix, like in Figure 1(e).

We propose a method that has precisely the above properties: it requires no "magic numbers," discovers row and column groups simultaneously (see Figure 1) and scales linearly with the problem size. We introduce a novel

approach and propose a general, intuitive model founded on compression and information-theoretic principles. In particular, unlike existing methods, we employ *lossless* compression and always operate at a zero-distortion level. Thus, we can use the MDL principle to automatically select the number of row and column groups. We provide an integrated framework to *automatically* find cross-associations. Also, our method is easily extensible to matrices with categorical values.

In Section 2, we survey the related work. In Section 3, we formulate our data description model starting from first principles. Based on this, in Section 4 we develop an efficient, parameter-free algorithm to discover cross-associations. In Section 5 we evaluate cross-associations demonstrating good results on several real and synthetic datasets. Finally, we conclude in Section 6.

## 2 Survey

In general, there are numerous settings where we want to find patterns, correlations and rules. There are several time-tested tools for most of these tasks. Next, we discuss several of these approaches, dividing them broadly into application domains. However, with few exceptions, all require tuning and human intervention, thus failing on property (P1).

**Clustering** We discuss work in the "traditional" clustering setting first. By that we mean approaches for grouping along the row dimension only: given a collection of $n$ points in $m$ dimensions, find "groupings" of the $n$ points. This setting makes sense in several domains (for example, if the $m$ dimensions have an inherent ordering), but it is different from our problem setting.

Also, most of the algorithms assume a user-given parameter. For example, the most popular approach, $k$-means clustering, requires $k$ from the user. The problem of finding $k$ is a difficult one and has attracted attention recently; for example X-means [1] uses BIC to determine $k$. Another more recent approach is G-means [2], which assumes a mixture of Gaussians (often a reasonable assumption, but which may not hold for binary matrices). Other interesting variants of $k$-means that improve clustering quality is $k$-harmonic means [3] (which still requires $k$) and spherical $k$-means (e.g., see [4]), which applies to binary data but still focuses on clustering along one dimension). Finally, there are many other recent clustering algorithms (CURE [5], BIRCH [6], Chameleon [7], [8]; see also [9]).

Several of the clustering methods might suffer from the dimensionality curse (like the ones that require a co-variance matrix); others may not scale up for large datasets.

Information Co-clustering (ITCC) [10] is a recent algorithm for simultaneously clustering rows and columns of a normalized contingency table or a two-dimensional probability distribution. Cross-associations (CA) also simultaneously group rows and columns of a binary (or categorical) matrix and, at the surface, bear similarity to ITCC. However, the two approaches are quite different:

**(1)** For each rectangular intersection of a row cluster with a column cluster, CA constructs a *lossless code*, whereas ITCC constructs a *lossy code* that can be thought of as a rank-one matrix approximation.

**(2)** ITCC generates a progressively finer approximation of the original matrix. More specifically, as the number of row and column clusters are increased, the Kullback-Leibler divergence (or, KL-divergence) between the original matrix and its lossy approximation tends to zero. In contrast, regardless of the number of clusters, CA always losslessy transmits the entire matrix. In other words, as the number of row and column clusters are increased, ITCC tries to sweep an underlying rate-distortion curve, where the *rate* depends upon the number of row and column clusters and *distortion* is the KL-divergence between the original matrix and its lossy approximation. In comparison, CA always operates at zero distortion.

**(3)** While both ITCC and CA use alternating minimization techniques, ITCC minimizes the KL-divergence between the original matrix and its lossy approximation, while CA minimizes the resulting codelength for the

original matrix.

**(4)** As our key contribution, in CA, we use the MDL principle to automatically select the number of row and column clusters. While MDL is well known for lossless coding which is the domain of CA, no MDL-like principle is yet known for lossy coding; for a very recent proposal towards this direction, see [11]. As a result, selecting the number of row and column clusters in ITCC is still an art. Note that ITCC is similar in spirit to the Information Bottleneck formulation [12].

Thus, to the best of our knowledge, our method is the first to study explicitly the problem of parameter-free, joint clustering of large binary matrices.

**Market-basket analysis / frequent itemsets** Frequent itemset mining brought a revolution [13] with a lot of follow-up work [9, 14]. However, they require the user to specify a "support." The work on "interestingness" is related [15], but still does not answer the question of "support."

**Information retrieval and LSI** The pioneering method of LSI [16] uses SVD on the term-document matrix. Again, the number $k$ of eigenvectors/concepts to keep is up to the user ([16] empirically suggest about 200 concepts). Additional matrix decompositions include the Semi-Discrete Decomposition (SDD) [17], PLSA [18], the clever use of random projections to accelerate SVD [19], and many more. However, they all fail on property (P1).

**Graph partitioning** The prevailing methods are METIS [20] and spectral partitioning [21]. These approaches have attracted a lot of interest and attention; however, both need the user to specify $k$, that is, the number of pieces to break the graph into. Moreover, they typically also require a measure of imbalance between the two pieces of each split.

**Other domains** Related to graphs in several settings is the work on conjunctive clustering [22]—which requires density (i.e., "homogeneity") and overlap parameters—as well as community detection [23], among many. Finally, there are several approaches to cluster micro-array data (e.g., [24]).

In conclusion, the above methods miss one or more of our prerequisites, typically (P1). Next, we present our method.

# 3 Cross-association and Compression

Our goal is to find patterns in a large, binary matrix, with no user intervention, as shown in Figure 1. How should we decide the number of row and column groups ($k$ and $\ell$, respectively) along with the assignments of rows/columns to their "proper" groups?

We introduce a novel approach and propose a general, intuitive model founded on compression, and more specifically, on the *MDL (Minimum Description Language)* principle [25]. The idea is the following: the binary matrix represents *associations* between objects (corresponding to rows and columns). We want to somehow summarize these in *cross-associations*, i.e., homogeneous, rectangular regions of high and low densities. At the very extreme, we can have $m \times n$ "rectangles," each really being an element of the original matrix, and having "density" of either 0 or 1. Then, each rectangle needs no further description. At the other extreme, we can have *one* rectangle, with a density in the range from 0 to 1. However, neither really is a summary of the data. So, the question is, how many rectangles should we have? The idea is that we penalize the number of rectangles, i.e., the complexity of the data description. We do this in a principled manner, based on a novel application of the MDL philosophy (where the costs are based on the number of bits required to transmit both the "summary" of the structure, as well as each rectangular region, given the structure).

| Symbol | Definition |
|---|---|
| $D$ | Binary data matrix |
| $m, n$ | Dimensions of $D$ (rows, columns) |
| $k, \ell$ | Number of row and column groups |
| $k^*, \ell^*$ | Optimal number of groups |
| $(\Phi, \Psi)$ | Cross-association |
| $D_{i,j}$ | Cross-associate (submatrix) |
| $a_i, b_j$ | Dimensions of $D_{i,j}$ |
| $n(D_{i,j})$ | Number of elements $n(D_{i,j}) := a_i b_j$ |
| $n_0(D_{i,j}), n_1(D_{i,j})$ | Number of 0, 1 elements in $D_{i,j}$ |
| $P_{D_{i,j}}(0), P_{D_{i,j}}(1)$ | Densities of 0, 1 in $D_{i,j}$ |
| $H(p)$ | Binary Shannon entropy function |
| $C(D_{i,j})$ | Code cost for $D_{i,j}$ |
| $T(D; k, \ell, \Psi, \Phi)$ | Total cost for $D$ |

Table 1: Table of main symbols.

This is an intuitive and very general model of the data, that requires *no* parameters. Our model allows us to find good cross-associations *automatically*. Next, we describe the theoretical underpinnings in detail.

### 3.1 Cross-association

Let $D = [d_{i,j}]$ denote a $m \times n$ ($m, n \geq 1$) binary data matrix. Let us index the rows as $1, 2, \ldots, m$ and columns as $1, 2, \ldots, n$.

