Algorithm for Low-Variance Biclusters to Identify Coregulation Modules in Sequencing Datasets

Zhen Hu
School of Computing Sciences and Informatics
University of Cincinnati
huze@mail.uc.edu

Raj Bhatnagar
School of Computing Sciences and Informatics
University of Cincinnati
Raj.Bhatnagar@uc.edu

ABSTRACT

High-throughput sequencing (CHIP-Seq) data exhibit binding events with possible binding locations and their strengths, followed by interpretations of the locations of peaks. Recent methods tend to summarize all CHIP-Seq peaks detected within a limited up and down region of each gene into one real-valued score in order to quantify the probability of regulation in a region. Applying subspace clustering (or biclustering) techniques on these scores would discover important knowledge such as the potential co-regulation or co-factors mechanisms. The ideal biclusters generated should contain subsets of genes, and transcription factors (TF) such that the cell-values in biclusters are distributed around a mean value with low variance. Such biclusters would indicate TF sets regulating gene sets with the same probability values. However, most existing biclustering algorithms are neither able to enforce variance as a strict limitation on the values contained in a bicluster, nor use variance as the guiding metric while searching for the desirable biclusters. An algorithm that uses search spaces defined by lattices containing all overlapping biclusters and a bound on variance values as the guiding metric is presented in this paper. The algorithm is shown to be an efficient and effective method for discovering the possibly overlapping biclusters under pre-defined variance bounds. We present in this paper our algorithm, its results with synthetic and CHIP-Seq and motif datasets, and compare them with the results obtained by other algorithms to demonstrate the power and effectiveness of our algorithm.

1. BACKGROUND AND MOTIVATION

Mining biclusters (or co-clusters) from sequencing datasets is one of the important ways to discover potential biological mechanisms. Transcription Factors (TF) binding related sequencing datasets, including High-throughput Chromatin Immunoprecipitation Sequencing (CHIP-Seq) [22] and motif searching [12] data, record potential matchings on genome with many different metrics. For example, CHIP-Seq peaks records intensity and position. By balancing contributions from several metrics, many researchers summarize them into unified scores to quantify the binding strengths for gene-TF pairs. These scores are very sensitive and minor differences may reflect quite different binding scenarios. Biclusters consisting of subsets of genes and TFs having very similar cell values can help provide insights into coregulation. However, traditional methods [29, 15, 14, 11] cannot be adapted easily to analyze the sequencing datasets because most of them do not seek biclusters with specificable bounds on statistical quantities such as the standard deviation (of the cell values). We present in this paper an algorithm to solve this problem. The generated biclusters are the largest possible in size such that the cell values contained in them are distributed with variance bounded by specified low thresholds.

High-throughput Chromatin Immunoprecipitation Sequencing (CHIP-Seq) experiments generate precise short DNA sequences bound to Transcription Factors. After mapping these short sequences back to the whole-genome sequence and searching for enriched regions, CHIP-Seq datasets provide precise binding information in terms of binding locations and strengths (or peaks) [23, 26, 27, 2, 13]. Many current methods summarize all peaks within up and down regions of each gene into a unified score by combinding the information of distances from peaks to transcription start sites (TSS) and the information of binding strengths together. For example, Ouyang et. al [21] compute the score by summing up all weighted peaks' strengths, influenced by the distances to TSS. Another similar type of sequencing dataset is generated while searching for motifs matching across the whole genome. The motifs are defined as position weighted matrices (TRANSFC), and the final matching scores are computed by using the method given in [12]. Both of these two types of sequencing scores are very sensitive; slight differences in scores indicate quite different binding scenarios. For example, based on the Ouyang et. al's method, same intensity peaks (E2F1) bound at positions 500 and 800 away from TSS may lead to differences of less than 1 between the final scores.

For illustration, we consider a very small synthetic dataset shown in Figure 1(a) in which the values are quite similar to the CHIP-Seq scores and motif matching scores. Biclusters shown in Figure 1(b) are such that the values in the selected cells are all about the same (std. dev. < 0.5) and also satisfy the size constraint, which is: biclusters should contain at least two rows and two columns. For binary datasets the theory of Formal Concept Analysis [17] treats all maximum sized sub-matrices containing only 1's as concepts and ar-

BIOKDD 2011, August 2011, San Diego, CA, USA

	a	b	c	d
$g1$	2.8	3.0	3.2	4.5
$g2$	3.0	2.7	2.7	1.6
$g3$	4.9	5.0	5.3	1.2
$g4$	2.1	5.2	4.8	0.8

(a) Data table

$< \{g1, g2\}, \{a, b, c\} >$
$< \{g1, g2\}, \{a, b\} >$
$< \{g1, g2\}, \{a, c\} >$
$< \{g1, g2\}, \{b, c\} >$
$< \{g3, g4\}, \{b, c\} >$

(b) Biclusters

Figure 1: Example Biclusters

ranges them in a partially ordered lattice. Here we consider all those maximum sized sub-matrices as *concepts* for which the standard deviation of all included cell values is below some thresholds. The parent-child relationship in the lattice is still defined by the superset-subset relationship among the attributes included in the bicluster. In our extension of the analogy to FCA lattices, each node of the lattice may contain more than one bicluster. Biclusters shown in figure 1(b) meet all the above requirements and are qualified to be *concepts* in the sense outlined above.

Potential co-factor or co-regulation mechanism could be discovered from these sequencing datasets by taking subsets of genes and subsets of TFs such that all TFs have similar binding probability with select genes (or low-variance cell values of the sub-matrices). The problem of discovering the qualified biclusters, including the ones that may overlap some other biclusters, is NP-Hard [20] and most of the proposed algorithms attack the problem in a greedy manner [11, 8]. These algorithms, however, do not emphasis the cell-values' variance or STD. Some other algorithms utilize pattern recognition techniques [28] to improve the quality of clusters but they miss out on the many potential good overlapping biclusters due to imposing hard pattern restrictions on real valued data. There are also many biclustering algorithms which are based on statistical theory [15, 14, 25]. These algorithms use their own optimized metric for clustering and it is still not clear how to control the variance of the cell-values in biclusters.

One critical issue with real-valued datasets is that the standard deviation of cell-values in any selected sub matrix depends on the distribution of all of these values. This means incremental addition of rows and/or columns to construct a larger bicluster cannot be guided, in an algorithm, by a monotonically increasing/decreasing variance of all the included cell-values. The variance itself is not one such monotonic metric and therefore, one challenge addressed by us in this paper is to develop such a monotonic metric and correlate it with the variance and standard deviation of a bicluster.

