Scientific Programming 20 (2012) 15–27
DOI 10.3233/SPR-2012-0336
IOS Press

Mining low-variance biclusters to discover coregulation modules in sequencing datasets

Zhen Hu and Raj Bhatnagar *
School of Computing Sciences and Informatics, University of Cincinnati, Cincinnati, OH, USA
E-mails: huze@mail.uc.edu, Raj.Bhatnagar@uc.edu

Abstract. High-throughput sequencing (CHIP-Seq) data exhibit binding events with possible binding locations and their strengths, followed by interpretation 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 techniques on these scores can help discover important knowledge such as the potential co-regulation or co-factor mechanisms. The ideal biclusters generated would contain subsets of genes and transcription factors (TF) such that the cell-values in biclusters are distributed around a mean value with very low variance. Such biclusters would indicate TF sets regulating gene sets with very similar probability values. However, most existing biclustering algorithms neither enforce low variance as the desired property of a bicluster, nor use variance as a guiding metric while searching for the desirable biclusters. In this paper we present an algorithm that searches a space of all overlapping biclusters organized in a lattice, and uses an upper bound on variance values of biclusters as the guiding metric. We show the algorithm 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, CHIP-Seq and motif datasets, and compare them with the results obtained by other algorithms to demonstrate the power and effectiveness of our algorithm.

Keywords: Classification, clustering

1. Background and motivation

Mining biclusters from sequencing datasets is one of the important ways to discover knowledge about potential biological mechanisms. Transcription Factors' (TF) bindings related sequencing datasets, including High-throughput Chromatin Immunoprecipitation Sequencing (CHIP-Seq) [21] and motif searching [10] datasets, record potential matchings on genome along with many different metrics. For example, CHIP-Seq peaks record intensity and position values. 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 very distinct binding scenarios. Biclusters consisting of subsets of genes and TFs and containing very similar values can help provide insights into coregulation. However, traditional methods [9,11,12,28] cannot be adapted easily to analyze the sequencing datasets because most of them do not seek biclusters with specifiable 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) [17,19,22,26,30]. Many current methods summarize all peaks within up and down regions of each gene into a unified score by combining the information of distances from peaks to transcription start sites (TSS) and the information of binding strengths together. For example, Ouyang et al. [20] 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.

*Corresponding author. E-mail: Raj.Bhatnagar@uc.edu

	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)

$\langle\{g1, g2\}, \{a, b, c\}\rangle$
$\langle\{g1, g2\}, \{a, b\}\rangle$
$\langle\{g1, g2\}, \{a, c\}\rangle$
$\langle\{g1, g2\}, \{b, c\}\rangle$
$\langle\{g3, g4\}, \{b, c\}\rangle$

(b) Biclusters

Fig. 1. Example biclusters. (a) Data table; (b) biclusters.

The motifs are defined as position weighted matrices (TRANSFC), and the final matching scores are computed by using the method given in [10]. Both of these 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 Fig. 1(a) in which the values are quite similar to the CHIP-Seq scores and motif matching scores. Biclusters shown in Fig. 1(b) are such that the values in the selected cells are all about the same (std. dev. < 0.5) and the biclusters also satisfy a selected size constraint, which in this case is: biclusters should contain at least two rows and two columns. For binary datasets the theory of Formal Concept Analysis [14] treats all maximum sized sub-matrices containing only 1's as concepts and arranges 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 threshold. The parent–child relationship in the lattice is still defined by the superset–subset relationship among the attributes included in the biclusters. In our extension of the analogy to FCA lattices, each node of the lattice may contain more than one bicluster. Biclusters shown in Fig. 1(b) meet all the above requirements and are qualified to be *concepts* in the sense outlined above.

Potential co-factor or co-regulation mechanisms could be discovered from sequencing datasets by taking subsets of genes and subsets of TFs such that all TFs have very similar binding probability with selected genes (or low-variance for cell values within each sub-matrix). The problem of discovering the qualified biclusters, including the ones that may overlap some other biclusters, is NP-Hard [18] and most of the proposed algorithms attack the problem in a greedy manner [6,9]. These algorithms, however, do not take into account the cell-values' variance. Some other algorithms utilize pattern recognition techniques [27] to improve the quality of clusters but they miss out on the many potential good overlapping biclusters due to imposing hard pattern and non-overlap restrictions on real valued data. There are also many biclustering algorithms which are based on various statistical properties of the data [11,12,24]. These algorithms use their own optimizing metrics for clustering and it is not possible to control the variance of the cell-values in biclusters by using these metrics.