Let $k$ denote the desired number of disjoint row groups and let $\ell$ denote the desired number of disjoint column groups. Let us index the row groups by $1, 2, \ldots, k$ and the column groups by $1, 2, \ldots, \ell$. Let

$$\Psi : \{1, 2, \ldots, m\} \rightarrow \{1, 2, \ldots, k\}$$
$$\Phi : \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, \ell\}$$

denote the assignments of rows to row groups and columns to column groups, respectively. We refer to $\{\Psi, \Phi\}$ as a *cross-association*. To gain further intuition about a given cross-association, given row groups $\Psi$ and column groups $\Phi$, let us rearrange the underlying data matrix $D$ such that all rows corresponding to group 1 are listed first, followed by rows in group 2, and so on. Similarly, let us rearrange $D$ such that all columns corresponding to group 1 are listed first, followed by columns in group 2, and so on. Such a rearrangement, implicitly, sub-divides the matrix $D$ into smaller two-dimensional, rectangular blocks. We refer to each such sub-matrix as a *cross-associate*, and denote them as $D_{i,j}$, $i = 1, \ldots, k$ and $j = 1, \ldots, \ell$. Let the dimensions of $D_{i,j}$ be $(a_i, b_j)$.

### 3.2 A Lossless Code for a Binary Matrix

With the intent of establishing a close connection between cross-association and compression, we first describe a lossless code for a binary matrix. There are several possible models and algorithms for encoding a binary matrix. With hindsight, we have simply chosen a code that allows us to build an efficient and analyzable cross-association algorithm. Throughout this paper, all logarithms are base 2 and all code lengths are in bits.

Let $A$ denote an $a \times b$ binary matrix. Define

$$n_1(A) := \text{number of nonzero entries in } A$$

$$n_0(A) := \text{number of zero entries in } A$$
$$n(A) := n_1(A) + n_0(A) = a \times b$$
$$P_A(i) := n_i(A)/n(A), \quad i = 0, 1.$$

Intuitively, we model the matrix $A$ such that its elements are drawn in an i.i.d. fashion according to the distribution $P_A$. Given the knowledge of the matrix dimensions $(a, b)$ and the distribution $P_A$, we can encode $A$ as follows. Scan $A$ in a fixed, predetermined ordering. Whenever $i$, $i = 0, 1$ is encountered, it can be encoded using $-\log P_A(i)$ bits, on average. The total number of bits sent (this can also be achieved in practice using, e.g., arithmetic coding [26, 27, 28]) will be

$$C(A) := \sum_{i=0}^{1} n_i(A) \log \left( \frac{n(A)}{n_i(A)} \right) = n(A) H\big(P_A(0)\big), \tag{1}$$

where $H$ is the binary Shannon entropy function.

For example, consider the matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

In this case, $n_1(A) = 4$, $n_0(A) = 12$, $n(A) = 16$, $P_A(1) = 1/4$, $P_A(0) = 3/4$. We can encode each $0$ element using roughly $\log(4/3)$ bits and each $1$ element using roughly $\log 4$ bits. The total code length for $A$ is: $4 * \log 4 + 12 * \log 4/3 = 16 * H(1/4)$.

### 3.3 Cross-association and Compression

We now make precise the link between cross-association and compression. Let us suppose that we are interested in transmitting (or storing) the data matrix $D$ of size $m \times n$ ($m, n \geq 1$), and would like to do so as efficiently as possible. Let us also suppose that we are given a cross-association $(\Psi, \Phi)$ of $D$ into $k$ row groups and $\ell$ column groups, with none of them empty.

With these assumptions, we now describe a two-part code for the matrix $D$. The first part will be a *description complexity* involved in describing the cross-association $(\Psi, \Phi)$. The second part will be the actual *code* for the matrix, given the cross-association.

#### 3.3.1 Description Complexity

The description complexity in transmitting the cross-association shall consist of the following terms:

1. Send the matrix dimensions $m$ and $n$ using, e.g., $\log^\star(m) + \log^\star(n)$, where $\log^\star$ is the universal code length for integers[1]. However, this term is independent of the cross-association. Hence, while useful for actual transmission of the data, it will not figure in our framework.
2. Send the row and column permutations using, e.g., $m\lceil \log m \rceil$ and $n\lceil \log n \rceil$ bits, respectively. This term is also independent of any given cross-association.
3. Send the number of groups $(k, \ell)$ using $\log^\star k + \log^\star \ell$ bits (or alternatively, using $\lceil \log m \rceil + \lceil \log n \rceil$ bits).

[1] It can be shown that $\log^\star(x) \approx \log_2(x) + \log_2 \log_2(x) + \ldots$, where only the positive terms are retained and this is the optimal length, if we do not know the range of values for $x$ beforehand [29]

4. Send the number of rows in each row group and also the number of columns in each column group. Let us suppose that $a_1 \geq a_2 \geq \ldots \geq a_k \geq 1$ and $b_1 \geq b_2 \geq \ldots \geq b_\ell \geq 1$. Compute

$$\bar{a}_i := \left(\sum_{t=i}^{k} a_t\right) - k + i, \quad i = 1, \ldots, k-1$$
$$\bar{b}_j := \left(\sum_{t=j}^{\ell} b_t\right) - \ell + j, \quad j = 1, \ldots, \ell-1.$$

   Now, the desired quantities can be sent using the following number of bits:

$$\sum_{i=1}^{k-1} \lceil \log \bar{a}_i \rceil + \sum_{j=1}^{\ell-1} \lceil \log \bar{b}_j \rceil.$$

5. For each cross-associate $D_{i,j}$, $i = 1, \ldots, k$ and $j = 1, \ldots, \ell$, send $n_1(D_{i,j})$, i.e., the number of ones in the matrix, using $\lceil \log(a_i b_j + 1) \rceil$ bits.

#### 3.3.2 The Code for the Matrix

Let us now suppose that the entire preamble specified above has been sent. We now transmit the each of the actual cross-associates $D_{i,j}$, $i = 1, \ldots, k$ and $j = 1, \ldots, \ell$, using $C(D_{i,j})$ bits according to Eq. 1.

#### 3.3.3 Putting It Together

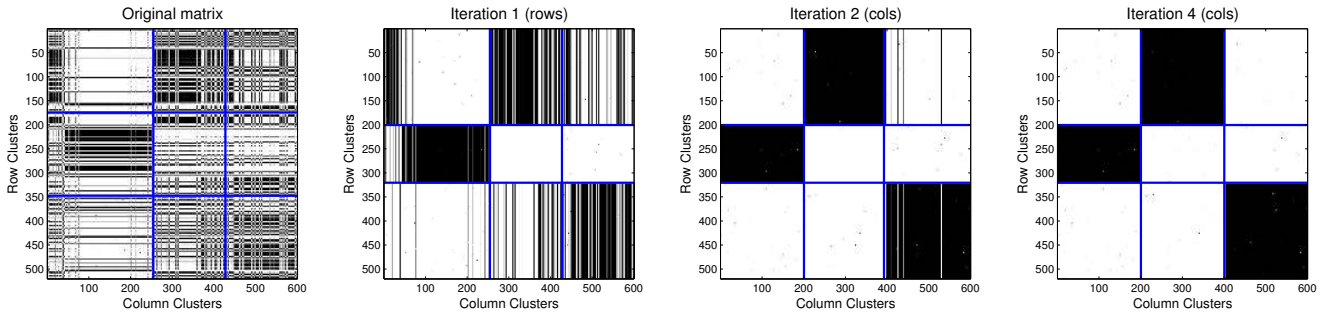
We can now write the total code length for the matrix $D$, with respect to a given cross-association as:

$$\begin{aligned} T(D; k, \ell, \Psi, \Phi) := \log^\star k + \log^\star \ell + \sum_{i=1}^{k-1} \lceil \log \bar{a}_i \rceil + \sum_{j=1}^{\ell-1} \lceil \log \bar{b}_j \rceil \\ + \sum_{i=1}^{k} \sum_{j=1}^{\ell} \lceil \log(a_i b_j + 1) \rceil + \sum_{i=1}^{k} \sum_{j=1}^{\ell} C(D_{i,j}), \end{aligned} \tag{2}$$

where we ignore the costs $\log^\star(m) + \log^\star(n)$ and $m\lceil \log m \rceil + n\lceil \log n \rceil$, since they do not depend upon the given cross-association.