A closed bicluster is one to which we cannot add either an attribute (column) or an object (row) and still maintain the standard deviation of all cells below the selected threshold. Our analogy with Formal concept analysis, and also our algorithms here, consider the lattice consisting of only the closed biclusters. A lattice of partially ordered closed biclusters is an efficient model of the search space in which a search algorithm may look for desirable closed biclusters. This approach has been adopted for finding biclusters in binary datasets [3, 7, 6, 30] and our work in this paper is the first attempt to advance the same idea to datasets with real-values entries in the cells.

In the following sections we formally define some ideas including a monotonic quality that can be used to bound the standard deviation of a non-closed bicluster. In section §3 we prove the relationship between our monotonic metric with standard deviation and present our algorithm; in section §3.5 we present results of our algorithm after some efficiency enhancing pruning is employed, and in section §4 we present results with a synthetic dataset and two genomic datasets.

2. PRELIMINARIES

We need a monotonic metric to help us guide the search for the best biclusters and we choose Range (max - min) of all the values in a biclusters to be this metric. In this paper we use dedicated symbols Δ (and δ) to denote the Range for a set of values in a submatrix.

Definition 1. The **Range** of a group of N data elements is the difference between the maximum and the minimum values of that group. That is,

$$\Delta = \max(N) - \min(N) \tag{1}$$

Given the range for a set of data elements we can derive an upper bound on the standard deviation for the data elements. This is possible because the standard deviation depends on the difference between an element and the mean and the value of *Range* is an upper bound on this difference value. Consequently, we can derive the relationship between standard deviation and range for single dimensional data in equation 2, which means by limiting Range the standard deviation is also limited.

$$s^2 \leq \delta^2 \tag{2}$$

From the point of view of formulating a search algorithm, we need a quantity that monotonically increases (or decreases) as the size of a potential biclusters increases. It is easy to see that as the size of a biclusters is enlarged, the *Range* of its values (and therefore the upper bound on its standard deviation) can only increase. This information, combined with the size of a potential biclusters, can be used to prune some potential search paths and also determine the most promising paths.

We represent a dataset as $D = (R, C)$, where R indicates the rows (or objects) of the table and C indicates the columns (or features). A bicluster is represented as $B = (sr, sc)$ where $sr \subseteq R$ and $sc \subseteq C$. We use $\widehat{B}$ to indicate the number of columns in a bicluster B, $\widetilde{B}$ to indicate its number of rows, and s_B to indicate its standard deviation.

There are many algorithms [11, 25, 5, 18, 19, 28, 14] using different metrics to define interesting biclusters. The quality of desired biclusters are based on those metrics. Here we give the definition of interesting biclusters which, compared with others, is based on the statistical restriction (standard deviation) directly:

Definition 2. A bicluster (B) is **an interesting bicluster** if it satisfies all of the following constraints: (i) $\widehat{B} \geq m$;

(ii) $\widetilde{B} \geq n$; and (iii) $s_B \leq S'$, where m and n are prespecified row and column sizes and S' is the threshold for the standard deviation.

In order to compare two biclusters, we also define several operators that will be used in pruning some branches of the search algorithm presented in section 3.3.

Definition 3. 1. A biclusters $B_1 = (sr_1, sc_1)$ is contained in biclusters $B_2 = (sr_2, sc_2)$, if and only if, $sr_1 \subseteq sr_2$ and $sc_1 \subseteq sc_2$.

2. A biclusters $B_1 = (sr_1, sc_1)$ is similar to $B_2 = (sr_2, sc_2)$, that is, $B_1 \approx B_2$ or $B_2 \approx B_1$, if and only if,

$$\frac{|sr_1 \cap sr_2|}{|sr_1 \cup sr_2|} \geq \theta; \;\; and \;\; \frac{|sc_1 \cap sc_2|}{|sc_1 \cup sc_2|} \geq \theta$$

where θ is a user defined threshold for similarity and has a value between 0 and 1.

For comparing two bicluster's interestingness based only on their sizes, we define an operator below which uses the number of cells included in each bicluster as the criterion. Intuitively, extending the **Range** bound for the biclusters to be found will lead our algorithm to generate biclusters with larger sizes. The trade-off between size and **Range** bound could be easily defined, if needed, and implemented in our algorithm.

Definition 4. A bicluster(B_1) is **more interesting than** a bicluster (B_2), that is, $B_1 \succ B_2$ or $B_2 \prec B_1$,if and only if

$$\widehat{B_1} \times \widetilde{B_1} \geq \widehat{B_2} \times \widetilde{B_2} \tag{3}$$

3. SEARCH ALGORITHM

The term *Range* is coined to restrict the statistical quality of biclusters and conduct our searching algorithm. We will prove its capability in restricting standard deviation and explain its usages in searching process. Relevant optimization strategies used in the searching algorithm are also discussed and analyzed.

3.1 Relating Range to Standard Deviation

Our criterion for choosing biclusters includes the standard deviation for all the values included in a bicluster. In order to construct relationship between range and standard deviation we define a few qualities computing standard deviations for individual rows and columns.

The *Range* for the i^{th} row of elements is denoted by $\delta_{i.}$, for the j^{th} column it is denoted by $\delta_{.j}$, and for whole bicluster it is denoted by Δ_B. The symbol $B_{i.}$ denotes all the data in the i^{th} rows of a bicluster B; $|B|_{i.}$ denotes the number of cells in the i^{th} row; $B_{.j}$ denotes the data in the j^{th} column of B; $|B|_{.j}$ denotes the number of data cells in j^{th} column; and μ and s denote the mean and standard deviation, defined as follows:

$$\begin{aligned} \mu_{i.} &= \frac{\sum_{d_{ip} \in B_{i.}} d_{ip}}{|B|_{i.}} \\ s^2_{i.} &= \frac{\sum_{d_{ip} \in B_{i.}} (d_{ip} - \mu_{i.})^2}{|B|_{i.}} \\ \mu_{.j} &= \frac{\sum_{d_{qj} \in B_{.j}} d_{qj}}{|B|_{.j}} \\ s^2_{.j} &= \frac{\sum_{d_{qj} \in B_{.j}} (d_{qj} - \mu_{.j})^2}{|B|_{.j}} \end{aligned} \tag{4}$$

Lemma 1. Given a bicluster $B = (sr, sc)$, if for $\{i \in sr|\ \delta_{i.} \leq S\}$ and $\{j \in sc|\ \delta_{.j} \leq S\}$, then $\Delta_B \leq 2 \times S$. (That is, if the *Range* for each row and each column of a bicluster is bounded by certain threshold S then the range for the whole bicluster is bounded by $2S$.)

