2013 IEEE 13th International Conference on Data Mining

Discovering Non-Redundant Overlapping Biclusters on Gene Expression Data

Duy Tin Truong*, Roberto Battiti†, Mauro Brunato‡
* truong@disi.unitn.it
† battiti@disi.unitn.it
‡ brunato@disi.unitn.it

***Abstract*—Given a gene expression data matrix where each cell is the expression level of a gene under a certain condition, *biclustering* is the problem of searching for a *subset* of genes that coregulate and coexpress only under a *subset* of conditions. As some genes can belong to different functional categories, searching for *non-redundant overlapping* biclusters is an important problem in biclustering. However, most recent algorithms can only either produce *disjoint* biclusters or *redundant* biclusters with significant overlap. In other words, these algorithms do not allow users to specify the maximum overlap between the biclusters.**

In this paper, we propose a novel algorithm which can generate K overlapping biclusters where the maximum overlap between them is below a predefined threshold. Unlike the other approaches which often generate all biclusters at once, our algorithm produces the biclusters sequentially, where each newly generated bicluster is guaranteed to be different from the previous ones but can still overlap with them. The experiments on real datasets confirm that different meaningful overlapping biclusters are successfully discovered. Besides, under the same constraints, our algorithm returns much larger and higher-quality biclusters compared to those of the other state-of-the art algorithms.

Keywords*—*non-redundant overlapping biclustering; gene expression data

I. Introduction

Gene expression is the process by which information from a gene is used in the synthesis of proteins. Microarray experiments provide us with the expression level of a large number of genes under different experimental conditions [2]. The conditions can be different time points, different types of tissues, or different individuals, etc. The gene expression results are presented as a matrix where each gene is a row and each condition is a column. Each cell is the expression level of a gene under a certain condition. The expression level measures the relative abundance of a gene, usually as the logarithmic ratio between the intensities of the dyes used in the experimental process.

Given the gene expression data, one would like to find a *subset* of genes that coregulate and coexpress (think “behave in a coherent manner”) only for a *subset* of conditions. This problem is called *biclustering* by Cheng and Church [5]. The objective is to find sub-matrices, i.e., maximal subgroups of genes and subgroups of conditions where the genes exhibit highly correlated activities over a range of conditions, and therefore often related to an underlying *gene regulatory network* of biological interest. Biclustering is also known as co-clustering [6], or subspace clustering [9], and used in other areas like text mining and collaborative recommendations, although with different underlying models. In the traditional clustering problem, only rows or columns of a data matrix are partitioned based on some geometric similarity measures like the cosine similarity, or the Euclidean distance. Meanwhile, in the biclustering problem, both rows and columns are clustered simultaneously and the identification of a relevant subset of genes and a subset of experimental conditions is a prerequisite to obtain a clustering of biological interest. Therefore, traditional clustering algorithms cannot be applied directly for biclustering.

Several algorithms have been proposed for biclustering [11], [5], [10]. The algorithms can return a single large and coherent bicluster or a set of biclusters. The coherence must be related to underlying gene regulatory networks of biological interest. An additive model of the gene expression is often used: the expression of a gene in a network is proportional to the sum of a term associated to the specific gene and a term associated to the specific experimental condition. The squared error w.r.t. this linear model, averaged over the entire bicluster expression levels, called *mean squared residue*, measures the lack of coherence of the network [5].

As some genes can belong to different functional categories, the biclusters extracted from a gene expression matrix can have overlap but the overlap should be below a predefined threshold. Otherwise, we obtain redundant or very similar biclusters. Many algorithms are proposed to find a set of biclusters. However, they can only either produce disjoint [5] or redundant biclusters [16], [10] where the overlap between them is arbitrary. This problem is similar to the alternative clustering problem where the goal is to generate high-quality clusterings but different from a given set of trivial clusterings [7], [13]. However, the alternative clustering algorithms can not be used directly for non-redundant biclustering because the redundancy is measured between clusterings, not biclusters. Günnemann et al. has also proposed an algorithm to search for non-redundant overlapping biclusters in general high dimensional data [8] However, their model is not suitable for analyzing gene expression data as the dissimilarity between the objects in a subspace is measured as the Euclidean distance (in that subspace) between them.