### 3.4 Problem Formulation

An *optimal cross-association* corresponds to the number of row groups $k^\star$, the number of column groups $\ell^\star$, and a cross-association $(\Psi^\star, \Phi^\star)$ such that the total resulting code length, namely, $T(D; k^\star, \ell^\star, \Psi^\star, \Phi^\star)$ is minimized. Typically, such problems are computationally hard. Hence, in this paper, we shall pursue feasible practical strategies. To determine the optimal cross-association, we must determine both the number of row and columns groups and also a corresponding cross-association. We break this joint problem into two related components: (i) finding a good cross-association for a given number of row and column groups; and (ii) searching for the number of row and column groups. In Section 4.1 we describe an alternating minimization algorithm to find an optimal cross-association for a fixed number of row and column groups. In Section 4.2, we outline an effective heuristic strategy that searches over $k$ and $\ell$ to minimize the total code length $T$. This heuristic is integrated with the minimization algorithm.




(a) Original groups (b) Row shifts (Step 2) (c) Column shifts (Step 4) (d) Column shifts (Step 4)

Figure 2: *Row and column shifting*: Holding $k$ and $\ell$ fixed (here, $k = \ell = 3$), we repeatedly apply Steps 2 and 4 of REGROUP until no improvements are possible (Step 6). Iteration 3 (Step 2) is omitted, since it performs no swapping. To potentially decrease the cost further, we must increase $k$ or $\ell$ or both, as in Figure 1.

# 4 Algorithms

In the previous section we established our goal: Among all possible $k$ and $l$ values, and all possible row- and column-groups, pick the arrangement with the smallest total compression cost, as MDL suggests (model plus data). Although theoretically pleasing, Eq. 2 does not tell us *how* to go about finding the best arrangement—it can only pinpoint the best one, among several candidates. The question is *how to generate good candidates*.

We answer this question in two steps:

1. REGROUP (inner loop): For a given $k$ and $\ell$, find a good arrangement (i.e., cross-association).
2. CROSSASSOCIATIONSEARCH (outer loop): Search for the best $k$ and $\ell$ ($k, \ell = 1, 2, \ldots$), re-using the arrangement so far.

We present each in the following sections.

## 4.1 Alternating Minimization (REGROUP)

Suppose we are given the number of row groups $k$ and the number of column groups $\ell$ and are are interested in finding a cross-association $(\Psi^\star, \Phi^\star)$ that minimizes

$$\sum_{i=1}^{k}\sum_{j=1}^{\ell} C(D_{i,j}), \tag{3}$$

where $D_{i,j}$ are the cross-associates of $D$, given $(\Psi^\star, \Phi^\star)$. We now outline a simple and efficient alternating minimization algorithm that yields a local minimum of Eq. 3. We should note that, in the regions we typically perform the search, the code cost dominates the total cost by far (see also Figure 3 and Section 5.1), which justifies this choice.

Algorithm REGROUP:

1. Let $t$ denote the iteration index. Initially, set $t = 0$. Start with an arbitrary cross-association $(\Psi^t, \Phi^t)$ of the matrix $D$ into $k$ row groups and $\ell$ column groups. For this initial partition, compute the cross-associate matrices $D^t_{i,j}$, and corresponding distributions $P_{D^t_{i,j}} \equiv P^t_{i,j}$.

2. For this step, we will hold column assignments, i.e., $\Phi^t$, fixed. For every row $x$, splice it into $\ell$ parts, each corresponding to one of the column groups. Denote them as $x^1, \ldots, x^\ell$. For each of these parts, compute $n_u(x^j)$, $u = 0, 1$, and $j = 1, \ldots, \ell$. Now, assign row $x$ to that row group $\Psi^{t+1}$ such that, for all $1 \leq i \leq k$:

$$\sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(x^j) \log \frac{1}{P^t_{\Psi^{t+1}(x),j}(u)} \leq \sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(x^j) \log \frac{1}{P^t_{i,j}(u)}. \tag{4}$$

3. With respect to cross-association $(\Psi^{t+1}, \Phi^t)$, recompute the matrices $D^{t+1}_{i,j}$, and corresponding distributions $P_{D^{t+1}_{i,j}} \equiv P^{t+1}_{i,j}$.

4–5. Similar to steps 2–3, but swapping columns instead and producing a new cross-association $(\Psi^{t+1}, \Phi^{t+2})$ and corresponding cross-associates $D^{t+2}_{i,j}$ with distributions $P_{D^{t+2}_{i,j}} \equiv P^{t+2}_{i,j}$.

6. If there is no decrease in total cost, stop; otherwise, set $t = t + 2$, go to step 2, and iterate.

---

Figure 2 shows the alternating minimization algorithm in action. The graph consists of three square sub-matrices ("caves" [30]) with sizes 280, 180 and 90, plus 1% noise. We permute this matrix and try to recover its structure. As expected, for $k = \ell = 3$, the algorithm discovers the correct cross-associations. It is also clear that the algorithm finds progressively better representations of the matrix.

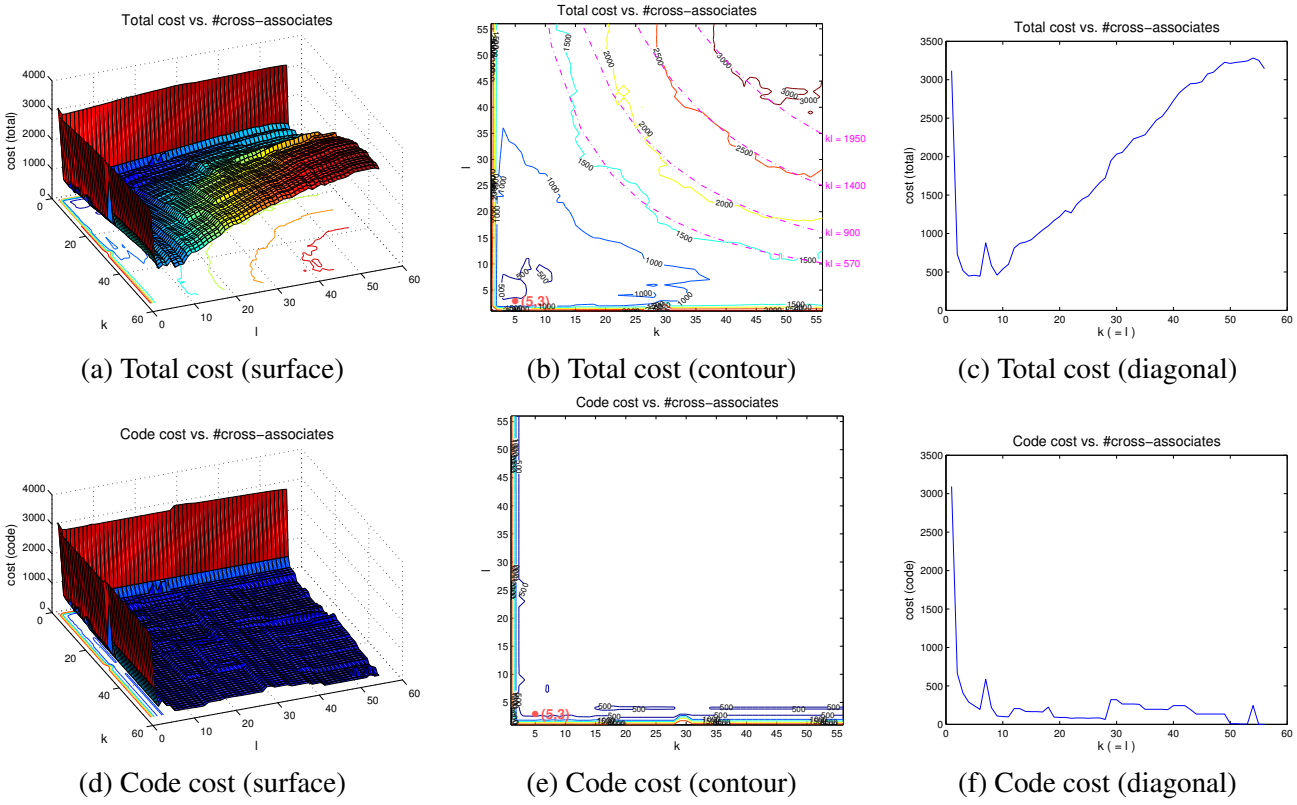
**Theorem 4.1.** *For $t \geq 1$,*

$$\sum_{i=1}^{k} \sum_{j=1}^{\ell} C(D^t_{i,j}) \geq \sum_{i=1}^{k} \sum_{j=1}^{\ell} C(D^{t+1}_{i,j}) \geq \sum_{i=1}^{k} \sum_{j=1}^{\ell} C(D^{t+2}_{i,j}).$$