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 variance or the standard deviation of the included 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 [1,4,5,29] and our work in this paper is the first attempt to advance the same idea to datasets with real-valued entries in the cells.

In the following sections we formally define some ideas including a quantitative monotonic property that can be used to bound the standard deviation of a non-closed bicluster. In Section 3 we prove the relationship between our proposed monotonic metric and the standard deviation of the cells 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 (and a vector).

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 differences between individual elements and the mean of all the elements, and the value of *Range* is an upper bound on the values of these differences. Consequently, we can derive the relationship between the standard deviation and the range for single dimensional data in Eq. (2), which captures the idea that by limiting the Range, the standard deviation (s) is also limited, and is given by:

$$s^2 \leqslant \delta^2. \tag{2}$$

From the point of view of formulating a search algorithm for the state space of biclusters, we need a quantity that monotonically increases (or decreases) as the size of potential biclusters increases (or decreases). 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 potential biclusters, can be used to prune some non-promising search paths and thus efficiently determine the most promising biclusters.

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 [3,9,11,15,16,24,27] using different metrics to define interesting biclusters. The metrics determine the desirability of a bicluster and are also used as a guiding heuristic for the search of the desired biclusters. We give here our definition of interesting biclusters which, in contrast to 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) $\widehat{B} \geqslant m$; (ii) $\widetilde{B} \geqslant n$; and (iii) $s_B \leqslant S'$, where m and n are pre-specified row and column minimum size thresholds and S' is the threshold for the standard deviation.

In order to compare two biclusters, we also define several bicluster comparison operators that will be used for pruning some non-promising branches by the search algorithm presented in Section 3.3.

Definition 3.

(1) A biclusters $B_1 = (sr_1, sc_1)$ is contained in bicluster $B_2 = (sr_2, sc_2)$, 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,

$$\frac{|sr_1 \cap sr_2|}{|sr_1 \cup sr_2|} \geqslant \theta \quad \text{and} \quad \frac{|sc_1 \cap sc_2|}{|sc_1 \cup sc_2|} \geqslant \theta;$$

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

For comparing two biclusters' 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, increasing the *Range* bound that makes biclusters acceptable will lead our algorithm to generate biclusters of larger sizes. The relationship between the bicluster size and the *Range* bound can be easily defined, if needed, and implemented in our algorithm to control the size of the desired biclusters.

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

$$\widehat{B_1} \times \widetilde{B_1} \geqslant \widehat{B_2} \times \widetilde{B_2}. \tag{3}$$

3. Search algorithm

The quantitative metric *Range* for the values included in a candidate bicluster is used to control the statistical quality of the enumerated biclusters and thus conduct our search for the desirable biclusters. We prove in a later section its ability to restrict the standard deviation of biclusters and explain its usages in the search 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 the relationship between the *range* and the *standard deviation* we define a few properties for computing the standard deviation for individual rows and columns.

The *Range* for the ith row of elements is denoted by $\delta_{i.}$, for the jth column it is denoted by $\delta_{.j}$, and for whole bicluster it is denoted by Δ_B. The symbol $B_{i.}$ denotes all the values in the ith row of a bicluster B; $|B|_{i.}$ denotes the number of cells in the ith row; $B_{.j}$ denotes the values in the jth column of B; $|B|_{.j}$ denotes the number of data cells in the jth 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 $\{i \in sr \mid \delta_{i.} \leqslant S\}$ and $\{j \in sc \mid \delta_{.j} \leqslant S\}$, then $\Delta_B \leqslant 2\times S$. (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 ith row and $\max(d_{.q})$ indicate the maximum value in the qth column. From Definition 1, we can derive the following relationships:

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

The left-hand side of each of the above two inequalities is smaller than S, and therefore, adding the two expressions on the left and the right-hand sides gives us:

$$d_{ij} - d_{pq} \leqslant 2 \times S. \qquad \square \tag{6}$$

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

Lemma 2. *Given a bicluster $B = (sr, sc)$, if for $\{i \in sr \mid \delta_{i.} \leqslant S\}$ and $\{j \in sc \mid \delta_{.j} \leqslant S\}$, then square of the standard deviation for the values in the bicluster B, 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 square of 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, using Eq. (2), we can derive that:

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

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 Eqs (4) and (8) we get:

$$\begin{aligned} s_B^2 &= \frac{\sum_{i=1}^{n}(s_i^2 + \mu_i^2 - \bar{\mu}^2)}{n} \\ &= \frac{\sum_{i=1}^{n} s_{i.}^2}{n} + s_\mu^2 \\ &\leqslant \frac{\sum_{i=1}^{n} \delta_{i.}^2}{n} + s_\mu^2. \end{aligned} \tag{11}$$

