# Finding biclusters in gene expression data by random projections

Stefano Lonardi
Dept. of Computer Science
University of California
Riverside, CA 92521

Wojciech Szpankowski
Dept. of Computer Sciences
Purdue University
West Lafayette, IN 47907

Qiaofeng Yang
Dept. of Computer Science
University of California
Riverside, CA 92521

**Abstract**

Given a matrix $X$ composed of symbols, a bicluster is a submatrix obtained by removing some of the rows and some of the columns of $X$ in such a way that each row of what is left reads the same string. In this paper, we are concerned with the problem of finding the bicluster with the largest area in a large matrix $X$. The problem is first proven to be **NP**-complete. We present a fast and efficient randomized algorithm that discovers the largest bicluster by random projections. A detailed probabilistic analysis of the algorithm and an asymptotic study on the statistical significance of the solutions are given. We report results of simulations on synthetic data and preliminary experimental results on the analysis of the leukemia gene expression dataset.

## 1 Introduction

Clustering refers to the problem of finding a partition of $n$ vectors in an $m$-dimensional space, such that the vectors in each cluster are "close" to one another (according to some predefined distance). The large majority of clustering algorithms do not perform well on high dimensional spaces because in such spaces data becomes very sparse (see, e.g., [2, 1]).

Recent research has been focused on the problem of finding hidden sub-structures in large matrices composed by thousands of high dimensional vectors (see, e.g., [4, 25, 24, 3, 27, 22, 18, 21]). This problem is known as *biclustering*. In biclustering, one is interested in determining the similarity in a *subset* of the dimensions (subset that has to be determined as well). Although there exists several distinct definitions of a bicluster, the problem can be informally described as the problem of finding a partition of the vectors and a subset of the dimensions such that the projections along those directions of the vectors in each cluster are close to one another. The problem requires to cluster the vectors and the dimensions simultaneously, thus the name "biclustering".

Biclustering has important applications in several areas, such as data mining, machine learning, and pattern recognition. Data arising from text analysis, market-basket data analysis, web logs, etc., is usually arranged in a contingency table or co-occurrence table, such as, a word-document table, a product-user table, a cpu-job table or a webpage-user table. Discovering a large bicluster in a product-user matrix indicates, for example, which users share the same preferences. Finding biclusters has therefore applications in recommender systems and collaborative filtering, identifying web communities, load balancing, discovering association rules, among others.

The application domain of primary interest here is the analysis of gene expression data obtained from microarray experiments. Gene expression data is typically arranged in a table with rows corresponding to genes, and columns corresponding to patients, tissues, time points, etc. The classical approach to analyze microarray data is clustering. The process of clustering partitions

genes into mutually exclusive clusters under the assumption that genes that are involved in the same genetic pathway behave similarly across all the testing conditions. The assumption might be true when the testing conditions are associated with time points. However, when the testing conditions are heterogeneous, such as patients or tissues, the previous assumption is not appropriate anymore. One would expect that a group of genes would exhibit similar expression patterns only in a subset of conditions, such as the subset of patients suffering from the same type of disease. Under this circumstance, biclustering becomes the alternative to the traditional clustering paradigm. The results of biclustering may enable one to discover hidden structures in gene expression data in which many genetic pathways might be embedded. It might also allow one to uncover unknown genetic pathways, or to assign functions to unknown genes in already known genetic pathways.

Biclustering is indeed, not a new problem. In fact, it is also known under several other names, namely "co-clustering", "two-way clustering" and "direct clustering". The problem was first introduced in the seventies in a paper by Hartigan [13]. Almost thirty years later, Cheng and Church [4] raised the interest on this problem for applications in gene expression data analysis.

Several other researchers studied the problem recently. Wang *et al.* propose the *pCluster* model that is capable of discovering shifting or scaling patterns from raw data sets [25]. Tanay *et al.* [24] combine a graph-theoretic approach with a statistical modeling of the data to discover biclusters in large gene expression datasets. Ben-Dor *et al.* [3] introduce a new notion of a bicluster called *order preserving submatrix*, which is a group of genes whose expression level induces a linear ordering across a subset of the conditions. Murali and Kasif [19] (see also [21]) propose the concept of *xmotif*, which is defined as a subset of genes whose expression is simultaneously conserved for a subset of samples.

As we were writing this document, we became aware of two other very recent contributions to the subject, by Sheng *et al.* [22], and Mishra *et al.* [18], that use a randomized approach similar with the work described here. Sheng *et al.* [22] propose a randomized algorithm based on Gibbs sampling to discover large biclusters in gene expression data. Their model of a bicluster is probabilistic, that is, each entry of the matrix is associated with a probability. Mishra *et al.* [18] are concerned with the problem of finding $\epsilon$-bicliques which maximizes the number of edges[1]. The authors give a randomized algorithm but no experimental results are reported.

As in the papers by Murali and Kasif [19] and Sheng *et al.* [22], our approach to biclustering gene expression data requires the discretization of the gene expression matrix into a matrix over a finite alphabet. The simplifying assumption is that the set of states in which each gene operates is finite, such as up-regulated, down-regulated or unchanged. Once the data is discretized into strings where each symbol corresponds to a state, the biclustering problem reduces to the problem of finding a subset of the rows and a subset of the columns such that the submatrix induced has the property that each row reads the same string. Such a submatrix would therefore correspond to a group of genes that exhibit a coherent pattern of states over a subset of conditions. This is indeed the formulation of the problem that we define in Section 2, which is first proven to be **NP**-complete. In Section 3 we present a randomized algorithm which is efficient, and easy to understand and implement. Section 4 presents an asymptotic analysis that allows one to determine the statistical significance of the solution. Finally, in Section 5 we report simulation results on synthetic data and preliminary results on the analysis of gene expression data.

[1] the connection between bicliques and bicluster will be explained in detail in Section 2

## 2 Notations and problem definition

We use standard concepts and notation about strings. The set $\Sigma$ denotes a nonempty *alphabet* of *symbols* and a *string* over $\Sigma$ is an ordered sequence of symbols from the alphabet. We use the variable $a$ as a shorthand for the cardinality of the set $\Sigma$, that is, $a = |\Sigma|$. Given a string $x$, the number of symbols in $x$ defines the *length* $|x|$ of $x$.

We write $x_{[i]}$, $1 \leq i \leq |x|$ to indicate the $i$-th symbol in $x$. We use $x_{[i,j]}$ shorthand for the substring $x_{[i]}x_{[i+1]} \dots x_{[j]}$ where $1 \leq i \leq j \leq |x|$, with the convention that $x_{[i:i]} = x_{[i]}$. Substrings in the form $x_{[1:j]}$ correspond to the *prefixes* of $x$, and substrings in the form $x_{[i:n]}$ to the *suffixes* of $x$.

Similarly, we can define a two-dimensional $n \times m$ string (or matrix) $X \in \Sigma^{n \times m}$ over the alphabet $\Sigma$. The element $(i, j)$ of $X$ is denoted by $X_{[i,j]}$. A contiguous submatrix from row $i_1$ to row $i_2$ and from column $j_1$ to column $j_2$ is denoted by $X_{[i_1:i_2,j_1:j_2]}$.