*In words,* REGROUP *never increases the objective function (Eq. 3).*

*Proof.* We shall only prove the first inequality, the second inequality will follow by symmetry between rows and columns.

$$\begin{aligned}
&\sum_{i=1}^{k} \sum_{j=1}^{\ell} C(D^t_{i,j}) \\
&= \sum_{i=1}^{k} \sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(D^t_{i,j}) \log \frac{1}{P^t_{i,j}(u)} \\
&= \sum_{i=1}^{k} \sum_{j=1}^{\ell} \sum_{u=0}^{1} \left[ \sum_{x:\Psi^t(x)=i} n_u(x^j) \right] \log \frac{1}{P^t_{i,j}(u)} \\
&= \sum_{i=1}^{k} \sum_{x:\Psi^t(x)=i} \left[ \sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(x^j) \log \frac{1}{P^t_{i,j}(u)} \right] \\
&\overset{(a)}{\geq} \sum_{i=1}^{k} \sum_{x:\Psi^t(x)=i} \left[ \sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(x^j) \log \frac{1}{P^t_{\Psi^{t+1}(x),j}(u)} \right] \\
&\overset{(b)}{=} \sum_{i=1}^{k} \sum_{x:\Psi^{t+1}(x)=i} \left[ \sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(x^j) \log \frac{1}{P^t_{\Psi^{t+1}(x),j}(u)} \right]
\end{aligned}$$




(a) Total cost (surface) (b) Total cost (contour) (c) Total cost (diagonal)

(d) Code cost (surface) (e) Code cost (contour) (f) Code cost (diagonal)

Figure 3: General shape of the total cost (number of bits) versus number of cross-associates (synthetic cave graph with three square caves of sizes 32, 16 and 8, with 1% noise). The "waterfall" shape (with the description and code costs dominating the total cost in different regions) illustrates the intuition behind our model, as well as why our minimization strategy is effective.

$$
\begin{aligned}
&= \sum_{i=1}^{k}\sum_{j=1}^{\ell}\sum_{u=0}^{1}\left[\sum_{x:\Psi^{t+1}(x)=i} n_u(x^j)\right]\log\frac{1}{P^t_{i,j}(u)} \\
&= \sum_{i=1}^{k}\sum_{j=1}^{\ell}\sum_{u=0}^{1} n_u(D^{t+1}_{i,j})\log\frac{1}{P^t_{i,j}(u)} \\
&\overset{(c)}{\geq} \sum_{i=1}^{k}\sum_{j=1}^{\ell}\sum_{u=0}^{1} n_u(D^{t+1}_{i,j})\log\frac{1}{P^{t+1}_{i,j}(u)} \\
&= \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(D^{t+1}_{i,j})
\end{aligned}
$$

where (a) follows from Step 2 of REGROUP; (b) follows by re-writing the outer two sums–since $i$ is not used anywhere inside the $[\cdots]$ terms; and (c) follows from the non-negativity of the Kullback-Leibler distance. □

**Remarks** Instead of batch updates, sequential updates are also possible. Also, rows and columns need not alternate in the minimization. We have many locally good moves available (based on Theorem 4.1) which require only linear time.

It is possible that REGROUP may cause some groups to be empty, i.e., $a_i = 0$ or $b_j = 0$ for some $1 \le i \le m$, $1 \le j \le n$ (to see that, consider e.g., a homogeneous matrix; then we always end up with one group). In other words, we may find $k$ and $\ell$ less than those specified. Finally, we can easily avoid infinite quantities in Eq. 4 by using, e.g., $(n_u(A) + 1/2)/(n(A) + 1)$ for $P_A(u)$, $u = 0, 1$.

**Initialization** If we want to use REGROUP (inner loop) by itself, we have to initialize the mappings $(\Phi, \Psi)$. For $\Phi$, the simplest approach is to divide the rows evenly into $k$ initial "groups," taking them in their original order. For $\Psi$ we do the initialization in the same manner. This often works well in practice. A better approach is to divide the "residual masses" (i.e., marginal sums of each column) evenly among $k$ groups, taking the rows in order of increasing mass (and similarly for $\Psi$). The initialization in Figure 2 is mass-based.

However, our CROSSASSOCIATIONSEARCH (outer loop) algorithm, described in the next section, is an even better alternative. We start with $k = \ell = 1$, increase $k$ and $\ell$ and create new groups, *taking into account* the cross-associations up to that point. This tightly integrated group creation scheme, that reuses current REGROUP row and column group assignments, yields much better results.

**Complexity** The algorithm is $O(n_1(D) \cdot (k + \ell) \cdot I)$ where $I$ is the number of iterations. In step (2) of the algorithm, we access each row and count their nonzero elements (of which there are $n_1(d)$ in total), then consider $k$ possible candidate row groups to place it into. Therefore, an iteration over rows is $O(n_1(D) \cdot k)$. Similarly, an iteration over columns (step 4) is $O(n_1(D) \cdot \ell)$. There is a total of $I/2$ row and $I/2$ column iterations. All this adds up to $O(n_1(D) \cdot (k + \ell) \cdot I)$.

### 4.2 Search for $k$ and $\ell$ (CROSSASSOCIATIONSEARCH)

The last part of our approach is an algorithm to look for good values of $k$ and $\ell$. Based on our cost model (Eq. 2), we have a way to attack this problem. As we discuss later, the cost function usually has a "waterfall" shape (see Figure 3), with a sharp drop for small $k$ and $\ell$, and an ascent afterwards. Thus, it makes sense to start with small values of $k, \ell$, progressively increase them, and keep rearranging rows and columns based on fast, local moves in the search space (REGROUP). We experimented with several search strategies, and obtained good results with the following algorithm.

Algorithm CROSSASSOCIATIONSEARCH:

1. Let $T$ denote the search iteration index. Start with $T = 0$ and $k^0 = \ell^0 = 1$.
2. **[Outer loop]** At iteration $T$, try to increase the number of row groups. Set $k^{T+1} = k^T + 1$. Split the row group $r$ with maximum entropy per row, i.e.,

$$r := \arg\max_{1 \le i \le k} \sum_{1 \le j \le \ell} \frac{n(D_{i,j}) H\big(P_{D_{i,j}}(0)\big)}{a_i}.$$

Construct an initial label map $\Psi_0^{T+1}$ as follows: For every row $x$ in row group $r$ (i.e., for every $1 \le x \le m$ such that $\Psi^T(x) = r$), place it into the new group $k^{T+1}$ (i.e., set $\Psi_0^{T+1}(x) = k^{T+1}$) if and only if it decreases the per-row entropy of group $r$, i.e., if and only if

$$\sum_{1 \le j \le \ell} \frac{n(D'_{r,j}) H\big(P_{D'_{r,j}}(0)\big)}{a_r - 1} < \sum_{1 \le j \le \ell} \frac{n(D_{r,j}) H\big(P_{D_{r,j}}(0)\big)}{a_r}, \tag{5}$$

where $D'_{r,j}$ is $D_{r,j}$ without row $x$. Otherwise, we let $\Psi_0^{T+1}(x) = r = \Psi^T(x)$. If we move the row to the new group, we update $D_{r,j}$ (for all $1 \le j \le \ell$) by removing row $x$ (for subsequent estimations of Eq. 5).

| Dataset | Dim. $(a \times b)$ | $n_1(A)$ |
|---|---|---|
| CAVE | 810×900 | 162,000 |
| CAVE-Noisy | 810×900 | 171,741 |
| CUSTPROD | 295×30 | 5,820 |
| CUSTPROD-Noisy | 295×30 | 5,602 |
| NOISE | 100×100 | 952 |
| CLASSIC | 3,893×4,303 | 176,347 |
| GRANTS | 13,297×5,298 | 805,063 |
| EPINIONS | 75,888×75,888 | 508,960 |
| CLICKSTREAM | 23,396×199,308 | 952,580 |
| OREGON | 11,461×11,461 | 65,460 |

Table 2: Dataset characteristics.