In this paper, we propose a novel algorithm, called **Core-Node**, which can return K biclusters such that the overlap between them is less than a threshold. While other algorithms often search for K biclusters at once [16], [10], our algorithm produces one bicluster in each of K iterations where each newly generated bicluster is different from the previous ones and still overlaps with them. In various experiments on real datasets, our algorithm can discover different meaningful overlapping biclusters (through gene ontology analysis). In addition, under the same constraints, our algorithm returns much larger and higher-quality biclusters.

The rest of this paper is organized as follows. In Section II, we summarize the related work. Then, we describe formally the non-redundant overlapping biclustering problem in Section III and our algorithm in Section IV. Finally, we present the experimental results in Section V.

II. Related Work

The algorithms proposed for solving the biclustering problem can be classified into different groups [11]. However, not all algorithms optimize the same "coherence" criterion, therefore we only compare our algorithm with those based on the *additive model*. One of the first additive models is proposed by Cheng and Church [5]. They define a bicluster (I, J) (with I and J are the row and column sets) as a δ-bicluster if and only if its mean squared residue $MSR(I, J)$ below a threshold δ with:

$$MSR(I,J) = \frac{1}{|I||J|} \sum_{i \in I,\ j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2$$

where a_{iJ}, a_{Ij} and a_{IJ} are the *ith* row mean, the *jth* column mean and the bicluster mean. Similarly, the residue of a row or a column are computed as the average residue of all cells in that row or column. The authors also propose an algorithm to search for the largest δ-bicluster. The algorithm starts from the initial bicluster containing all rows and columns and iteratively deletes a set of nodes (rows or columns) with large residues until the mean squared residue of that bicluster is below the threshold. A set of outside nodes with residues below the bicluster residue is then added to the bicluster to increase its volume. The process is repeated until no nodes can be added. This algorithm is deterministic and very fast, as a set of nodes can be deleted or added at the same time. Its complexity is $O((M + N)MN)$ where M and N are the number of genes and conditions. However, modifying a set of nodes simultaneously can also leads to non-optimal solutions. Besides, in each run, the algorithm can only generate one largest bicluster. In order to generate a set of biclusters, the cell values in the previously discovered biclusters are replaced by random values and the algorithm is rerun on the new data. This technique often results in disjoint biclusters because the random values have removed the coherence of rows and columns in replaced biclusters. In this paper, we will denote the Cheng and Church's algorithm as **ChengChurch**.

To overcome this limitation, Yang et al.[16] propose a probabilistic algorithm named **FLOC** (Flexible Overlapped Clusters) which can discover a set of K biclusters in a run. The algorithm starts from a set of random initial biclusters. Each initial bicluster is formed by selecting randomly a subset of rows and a subset of columns from the dataset, such that the bicluster residue is below a predefined threshold. It next iteratively performs the best action for each row and column to improve the bicluster quality. The actions are deleting or adding a row or a column to one of K biclusters. The best action is the one that gives the highest improvement in a gain function which is the sum of the reduction ratio in mean squared residue and the increase ratio in volume. As two objectives (volume and residue) are considered at the same time, **FLOC** can return biclusters with very small volume while their residues are much lower than the threshold. Besides, **FLOC** is very sensitive to the initial biclusters and its complexity is $O((M + N)^2 \times K \times p)$ where M, and N are the number of genes and conditions, K is the number of biclusters, and p is the number of iterations the algorithm runs until convergence.

BICRELS is another algorithm which aims at discovering the largest δ-bicluster proposed by Truong et al. [14]. **BICRELS** is a repeated local search algorithm for biclustering. In each iteration, the algorithm first reduces the residue of a bicluster (I, J) by replacing rows or columns with largest residues in (I, J) by rows or columns with smallest residues outside (I, J). Then, it removes some other largest rows or columns to guarantee the residue below the threshold. Finally, it keeps adding rows or columns with smallest residues to increase the bicluster volume until the residue exceeds the threshold. To escape from the local minima, the algorithm restarts for a number of times from random solutions in a set of potential solutions. The algorithm complexity is $O((M + N)MN)$ where M and N are the number of rows and columns. Although **BICRELS** was shown to produce larger biclusters compared to those of other state-of-the-art algorithms, it can only generate one largest bicluster in each run.