A *row selection* of size $k$ of $X$ is defined as the subset of the rows $R = \{i_1, i_2, \dots, i_k\}$, where $1 \leq i_s \leq n$ for all $1 \leq s \leq k$. Similarly, a *column selection* of size $l$ of $X$ is defined as a subset of the columns $C = \{j_1, j_2, \dots, j_l\}$, where $1 \leq j_t \leq m$ for all $1 \leq t \leq l$.

The submatrix $X_{(R,C)}$ *induced* by the pair $(R, C)$ is defined as the matrix

$$X_{(R,C)} = \begin{vmatrix} X_{[i_1,j_1]} & X_{[i_1,j_2]} & \cdots & X_{[i_1,j_l]} \\ X_{[i_2,j_1]} & X_{[i_2,j_2]} & \cdots & X_{[i_2,j_l]} \\ \cdots & \cdots & \cdots & \cdots \\ X_{[i_k,j_1]} & X_{[i_k,j_2]} & \cdots & X_{[i_k,j_l]} \end{vmatrix}$$

Given a selection of rows $R$, we say that a column $j$, $1 \leq j \leq m$, is *clean* with respect to $R$ if the symbols in the $j$-th column of $X$ restricted to the rows $R$, are identical.

**Example 1** *Given the* $6 \times 6$ *matrix* $X = \begin{vmatrix} \mathbf{001020} \\ \mathbf{100100} \\ \mathbf{301120} \\ \mathbf{201020} \\ \mathbf{131111} \\ \mathbf{110120} \end{vmatrix}$ *over the alphabet* $\Sigma = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$*, a selection* $(R, C) = (\{2, 5, 6\}, \{1, 4, 6\})$ *results in the matrix* $X_{(R,C)} = \begin{vmatrix} \mathbf{110} \\ \mathbf{111} \\ \mathbf{110} \end{vmatrix}$*. Given the row selection* $R = \{2, 5, 6\}$*, columns 1 and 4 are clean.*

The problem addressed in this paper is defined as follows.

LARGEST BICLUSTER($f$) problem
**Instance:** A matrix $X \in \Sigma^{n \times m}$ over the alphabet $\Sigma$.
**Question:** Find a row selection $R$ and a column selection $C$ such that the rows of $X_{(R,C)}$ are identical strings and the objective function $f(X_{(R,C)})$ is maximized.

Some examples of objective functions are the following.

- $f_1\left(X_{(R,C)}\right) = |R| + |C|$
- $f_2\left(X_{(R,C)}\right) = |R|$ provided that $|C| = |R|$
- $f_3\left(X_{(R,C)}\right) = |R|\,|C|$

**Example 2** *Assume $X$ to be the matrix defined in Example 1. If we choose the objective functions $f_2$ there are three solutions, namely* $(\{1,3,4\},\{2,3,5\})$ *(corresponding to the string* ***012****),* $(\{1,3,4\},\{2,3,6\})$ *(corresponding to* ***010****) and* $(\{1,3,4\},\{3,5,6\})$ *(corresponding to* ***120****). If we choose the objective functions $f_1$ or $f_3$, the solution is* $(\{1,3,4\},\{2,3,5,6\})$ *(corresponding to* ***0120****).*

The problem in general may have multiple solutions which optimize the objective function. The solutions may also "overlap", that is, they may share some elements of the original matrix.

The computational complexity of this family of problems depends on the objective function $f$. It has been studied mostly from a graph-theoretical viewpoint which corresponds to the special case $\Sigma = \{0,1\}$. In fact, observe that a matrix $X \in \{0,1\}^{n\times m}$ is the adjacency matrix of a bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = n$ and $|V_2| = m$. An edge $(i,j) \in E$ connects node $i \in V_1$ to node $j \in V_2$ if $X_{i,j} = 1$. Thus, a submatrix of 1's in $X$ corresponds to a subgraph of $G$ which is completely connected. Such a subgraph is called a *biclique*. Because of this relation, we will use the terms "submatrix", "biclique", and "bicluster" interchangeably.

When the alphabet is binary, the problem associated with objective function $f_1$ is known as the MAXIMUM VERTEX BICLIQUE problem, and it can be solved in polynomial time because it is equivalent to the maximum independent set in bipartite graphs which, in turn, can be solved by a minimum cut algorithm (see, e.g., [15]). The same problem with objective function $f_2$ over a binary alphabet is called BALANCED COMPLETE BIPARTITE SUBGRAPH problem or BALANCED BICLIQUE problem and it is listed as GT24 among the **NP**-complete problems in Garey & Johnson's book [8] (see also [10]).

The LARGEST BICLUSTER problem with objective function $f_3$ and $\Sigma = \{0,1\}$ is called MAXIMUM EDGE BICLIQUE problem. The problem requires to find the biclique which has the maximum number of edges. The problem was proven to be **NP**-complete in [20] by reduction to 3SAT. The weighted version of this problem was shown **NP**-complete by Dawande *et al.* [5]. A 2-approximation algorithm based on LP-relaxation was given in [15].

**Theorem 1** *The decision problem associated with* LARGEST BICLUSTER$(f_3)$ *is* ***NP****-complete.*

**Proof** The problem is trivially in **NP**. We reduce LARGEST BICLUSTER$(f_3)$ to MAXIMUM EDGE BICLIQUE. Note that LARGEST BICLUSTER is more general than the biclique problems on graphs, for two reasons (1) in the LARGEST BICLUSTER problem the alphabet is not necessarily binary and (2) a biclique corresponds to a submatrix of 1's in the corresponding adjacency matrix, while LARGEST BICLUSTER can return submatrices with 0's as well.

Assume that we are given an instance of the biclique problem represented as an adjacency matrix $X \in \{0,1\}^{n\times m}$. Process $X$ as follows. Replace each 0 in $X$ by a unique new symbol. Now LARGEST BICLUSTER will be forced to return a matrix of 1's, which corresponds to a biclique in the bipartite graph. □

By the same approach, LARGEST BICLUSTER$(f_2)$ can also be proven to be **NP**-complete. Henceforth our objective function will be $f_3$, which is associated with the area of the submatrix.

## 3 Randomized search

Given that LARGEST BICLUSTER$(f_3)$ problem is **NP**-complete, it is unlikely that a polynomial time algorithm could be found. In this paper, we present a randomized algorithm which is guaranteed to find the optimal solution with probability $1-\epsilon$, where $0 < \epsilon < 1$.

Assume that we are given a large matrix $X \in \Sigma^{n\times m}$ in which a submatrix $X_{(R^*,C^*)}$ is implanted. Assume also that the submatrix $X_{(R^*,C^*)}$ is maximal. To simplify the notation, let $r^* \equiv |R^*|$ and $c^* \equiv |C^*|$.