3. **[Inner loop]** Use REGROUP with initial cross-associations $(\Psi_0^{T+1}, \Phi^T)$ to find new ones $(\Psi^{T+1}, \Phi^{T+1})$ and the corresponding total cost.
4. If there is no decrease in total cost, stop and return $(k^*, \ell^*) = (k^T, \ell^T)$—with corresponding cross-associations $(\Psi^T, \Phi^T)$. Otherwise, set $T = T + 1$ and continue.

5–7. Similar to steps 2–4, but trying to increase column groups instead.

---

Figure 1 shows the search algorithm in action. Starting from the initial matrix (CAVES), we successively increase the number of column and row groups. For each such increase, the columns are shifted using REGROUP. The algorithm successfully stops after iteration pair 4 (Figure 1(e)).

**Lemma 4.1.** *If $D = [D_1 D_2]$, then $C(D_1) + C(D_2) \leq C(D)$.*

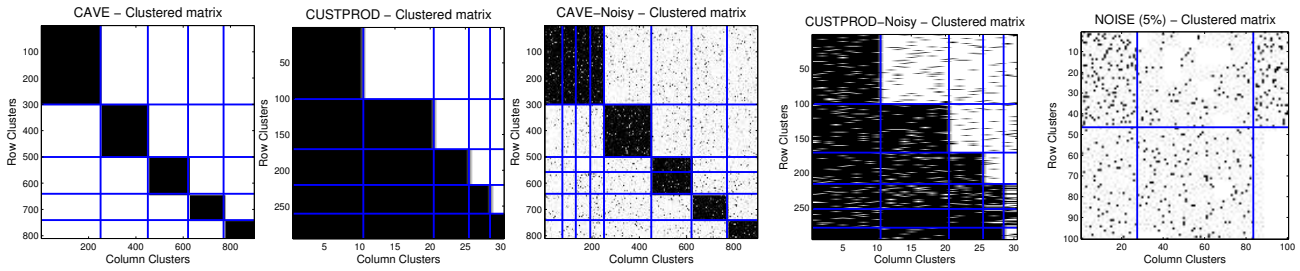
*Proof.* We have

$$\begin{aligned} C(D) &= n(D)H\big(P_D(0)\big) = n(D)H\left(\frac{n_0(D)}{n(D)}\right) \\ &= n(D)H\left(\frac{P_{D_1}(0)n(D_1) + P_{D_2}(0)n(D_2)}{n(D)}\right) \\ &\geq n(D_1)H\big(P_{D_1}(0)\big) + n(D_2)H\big(P_{D_1}(0)\big) \\ &= C(D_1) + C(D_2), \end{aligned}$$

where the inequality follows from the concavity of $H(\cdot)$ and the fact that $n(D_1)+n(D_2) = n(D)$ or $n(D_1)/n(D)+ n(D_2)/n(D) = 1$. □

Note that the original code cost is zero only for a completely homogeneous matrix. Also, the code length for $(k, l) = (a, b)$ is, by definition, zero. Therefore, provided that the fraction of non-zeros is not the same for every column (and since $H(\cdot)$ is strictly concave), the next observation follows immediately.

**Corollary 4.1.** *For any $k_1 \geq k_2$ and $\ell_1 \geq \ell_2$, there exists cross-associations such that $(k_1, \ell_1)$ leads to a shorter code (Eq. 3).*




(a) CAVE (b) CUSTPROD (c) CAVE−Noisy (d) CUSTPROD−Noisy (e) NOISE (5%)

Figure 4: *Cross-associations on synthetic datasets*: Our method gives the intuitively correct cross-associations for (a) CAVE and (b) CUSTPROD. In the noisy versions (c, d), few extra groups are found due to patterns that emerge, such as the "almost-empty" and "more-dense" cross-associations for pure NOISE (e).

By Corollary 4.1, the outer loop in CROSSASSOCIATIONSEARCH decreases the objective cost function. By Theorem 4.1 the same holds for the inner loop (REGROUP). Therefore, the entire algorithm CROSSASSOCIATIONSEARCH also decreases the objective cost function (Eq. 3). However, the description complexity evidently increases with $(k, \ell)$. We have found that, in practice, this search strategy performs very well. Figure 3 (discussed in Section 5.1) provides an indication why this is so.

**Complexity** Since at each step of the search we increase either $k$ or $\ell$, the sum $k + \ell$ always increases by one. Therefore, the overall complexity of the search is $O(n_1(D)(k^* + \ell^*)^2)$, if we ignore the number of REGROUP iterations $I$ (in practice, $I \leq 20$ is always sufficient).

# 5 Experiments

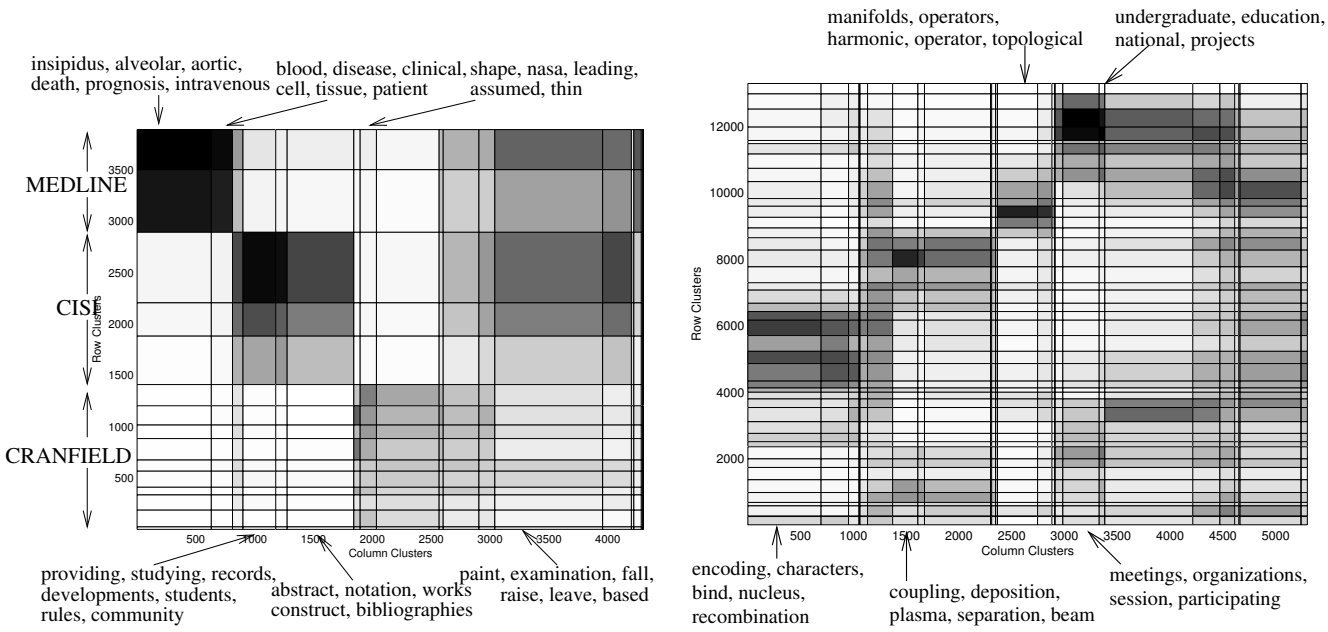
We did experiments to answer two key questions: (i) how good is the quality of the results (which involves both the proposed criterion and the minimization strategy), and (ii) how well does the method scale up. To the best of our knowledge, in the literature to date, no other method has been explicitly proposed and studied for parameter-free, joint clustering of binary matrices.

We used several datasets (see Table 2), both real and synthetic. The synthetic ones were: **(1)** CAVE, representing a social network of "cavemen" [30], that is, a block-diagonal matrix of variable-size blocks (or "caves"), **(2)** CUSTPROD, representing groups of customers and their buying preferences[2], **(3)** NOISE, with pure white noise. We also created noisy versions of CAVE and CUSTPROD (CAVE−Noisy and CUSTPROD−Noisy), by adding noise (10% of the number of non-zeros).

The real datasets are: **(1)** CLASSIC, Usenet documents (Cornell's SMART collection [10]), **(2)** GRANTS, 13,297 documents (NSF grant proposal abstracts) from several disciplines (physics, bio-informatics, etc.), **(3)** EPINIONS, a who-trusts-whom social graph of www.epinions.com users [31], **(4)** CLICKSTREAM, with users and URLs they clicked on [32], and **(5)** OREGON, with connections between Autonomous Systems (AS) in the Internet.