Lazzeroni and Owen introduce the plaid model to analyze the underlying structure of a gene expression matrix [10]. The plaid model allows the overlap between the biclusters. In this model, the data matrix is described as the sum of different layers where each layer corresponds to a bicluster. In detail, a gene expression level a_{ij} of the *ith* gene and the *jth* condition is modelled as:

$$a_{ij} = \sum_{k=0}^{K} (\mu_k + \alpha_{ik} + \beta_{jk}) \rho_{ik} \kappa_{jk} \tag{1}$$

where μ_k, α_{ik} and β_{jk} are the mean, the *ith* gene and the *jth* condition effects in the *kth* layer. ρ_{ik} and κ_{jk} are binary variables describing the membership of the *ith* gene and the *jth* condition. The layer zero ($k = 0$) is used to describe the background. This background layer is fitted before adding the next layers one by one until the predefined number of biclusters is reached or no more significant biclusters are found. While fitting the model parameters, the authors relax the membership variables ρ_{ik} and κ_{jk} as real variables shifted towards binary solutions during the optimizing process. Turner et al. find that "the gradual shifting of estimates introduces a discontinuity which may prevent the algorithm converging to a superior solution to the final result" [15]. Therefore, they propose an optimization method with the use of binary bicluster membership to preserve continuity and speed up the convergence. We will refer this improved version as **ImpPlaid** in this paper.

III. Non-Redundant Overlapping BiClustering

In this section, we recall the problem of traditional biclustering and then couple it with the overlap constraint, thereby presenting the problem of non-redundant overlapping biclustering.

A. The Biclustering Problem

We follow the same notation used in [5], [14]. The biological motivation for the model is that, in a gene regulatory network, the gene expression level is proportional to a sum of a term characterizing the gene plus a term characterizing the experimental condition which is activating the specific network. Let's note that, if logarithms of the original measures are taken, the model is multiplicative in the original measures. Fig.1 shows an example of a bicluster with 9 genes and 6 conditions.

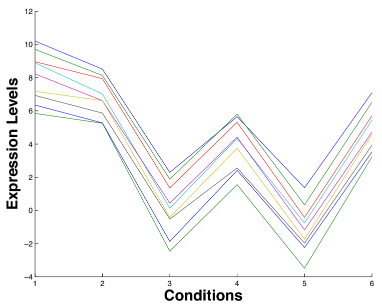


Fig. 1: An example of a bicluster with 9 genes and 6 conditions.

Let X be the set of genes and Y be the set of conditions; a_{ij} be the element of the expression matrix A representing the expression level of the gene i under condition j; $I \subset X$ and $J \subset Y$ be subsets of genes and conditions. The pair (I, J) specifies a submatrix A_{IJ} with the following *mean squared residue* score:

$$MSR(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \quad (2)$$

where:

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}; \qquad a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}$$

$$a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I} \sum_{j \in J} a_{ij}$$

Similarly, the mean squared residue of rows and columns are computed as follows:

$$rowMSR_{I,J}(i) = \frac{1}{|J|} \sum_{j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \quad (3)$$

$$colMSR_{I,J}(j) = \frac{1}{|I|} \sum_{i \in I} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2 \quad (4)$$

The traditional biclustering problem is defined as follows [5].

Definition 1: The biclustering problem is the problem of searching for the largest bicluster (I, J) in a data matrix $A_{X,Y}$ such that its mean squared residue is below a threshold:

$$(I', J') = \underset{I \subset X, J \subset Y}{\arg\max} |I||J| \quad (5)$$

subject to:

$$MSR(I,J) \le \delta$$

As the biclusters containing only one row (with arbitrary number of columns), or only one columns (with arbitrary number of rows) having the residue of zero, the biclustering problem in Equation (6) always has solutions (when $\delta \ge 0$).