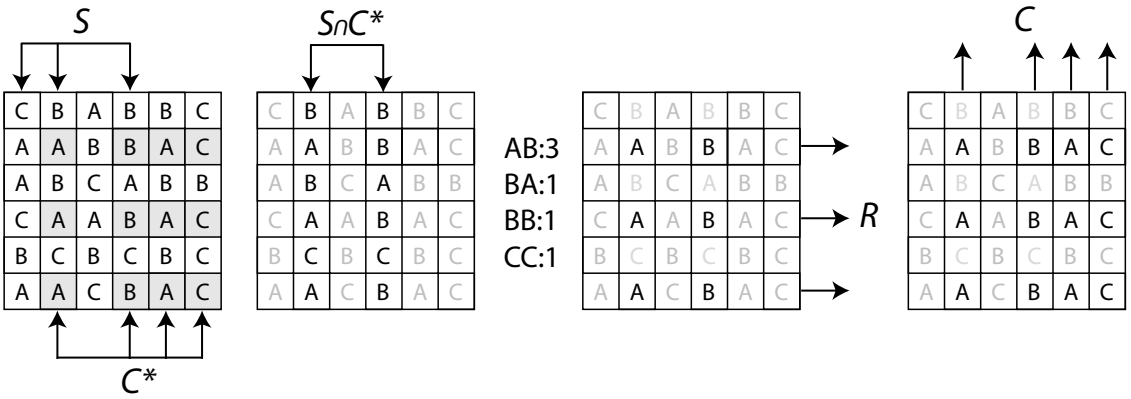


Figure 1: An illustration of a recovery of the embedded matrix by random projections. $C^*$ is the set of columns containing the embedded submatrix. $S$ is a random selection of columns. By following the steps described in the text, the correct solution can be easily retrieved.

The idea behind the algorithm comes from the following simple observation. Observe that if we knew $R^*$, then $C^*$ could be determined by selecting the clean columns with respect to $R^*$. If instead we knew $C^*$, then $R^*$ could be obtained by taking the maximal set of rows which read the same string. Unfortunately, neither $R^*$ nor $C^*$ is known. Our approach is to "sample" the matrix by random projections, with the expectation that at least some of the projections will overlap with the solution $(R^*, C^*)$. Clearly, one can project either rows or columns. In what follows we describe how to retrieve the solution by sampling columns.

The algorithm works as follows. Select a random subset $S$ of size $k$ uniformly from the set of columns $\{1, 2, \ldots, m\}$. Assume for the time being that $S \cap C^* \neq \emptyset$. If we knew $S \cap C^*$, then $(R^*, C^*)$ could be determined by the following three steps (1) select the string(s) $w$ that appear exactly $r^*$ times in the rows of $X_{[1:n, S \cap C^*]}$, (2) set $R^*$ to be the set of rows in which $w$ appears and (3) set $C^*$ to be the set of clean columns corresponding to $R^*$.

**Example 3** *For example, consider the matrix in Figure 1. The shaded cells contain the embedded submatrix, and $r^* = 3$, $c^* = 4$. Once $S \cap C^*$ is determined, the string* `AB` *is chosen since it is the only one that has frequency equal to $r^*$. Then, $R$ is determined by the location of* `AB` *and $C$ is determined by selecting the clean columns of $R$.*

The algorithm would work, but there are a few problems that are still unresolved. First, the set $S \cap C^*$ could be empty. The solution is to try different random projections $S$, relying on the argument that the probability that $S \cap C^* \neq \emptyset$ *at least once* will approach one with more and more projections. The second problem is that we do not really know $S \cap C^*$. But, certainly $S \cap C^* \subseteq S$, so our approach is to check all possible subsets $U \subseteq S$ such that $|U| \geq k_{\min}$, where $1 \leq k_{\min} \leq k$ is a user-defined parameter. The final problem is that we assumed that we knew $r^*$, but we do not. The solution is to introduce a *row threshold* parameter, called $\hat{r}$, that replaces $r^*$.

As it turns out, we need another parameter to avoid producing solutions with too few columns. The *column threshold* parameter $\hat{c}$ is used to discard submatrices whose number of columns is smaller than $\hat{c}$. The algorithm considers all the submatrices which satisfy the user-defined row and column threshold as candidates. Among all candidate submatrices, only the ones that maximize the total area are kept. A sketch of the algorithm is shown in Figure 2.

The algorithm depends on five key parameters, namely the projection size $k$, the minimum subset size $k_{\min}$, the row threshold $\hat{r}$, the column threshold $\hat{c}$, and the number of iterations $t$. We

```
LARGEST_BICLUSTER_C(X, t, k, k_min, r̂, ĉ)
INPUT: X is a n × m matrix over Σ
       t is the number of iterations
       k is the projection size
       k_min is the size of the smallest subset of the projection
       r̂, ĉ are the "thresholds" on the number of rows and columns, resp.
1  repeat t times
2      select randomly a subset S of columns such that |S| = k
3      for all subsets U ⊆ S such that |U| ≥ k_min do
4          D ← all strings induced by X_[1:n,U] that appear at least r̂ times
5          for each string w in D
6              V ← rows corresponding to w
7              Z ← all "clean" columns corresponding to V
8              if |Z| ≥ ĉ then save (V, Z)
9  return the (V, Z) that maximizes f
```

Figure 2: A sketch of the algorithm that discovers large biclusters (sampling columns)

discuss how to choose each of these in the rest of the section.

The projection size $k$ is determined by a probabilistic argument. It is well-known that in a random string of size $m$ over an alphabet of size $a$, the number of occurrences of substrings has two different probabilistic regimes (1) Gaussian distributed for strings shorter than $\log_a m$ and (2) Poisson distributed for strings longer than $\log_a m$. Based on this observation, when $k_{\min} = k$ we argue that $k = \log_a m$ is the optimal tradeoff between generating too many trivial solutions ($k$ too small) and potentially missing the solution ($k$ too large). As it turns out, $k = \log_a m$ not only corresponds to the minimum in the graph (not shown) of the function $\tilde{\alpha}$ discussed at the end of this section, but it has been confirmed to be the optimal choice in our simulations. When $k_{\min} = 1$, then $k$ can be chosen significantly larger, but this will adversely affect the running time. An experimental comparison between $k_{\min} = k$ (i.e., no subsets), and $k_{\min} = 1$ (i.e., all subsets) is reported in Section 5.1.

The thresholds $\hat{r}$ and $\hat{c}$ are associated with the uncertainty on the size of the largest submatrix $r^*, c^*$ for a particular input instance. There may be situations in which the user has already a good idea about $r^*, c^*$. If however $r^*$ and $c^*$ are completely unknown, then we propose the following trial-and-error strategy. Set $\hat{r}$ to some value, and use Theorem 2 (Section 4) to set the value $\hat{c}$. Run the algorithm. If the algorithm returns too many solutions, try to increase $\hat{r}$ and update $\hat{c}$ correspondingly. If there are no solutions, lower the value of $\hat{r}$ and repeat. Observe that the number of choices for $\hat{r}$ is finite since $\hat{r} \in [1, n]$. The rationale behind using Theorem 2 to set $\hat{c}$ is that it gives the expected number of columns of the largest submatrix in a random matrix, when the number of rows is fixed. Since we want to find statistically significant biclusters, we are not interested in submatrices whose size is such that they can be found in totally random matrices.