Our implementation was done in MATLAB (version 6.5 on Linux) using sparse matrices. The experiments were performed on an Intel Xeon 2.8GHz machine with 1GB RAM.

[2]We try to capture market segments with heavily overlapping product preferences, like, say, "single persons", buying beer and chips, "couples," buying the above plus frozen dinners, "families," buying all the above plus milk, etc.




(a) CLASSIC cross-associates ($k^* = 15, \ell^* = 19$)

(b) GRANTS cross-associates ($k^* = 41, \ell^* = 28$)

Figure 5: *Cross-associations for* CLASSIC *and* GRANTS: Due to the dataset sizes, we show the Cross-associations via shading; darker shades correspond denser blocks (more ones). We also show the most frequently occurring words for several of the word (column) groups.

## 5.1 Quality

**Total code length criterion** Figure 3 illustrates the intuition behind both our information-theoretic cost model, as well as our minimization strategy. It shows the general shape of the total cost (in number of bits) versus the number of cross-associates. For this graph, we used a "caveman" matrix with three caves of sizes 32, 16 and 8, adding noise (1% of non-zeros). We used REGROUP, forcing it to never empty a group. The slight local jaggedness in the plots is due to the presence of noise and occasional local minima hit by REGROUP.

However, the figure reveals nicely the overall, global shape of the cost function. It has a "waterfall" shape, dropping very fast initially, then rising again as the number of cross-associates increases. For small $k, \ell$, the code cost dominates the description cost (in bits), while for large $k, \ell$ the description cost is the dominant one. The key points, regarding the model as well as the search strategies, are:

- The optimal $(k^*, \ell^*)$ is the "sweet spot" balancing these two. The trade-off between description complexity and code length indeed has a desirable form, as expected.
- As expected, cost iso-surfaces roughly correspond to $k \cdot \ell = \text{const.}$, i.e., to constant number of cross-associates.
- Moreover, for relatively small $(k, \ell)$, the code cost clearly dominates the total cost by far, which justifies our choice of objective function (Eq. 3).
- The overall, well-behaved shape also demonstrates that the cost model is amenable to efficient search for a minimum, based on the proposed linear-time, local moves.
- It also justifies why starting the search with $k = \ell = 1$ and gradually increasing them is an effective approach: we generally find the minimum after a few CROSSASSOCIATIONSEARCH (outer loop) iterations.

| Clusters found | Document class | | | Precision |
|---|---|---|---|---|
| | CRANFIELD | CISI | MEDLINE | |
| 1 | 0 | 1 | 390 | 0.997 |
| 2 | 2 | 676 | 9 | 0.984 |
| 3 | 0 | 0 | 610 | 1.000 |
| 4 | 1 | 317 | 6 | 0.978 |
| 5 | 188 | 0 | 0 | 1.000 |
| 6 | 207 | 0 | 0 | 1.000 |
| 7 | 3 | 452 | 16 | 0.960 |
| 8 | 131 | 0 | 0 | 1.000 |
| 9 | 209 | 0 | 0 | 1.000 |
| 10 | 107 | 2 | 0 | 0.982 |
| 11 | 152 | 3 | 2 | 0.968 |
| 12 | 74 | 0 | 0 | 1.000 |
| 13 | 139 | 9 | 0 | 0.939 |
| 14 | 163 | 0 | 0 | 1.000 |
| 15 | 24 | 0 | 0 | 1.000 |
| Recall | 0.996 | 0.990 | 0.968 | |

Table 3: The clusters for CLASSIC (see Figure 5(a)) recover the known document classes. Furthermore, our approach also captures *unknown* structure (such as the "technical" and "everyday" medical terms).

**Results—synthetic data** Figure 4 depicts the cross-associations found by our method on several synthetic datasets. For the noise-free synthetic matrices CAVE and CUSTPROD, we get exactly the intuitively correct groups. This serves as a sanity check for our whole approach (criterion plus heuristics). When noise is present, we find some extra groups which, on closer examination, are picking up patterns in the noise. This is expected: it is well known that spurious patterns emerge, even when we have pure noise. Figure 4(e) confirms it: even in the NOISE matrix, our algorithm finds blocks of clearly lower or higher density.

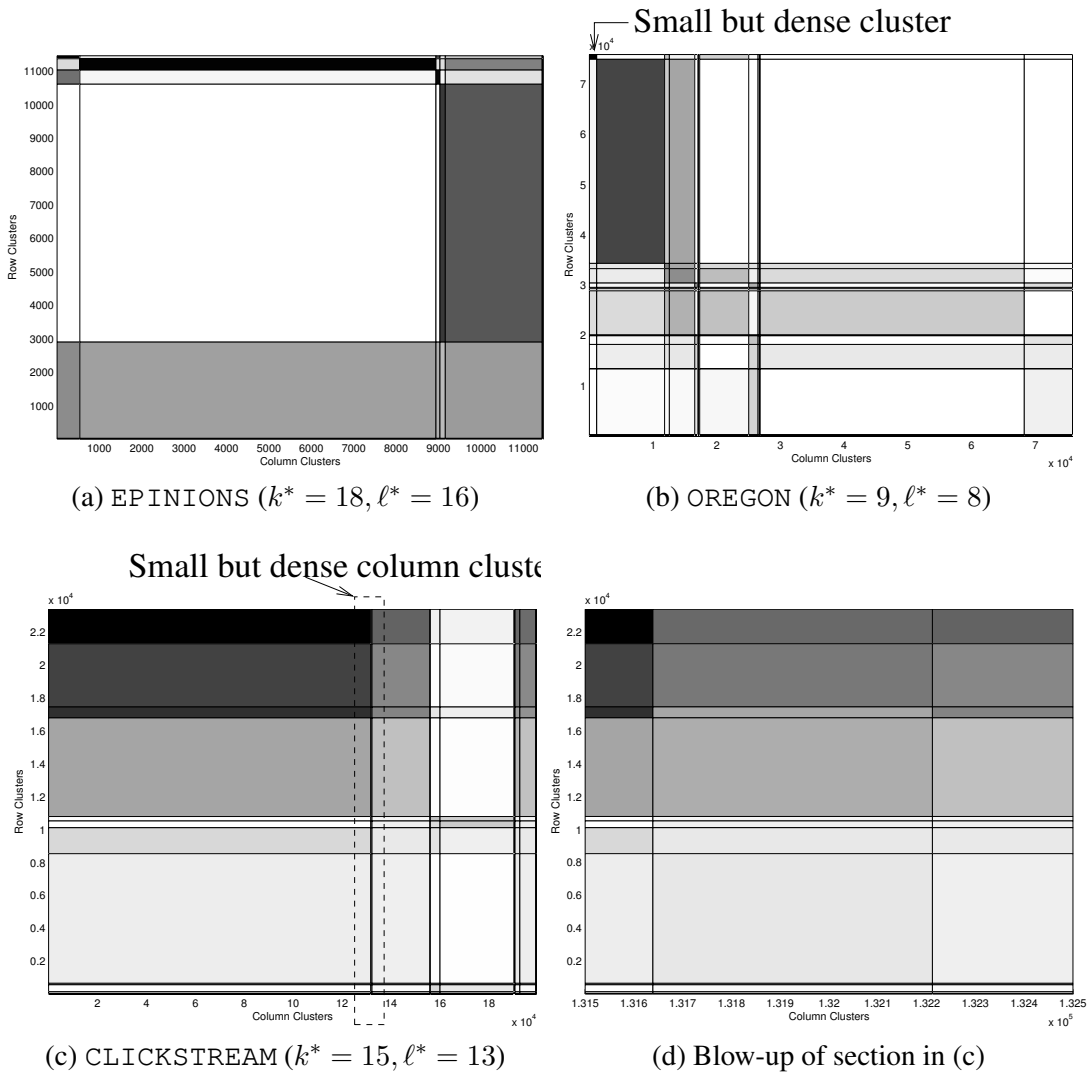
**Results—real data** Figures 5 and 6 show the cross-associations found on several real-world datasets. They demonstrate that our method gives intuitive results.

Figure 5(a) shows the CLASSIC dataset, where the rows correspond to documents from MEDLINE (medical journals), CISI (information retrieval) and CRANFIELD (aerodynamics); and the columns correspond to words.