B. The Non-Redundant Overlapping Biclustering Problem

Definition 2: The overlap of a bicluster (I_1, J_1) with a bicluster (I_2, J_2) is defined as the ratio between the common cells and the volume of (I_1, J_1):

$$Overlap((I_1,J_1),(I_2,J_2)) = \frac{|I_1 \cap I_2||J_1 \cap J_2|}{|I_1||J_1|} \quad (6)$$

Definition 3: The overlap of a bicluster (I_1, J_1) with a set $\boldsymbol{B}$ of biclusters is computed as the maximal overlap between (I_1, J_1) with other biclusters in $\boldsymbol{B}$:

$$Overlap((I_1,J_1),\boldsymbol{B}) = \max_{(I_2,J_2) \in \boldsymbol{B}} Overlap((I_1,J_1),(I_2,J_2)) \quad (7)$$

Definition 4: Given a set $\boldsymbol{B}$ of biclusters, non-redundant overlapping biclustering is the problem of searching for the largest bicluster (I, J) in a data matrix $A_{X,Y}$ such that its mean squared residue is less than a threshold δ and the overlap between that bicluster and all biclusters in $\boldsymbol{B}$ is less than γ:

$$(I', J') = \underset{I \subset X, J \subset Y}{\arg\max} |I||J| \quad (8)$$

subject to:

$$MSR(I,J) \le \delta$$
$$Overlap((I,J),\boldsymbol{B}) \le \gamma$$

Unless all biclusters in $\boldsymbol{B}$ covers all cells in the data matrix, for $\delta \ge 0$ and $0 \le \gamma < 1$, the problem in Equation (9) always have solutions as the biclusters with single row or single column have residues of zero.

To generate K non-redundant biclusters, we run an algorithm to solve the problem in Definition 4 for K times. The algorithm starts from an empty set $\boldsymbol{B}$ of biclusters. At each run, it adds the discovered bicluster to $\boldsymbol{B}$ so that in the next run, it can produce a different bicluster which can overlap with the previous biclusters in $\boldsymbol{B}$. Since a clustering problem is NP-hard in general, we propose in the sequel a heuristic algorithm which can find reasonably good solutions in polynomial time.

IV. A Core-Node Biclustering Algorithm

We develop our algorithm based on the notion of core rows or columns in a bicluster. For the sake of convenience, we shall refer to a row or a column as a node hereafter. The largest (or smallest) node is the one with the highest (or lowest) residue.

A. The core nodes

In a δ-bicluster (I, J), the nodes can be divided into two groups which are core nodes and their complement: loose nodes. In detail, a row $i \in I$ is a core row if and only if:

$$rowMSR_{I,J}(i) \le \beta_r$$

where:

$$0 \le \beta_r < \max_{i \in I}(rowMSR_{I,J}(i))$$

Similarly, a column $j \in I$ is a core column if and only if:

$$colMSR_{I,J}(j) \leq \beta_c$$
where:
$$0 \leq \beta_c < \max_{j \in J}(colMSR_{I,J}(j))$$

For a pair of (β_r, β_c), we can identify the boundary between the core and loose nodes. Fig.2 shows an example of core nodes for a specific pair of (β_r, β_c). The blue genes and the conditions from 1 to 6 are core nodes because these genes show more similar patterns in these conditions. The remaining genes and conditions are loose nodes.

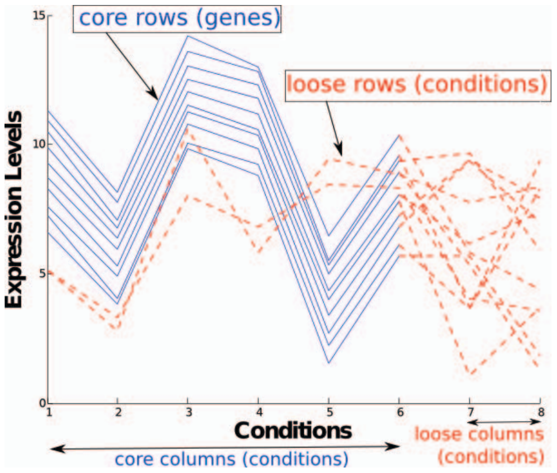
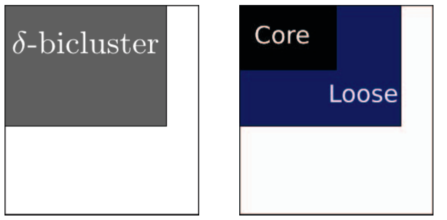


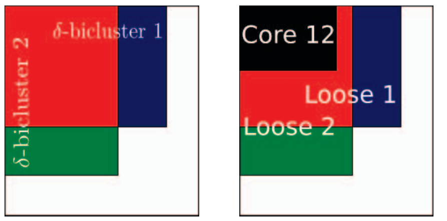
Fig. 2: A core node example of a δ-bicluster.