Because of the randomized nature of the approach, there is no guarantee that the algorithm will find the solution after a given number of iterations. We therefore need to choose $t$ so that the probability that the algorithm will recover the solution in at least one of the $t$ trials is $1 - \epsilon$, where $0 < \epsilon < 1$ is a user-defined parameter.

Let $\alpha(n, m, k, r^*, c^*, a)$ be the probability of missing the solution in one of the trials assuming that $r^*$ and $c^*$ are known and that $k_{\min} = 1$. There are two disjoint cases in which the algorithm can miss $(R^*, C^*)$. The first is when the random projection $S$ misses completely $C^*$, i.e., $S \cap C^* = \emptyset$. The second is when $S \cap C^* = U \neq \emptyset$ but the string $w$ chosen by the algorithm among the rows $X_{[1:n,U]}$ also appears in another row that does not belong to the set $R^*$ of the real solution. In this

case, the algorithm will select a set of rows larger than $R^*$ and thus miss the solution. Hence, we have

$$\begin{aligned}\alpha(n,m,k,r^*,c^*,a) &= \mathsf{Pr}\{S\cap C^*=\emptyset\}+\sum_{i=1}^{k}\mathsf{Pr}\{|S\cap C^*|=i \text{ and } |R|>r^*\}\\ &= \mathsf{Pr}\{S\cap C^*=\emptyset\}+\sum_{i=1}^{k}\mathsf{Pr}\{|R|>r^* \text{ given } |S\cap C^*|=i\}\mathsf{Pr}\{|S\cap C^*|=i\}\end{aligned}$$

Let $Y$ be the random variable associated with the size of the set $S\cap C^*$, that is, $Y=|S\cap C^*|$. Since we are sampling without replacement, $Y$ follows the hypergeometric distribution.

$$\mathsf{Pr}\{Y=0\}=\binom{m-c^*}{k}\Big/\binom{m}{k} \quad \text{and} \quad \mathsf{Pr}\{Y=i\}=\binom{c^*}{i}\binom{m-c^*}{k-i}\Big/\binom{m}{k}$$

In order to compute the probability of missing the solution given $|S\cap C^*|=i$, we have to estimate how likely a string $w$ belonging to some of the rows of $X_{[1:n,U]}$ is more frequent than $r^*$. Assuming the symbols in the matrix $X$ are generated by a symmetric Bernoulli i.i.d. model, the probability that $w$ will never appear in the other $n-r^*$ rows is $\left(1-\frac{1}{a^i}\right)^{n-r^*}$ and therefore

$$\mathsf{Pr}\{|R|>r^* \text{ given } |S\cap C^*|=i\}=1-\left(1-\frac{1}{a^i}\right)^{n-r^*}$$

Combining all together, the probability of missing the solution in one iteration is given by

$$\alpha(n,m,k,r^*,c^*,a)=\left(\binom{m-c^*}{k}+\sum_{i=1}^{k}\left(1-\left(1-\frac{1}{a^i}\right)^{n-r^*}\right)\binom{c^*}{i}\binom{m-c^*}{k-i}\right)\Big/\binom{m}{k}$$

Now suppose we want the probability of missing the solution to be smaller than a given $\epsilon$, $0<\epsilon<1$. We can obtain the number of iterations $t$ by solving the inequality $(\alpha(n,m,k,r^*,c^*,a))^t\le\epsilon$, which gives

$$t\ge\frac{\log\epsilon}{\log\alpha(n,m,k,r^*,c^*,a)} \tag{1}$$

This bound on the number of iterations has been verified by our experimental results (compare Table 1 in the Appendix with our experimental results shown in Figure 3). For example, by setting $a=4$, $k=4$, $\epsilon=0.7$, equation (1) gives $t=90$ iterations whereas the experimental results show that with 90 iterations we obtain a performance of $\epsilon=0.689$.

The worst case time complexity of LARGEST_BICLUSTER_C is $O\left(t\sum_{j=k_{\min}}^{k}\binom{k}{j}(kn+nm)\right)$.

The probability of missing the solution changes significantly when we set $k_{\min}=k$. In this case, we are not checking any of the subsets of $S$, but we simply rely on the fact that eventually one of the random projections $S$ will end up completely contained in $C^*$, in which case we have a chance to find the solution.

Since we avoid checking the $O(2^k)$ subsets of $S$, the number of iterations $t$ to achieve the same level of performance of the case $k_{\min}=1$ must be significantly larger. Indeed, by a similar argument as we did for $k_{\min}=1$, the probability of missing the solution when $k_{\min}=k$ can be estimated by the following formula

$$\begin{aligned}\tilde{\alpha}(n,m,k,r^*,c^*,a) &= \mathsf{Pr}\{S\not\subseteq C^*\}+\mathsf{Pr}\{S\subseteq C^* \text{ and } |R|>r^*\}\\ &= 1-\frac{c^*!}{(c^*-k)!m^k}+\left(1-\left(1-\frac{1}{a^k}\right)^{n-r^*}\right)\frac{c^*!}{(c^*-k)!m^k}\\ &= 1-\left(1-\frac{1}{a^k}\right)^{n-r^*}\frac{c^*!}{(c^*-k)!m^k}\end{aligned}$$

As mentioned above, we also have the option to project the rows instead of the columns, which would result in a slightly different algorithm, called LARGEST_BICLUSTER_R. For lack of space we sketch the algorithm in Figure 4 in the Appendix.

The worst case time complexity of LARGEST_BICLUSTER_R is $O\left(t\sum_{j=k_{\min}}^{k}\binom{k}{j}(km+nm)\right)$. The probability of missing a solution can be obtained using the same argument as above, and it is given by

$$\beta(n,m,k,r^*,c^*,a) = \left(\binom{n-r^*}{k} + \sum_{i=1}^{k}\left(1-\left(1-\frac{1}{a^{i-1}}\right)^{n-c^*}\right)\binom{r^*}{i}\binom{n-r^*}{k-i}\right)/\binom{n}{k}$$

# 4 Statistical analysis

We now analyze the statistical significance of finding a large submatrix of size $r \times c$ hidden into a random $n \times m$ matrix over an alphabet of cardinality $a$. More specifically, we randomly generate a matrix $X \in \Sigma^{n\times m}$ using a memoryless source with parameters $\{p_1, \ldots, p_a\}$ where $p_i$ is the probability of the $i$-th symbol in $\Sigma$. Given $X$, the goal is to characterize asymptotically the size of the largest submatrix in $X$.

For convenience of notation, let us call $P_r = p_1^r + p_2^r + \ldots + p_a^r$ the probability of observing a clean column over $r$ rows, and let us define $H(x) = -x\log x - (1-x)\log(1-x)$.