First, we observe that the cross-associates are in agreement with the known document classes (left axis annotations). We also annotated some of the column groups with their most frequent words. Cross-associates belonging to the same document (row) group clearly follow similar patterns with respect to the word (column) groups. For example, the MEDLINE row groups are most strongly related to the first and second column groups, both of which are related to medicine. ("insipidus," "alveolar," "prognosis" in the first column group; "blood," "disease," "cell," etc, in the second).

Besides being in agreement with the known document classes, the cross-associates *reveal further structure* (see Table 3). For example, the first word group consists of more "technical" medical terms, while second group consists of "everyday" terms, or terms that are used in medicine often, but not exclusively[3]. Thus, the second word group is more likely to show up in other document groups (and indeed it does, although not immediately apparent in the figure), which is why our algorithm separates the two.

[3]This observation is also true for nearly all of the (approximately) 600 and 100 words belonging to each group, not only the most frequent ones shown here.




(a) EPINIONS ($k^* = 18, \ell^* = 16$) (b) OREGON ($k^* = 9, \ell^* = 8$)

(c) CLICKSTREAM ($k^* = 15, \ell^* = 13$) (d) Blow-up of section in (c)

Figure 6: *Cross-associations for* EPINIONS*,* OREGON *and* CLICKSTREAM: The matrices are organized successfully in homogeneous regions. (d) shows that our method captures dense clusters, irrespective of their size.

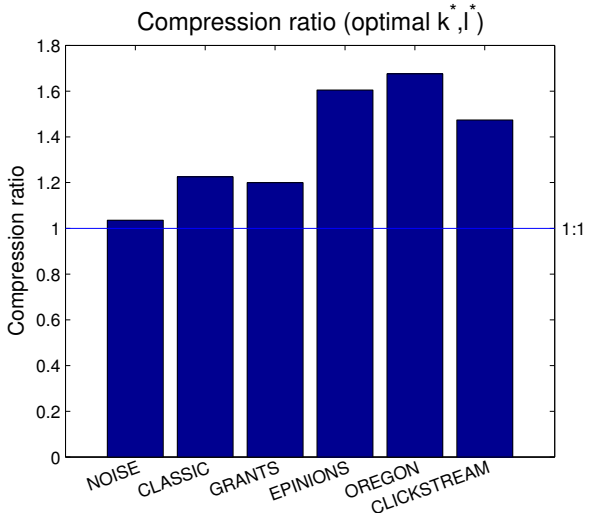
Figure 5(b) shows GRANTS, which consists of NSF grant proposal abstracts in several disciplines, such as genetics, mathematics, physics, organizational studies. Again, the terms are meaningfully grouped: e.g., those related to biology ("encoding," "recombination," etc.), to physics ("coupling," "plasma," etc.) and to material sciences.

We also present experiments on matrices from other settings: social networks (EPINIONS), computer networks (OREGON) and web visit patterns (CLICKSTREAM). In all cases, our algorithm organizes the matrices in homogeneous regions. Also, in EPINIONS, notice that there is a small but dense cluster, probably corresponding to a dense clique of experts that they mainly trust each other. The large gray rectangle should correspond to another, much larger, but less coherent, group of people.

**Compression and density** Figure 7 lists the compression ratios achieved by our cross-association algorithms for each dataset. Figure 8 shows how our algorithm effectively divides the CLASSIC matrix in sparse and dense

| Dataset | Average cost per element | | |
|---|---|---|---|
| | $k = \ell = 1$ | Optimal $k^*, \ell^*$ | |
| CAVE | 0.766 | 0.00065 | (1:1178) |
| CAVE-Noisy | 0.788 | 0.1537 | (1:5.1) |
| CUSTPROD | 0.930 | 0.0320 | (1:29) |
| CUSTPROD-Noisy | 0.952 | 0.3814 | (1:2.5) |
| NOISE | 0.2846 | 0.2748 | (1:1.03) |
| CLASSIC | 0.0843 | 0.0688 | (1:1.23) |
| GRANTS | 0.0901 | 0.0751 | (1:1.20) |
| EPINIONS | 0.0013 | 0.00081 | (1:1.60) |
| OREGON | 0.0062 | 0.0037 | (1:1.68) |
| CLICKSTREAM | 0.0028 | 0.0019 | (1:1.47) |

(a) All datasets




(b) Real datasets

Figure 7: Compression ratios.

regions (i.e., cross-associates), thereby summarizing its structure.

### 5.2 Scalability

Figure 9 shows wall-clock times (in seconds) of our MATLAB implementation. In all plots, the datasets were cave-graphs with three caves. For the noiseless case (b), times for both REGROUP and CROSSASSOCIATIONSEARCH increase linearly with respect to number of non-zeros. We observe similar behavior for the noisy case (c). The "sawtooth" patterns are explained by the fact that we used a new matrix for each case. Thus, it was possible for some graphs to have different "regularity" (spuriously emerging patterns), and thus compress better and faster. Indeed, when we approximately scale by the number of inner loop iterations in CROSSASSOCIATIONSEARCH, an overall linear trend (with variance due to memory access overheads in MATLAB) appears.

Finally, Figure 10 shows the progression of total cost (in bits) for every iteration of CROSSASSOCIATION-SEARCH (outer loop). We clearly see that our algorithm quickly finds better cross-associations. These plots are from the same wall-clock time experiments.

## 6 Conclusions

We have proposed one of the few methods for clustering and graph partitioning, that needs *no* "magic numbers."

- Besides being fully automatic, our approach satisfies all properties (P1)–(P3): it finds row and column groups simultaneously and scales linearly with problem size.
- We introduce a novel approach and propose a general, intuitive model founded on compression and information-theoretic principles.
- We provide an integrated, two-level framework to find cross-associations, consisting of REGROUP (inner loop) and CROSSASSOCIATIONSEARCH (outer loop).
- We give an effective search strategy to minimize the total code length, taking advantage of the cost function properties ("waterfall" shape).

Also, our method is easily extensible to matrices with categorical values. We evaluate our method on several real and synthetic datasets, where it produces intuitive results.

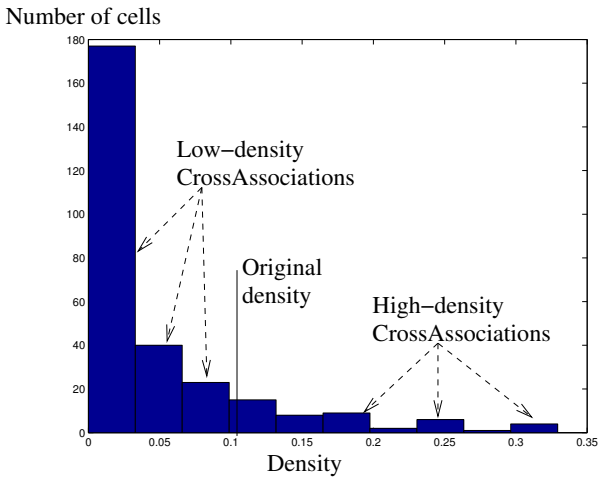



Figure 8: The algorithm splits the original CLASSIC matrix into homogeneous, high and low density Cross-associations.

## References

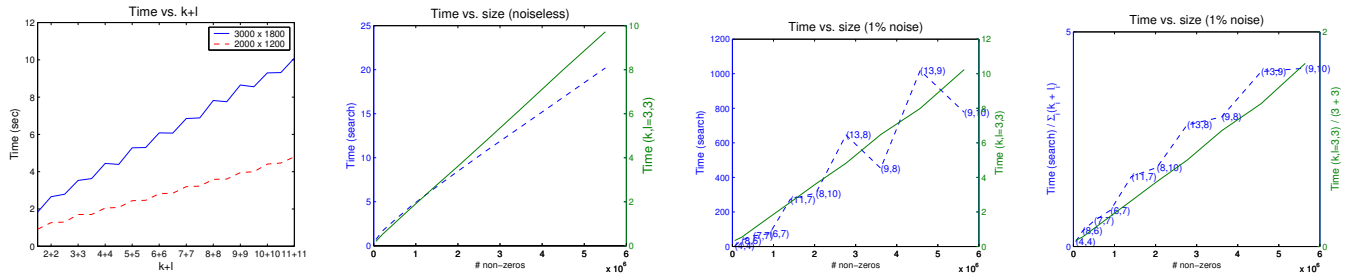

[1] D. Pelleg and A. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *Proc. 17th ICML*, pp. 727–734, 2000.