As the rows or columns in a bicluster can be non-sequential, for the sake of presentation, we often rearrange them in a sequential order as Fig.3a. When the rows and columns of a δ-bicluster are sorted in the ascending order of their residues, the core and loose nodes would be traversed from left to right and top to bottom as Fig.3b. Because of the tight relation between core nodes, i.e., some genes can only show similar behaviors through some conditions, the core-node set of a bicluster is also often included in other locally optimal biclusters which are in the neighborhood of that bicluster. Fig.3c and 3d illustrate this relation (assuming that we can rearrange the rows and columns of two biclusters). The first bicluster consists of the red and blue part. The second bicluster is the combination of the red and green part. These biclusters share a core set denoted as the black part. Therefore, when we expand a bicluster around its core set (in the column or row direction), we can quickly identify a family of locally optimal biclusters. This is because we do not have to build or search for such biclusters from scratch. From the local search view, an optimal solution is often surrounded by other near locally optimal solutions and shares some common structures with them [3]. In order to discover such optimal solutions from a local minimum, an effective algorithm should modify the current solution so as to move to the next local optima rather than restarting over. Besides, different core sets can exist in different regions of the data matrix. Discovering these core sets and searching for the biclusters surrounding them help extract the representative biclusters of the data matrix.



(a) The arrangement of all nodes in a δ-bicluster in a sequential order.

(b) The arrangement of all nodes in a δ-bicluster according to their residues.



(c) The arrangement of all nodes of two neighborhood δ-biclusters in a sequential order.

(d) The arrangement of all nodes of two δ-biclusters according to their residues.

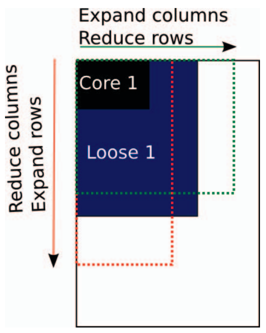
Fig. 3: The arrangement of all nodes in δ-biclusters.

B. The Core-Node Biclustering Algorithm

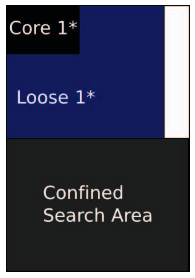
Our core-node search algorithm has three main steps which can be illustrated as in Fig.4. At the beginning, a local search algorithm is used to identify the first locally optimal bicluster denoted by $Core\ 1$ and $Loose\ 1$ in Fig.4a. From there, it searches for other locally optimal solutions around the core set by expanding columns (and reducing rows) or reducing columns (and expanding rows). After identifying the optimal bicluster (denoted by $Core\ 1^*$ and $Loose\ 1^*$ in Fig.4b) around the first core set, the algorithm restricts the search area in the largest submatrix (the grey area in this case) having no intersection with the first optimal bicluster. The whole process is repeated in order to identify the next optimal bicluster as in Fig.4c and Fig.4d. Note that the second optimal bicluster (consisting of $Loose\ 1^*, 2^*$, $Core\ 2^*$, $Loose\ 2^*$ in Fig.4d) *overlaps* with the first optimal bicluster (in the $Loose\ 1^*, 2^*$ part) because of the expansion steps.

Intuitively, restricting the search area guarantees that the sub-matrix formed by the new core set is disjoint from those formed by previous core sets. This new sub-matrix thus reveals a non-redundant view of the data matrix. The algorithm subsequently expands around this core set to search for overlapping nodes. In addition, the core node thresholds (β_r, β_c) are identified through searching in some potential intervals. The pseudo-code of our solution is shown in Algorithm 1. In each iteration, the algorithm searches for the largest bicluster satisfied the residue and overlap constraints.