The first result characterizes the random variable associated with the number of columns of the largest bicluster, when we fix the number of rows.

**Theorem 2** *Let $\boldsymbol{C}_{n,m,r,a}$ be the random variable associated with the number of columns of the submatrix with the largest area in a matrix $X \in \Sigma^{n\times m}$ generated from a memoryless source, once the number of rows $r$ is fixed. Then*

$$\boldsymbol{C}_{n,m,r,a} \leq mP_r + \sqrt{2P_r(1-P_r)mF(n,r)} \equiv C_{\max}$$

*with high probability and as $n \to \infty$, where*

$$F(n,r) = \begin{cases} r\log n & \text{if } r = o(n) \\ nH(\alpha) & \text{if } r = \alpha n \text{ where } 0 < \alpha < 1 \end{cases}$$

When $r = o(n)$ the error term is $O(1/\log^d n)$ for some $d > 1$ that may depend on $a$, whereas the error becomes $O(1/\sqrt{n})$ when $r = \alpha n$. The prediction on random matrices is indeed quite accurate as reported in Table 2 (Section 5.1). We claim that the upper bound is actually an equality, that is, asymptotically and with high probability $\mathbf{C}_{n,m,r,a} = C_{\max}$.

The practical implications of Theorem 2 are twofold. First, the expected number of columns can be used to set the column threshold parameter $\hat{c} \gg \max\{C_{\max}, 1\}$. That allows the algorithm to avoid considering statistically non-significant submatrices. Second, observe that when $\log n = o(m)$, then the dominant term of $C_{\max}$ is the average, say $\mathbf{E}[C]$, of the number of clean columns, that is, $\mathbf{E}[C] = mP_r$. This implies $C_{\max}/\mathbf{E}[C] \leq 1 + o(1)$ for $\log n = o(m)$, and therefore with high probability any algorithm is asymptotically optimal. Clearly, this is not true for $r = \alpha n$. Finally, in passing we add that when we restrict the search to largest *squared* matrix (see objective function $f_2$ above), then its side is asymptotically equal to $2\log\left(n/(2\log n)\right)/\log P_r^{-1}$.

The second result characterizes the random variable associated with the area of the solution, under some general assumptions on the number of rows and columns.

**Theorem 3** *Let $\boldsymbol{A}_{n,m,a}$ be the random variable associated with the area of the largest submatrix in a matrix $X \in \Sigma^{n \times m}$ generated from a memoryless source. Then, with high probability (i.e., $1 - \Theta(1/\sqrt{n})$) and as $n \to \infty$,*

- *if $r = \alpha n$ and $c = O(1)$, then $\boldsymbol{A}_{n,m,a} \leq \alpha^* nc$*
- *if $r = O(1)$ and $c = \beta n$, then $\boldsymbol{A}_{n,m,a} \leq \beta^* nr$*

*where $\alpha^*$ and $\beta^*$ are the solution of the following equations*

$$\begin{aligned} H(\alpha^*) &= c\alpha^* \log P_{\alpha r}^{-1} \\ H(\beta^*) &= \beta \log P_r^{-1} \end{aligned}$$

The intuition behind Theorem 3 is that on random matrices one should expect the largest submatrix to be "skinny", that is, a few columns and lots of rows, or vice versa. For example, the largest submatrix in a random $\{0,1\}$-matrix of size $256 \times 256$ is of size $2 \times 160$ (see Table 2). This phenomenon was also observed in the experiments on microarray data.

For lack of space, proofs of Theorem 2 and Theorem 3 can be found in the Appendix.

# 5 Implementation and Experiments

We implemented column- and row-sampling algorithms in C++ and tested the programs on a desktop PC with a 1.2GHz Athlon CPU and 1GB of RAM, under Linux. Although the algorithms do not require sophisticated data structures, in order to carry out step 4 in the algorithm of Figure 2, one needs a data structure to store the strings and their frequencies. Since $k$ and $a$ are usually not very large, our experience shows that a simple hash table (of size $a^k$) is a good choice. If $a^k$ becomes too large, a trie would be a better data structure. If one uses the hash table, it is important to keep track of the non-zero entries in another balanced data structure. That would avoid the algorithm to spend $O(a^k)$ to search for the frequently occurring strings. Observe also that row-sampling algorithm (Figure 4) does not require any hash table, or any other data structure. However, our experiments show that in order to get the same level of performance of the column sampling, it needs a significantly larger projection $k$ which adversely affects the running time.

Another issue is to whether one should keep track of the projections generated so far to avoid generating duplicates. We studied this matter experimentally, and found that it is worthwhile to keep track of the projections in some balanced data structure only when $k$ is small. If $k$ is large, the overhead required to keep the data structure updated is much higher than the time wasted in processing the same projection multiple times.

## 5.1 Simulations

In order to evaluate the performance of the algorithms, we designed several simulation experiments. In these experiments we randomly generated one thousand $256 \times 256$ matrices of symbols drawn from an symmetric i.i.d. distribution over an alphabet of cardinality $a$. Then, in each matrix we embedded a random $64 \times 64$ submatrix at random columns and random rows. We ran the algorithms for a few tens of iterations ($t = 5, \ldots, 100$), and for each choice of $t$ we measured the number of successes out of the 1,000 distinct instances. Figure 3 summarizes the performance of LARGEST_BICLUSTER_C, for several choices of alphabet size $a$ and projection size $k$, and minimum subset size $k_{\min}$.

In order to make a fair comparison between $k_{\min} = k$ and $k_{\min} = 1$, the number of iterations for the case $k_{\min} = k$ was multiplied by $2^k - 1$. Note that by doing so, we are assuming that one

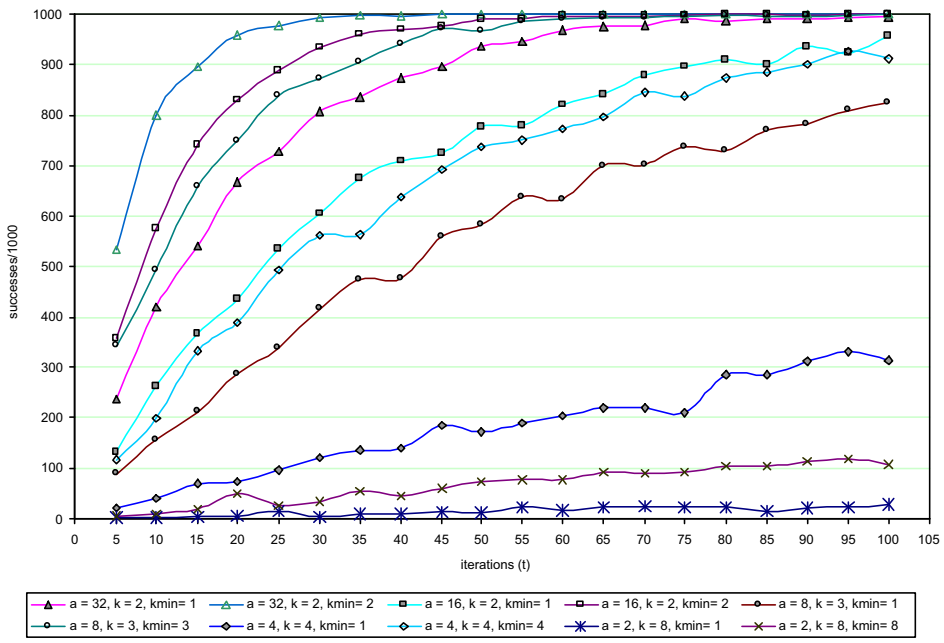


Figure 3: Comparing the performance of the randomized algorithm LARGEST_BICLUSTER_C when $k_{\min} = k$ versus $k_{\min} = 1$, for different choices of the alphabet size $a$. The projection size is $k = \log_a m$