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

$$\begin{aligned} &\left(\frac{\sum_{j=1}^{m}(d_{pj} - \mu_{.j})}{m}\right)^2 \\ &\leqslant \frac{\sum_{j=1}^{m}(d_{pj} - \mu_{.j})^2}{m}. \end{aligned} \tag{12}$$

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 $k = 2$, Eq. (12) is satisfied. Now assuming Eq. (12) to be correct for $k = \tau$, we get:

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

Then for $k = \tau + 1$

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

The derivation in the previous step employs the following inequalities: $2 * a * b \leqslant a^2 + b^2 \leqslant a^2 + n * b^2$.

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

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

Combining conclusions from Eqs (7), (11) and (15) we can derive that:

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

This means that by bounding the *Range* for each row and column ($\delta \leqslant S$), the standard deviation of the whole bicluster also gets bounded ($s_B \leqslant \sqrt{2}S$) which is also denoted as S' in Definition 2. This conclusion is an important analytical 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 of incorporating the *Range* metric in the biclustering procedure. What we are interested in, and is also very useful for many real problems, is to discover the most interesting biclusters as given by Definitions 2 and 4. In order to discover biclusters with largest possible size, we need to maximize the size while limiting the range for individual rows and columns individually as required by Lemma 2. We start our search process by setting every single column of the data table, with all rows included, as one branch of the state space; and each such branch corresponds to a high level node in the lattice of all possible overlapping biclusters. The search can also be formulated by interchanging the roles of rows and columns of the dataset. The basic operations of our search algorithm performed on intermediate biclusters, for enumerating successor states, are adding columns and removing rows. Biclusters that are completely subsumed by oth-

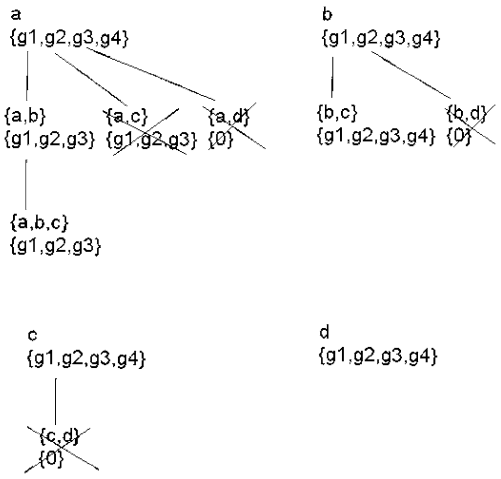
ers generated earlier are dropped from further consideration. For example, in Fig. 1(b), $\langle\{g1, g2\}, \{a, b\}\rangle$, $\langle\{g1, g2\}, \{a, c\}\rangle$ and $\langle\{g1, g2\}, \{b, c\}\rangle$ will not appear since they are covered by $\langle\{g1, g2\}, \{a, b, c\}\rangle$.

Prefix-based equivalence classes have been used to formulate many search algorithms and the same approach is used by us in this formulation. 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 [1] and [29] for searching for biclusters in binary datasets.

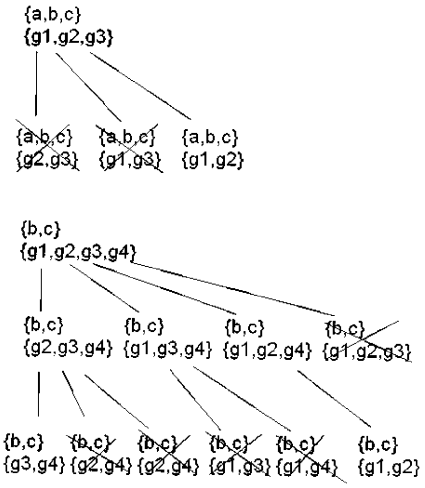
Our search process can be viewed as made up of two independent phases. In the first phase, we generate all children subspaces by adding new single columns to each parent subspace, in the prefix tree ordering, updating the range value for each row in each subspace enumerated by the newly added columns, and removing those rows from each subspace whose range values exceed the specified threshold. Such prefix tree based enumeration of subspaces guarantees that every possible combination of columns will be examined by the search algorithm. The first phase of the search algorithm for the example given earlier in Fig. 1 is shown in Fig. 2(a). This phase creates all those subspaces within which rows having values within the range-threshold exist. Each such subspace may contain multiple rows, and different subsets of these rows may form biclusters with each column obeying the range-threshold. Therefore, in the second phase we re-examine all the enumerated subspaces and evaluate the range values for each column. Each feasible and maximal subset of rows in a subspace, all of whose columns obey the range threshold, are retained as candidate biclusters.