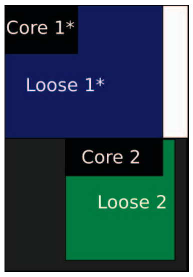
Given a set $\boldsymbol{B}$ of previous optimal biclusters, the algorithm finds the largest submatrix having no common cells with any bicluster in $\boldsymbol{B}$ by calling procedure $restrictRegion$ of which pseudo-code is presented in Algorithm 2. Starting from the full matrix, this procedure identifies the bicluster in $\boldsymbol{B}$ having the largest overlap. If the ratio of common rows is greater



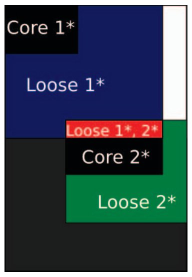
(a) Identify the first core node set and search around this core set



(b) Confine search area to the largest disjoint submatrix



(c) Identify a new core node set



(d) Search around this core set

Fig. 4: Main steps of the core-node based search algorithm.

than the ratio of common columns, the common columns are removed from the restricted matrix. Otherwise, the common rows are removed. The whole process is repeated until there is no overlap. Through incremental computation, the complexity of this procedure is $O(M+N)$ where M and N are the number of rows and columns, respectively.

To find a locally optimal bicluster in the restricted area, any biclustering algorithm returning the largest bicluster with residue below a threshold can be used. In this paper, we use **BICRELS** because it is reasonably simple and effective in terms of implementation and performance.

The characteristic of the bicluster (I, J) returned by **BICRELS** can fall into two cases. First, only a subset of columns in the current biclusters are core nodes. The number of core columns is then identified by each core column threshold β_c. However, setting precisely β_c is difficult, therefore we assume that the number of core columns varies from $(1-\alpha)|J|$ to $|J|$. The second situation is where all columns of the current bicluster are core nodes and included in a larger core set. In this case, we assume that the number of core columns can be from $|J|$ to $(1+\alpha)|J|$. In other words, the algorithm tries reducing and expanding the column size from $(1-\alpha)|J|$ to $(1+\alpha)|J|$ where α is a parameter controlling the reduction and expansion size. A large value of α means that the algorithm can search for more potential solutions and consume more run-time. To guarantee the overlap limit, α must satisfy $\alpha \leq \frac{\gamma}{1-\gamma}$ (where γ is the overlap threshold). To prove this, we observe that the core bicluster has no common cells with other biclusters, and the overlapping cells can only come from adding $\alpha|J|$ columns to the current bicluster. When the column size is $(1+\alpha)|J|$, the maximum overlap between the current bicluster and the other biclusters in $\boldsymbol{B}$ is:

$$\frac{\alpha|J||I|}{(1+\alpha)|J||I|} = \frac{\alpha}{1+\alpha} \leq \gamma$$
$$\Leftrightarrow \alpha \leq \frac{\gamma}{1-\gamma}$$

In order to reduce (or increase) the columns size, our algorithm removes (or adds) the columns with largest MSR from (or to) the core bicluster. It then calls procedure $replaceNodes2$ of which pseudo-code is shown in Algorithm 3 to reduce the bicluster residue. This procedure is an extension of procedure $replaceNodes$ in **BICRELS** for controlling the overlap limit. Specifically, the $replaceNodes2$ procedure replaces nodes which are not in $\sqrt{\tau}|I|$ rows and $\sqrt{\tau}|J|$ columns with smallest MSR where $\tau = (1-\gamma)\frac{n}{|J|} \leq 1$ and n is the number of columns in the new bicluster. As the submatrix formed by these rows and columns is included in the core bicluster, therefore it has no overlap with other biclusters. This guarantees that the overlap is always below the overlap threshold γ. This is because when the column size is n and we keep $\sqrt{\tau}|I|$ and $\sqrt{\tau}|J|$ smallest rows and columns, the maximum overlap is:

$$1 - \frac{\sqrt{\tau}|I|\sqrt{\tau}|J|}{|I|n} = 1 - \frac{\tau|J|}{n} \leq \gamma$$
$$\Leftrightarrow \tau \geq (1-\gamma)\frac{n}{|J|}$$

After performing the $replaceNodes2$ procedure, if the bicluster residue is smaller than the threshold δ, our algorithm traverses the rows in the ascending order of MSR and adds the ones such that adding them to the bicluster does not violate the overlap limit. The process is repeated until the bicluster residue exceeds the threshold or no such rows exist. The last added row is removed to guarantee the residue constraint if necessary. Similarly, if the bicluster residue is larger than the threshold δ, the algorithm keeps deleting largest rows from the bicluster so as not to violate the overlap constraint and until its residue is below the threshold or no such rows exist. Finally, if the current bicluster is valid and better, the best bicluster will be updated and added to the bicluster set $\boldsymbol{B}$ before moving to the next iteration.