projection for $k_{\min} = 1$ takes about the same time as one projection for $k_{\min} = k$, which is not necessarily very accurate. Under this assumption, however, $k_{\min} = k$ outperforms $k_{\min} = 1$ (see Figure 3). This not necessarily true in the row sampling strategy (see Figure 5 in the Appendix).

By comparing the performance of row sampling against column sampling, we observed that if one uses the same set of parameters, column sampling always outperforms row sampling.

### 5.2 Preliminary Experimental Results

We applied our algorithm on gene expression data from the leukemia gene expression study [9]. This data set contains 7129 genes from 72 samples collected from acute leukemia patients at the time of diagnosis. Out of the 72 samples, 47 were diagnosed as ALL (Acute Lymphoblastic Leukemia) and the rest 25 as AML (Acute Myeloid Leukemia). After calculating the standard deviation of the expression levels of each gene, we selected only those genes whose standard deviation is greater than 200. As a result of this selection procedure 3304 genes were left for further analysis. Then we discretized the value of the normalized matrix row-by-row by the equal frequency principle into $a = 4$ bins.

Figure 6 shows a $42 \times 10$ bicluster found by running two million iterations of the algorithm LARGEST_BICLUSTER_C with parameters $k = 10$, $k_{\min} = 10$, $\hat{r} = 37$, $\hat{c} = 10$. The bicluster is quite significant not only because of its size, but also because it contains only the symbols `A` and `D` which correspond respectively to the down-regulated and up-regulated states.

A comprehensive report of experimental results will be included in the final version of the paper.

## References

[1] AGGARWAL, C. C., PROCOPIUC, C., WOLF, J. L., YU, P. S., AND PARK, J. S. Fast algorithms for

projected clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-99)* (New York, June 1–3 1999), vol. 28,2 of *SIGMOD Record*, ACM Press, pp. 61–72.

[2] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)* (New York, June 1–4 1998), vol. 27,2 of *ACM SIGMOD Record*, ACM Press, pp. 94–105.

[3] Ben-Dor, A., Chor, B., Karp, R., and Yakhini, Z. Discovering local structure in gene expression data: The order-preserving submatrix problem. In *Proceedings of Sixth International Conference on Computational Molecular Biology (RECOMB 2002)* (2002), ACM Press, pp. 45–55.

[4] Cheng, Y., and Church, G. M. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-00)* (Menlo Park, CA, Aug. 16–23 2000), AAAI Press, pp. 93–103.

[5] Dawande, M., Keskinocak, P., Swaminathan, J. M., and Tayur, S. On bipartite and multipartite clique problems. *Journal of Algorithms 41* (2001), 388–403.

[6] Dhillon, I. Co-clustering documents and words using bipartite spectral graph parititioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)* (New York, Aug. 26–29 2001), ACM Press, pp. 269–274.

[7] Dhillon, I., Mallela, S., and Modha, D. Information-theoretic co-clustering. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)* (2001), ACM Press, p. to appear.

[8] Garey, M. R., and Johnson, D. S. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, New York, NY, 1979.

[9] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science 286*, 5439 (1999), 531–537.

[10] Grigni, M., and Manne, F. On the complexity of the generalized block distribution. In *In proceedings of Irregular'96, The third international workshop on parallel algorithms for irregularly structured problems* (1996), vol. 1117, Lecture Notes in Computer Science, Springer, pp. 319–326.

[11] Gu, M., Zha, H., Ding, C., He, X., and Simon, H. Spectral relaxation models and structure analysis for $k$-way graph clustering and bi-clustering. Tech. Rep. CSE-01-007, Department of Computer Science and Engineering, Pennsylvania State University, 2001.

[12] Hanisch, D., Zien, A., Zimmer, R., and Lengauer, T. Co-clustering of biological networks and gene expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular (ISMB-02)* (2002), AAAI Press, pp. 145–154.

[13] Hartigan, J. A. Direct clustering of a data matrix. *Journal of the American Statistical Association 67*, 337 (1972), 123–129.

[14] Hastie, T., Tibshirani, R., Eisen, M., Alizadeh, A., Levy, R., Staudt, L., Chan, W., Botstein, D., and Brown, P. 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biol. 1*, 2 (2000), 1–21.

[15] Hochbaum, D. S. Approximating clique and biclique problems. *Journal of Algorithms 29*, 1 (Oct. 1998), 174–200.

[16] Kluger, Y., Basri, R., Chang, J., and Gerstein, M. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res. 13*, 4 (2003), 703–16.

[17] Lazzeroni, L., and Owen, A. Plaid models for gene expression data. *Statistica Sinica 12*, 1 (2002), 61–86.

[18] Mishra, N., Ron, D., and Swaminathan, R. On finding large conjunctive clusters. In *Proc. of the ACM Conference on Computational Learning Theory (COLT'03)* (2003), p. to appear.

[19] Murali, T. M., and Kasif, S. Extracting conserved gene expression motifs from gene expression data. In *Proceedings of the Pacific Symposium on Biocomputing (PSB'03)* (2003), pp. 77–88.

[20] Peeters, R. The maximum-edge biclique problem is NP-complete. Tech. Rep. 789, Tilberg University: Faculty of Economics and Business Adminstration, 2000.

[21] Procopiuc, M., Jones, M., Agarwal, P., and Murali, T. M. A Monte-Carlo algorithm for fast projective clustering. In *Proceedings of the 2002 International Conference on Management of Data (SIGMOD'02)* (2002), pp. 418–427.

[22] Q., S., Y., M., and B., D. M. Biclustering microarray data by gibbs sampling. In *Proceedings of European Conference on Computational Biology (ECCB'03)* (2003), p. to appear.

[23] Szpankowski, W. *Average Case Analysis of Algorithms on Sequences.* Wiley Interscience, 2001.

[24] Tanay, A., Sharan, R., and Shamir, R. Discovering statistically significant biclusters in gene expression data. In *Proceedings of the 10th International Conference on Intelligent Systems for Molecular Biology (ISMB'02), in Bioinformatics* (2002), vol. 18, pp. S136–S144.

[25] Wang, H., Wang, W., Yang, J., and Yu, P. S. Clustering by pattern similarity in large data sets. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD-02)* (New York, June 3–6 2002), ACM Press, pp. 394–405.