PROOF. Let d_{ij} indicate the maximum value in the bicluster B, d_{pq} indicate the minimum value, $min(d_{i.})$ indicate the minimum value in the i^{th} row and $max(d_{.q})$ indicate the maximum value in the q^{th} column. From the definition of 1, we can derive the following inequalities:

$$\begin{aligned} d_{ij} - min(d_{i.}) &\geq d_{ij} - d_{iq} \\ max(d_{.q}) - d_{pq} &\geq d_{iq} - d_{pq}. \end{aligned} \tag{5}$$

The left hand side of each inequality is smaller than S and adding the two expressions on the left and right hand sides gives us:

$$d_{ij} - d_{pq} \leq 2 \times S. \tag{6}$$

□

Then a stronger conclusion about the relationship between the bound of the standard deviation and *Range* for the values in a bicluster can be derived.

Lemma 2. Given a bicluster $B = (sr, sc)$, if for $\{i \in sr|\ \delta_{i.} \leq S\}$ and $\{j \in sc|\ \delta_{.j} \leq S\}$, then the standard deviation s^2_B is less than $2 \times S^2$. (That is, if the range for each row and each column of a bicluster is less than S then the standard deviation of the bicluster is less than $2 \times S^2$.)

PROOF. When a bicluster $B = (sr, sc)$ has n rows and m columns, each of which has an upper bound of S on its Range. From equation 2, we can derive that:

$$\begin{aligned} s^2_{i.} &\leq \delta^2_{i.} \leq S^2 \\ s^2_{.j} &\leq \delta^2_{.j} \leq S^2 \end{aligned} \tag{7}$$

We use $\bar{\mu}$ to denote the mean of all individual row means, $\mu_{i.}$'s, s_μ to denote the standard deviation of all the individual row means, and μ is the mean of all the elements in the bicluster. Then we can say that:

$$\begin{aligned} s^2_B &= \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} (d_{ij} - \mu)^2}{nm} \\ &= \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} d^2_{ij} - nm\mu^2}{nm} \end{aligned} \tag{8}$$

$$\begin{aligned} \bar{\mu} &= \frac{\sum_{i=1}^{n} \mu_{i.}}{n} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij}}{nm} \\ &= \frac{\sum_{j=1}^{m} \mu_{.j}}{m} = \mu \end{aligned} \tag{9}$$

$$s^2_\mu = \frac{\sum_{i=1}^{n} (\mu_{i.} - \bar{\mu})^2}{n} = \frac{\sum_{i=1}^{n} \mu^2_{i.} - n\bar{\mu}^2}{n} \tag{10}$$

Combining equations 4 and 8 we get:

$$\begin{aligned} s^2_B &= \frac{\sum_{i=1}^{n} (s^2_i + \mu^2_i - \bar{\mu}^2)}{n} \\ &= \frac{\sum_{i=1}^{n} s^2_{i.}}{n} + s^2_\mu \\ &\leq \frac{\sum_{i=1}^{n} \delta^2_{i.}}{n} + s^2_\mu \end{aligned} \tag{11}$$

Also, for any row $p \in n$ we claim the following and then prove it by induction.

$$\left(\frac{\sum_{j=1}^{m}(d_{pj}-\mu_{.j})}{m}\right)^2 \leq \frac{\sum_{j=1}^{m}(d_{pj}-\mu_{.j})^2}{m} \tag{12}$$

The main steps of the induction proof, done on the number of columns, are as follows. Let $\phi_q = d_{pq} - \mu_{.q}$, and let k indicate the number of columns. When $k = 2$, equation 12 is satisfied. Now assuming the equation 12 to be correct for $k = \tau$, we get:

$$\left(\sum_{q=1}^{\tau}\phi_q\right)^2 \leq \tau * \sum_{q=1}^{\tau}\phi_q^2 \tag{13}$$

Then for $k = \tau + 1$

$$\begin{aligned}\left(\sum_{q=1}^{\tau+1}\phi_q\right)^2 &= \left(\sum_{q=1}^{\tau}\phi_q\right)^2 + 2*\left(\sum_{q=1}^{\tau}\phi_q\right)*\phi_{\tau+1} + \phi_{\tau+1}^2 \\ &\leq \left(\sum_{q=1}^{\tau}\phi_q\right)^2 + \sum_{q=1}^{\tau}\phi_q^2 + \tau*\phi_{\tau+1}^2 + \phi_{\tau+1}^2 \\ &\leq (\tau+1)*\sum_{q=1}^{\tau+1}\phi_q^2 \end{aligned} \tag{14}$$

The above is done for a single row, and summing for all rows we can derive that:

$$\begin{aligned} s_\mu^2 &= \frac{\sum_{i=1}^{n}(\mu_{i.}-\bar{\mu})^2}{n} \\ &= \frac{\sum_{i=1}^{n}(\sum_{j=1}^{m}(d_{ij}-\mu))^2}{m^2*n} \\ &\leq \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(d_{ij}-\mu)^2}{m*n} \\ &= \frac{\sum_{j=1}^{m}s_{.j}^2}{m} \leq \frac{\sum_{j=1}^{m}\delta_{.j}^2}{m} \end{aligned} \tag{15}$$

Combining conclusions from equations 7,11 and 15 we can derive that:

$$s_B^2 \leq 2 * S^2. \tag{16}$$

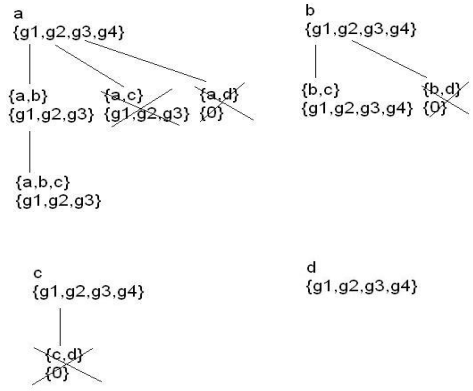
This means that by bounding the *Range* for each row and column ($\delta \leq S$), the standard deviation of the whole bicluster also gets bounded ($s_B \leq \sqrt{2}S$) which is also denoted as S' in definition 2. This conclusion is an important theoretical support for our search algorithm, which looks at biclusters as combinations of rows and columns and advances in the search space by adding columns or deleting rows. If the search algorithm wants to find biclusters with some bound on the standard deviation of its values, it could focus on a bound on the Range for each row and column separately. □