[2] G. Hamerly and C. Elkan, "Learning the $k$ in $k$-means," in *Proc. 17th NIPS*, 2003.

[3] B. Zhang, M. Hsu, and U. Dayal, "K-harmonic means—a spatial clustering algorithm with boosting," in *Proc. 1st TSDM*, pp. 31–45, 2000.

[4] I. S. Dhillon and D. S. Modha, "Concept decom- positions for large sparse text data using clustering," *Mach. Learning*, vol. 42, pp. 143–175, 2001.

[5] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," in *Proc. SIGMOD*, pp. 73–84, 1998.


(a) Time vs. $k + \ell$ (b) Noiseless (c) Noisy (1%) (d) Noisy, scaled

Figure 9: (a) Wall-clock time for one row and column swapping (step (2) and (4)) vs. $k+\ell$ is linear (shown for two different matrix sizes, with $n_1(D) = 0.37n(D)$). (b,c) Wall-clock time vs. number of non-zeros, for CROSSASSOCIATIONSEARCH (dashed) and for REGROUP with $(k, l) = (3, 3)$ (solid). The stopping values $(k^*, \ell^*)$ are shown on plots (c,d), if different from $(3, 3)$. (d) Wall-clock times of plot (c), scaled by $\propto 1/(k^* + \ell^*)^2$.

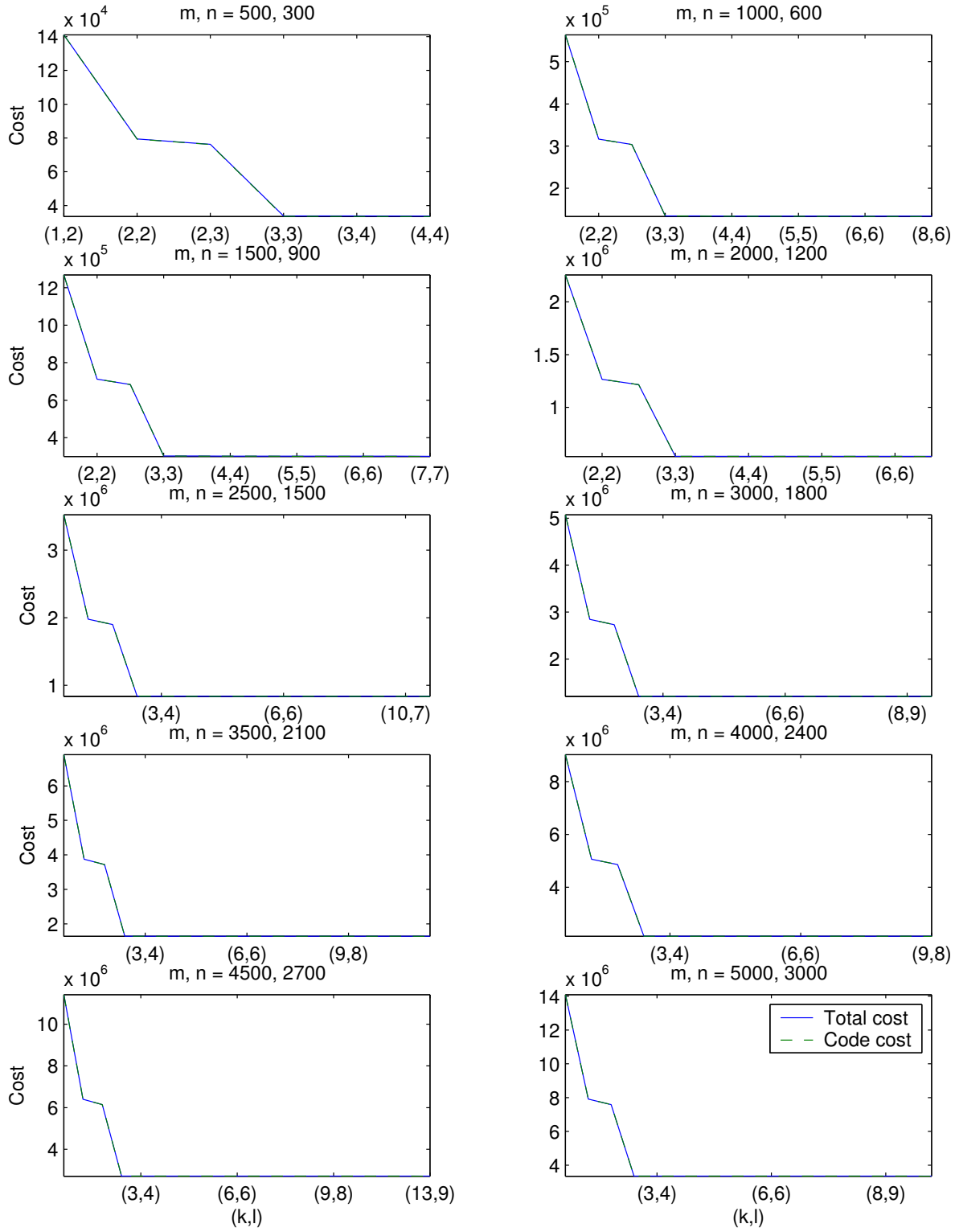



Figure 10: Progression of cost during search on synthetic cave graphs (for varying sizes); see also Figure 9(b).

[6] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. SIGMOD*, pp. 103–114, 1996.

[7] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *IEEE Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[8] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. 4th KDD*, pp. 58–65, 1998.

[9] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[10] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proc. 9th KDD*, pp. 89–98, 2003.

[11] M. M. Madiman, M. Harrison, and I. Kontoyiannis, "A minimum description length proposal for lossy data compression," in *Proc. IEEE ISIT*, 2004.

[12] N. Friedman, O. Mosenzon, N. Slonim, and N. Tishby, "Multivariate information bottleneck," in *Proc. 17th UAI*, pp. 152–161, 2001.

[13] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th VLDB*, pp. 487–499, 1994.

[14] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Min. Knowl. Discov.*, vol. 8, no. 1, pp. 53–87, 2004.

[15] A. Tuzhilin and G. Adomavicius, "Handling very large numbers of association rules in the analysis of microarray data," in *Proc. 8th KDD*, 2002.

[16] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *JASI*, vol. 41, pp. 391–407, 1990.

[17] T. G. Kolda and D. P. O'Leary, "A semidiscrete matrix decomposition for latent semantic indexing information retrieval," *ACM Transactions on Information Systems*, vol. 16, no. 4, pp. 322–346, 1998.

[18] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. 22nd SIGIR*, pp. 50–57, 1999.

[19] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," in *Proc. 17th PODS*, 1998.

[20] G. Karypis and V. Kumar, "Multilevel algorithms for multi-constraint graph partitioning," in *Proc. SC98*, pp. 1–13, 1998.

[21] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. NIPS*, pp. 849–856, 2001.

[22] N. Mishra, D. Ron, and R. Swaminathan, "On finding large conjunctive clusters," in *Proc. 16th COLT*, 2003.

[23] P. K. Reddy and M. Kitsuregawa, "An approach to relate the web communities through bipartite graphs," in *Proc. 2nd WISE*, pp. 302–310, 2001.

[24] C. Tang and A. Zhang, "Mining multiple phenotype structures underlying gene expression profiles," in *Proc. CIKM03*, pp. 418–425, 2003.

[25] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

[26] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Res. Dev.*, vol. 20, no. 3, pp. 198–203, 1976.

[27] J. Rissanen and G. G. Langdon Jr., "Arithmetic coding," *IBM J. Res. Dev.*, vol. 23, pp. 149–162, 1979.

[28] I. H. Witten, R. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Comm. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[29] J. Rissanen, "Universal prior for integers and estimation by minimum description length," *Annals of Statistics*, vol. 11, no. 2, pp. 416–431, 1983.

[30] D. J. Watts, *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton Univ. Press, 1999.

[31] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *Proc. 2nd ISWC*, pp. 351–368, 2003.

[32] A. L. Montgomery and C. Faloutsos, "Identifying web browsing trends and patterns," *IEEE Computer*, vol. 34, no. 7, pp. 94–95, 2001.