The complexity of **BICRELS** procedure is $O((M+N)MN)$ where M and N are the number of rows and columns. Each step of removing or adding a column from line 11 to 14 requires the computation of all column MSRs with the complexity of $O(MN)$. Thus, the complexity of this step is $O(MN^2)$. As only one column is modified each time, the incremental update can be used to reduce the complexity from $O(MN^2)$ to $O(N^2)$. The complexity of the $replaceNodes2$ procedure is $O((M+N)MN)$ because the maximum number of rows and columns can be replaced is $M+N$. For each replacement step, we need to compute all row or column residues with the complexity of $O(MN)$. Besides, from line 20 to 23, searching for the smallest or largest rows requires the computation of all row MSRs with the complexity of $O(MN)$. In the worst case, all rows can be added therefore complexity of this part is $O(M^2N)$. However, in each step, only one row

Algorithm 1: CoreNode

```
1  $B = \emptyset$
2  for $k = 1$ to $numberOfBiclusters$ do
3      // Restrict search region
4      $(\hat{I}, \hat{J}) = restrictRegion(A, B)$
5      // Identify the core bicluster
6      $(I, J) = $ BICRELS$(A_{\hat{I}\hat{J}}, \delta)$
7      $(I^*, J^*) = (I, J)$
8      // Search for variants of the core bicluster
9      for $n = (1-\alpha)|J|$ to $(1+\alpha)|J|$ do
10         $(I_1, J_1) = (I, J)$
11         if $n < |J|$ then
12             Remove $|J| - n$ smallest columns from $(I_1, J_1)$
13         else
14             Add $n - |J|$ smallest columns to $(I_1, J_1)$
15         $\tau = (1-\gamma)\frac{n}{|J|}$
16         $\overline{I} = \sqrt{\tau}|I|$ smallest rows in $I$
17         $\overline{J} = \sqrt{\tau}|J|$ smallest columns in $J$
18         $(I_1, J_1) = replaceNodes2(I_1, J_1, A, \overline{I}, \overline{J})$
19         if $MSR(I_1, J_1) < \delta$ then
20             Add smallest rows to $(I_1, J_1)$ s.t. adding them does not violate the overlap limit until $MSR(I_1, J_1) \geq \delta$ or no such rows exist.
21             Remove the last added row from $(I_1, J_1)$ if $MSR(I_1, J_1) > \delta$ .
22         else
23             Delete largest rows from $(I_1, J_1)$ s.t. deleting them does not violate the overlap limit until $MSR(I_1, J_1) \leq \delta$ or no such rows exist.
24         if $|I^*||J^*| > |I_1||J_1|$ and $MSR(I_1, J_1) \leq \delta$ then
25             $(I^*, J^*) = (I_1, J_1)$
26     $B = B \cup \{(I^*, J^*)\}$
27 return $B$
```

is added or removed, thus their MSRs can be updated incrementally to reduce the complexity from $O(M^2N)$ to $O(M^2)$. In total, the complexity of our algorithm is $O((M+N)MN)$.

V. Experiments

In this section, we compare the performance of all algorithms on finding the overlapping biclusters, the largest biclusters and the coverage of K biclusters. To guarantee the repeatability of the experiments, the source code of our algorithm **CoreNode** and the datasets are made available on our website at http://lion.disi.unitn.it/intelligent-optimization/coreNode/.