At the top level of each search branch, we enumerate every single column containing all the rows as forming one of the initial candidate biclusters. 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 $\langle\{g1, g2, g3\}, \{a, c\}\rangle$ is d. After adding some columns and then removing some rows, some candidate biclusters may violate the minimum-size constraint, such as $\langle\{\phi\}, \{a, d\}\rangle$, $\langle\{\phi\}, \{b, d\}\rangle$, and may be removed. The candidate $\langle\{g1, g2, g3\}, \{a, c\}\rangle$ is removed because it is contained in an already generated bicluster.

Search completeness: The first phase guarantees that all prefix combinations of columns will be enumerated as subspaces but all the rows in each subspace may not



(a) Phase 1



(b) Phase 2

Fig. 2. Prefix tree.

comply with the constraint on the range value for each column. In the second phase the algorithm enumerates all possible maximal subsets of rows in each subspace such that each column of these maximal subsets obeys the range threshold, and all possible combinations of row subsets are enumerated. For example, in Fig. 2(b) the lower tree shows that $\langle\{g1, g2, g3, g4\}, \{b, c\}\rangle$ generates $\langle\{g1, g2, g4\}, \{b, c\}\rangle$ etc., four subsets containing three rows each. If the columns of a subset of rows at this level in the tree follow the range limit then its further subsets are not enumerated; otherwise the smaller subsets of size two are enumerated, as shown in the figure. Since all possible combinations of columns are potentially enumerated, and within each column combination all possible row combinations are enumerated, we can be sure of that all possible biclus-

ter candidates, if following the range threshold, will be enumerated and evaluated by the search algorithm. However, this extensive enumeration is not needed. Only those column subspaces are retained that contain some minimum number of rows each of which follows the range threshold. This eliminates a large number of column combinations. Also, only maximal subsets of rows are then enumerated in each subspace, many smaller candidates for biclusters are not enumerated. A few more pruning strategies are adopted to make the search process efficient and they are described below.