3.2 Enumerate Biclusters

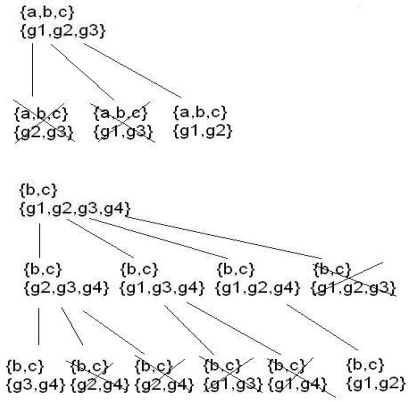
There are many ways incorporating *Range* in clustering procedure. What we are interested, also believed to be more practical to real problems, is to discover most interesting clusters base on definition 2 and 4. In order to discover biclusters with largest possible size, we need to limit the range of rows and columns separately which is also the reason we need supports from Lemma 2. We start our searching process by setting every single column with all rows as one searching branch of lattice. It is also applied for setting rows as one search branch. The basic operations for our search algorithm performed on intermediate biclusters are adding columns and removing rows. Biclusters which is covered by larger ones will not occurred in the final results. For example, in figure 1(b), $< \{g1, g2\}, \{a, b\} >$, $< \{g1, g2\}, \{a, c\} >$ and $< \{g1, g2\}, \{b, c\} >$ will not appear since they are covered by $< \{g1, g2\}, \{a, b, c\} >$.

Prefix-based equivalence classes have been used to formulate many search algorithms. We can form prefixes either from column headings or from row headings and in our case we have chosen to use the column headings. This helps divide the search space into independent sub-spaces of the search space at each level. This approach has been successfully adopted in [30] and [3] for searching biclusters in binary datasets.

Our search process can be viewed as made up of two independent phases. In the first phase, we generate children candidates by adding columns to each parent bicluster, updating the range for each row in the context of newly added columns, and removing those rows whose range values exceed the specified threshold. Such prefix tree based enumeration guarantees that every possible combinations of



(a) Phase1



(b) Phase2

Figure 2: Prefix Tree

columns will be examined. The first phase of the search algorithm for the example given earlier in figure 1 is shown in figure 2(a) here. In the second phase we re-examine all the generated candidate biclusters and check the Range values for each column. If a column's Range exceeds the specified threshold, the offending rows are removed from the candidate to make each column comply with the Range threshold.

At top level of each search branch, we enumerate every single column combined with all rows as one candidate bicluster. Since the order of the columns (a, b, c, d) are fixed, each candidate can only add columns that follow it. For example the only column that could be added to candidate $< \{g1, g2, g3\}, \{a, c\} >$ is d. After adding some columns and then removing rows, some candidate biclusters may violate the size constraint, such as $< \{\phi\}, \{a, d\} >$, $< \{\phi\}, \{b, d\} >$, and may be removed. The candidate $< \{g1, g2, g3\}, \{a, c\} >$ is removed because it is contained in an already generated bicluster.

Phase-one guarantees that all prefix combinations will be enumerated but the candidates may not comply with the constraint on the Range value for each column; In phase-two the algorithm removes some rows to bring each column within the acceptable Range limit. For example, in figure 2(b) $< \{g1, g2, g3, g4\}, \{b, c\} >$ generates $< \{g1, g2, g4\}, \{b, c\} >$ by removing $g3$, thus the only row could drop next is $g4$. After the removal the final results are $< \{g1, g2, g3\}, \{a, c\} >$, $< \{g1, g2\}, \{b, c\} >$ and $< \{g3, g4\}, \{b, c\} >$.

3.3 Pruning

To reduce the computational cost of the search, we employ a number of pruning strategies while still guaranteeing that all interesting biclusters will still be retained. These strategies are outlined below.

Pruning based on containment: In figure 2(a), our algorithms prune the candidate $< \{g1, g2, g3\}, \{a, c\} >$ since the depth first ordering of the search has already generated the hypothesis $< \{g1, g2, g3\}, \{a, b, c\} >$ which contains the former.

Pruning based on size: As stated in definition 4 we may compare the sizes of candidate biclusters and if only top k biclusters were needed from the entire search, we may without any loss, keep only top k candidates in each top level branch of the search and prune the rest.

Pruning based on similarity: In most real world datasets, a large number of the biclusters are *similar* to each other (definition 3). Our algorithm prunes the smaller sized biclusters among similar pairs of biclusters. All of the experiments reported in this paper have a threshold value of $\theta = 0.8$ as cutoff for pruning.

Pruning based on redundancy: Many real world datasets contain biclusters with very large number of rows. Instead of keeping all permutations of fewer rows as biclusters hypotheses, we delete from the parent bicluster those rows that reduce the Range the most and keep the rest of the rows in the hypotheses.

Even after previous prunings, the searching bicluster still could be more succinct. During the whole process, lattice could generate millions of biclusters. However what scientists really want to find is the biclusters which satisfy their own interestingness definitions. With modifications of definition 4, our algorithm could cut off the search branch which can not generate more interesting biclusters than those already had. Although this kind of pruning bring in some computations, compared to its reduction of searching space, it is really deserve. We also paralleled our algorithm by setting each searching branch as one thread. Hence the algorithm top K interesting biclusters for each thread and generate final biclusters by comparing those biclusters from each threads.

```
   input  : Data matrix DMX, Range δ, Share
            memory all current final biclusters Result,
            top interesting biclusters number K
   output: semi-qualified bicluster set BS
 1 begin
 2   Initialize BS each bs ∈ BS has one column with
     all rows ;
 3   while ∃bs ∈ BS can add more column do
 4     c = Next column id satisfying depth-first
       search prefix-tree;
 5     bs' = Add c to bs ;
 6     bs' remove rows which exceed δ;
 7     if bs' is interesting then
 8       Remove
         {bs''|bs'' ∈ BS ∧ bs'' ≈ bs' ∧ bs'' ≺ bs'} ;
 9       if no
         {bs''|bs'' ∈ BS ∧ bs'' ≈ bs' ∧ bs'' ≻ bs'}
         then
10         Add bs' to BS ;
11   if Result has K members then
12     for bs' ∈ Result do
13       Remove {bs''|bs'' ∈ BS ∧ bs'' ≺ bs'} ;
14   Return(BS)
15 end
```

Procedure 1. `Adding Columns`

3.4 Pseudo Code

The prefixes are constructed by each column and its combinations with those that follow it in the column ordering. Here we give the pseudo code of the algorithm to show how one of the prefix branches is pursued by the search algorithm (we call each branch a thread). The complete algorithm can be easily parallelized by having each thread run on a separate processor which also saves running times. We search biclusters from a real genetic dataset which contains 24190 rows and 5 columns on a computer equipped with a Intel Core 2 Quad 2.66GHz processor. By setting size limitation as 2000 rows, 2 column, range limitation as 2.1 and 4 threads working simultaneously, the searching process finishes in only 59 seconds.