[26] Yang, J., Wang, H., Wang, W., and Yu, P. S. Enhanced biclustering on gene expression data. In *IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03)* (2003), p. to appear.

[27] Zhang, L., and Zhu, S. A new clustering method for microarray data analysis. In *Proceedings of the First IEEE Computer Society Bioinformatics Conference (CSB'02)* (2002), IEEE Press, pp. 268– 275.

# Appendix

```
LARGEST_BICLUSTER_R(X, t, k, k_min, r̂, ĉ)
INPUT: X is a n × m matrix over Σ
       t is the number of iterations
       k is the projection size
       k_min is the size of the smallest subset of the projection
       ĉ, r̂ are the "thresholds" on the number of columns and rows, resp.
1  repeat t times
2      select randomly a subset S of rows such that |S| = k
3      for all subsets U ⊆ S such that |U| ≥ k_min do
4          Z ← all columns induced by X_[U,1:m] that are clean
5          w ← string induced by any of the row of X_[U,Z]
6          if |Z| ≥ ĉ then
7              V ← rows of X_[1:n,Z] which contain w
8              if |V| ≥ r̂ then save (V, Z)
9  return the (V, Z) that maximizes f
```

Figure 4: A sketch of the algorithm that discovers large biclusters (sampling row)

| $\epsilon$ | $a=2,k=8$ | $a=4,k=4$ | $a=8,k=3$ | $a=16,k=2$ | $a=32,k=2$ |
|---|---|---|---|---|---|
| 0.005 | 18794 | 1342 | 306 | 179 | 99 |
| 0.05 | 10626 | 759 | 173 | 101 | 56 |
| 0.1 | 8168 | 583 | 133 | 78 | 43 |
| 0.2 | 5709 | 408 | 93 | 54 | 30 |
| 0.3 | 4271 | 305 | 70 | 41 | 23 |
| 0.4 | 3250 | 232 | 53 | 31 | 17 |
| 0.5 | 2459 | 176 | 40 | 23 | 13 |
| 0.6 | 1812 | 129 | 29 | 17 | 10 |
| 0.7 | 1265 | 90 | 21 | 12 | 7 |
| 0.8 | 792 | 57 | 13 | 8 | 4 |
| 0.9 | 374 | 27 | 6 | 4 | 2 |

Table 1: The estimated number of iterations for a matrix $256 \times 256$ with a submatrix $64 \times 64$, for different choices of $\epsilon$, alphabet size $a$, and projection size $k$ (sampling columns)

| rows | columns observed | columns predicted |
|---|---|---|
| 1 | 256 | 256 |
| 2 | 160 | 165.6771209 |
| 3 | 100 | 103.9626215 |
| 4 | 67 | 67.24371945 |
| 5 | 45 | 44.84053788 |
| 6 | 31 | 30.70906224 |
| 7 | 23 | 21.48364693 |
| 8 | 16 | 15.26873716 |

Table 2: The statistics of large submatrices in a random $\{0,1\}$-matrix of size $256 \times 256$. The second column reports the number of columns of the submatrices observed in a random matrix, whereas the third reports the prediction based on Theorem 2

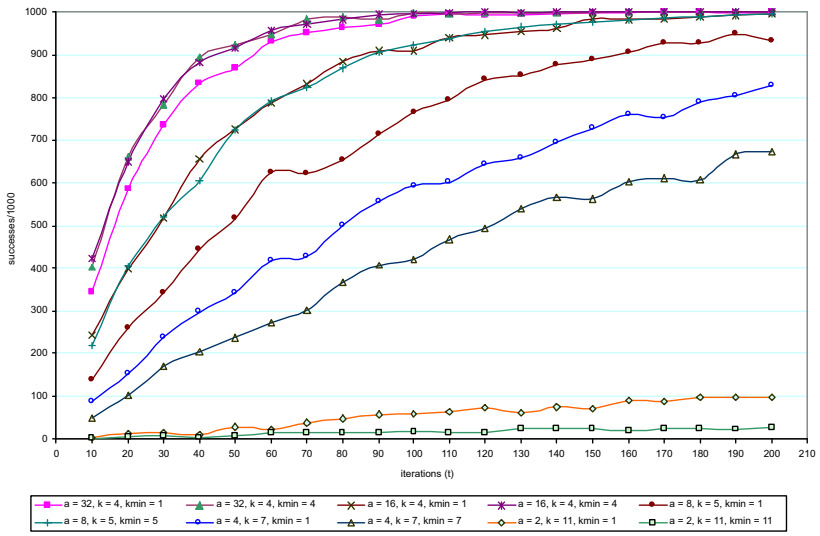


Figure 5: Comparing the performance of the randomized algorithm LARGEST_BICLUSTER_R for different choices of the alphabet size $a$ and projection size $k$

## Proofs

**Theorem 1** *Let $\boldsymbol{C}_{n,m,r,a}$ be the random variable associated with the number of columns of the submatrix with the largest area in a matrix $X \in \Sigma^{n \times m}$ generated from a memoryless source, once that the number of rows $r$ is fixed. Then*

$$\boldsymbol{C}_{n,m,r,a} \leq mP_r + \sqrt{2P_r(1-P_r)mF(n,r)}$$

*with high probability and as $n \to \infty$, where*

$$F(n,r) = \begin{cases} r \log n & \text{if } r = o(n) \\ nH(\alpha) & \text{if } r = \alpha n \text{ where } 0 < \alpha < 1 \end{cases}$$

**Proof** Recall from Section 4 that $P_r = p_1^r + p_2^r + \ldots + p_a^r$ for an alphabet $\Sigma$ of cardinality $a$. Let us define the random variable $\mathbf{Z}_{j_1,\ldots,j_r}$ associated with the number of clean columns in rows $1 \leq j_1 < \ldots < j_r \leq n$. Our goal is to find

$$\mathbf{C}_{n,m,r,a} = \max_{1 \leq j_1 < \ldots < j_r \leq n} \{\mathbf{Z}_{j_1,\ldots,j_r}\}$$

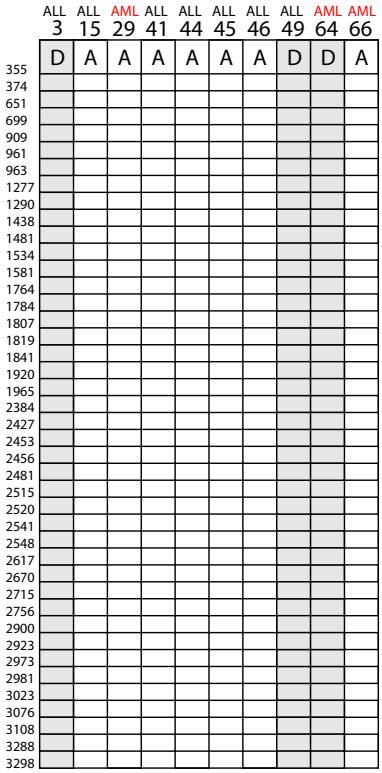


Figure 6: A $42 \times 10$ bicluster found in the $3304 \times 72$ leukemia dataset (discretized over an alphabet of cardinality $a = 4$), by running $t = 2,000,000$ iterations of the LARGEST_BICLUSTER_C algorithm with parameters $k = 10$, $k_{\min} = 10$, $\hat{r} = 37$, $\hat{c} = 10$

Clearly,

$$\mathsf{Pr}\{\mathbf{Z}_{j_1,\ldots,j_r} = k\} = \binom{m}{k} P_r^k (1 - P_r)^{m-k},$$

and in particular, $\mathbf{E}[\mathbf{Z}_{j_1,\ldots,j_r}] = mP_r$ and $\mathbf{Var}[\mathbf{Z}_{j_1,\ldots,j_r}] = mP_r(1 - P_r)$.