3.3. Pruning

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

Pruning based on containment: In Fig. 2(a), our algorithms prune the candidate $\langle\{g1, g2, g3\}, \{a, c\}\rangle$ since the depth first ordering of the search has already generated the hypothesis $\langle\{g1, g2, g3\}, \{a, b, c\}\rangle$ 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 applying the above prunings, the search for biclusters can be made even more efficient. During the search process, the lattice shaped search space could enumerate millions of biclusters. However, biomedical scientists in the case of our datasets, and domain experts in general, can define their own domain specific interestingness definitions. Monotonicity preserving versions of these interestingness conditions can be utilized to further prune the search space. Our algorithm is also very easily amenable for parallel implementations and this is achieved by setting each search branch as one thread. Hence, the algorithm can find the top K interesting biclusters for each thread and then generate the final top K biclusters by comparing the biclusters from each thread.

```
   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 saves the running time. In the results reported here we search biclusters from a real genetic dataset which contains 24 190 rows and 5 columns on a computer equipped with an Intel Core 2 Quad 2.66 GHz processor. By setting size limitation as 2000 rows, 2 column, range limitation as 2.1, and 4 threads working simultaneously, the search process took at most 30 s across various runs of the algorithm.

input : Data matrix DMX, Range δ, semi-qualified bicluster set BS, top interesting biclusters number K,
output: Share memory all current final biclusters *Result*

begin
  while $\exists bs \in BS$ **do**
    if *System memory is not enough* **then**
      r = Next row id increasing interest the most;
      bs' = Remove r from bs ;
      Remove bs ;
    else
      r = Next row id satisfying depth-first search prefix-tree;
      bs' = Remove r from bs ;
    if bs' *is interesting* **then**
      Remove $\{bs''|bs'' \in BS \wedge bs'' \approx bs' \wedge bs'' \prec bs'\}$;
      if *no* $\{bs''|bs'' \in BS \wedge bs'' \approx bs' \wedge bs'' \succ bs'\}$ **then**
        Add bs' to BS ;
  Add BS to *Result*;
  Keep top K interesting biclusters;
  `Return(`*Result*`)`
end

Procedure 2. Removing rows.

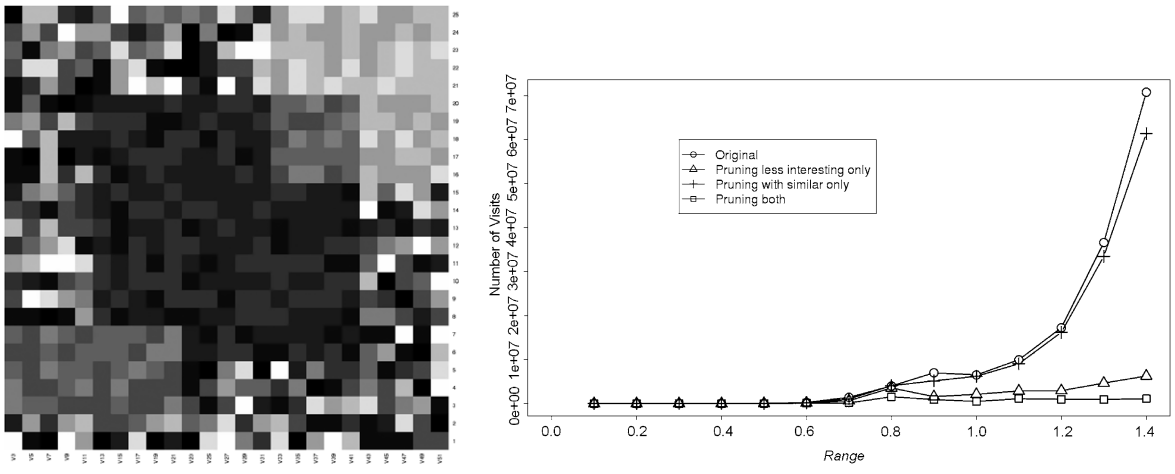
3.5. Pruning efficiency

In order to analyze the effect of pruning we created a synthetic dataset with 25 rows and 25 columns, as shown in Fig. 3(a); with the gray scale reflecting the cell values. There are four big blocks (biclusters) embedded in Fig. 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 final biclusters are output. The performance for various pruning strategies is shown in Fig. 3(b). The x-axis shows the value of prespecified Range value and y-axis shows the number of intermediate biclusters. There are four cases plotted in Fig. 3(b): the line with circles represents the performance of the original search algorithm that uses pruning based only on containment; the line with triangles represents the performances of search algorithm 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.

4. Empirical evaluation

There are many biclustering algorithms that can be compared with our algorithm. We choose Cheng et al.'s algorithm [9] representing a direct biclustering algo-



(a) (b)

Fig. 3. Efficiency test. (a) 25 × 25 (uniform); (b) number of visits to candidate biclusters.

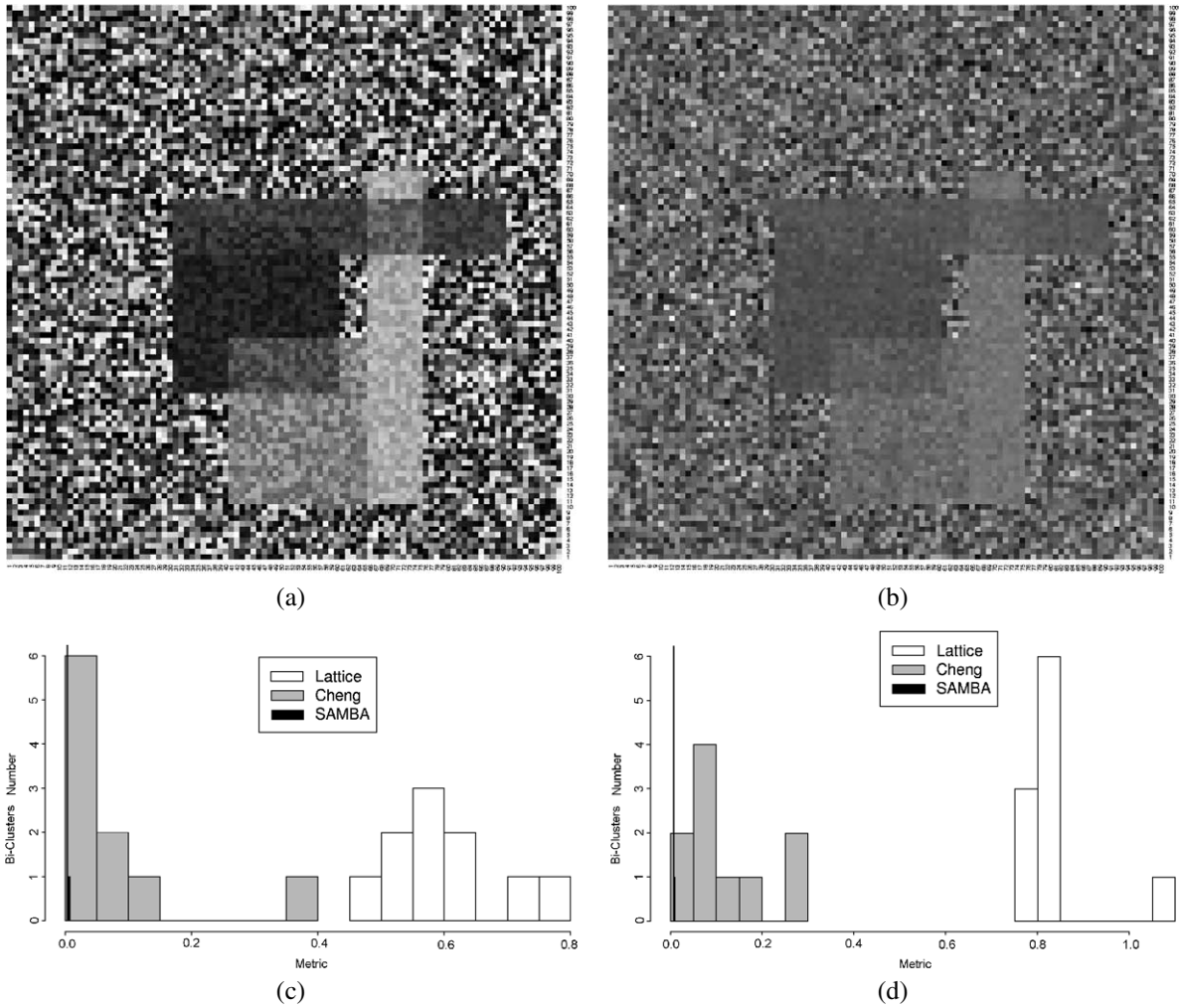