3.5 Pruning Efficiency:

In order to analyze the effect of pruning we created a synthetic dataset with 25 rows and 25 columns, as shown in Figure 3(a); with the gray scale reflecting the cell values. There are four big blocks (biclusters) embedded in Figure 3(a), right-top, center, left-bottom and background, and the cell values within each block are distributed uniformly within a narrow range.

We count the number of intermediate candidate biclusters generated before the biclusters are output. The performances for various pruning strategies are shown in Figure 3(b). The x-axis shows the value of pre-specified Range

```
input  : Data matrix DMX, Range δ,
         semi-qualified bicluster set BS, top
         interesting biclusters number K,
output: Share memory all current final biclusters
         Result
1  begin
2    while ∃bs ∈ BS do
3      if System memory is not enough then
4        r = Next row id increasing interest the
         most;
5        bs' = Remove r from bs ;
6        Remove bs ;
7      else
8        r = Next row id satisfying depth-first
         search prefix-tree;
9        bs' = Remove r from bs ;
10     if bs' is interesting then
11       Remove
         {bs''|bs'' ∈ BS ∧ bs'' ≈ bs' ∧ bs'' ≺ bs'} ;
12       if no
         {bs''|bs'' ∈ BS ∧ bs'' ≈ bs' ∧ bs'' ≻ bs'}
         then
13         Add bs' to BS ;
14   Add BS to Result;
15   Keep top K interesting biclusters;
16   Return(Result)
17 end
```

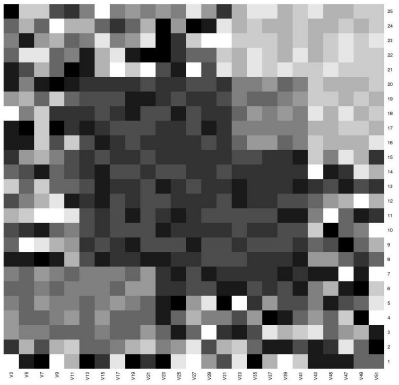
Procedure 2. `Removing Rows`

value and y-axis shows the number of intermediate biclusters. There are four cases plotted in Figure 3(b): the line with circles represents the performance of the original search algorithm based on pruning based on containment only; the line with triangles represents the performances of search using pruning based on size and containment; the line with crosses represents search with pruning based on similarity and containment; and the line with rectangles represents search with all the pruning strategies combined. Thus we can see that each pruning strategy has its impact on reducing the number of intermediate biclusters and using all the pruning techniques simultaneously performs the best.

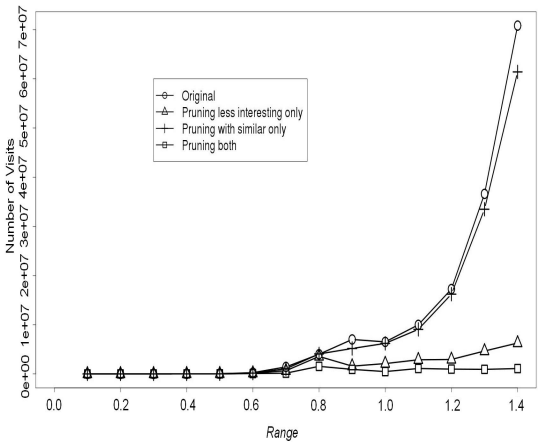
In real world dataset, the strategy of *pruning based on similarity* will greatly improve the performance. The reason why it dose not achieve much optimizations is that the the standard judging two biclusters are similar is very critical for this synthetic dataset. In our algorithm we set θ equals to 0.8 which means if two biclusters both have 9 columns and 9 rows, they are similar to each other only when 8 columns and 8 rows are the same.

4. EMPIRICAL EVALUATION

There are many biclustering algorithms can be compared with, we choose Cheng et al.'s algorithm [11] delegating direct biclustering algorithm and SAMBA[25] algorithm delegating graph theory based biclustering algorithm for synthetic dataset. We compare both accuracy and effectiveness of our algorithm with them. We also test our algorithm with two datasets from genomics domain to show the biological significance of output biclusters and compared with



(a) 25x25(uniform)



(b) Number Of Visits

Figure 3: Efficiency Test

two more recent algorithms: Co-clustering [14], OPSM [5] and ISA [18, 19].

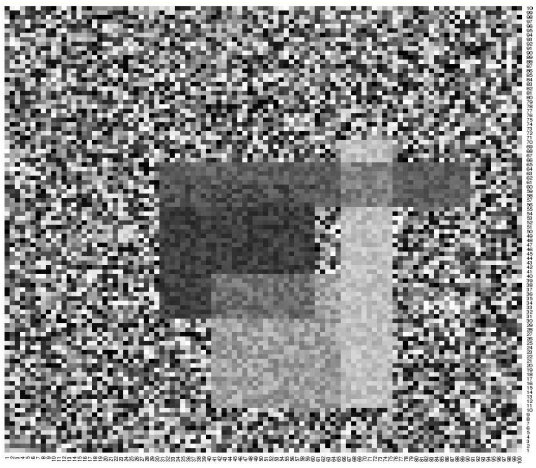
4.1 Synthetic Data Analysis

We consider the following metric for determining the quality of a bicluster found in a dataset. This evaluation metrics is not objective function for our algorithms and we do permit users to define their own interestingness setting by altering definition 4.

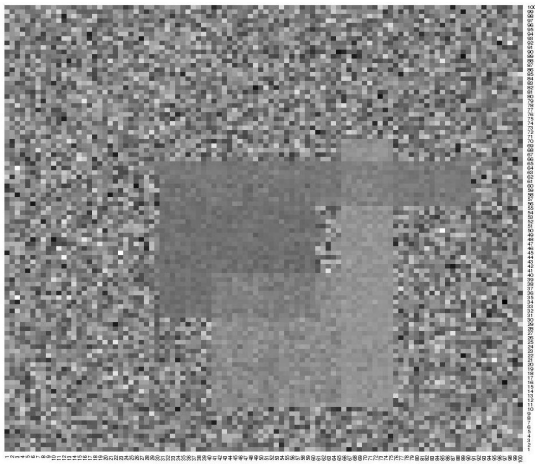
$$\lambda = \frac{\widehat{B_i} \times \widetilde{B_i} / s^2_{B_i}}{\widehat{D} \times \widetilde{D} / s^2_D}. \tag{17}$$

Here D represents the whole dataset, $\widehat{D}$ denotes its number of columns, $\widetilde{D}$ denotes its number of rows, and s_D denotes the standard deviation of D. This metric gives larger values for biclusters with larger sizes and smaller standard deviations. The metric is also normalized by s_D so that it is still meaningful across different datasets.