We now proceed as follows.

$$\begin{aligned}
\mathbf{C}_{n,m,r,a} &= \max_{1 \le j_1 < \ldots < j_r \le n} \{\mathbf{Z}_{j_1,\ldots,j_r}\} \\
&= \sqrt{\mathbf{Var}[\mathbf{Z}_{j_1,\ldots,j_r}]} \max_{1 \le j_1 < \ldots < j_r \le n} \left\{ \frac{\mathbf{Z}_{j_1,\ldots,j_r} - \mathbf{E}[\mathbf{Z}_{j_1,\ldots,j_r}]}{\sqrt{\mathbf{Var}[\mathbf{Z}_{j_1,\ldots,j_r}]}} \right\} + \mathbf{E}[\mathbf{Z}_{j_1,\ldots,j_r}] \\
&= \mathbf{Var}[\mathbf{Z}_{j_1,\ldots,j_r}] \max_{1 \le j_1 < \ldots < j_r \le n} \{\mathbf{Y}_{j_1,\ldots,j_r}\} + \mathbf{E}[\mathbf{Z}_{j_1,\ldots,j_r}], \qquad (2)
\end{aligned}$$

where $\mathbf{Y}_{j_1,\ldots,j_r}$ are (approximately) normally distributed $N(0,1)$. Observe that $\mathbf{Y}_{j_1,\ldots,j_r}$ are not independent (but we shall show below that they are only weakly dependent).

We now estimate $\mathbf{C'}_{n,m,r,a} = \max_{1 \le j_1 < \ldots < j_r \le n} \{\mathbf{Y}_{j_1,\ldots,j_r}\}$. Since $\mathbf{Y}_{j_1,\ldots,j_r}$ are normally distributed we have, when $r = o(n)$

$$\begin{aligned}
\mathsf{Pr}\{\mathbf{C'}_{n,m,r,a} > x\} &\le \binom{n}{r} \mathsf{Pr}\{\mathbf{Y}_{j_1,\ldots,j_r} > x\} \\
&\sim \frac{n^r}{r!} \frac{e^{-x^2/2}}{x}. \qquad (3)
\end{aligned}$$

Hence for $x = \sqrt{2 \log n^r}$ we find that

$$\mathsf{Pr}\{\mathbf{C'}_{n,m,r,a} > \sqrt{2 \log n^r}\} \leq O(1/\log n) \to 0$$

as $n \to \infty$. We have

$$\mathbf{C}_{n,m,r,a} \leq mP_r + \sqrt{2P_r(1-P_r)mr \log n}$$

with high probability.

If instead $r = \alpha n$. In this case, equation (3) should be replaced by

$$\begin{aligned} \mathsf{Pr}\{\mathbf{C'}_{n,m,r,a} > x\} &\leq \binom{n}{r} \mathsf{Pr}\{\mathbf{Y}_{j_1,\cdots,j_r} > x\} \\ &\leq \frac{1}{\sqrt{2\pi n\alpha(1-\alpha)}} \frac{e^{nH(\alpha)-x^2/2}}{x} \end{aligned}$$

where $H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$. Thus $x = \sqrt{2nH(\alpha)}$ makes the above probability small and $\mathbf{C'}_{n,m,r,a} \sim \sqrt{2nH(\alpha)}$ (pr.). Finally,

$$\mathbf{C}_{n,m,r,a} \leq mP_r + \sqrt{2P_r(1-P_r)mnH(\alpha)}$$

with high probability. □

**Theorem 2** *Let $\boldsymbol{A}_{n,m,a}$ be the random variable associated with the area of the submatrix with the largest area in a matrix $X \in \Sigma^{n \times m}$ generated from a memoryless source. Then, with high probability and as $n \to \infty$,*

- *if $r = \alpha n$ and $c = O(1)$, then $\boldsymbol{A}_{n,m,a} \leq \alpha^* nc$*
- *if $r = O(1)$ and $c = \beta n$, then $\boldsymbol{A}_{n,m,a} \leq \beta^* nr$*

*where $\alpha^*$ and $\beta^*$ are the solution of the following equations*

$$\begin{aligned} H(\alpha^*) &= c\alpha^* \log P_{\alpha n}^{-1} \\ H(\beta^*) &= \beta \log P_r^{-1} \end{aligned}$$

**Proof** First, observe that

$$\mathsf{Pr}\{\mathbf{A}_{n,m,a} > r \cdot c\} \leq \binom{n}{r}\binom{m}{c} P_r^c. \tag{4}$$

Now one needs to select $r$ and $c$ such that the right-hand side of equation (4) is small.

Let $r = \alpha n$ and $c = O(1)$. As in previous proof, we observe that

$$\binom{n}{\alpha n} \sim \frac{e^{nH(\alpha)}}{\sqrt{2\pi n\alpha(1-\alpha)}}$$

where $H(\alpha)$ is a natural entropy. Then

$$\mathsf{Pr}\{\mathbf{A}_{n,m,a} > \alpha cn\} \leq \frac{e^{nH(\alpha)+c\log P_{\alpha n}^{-1}}}{\sqrt{2\pi n\alpha(1-\alpha)}}.$$

We choose such $\alpha^*$ that

$$H(\alpha^*) = c\alpha^* \log P_{\alpha n}^{-1}. \tag{5}$$

Then

$$\mathsf{Pr}\{\mathbf{A}_{n,m,a} > \alpha^* cn\} \leq \frac{1}{\sqrt{2\pi n\alpha^*(1-\alpha^*)}} \to 0.$$

Let now $r = O(1)$ and $c = \beta n$. Repeating the same calculation we have

$$\mathsf{Pr}\{\mathbf{A}_{n,m,a} > \beta cn\} \leq \frac{1}{\sqrt{2\pi n\beta(1-\beta)}} e^{nH(\beta)+\beta n \log P_r^{-1}}.$$

Finding $\beta^*$ such that

$$H(\beta^*) = \beta \ln P_r^{-1} \tag{6}$$

we arrive finally at

$$\mathsf{Pr}\{\mathbf{A}_{n,m,a} > \beta^* cn\} \leq \frac{1}{\sqrt{2\pi n\beta^*(1-\beta^*)}} \to 0.$$

□