Fig. 4. Synthetic data. (a) 100 × 100 (uniform); (b) 100 × 100 (normal); (c) 100 × 100 (uniform distribution); (d) 100 × 100 (normal distribution).

rithm and SAMBA [24] algorithm representing graph-theory based biclustering algorithm for comparison of results with our synthetic dataset. We compare both accuracy and effectiveness of our algorithm with these two other algorithms. We also test our algorithm with two datasets from genomics domain to show the biological significance of output biclusters and compare these results with three other recent algorithms: Co-clustering [11], OPSM [3] and ISA [15,16].

4.1. Synthetic data analysis

We consider the following metric for determining the quality of a bicluster found in a dataset.

$$\lambda = \frac{\widehat{B_i} \times \widetilde{B_i}/s_{B_i}^2}{\widehat{D} \times \widetilde{D}/s_D^2}. \tag{17}$$

Here D represents the complete 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 meaningful for comparisons across different datasets.

We have used two synthetic datasets, shown in Fig. 4(a) and (b). The dataset in Fig. 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 centers and values are within a range of 1.2. The background cluster is distributed with a range of 2.0. Dataset in Fig. 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 also 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 ran SAMBA algorithm 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 to default values. We recorded the top 10 interesting biclusters for our clustering algorithm, the first 10 biclusters generated by Cheng's algorithm and the top 10 best biclusters from SAMBA based on the metric value described above. The performances are presented in Fig. 4(c). The x-axis in this figure shows the metric value (λ) and the y-axis shows the number of biclusters with that metric value. There are three types of bars in the figure: the white bars represent the histogram of metric value for biclusters discovered by our algorithms; the gray bars represent histogram of output from Cheng's algorithm and the black bars represent the histogram of output from the SAMBA algorithm. Better quality biclusters have larger metric values and the plot shows that the biclusters discovered by our algorithm achieve the best quality as per the metric.

Figure 4(d) shows the clustering comparison for the normally distributed dataset shown in Fig. 4(b). For Cheng et al.'s algorithm and SAMBA, parameter are set in the same way as for the dataset in Fig. 4(c). For our algorithm we extend the 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 than the other two.

The impact of selecting different parameter values on the performance of our algorithm can be analyzed by examining the resulting values of the metric given by Eq. (17) by running our algorithm 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 shown in Fig. 4(b) to analyze the performance trends by modifying the parameters. For each parameter combination we recorded 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 the size criterion; the line with the circles shows the performance while keeping the top 10; and the line with the triangles shows the performance when we keep the top 15 biclusters. When we extend the range parameter from 2.2 to 2.5, the metric value increases due to a larger 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 smaller than the increase in the variances; see Fig. 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 cannot compensate for the decrease of uniformity of values (or increase in standard deviation). 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 more stable.