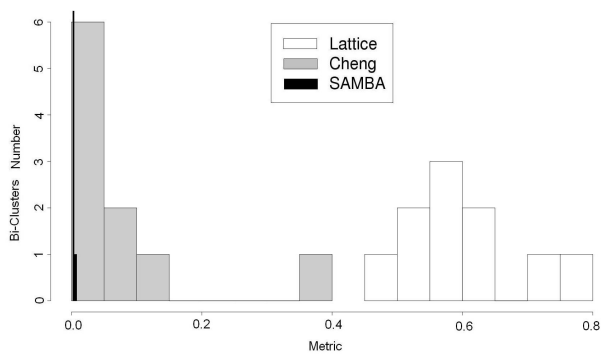
We have used two synthetic datasets, shown in Figure 4(a) and 4(b). The dataset in Figure 4(a) is 100 rows by 100 columns and values are reflected by the gray code intensity. There are five biclusters embedded in this dataset and all of them follow a uniform distribution of values. Four of the clusters are uniformly distributed around different cen-



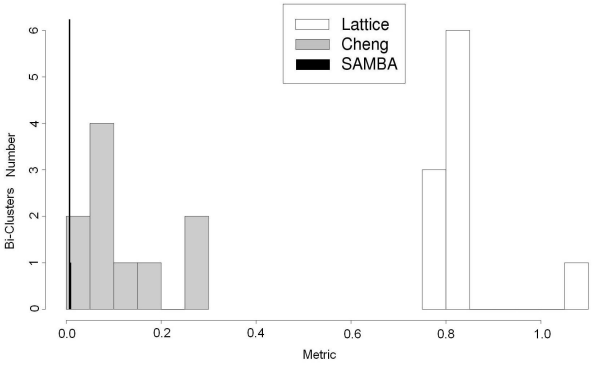
(a) 100x100(uniform)



(b) 100x100(normal)



(c) 100x100(uniform distribution)



(d) 100x100(normal distribution)

Figure 4: Synthetic Data

ters and values are within a range 1.2. The background cluster is distributed in the range of 2. Dataset in Figure 4(b) has the same size but the values of data cells in each biclusters follow normal distributions. Four overlapping biclusters are distributed normally with different μ and σ (less than 1.2). The background cluster is distributed normally with μ equals to zero and σ less than 2. We ran Cheng et al's algorithm by setting the size limit to 20 rows by 20 columns and our algorithm by setting the Range limit to 1.3. We also run SAMBA by setting option files type valsp_3p, with an overlap factor of 0.8, hashing kernel range from 1 to 7, and all other parameters as default value. We record the top 10 interesting biclusters for our clustering algorithm, first 10 biclusters generated by Cheng's algorithm and top 10 best biclusters based on metric value. The performance are presented in Figure 4(c). The x-axis in figure shows the metric value (λ) and y-axis shows the number of biclusters. There are three kind of bars in the figure: white bars represent the histogram of metric value for biclusters discovered by our algorithms; gray bars represent histogram of Cheng's algorithm and black bars represent histogram of SAMBA algorithm. biclusters discovered by our algorithm are shown to achieve the best quality as per the above metric.

Figure 4(d) shows the clustering comparisons for dataset in Figure 4(b). For Cheng et al's algorithm and SAMBA, parameter are set the same as in dataset in Figure 4(c). For our algorithm we extend range limit to 2.5 which covers more than two times the standard deviation for the normal distribution of biclusters ($\pm 2\sigma$). We see again from the second histogram that our algorithm performs significantly better that the other two. SAMBA achieve worse performance in both cases since it can not find biclusters with large size.

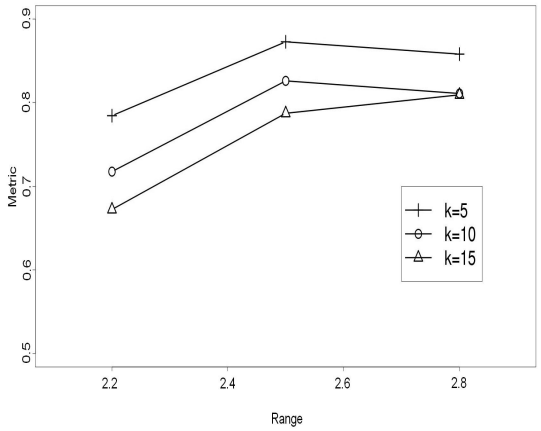


Figure 5: Effect of Parameter Changes

The impact of parameters on the performance of our algorithm can be analyzed by examining the evaluation metric

Algorithm	CLEAN Score	Variance Of Biclusters	Size (Rows X Cols)	Average Column STD
Lattice(our algorithm)	80.95	0.21	97x2	0.43
CC	33.65	1.27	847x12	0.89
OPSM	50.15	0.49	1726x6	0.53
Co-Clustering	40.44	0.99	469x2	0.99

Figure 6: Mouse Embryonic Stem Cell

Equation 17 for different values of the range and the value of k used for selecting the top k candidates. Apparently, our algorithm should generate more biclusters if we extend the range limit and/or increase the number k. Intuitively, extending range limit will increase allowable standard deviation but may also increase the size of the bicluster, and thus the metric may or may not be affected much. Meanwhile keeping more of the less interesting biclusters will reduce average of the metric value for the resulting biclusters. We use the dataset in Figure 4(b) to analyze the performance trend by modifying the parameters. For each parameter combination we record the average metric value for the obtained biclusters. Figure 5 shows the relationship between the Range parameter and the average of the metric value obtained. There are three lines in the figure: the top line with crosses shows the performance when we keep the top 5 most interesting biclusters based on size criterion; the line with circles shows the performance while keeping the top 10; and line with triangles shows the performance while keeping the top 15. When we extend the range from 2.2 to 2.5, the metric value increase due to a faster increase in the sizes of the resulting biclusters than the increases in their variances. However, for range values larger than 2.5 the increase in sizes is slower than the increase in the variances; see Figure 5; (This dataset consists of values that follow a normal distribution). Consequently, we can conclude that extending the **Range** does not always improve the performance because after certain point the increase of bicluster's size can not compensate for the decrease of accuracy (or increase in STD). Also, we see that keeping a smaller number of most interesting biclusters will always increase the performance for a specific value of Range because the standard deviation of those biclusters is relatively stable.