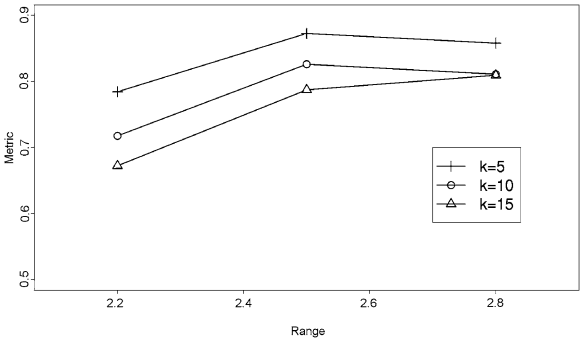


Fig. 5. Effect of parameter changes.

4.2. Mouse embryonic stem cell dataset

The authors Ouyang et al. [20] have reported their CHIP-Sequence scores on mouse embryonic stem cell in [8]. 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 18 936 genes included in this dataset have been known to show correlations in some other studies. Using our algorithm on their original data with-

Algorithm	CLEAN score	Variance of biclusters	Size (rows × cols)	Average column STD
Lattice (our algorithm)	80.95	0.21	97×2	0.43
CC	33.65	1.27	847×12	0.89
OPSM	50.15	0.49	1726×6	0.53
Co-clustering	40.44	0.99	469×2	0.99

Fig. 6. Mouse embryonic stem cell.

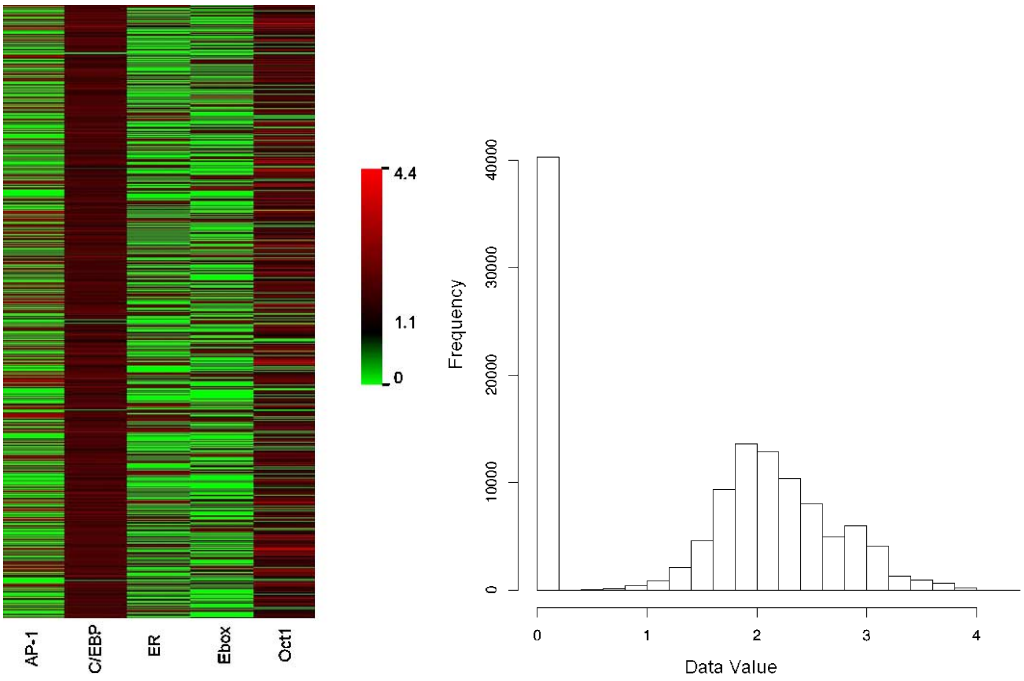
out any normalization, we seek to discover underlying co-factor mechanisms. One bicluster unveiled co-regulated TFs (Nanog and Oct4) with a variance of 0.21 and that is well corroborated by [8]. In order to demonstrate the functional coherence of the genes co-regulated in the bicluster, we use the CLEAN [13] metric to check the functional enrichment with Gene Ontology terms [2]. 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 Fig. 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 (Fig. 6).

4.3. Human genomic dataset

We consider the dataset from human genome from hg18 [25] and calculate the maximum possible relative probability associated with each gene–motif pair using the Sequence Motif-Matching Scoring model [10]. The data contains 24 190 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 [23]. The other source of data that we use is based on experiments [7]. 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 24 190 genes is shown in Fig. 7(a) and the distribution of values is shown in Fig. 7(b) in the format of a histogram.

We want to see whether the biclusters found by our algorithm in the theoretically obtained data match the ones reported in the experimental results. We use *Fisher's Exact Test* to determine whether our clustering algorithm could really identify the biclusters predicted by the experimental results. The criterion used here is the negative of the $\log_{10}$ 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 Fig. 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 value-uniformity and thus the inference of similar behavior by the row elements, worse (the second, third, and fourth rows). The best p-value of this test is 2.90×10^{-7} (or 6.54 for $-\log_{10}(pvalue)$) which is much smaller than the 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 [3,9,11,15,16] using the metric defined in Eq. (17). Parameters used in all of the algorithms are kept as the default ones. For Cheng et al.'s algorithm, we retained the first bicluster that is generated; for biclustering algorithm ISA [15,16], we could not find any biclusters; for biclustering algorithm OPSM [3], 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 [11] algorithm we keep the same number of biclusters as our algorithm. The biclustering results are summarized using the metric (λ from Eq. (17)) in Fig. 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 include it 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 metric value.



(a) (b)

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)

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)

Fig. 7. Genomic data validation. (a) Heatmap of motif data; (b) motif data distribution; (c) motif bicluster comparison; (d) motif bicluster statistics. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/SPR-2012-0336.)

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.

References

[1] 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*, 2008, pp. 1103–1112.

[2] M. Ashburner, C. Ball, J. Blake, D. Botstein, H.B.J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig et al., Gene ontology: tool for the unification of biology, *Nature Genetics* **25**(1) (2000), 25–29.

[3] 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)*, 2002, pp. 49–57.

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

[5] H. Bian, R. Bhatnagar and B. Young, An efficient constraint-based closed set mining algorithm, in: *Proceedings of the 6th International Conference on Machine Learning*, 2007, pp. 172–177.

[6] K. Bryan, P. Cunningham and N. Bolshakova, Biclustering of expression data using simulated annealing, in: *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, 2005, pp. 383–388.

[7] 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** (2006), 1289–1297.

[8] 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.R. 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** (2008), 1106–1117.

[9] Y. Cheng and G. Church, Biclustering of expression data, in: *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 2000, pp. 93–103.

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

[11] 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.

[12] 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)*, 2003, pp. 89–98.

[13] J.M. Freudenberg, V.K. Joshi, Z. Hu and M. Medvedovic, CLEAN: CLustering Enrichment ANalysis, *BMC Bioinformatics* **10**(234) (2009).

[14] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer-Verlag, Heidelberg, 1999.

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

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

[17] T. Laajala, S. Raghav, S. Tuomela, R. Lahesmaa, T. Aittokallio et al., A practical comparison of methods for detecting transcription factor binding sites in chip-seq experiments, *BMC Genomics* **10** (2009), 618.

[18] 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) (2004), 24–45.

[19] D. Nix, S. Courdy and K. Boucher, Empirical methods for controlling false positives and estimating confidence in ChIP-Seq peaks, *BMC Bioinformatics* **9** (2008), 523.

[20] 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) (2009), 21521–21526.

[21] P.J. Park, Chip-seq: advantages and challenges of a maturing technology, *Nat. Rev. Genet.* **10** (2009), 669–680.

[22] S. Pepke, B. Wold and A. Mortazavi, Computation for ChIP-seq and RNA-seq studies, *Nat. Methods* **6** (2009), S22–S32.

[23] 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).

[24] A. Tanay, R. Sharan and R. Shamir, Discovering statistically significant biclusters in gene expression data, *Bioinformatics* **18** (2002), 136–144.

[25] UCSC, UCSC genome browser website: available at: http://genome.ucsc.edu/.

[26] A. Valouev, D. Johnson, A. Sundquist, C. Medina, E. Anton et al., Genome-wide analysis of transcription factor binding sites based on chip-seq data, *Nat. Methods* **5** (2008), 829–834.

[27] 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*, 2002, pp. 517–528.

[28] S. Yoon, L. Benini and D.M.G., Co-clustering: A versatile tool for data analysis in biomedical informatics, *IEEE Transactions on Information Technology in Biomedicine* **11** (2007), 493–494.

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

[30] Y. Zhang, T. Liu, C. Meyer, J. Eeckhoute, D. Johnson et al., Model-based analysis of CHIP-SEQ (MACS), *Genome Biology* **9** (2008), R137.