4.2 Mouse Embryonic Stem Cell Dataset

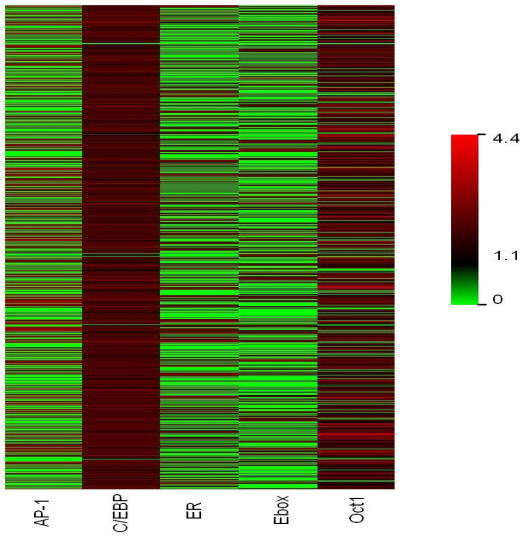
The authors Ouyang et al. [21] have reported their CHIP-Sequence scores on mouse embryonic stem cell in [10]. In this dataset the rows represent genes, the columns represent transcription factors (proteins), and the cell-values represent the strength of binding between the row and column elements. Twelve proteins and 18936 genes included in this dataset have been known to show correlations in some other studies. Using our algorithm on their original data without any normalization, we seek to discover underlying co-factor mechanisms. One bicluster unveiled co-regulated TFs (Nanog and Oct4) with a variance of 0.21 and that is well corroborated by [10]. In order to demonstrate the functional coherence of the genes co-regulated in the bicluster, we use the CLEAN [16] metric to check the functional enrichment with Gene Ontology terms [4]. The higher the CLEAN score the better is the functional coherence of the genes. Low-Variance biclusters found by our algorithm show the highest CLEAN-Score values and the lowest variance when compared with the biclusters found by other methods (first row in table of Figure 6 shows our results). It should be noted that traditional biclustering algorithms could discover biclusters with relatively low standard deviations within each column of a bicluster but the variance of the whole data blocks is larger, and therefore they could not find highly functionally correlated gene sets; and therefore their CLEAN scores are lower. For each algorithm, the data shown in the table represents the bicluster with highest CLEAN score (if many biclusters were found then the best one was selected and reported); the table also lists the bicluster variance and the average column standard deviation for each reported bicluster (Figure 6).

4.3 Human Genomic Dataset

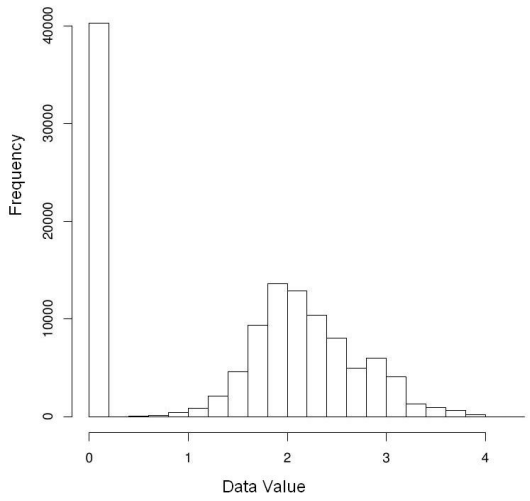
We consider the dataset from human genome from hg18 [1] and calculate the maximum possible relative probability associated with each gene-motif pair using the Sequence Motif-Matching Scoring model [12]. The data contains 24190 genes (rows) and 287 motifs (columns). Relative probability values in data cells are in the range of $[0, 4.3]$. The data is taken from [24]. The other source of data that we use is based on experiments [9]. They present the distributions of five motifs (ERE, AP-1, Oct, FKH and C/EBP) for these genes and also the pair-wise relationships between those motifs. The heat map of these five motifs with 24190 genes is shown in Figure 7(a) and the distribution of values is shown in Figure 7(b) in the format of histogram.

We want to see whether biclusters found by our algorithm in the theoretically obtained data match the ones reported in the experimental results. We use the *Fisher's Exact Test* to determine whether our clustering algorithm could really generate the biclusters predicted by the experimental results. The criteria used here is the negative of log 10 based p-value, meaning the higher the value the more significantly the two sets match. Results of our algorithm for various parameter values are shown in Figure 7(d). As we expected, keeping Range the same and increasing the minimum row-limit size reduces the number of clusters discovered (the first row and the second row), and increasing the Range bound discovers more biclusters but makes the accuracy worse (the second, third, and fourth rows). The best p-value of this test is 2.90×10^{-7} (or 6.54 for $-\log_{10}(pvalue)$) in the third row which is much smaller than conventional cutoff 0.05 (or 1.3 for $-\log_{10}(pvalue)$). Therefore we conclude that the biclusters discovered by our algorithm significantly overlap with the experimental results.

We also compare the results with some well known algorithms [11, 5, 18, 19, 14] using the metric defined in equation 17. Parameters used in all of the algorithms are kept as the default ones. For Cheng et al.'s algorithm, we re-



(a) Heatmap of Motif Data



(b) Motif Data Distribution

Algorithm	λ	$\widehat{\lambda}$	Average Column STD
Lattice(our algorithm)	0.91;0.80;0.81	0.84	0.31;0.25;0.32
CC	0.45	0.45	0.94
OPSM	0.19;0.22;0.23;0.23	0.22	0.31;0.57;0.77;0.87
Co-Clustering	0.43;0.29;0.19	0.30	0.27;0.63;0.34

(c) Motif Bicluster Comparison

δ	Row Limit	biclusters	$-\log_{10}(pvalue)$
2.1	6000	2	3.56
2.1	4000	3	6.54
2.3	4000	7	5.17
2.5	4000	8	5.35

(d) Motif Bicluster Statistics

Figure 7: Genomic Data Validation

tained the first bicluster that is generated; for biclustering algorithm ISA [18, 19], we could not find any biclusters; for biclustering algorithm OPSM [5], we kept all the biclusters generated; for our algorithm, we kept the top 3 biclusters with minimum size of 4000 rows and 2 columns in which data values have a range of 2.1 and for Co-clustering [14] algorithm we keep the same number of biclusters as our algorithm. The biclustering results are summarized using the metric (λ from equation 17) in Figure 7(c). We first listed all metric values for each bicluster generated in the second column, then we took the average of those metric values for each algorithm and it is reported in the third column of the table. ISA did not find any biclusters, so we did not put in the table. It is clear from the results that our algorithm could discover biclusters with largest metric values, either considering individual biclusters or their average.

5. CONCLUSION

We have presented a search based algorithm for discovering low-variance biclusters in sequencing datasets and have shown that it performs much better than several other competing algorithms using a statistical metric for merit. Our algorithm can enumerate overlapping biclusters and generate the top K interesting biclusters based on the specified size and standard deviation requirements. Other algorithms are not capable of discovering all overlapping biclusters and controlling the variance at the same time. Challenges still exist for discovering the complete set of low-variance biclusters because our algorithm presented here generates only those biclusters that satisfy the low-variance criterion but it cannot discover all low-variance biclusters - particularly those that have low variance despite a large range for the included values. But the combination of large range and low variance is not desirable for the sequencing data applications and our algorithm is therefore very suitable.

6. REFERENCES

[1] Ucsc genome browser website:. *http://genome.ucsc.edu/*.

[2] V. A, J. DS, S. A, M. C, A. E, and et al. Genome-wide analysis of transcription factor binding sites based on chip-seq data. *Nat Methods*, 5:829–834, 2008.

[3] F. Alqadah and R. Bhatnagar. An effective algorithm for mining 3-clusters in vertically partitioned data. *In Proceeding of the 17th ACM conference on*

Information and knowledge management, pages 1103–1112, 2008.

[4] M. Ashburner, C. Ball, J. Blake, D. Botstein, H. B. J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, and et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1), 2000.

[5] B. C. Ben-Dor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. *In Proceedings of the 6th International Conference on Computational Biology (RECOMB-02)*, pages 49–57, 2002.

[6] H. Bian and R. Bhatnagar. An algorithm for lattice-structured subspace clustering. *Proceedings of the SIAM International Conference on Data Mining*, 2005.

[7] H. Bian, R. Bhatnagar, and B. Young. An efficient constraint-based closed set mining algorithm. *In Proceedings of the 6th international confernce on Machine Learning*, pages 172–177, 2007.

[8] K. Bryan, P. Cunningham, and BolshakovaN. Biclustering of expression data using simulated annealing. *In Proceedings of the 18th IEEE symposium on computer-based medical systems*, pages 383–388, 2005.

[9] J. S. Carroll, C. A. Meyer, J. Song, W. Li, T. R. Geistlinger, J. Eeckhoute, A. S. Brodsky, E. K. Keeton, K. C. Fertuck, G. F. Hall, Q. Wang, S. Bekiranov, V. Sementchenko, E. A. Fox, P. A. Silver, T. R. Gingeras, X. S. Liu, and M. Brown. Genome-wide analysis of estrogen receptor binding sites. *Nature Genetics*, 38:1289–1297, 2006.

[10] X. Chen, H. Xu, P. Yuan, F. Fang, M. Huss, V. B. Vega, E. Wong, Y. L. Orlov, W. Zhang, J. Jiang, Y.-H. Loh, H. C. Yeo, Z. X. Yeo, V. Narang, K. Ramamoorthy, Govindarajan, B. Leong, A. Shahab, Y. Ruan, G. Bourque, W.-K. Sung, N. D. Clarke, C.-L. Wei, and H.-H. Ng. Integration of external signaling pathways with the core transcriptional network in embryonic stem cells. *Cell*, 133:1106âĂŞ1117, 2008.

[11] Y. Cheng and G. Church. Biclustering of expression data. *In Proceedings of the 8th international conference on intelligent systems for molecular biology*, pages 93–103, 2000.

[12] E. Conlon, X. Liu, J. Lieb, and J. Liu. Integrating regulatory motif discovery and genome-wide expression analysis. *Proc. Natl. Acad. Sci. U.S.A.*, 100(6):3339–3344, 2003.

[13] N. D, C. S, and B. K. Empirical methods for controlling false positives and estimating confidence in chip-seq peaks. *BMC Bioinformatics*, 9:523, 2008.

[14] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. *In Proceedings of the 7th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining(KDD)*, 2001.

[15] I. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. *In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 89–98, 2003.

[16] J. M. Freudenberg, V. K. Joshi, Z. Hu, and M. Medvedovic. Clean: Clustering enrichment analysis. *BMC Bioinformatics*, 10(234), 2009.

[17] B. Ganter and R. Wille. Formal concept analysis: Mathematical foundations. *Springer-Verlag,Heidelber*, 1999.

[18] J. Ihmels, S. Bergmann, and N. Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20:1993 – 2003, 2004.

[19] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31:370–377, 2002.

[20] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[21] Z. Ouyang, Q. Zhou, and W. H. Wong. Chip-seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells. *PNAS*, 106(51):21521–21526, 2009.

[22] P. PJ. Chip-seq: advantages and challenges of a maturing technology. *Nat Rev Genet*, 10:669âĂŞ680, 2009.

[23] P. S, W. B, and M. A. Computation for chip-seq and rna-seq studies. *Nat Methods*, 6:S22âĂŞ32, 2009.

[24] K. Shinde, M. Phatak, J. M. Freudenberg, J. Chen, Q. Li, V. Joshi, Z. Hu, K. Ghosh, J. Meller, and M. Medvedovic. Genomics portals: Integrative web-platform for mining genomics data. *BMC Genomics*, 11(1), 2010.

[25] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant bilcusters in gene expression data. *Bioinformatics*, 18:136–144, 2002.

[26] L. TD, R. S, T. S, L. R, A. T, and et al. A practical comparison of methods for detecting transcription factor binding sites in chip-seq experiments. *BMC Genomics*, 10:618, 2009.

[27] Z. Y, L. T, M. C, E. J, J. D, and et al. Model-based analysis of chip-seq (macs). *Genome Biology*, 9:R137, 2008.

[28] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: capturing subspace correlation in a large data set. *In Proceedings of the 18th IEEE International Conference On Data Engineering*, pages 517–528, 2002.

[29] S. Yoon, L. Benini, and D. M. G. Co-clustering: A versatile tool for data analysis in biomedical informatics. *Information Technology in Biomedicine,IEEE Transactions on*, 11:493–494.

[30] M. J. Zaki and K. Gouda. Fast vertical mining using diffsets. *In 9th International Conference on Knowledge Discovery and Data Mining*, 2003.