A. Experimental Setup:

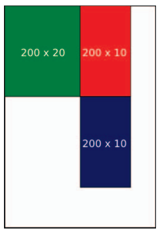
In the following experiments, we set the number of biclusters generated by each algorithm to $K = 10$. The parameter α and γ of **CoreNode** are both set to 0.5. The parameter α of **ChengChurch** is set to its default value ($\alpha = 1.2$). The number of initial rows and columns in the **FLOC** algorithm are set to 9 and 2, respectively as in [14]. The number of biclusters produced by **FLOC** in each run is set to 10. **FLOC** is run for 10 times to eliminate the randomness effect. In each run, **BICRELS** is restarted for 10 times by setting its parameter *numberOfRestarts* to 10. As **BICRELS** can only return the largest bicluster in each run, to generate $K = 10$ biclusters, we also apply the random masking process of **ChengChurch** on **BICRELS**. Because **ImpPlaid** is non-deterministic and the number of biclusters is identified automatically, we run **ImpPlaid** 10 times on each dataset with different random seeds, and select the result from the run which produces at least two biclusters and has the highest coverage. The parameters of **ImpPlaid** are set to the default values in the source code

Algorithm 2: restrictRegion

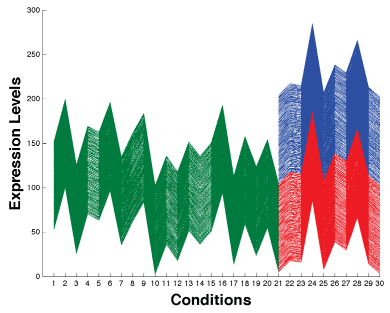
```
   Input: data matrix $A$ with rows $X$ and columns $Y$, bicluster set $B$
1  $(\hat{I}, \hat{J}) = (X, Y)$
2  repeat
3      $(I_2, J_2) = \arg\max_{(I_1,J_1)\in B} |(I_1, J_1) \cap (\hat{I}, \hat{J})|$
4      if $\frac{|I_2 \cap \hat{I}|}{|\hat{I}|} > \frac{|J_2 \cap \hat{J}|}{|\hat{J}|}$ then
5          $\hat{J} = \hat{J} \setminus (J_2 \cap \hat{J})$
6      else
7          $\hat{I} = \hat{I} \setminus (I_2 \cap \hat{I})$
8  until $I_2 = \emptyset$ or $J_2 = \emptyset$;
9  return $(\hat{I}, \hat{J})$
```

Algorithm 3: replaceNodes2

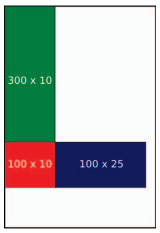
```
   Input: a bicluster $(I, J)$, tabu row and column set $\overline{I}, \overline{J}$
1  $\hat{I} = X \setminus I, \quad \hat{J} = Y \setminus J$
2  repeat
3      // Replace columns
4      repeat
5          $maxJ = \arg\max_{j \in J \setminus \overline{J}} colMSR_{I,J}(j)$
6          $minJ = \arg\min_{j \in \hat{J}} colMSR_{I,J}(j)$
7          $J' = J \cup \{minJ\} \setminus \{maxJ\}$
8          if $MSR(I, J') < MSR(I, J)$ then
9              $J = J'$
10             $\hat{J} = \hat{J} \setminus \{minJ\}$
11     until $J$ is not modified or $\hat{J} = \emptyset$;
12     // Replace rows
13     repeat
14         $maxI = \arg\max_{i \in I \setminus \overline{I}} rowMSR_{I,J}(i)$
15         $minI = \arg\min_{i \in \hat{I}} rowMSR_{I,J}(i)$
16         $I' = I \cup \{minI\} \setminus \{maxI\}$
17         if $MSR(I', J) < MSR(I, J)$ then
18             $I = I'$
19             $\hat{I} = \hat{I} \setminus \{minI\}$
20     until $I$ is not modified or $\hat{I} = \emptyset$;
21 until $I, J$ are not modified;
22 return $(I, J)$
```



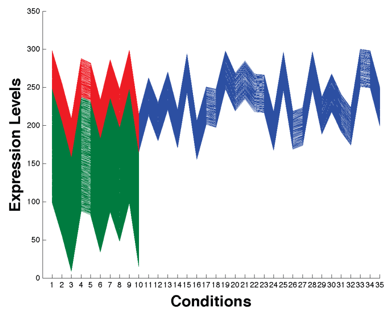
(a) *Dataset 1* for testing row expansion



(b) *Dataset 1*, two biclusters: 6000(200 × 30), 4000(400 × 10)



(c) *Dataset 2* for testing column expansion



(d) *Dataset 2*, two biclusters: 4000(400 × 10), 3500(100 × 35)

Fig. 5: Synthetic datasets with overlapping biclusters. Each bicluster's information is presented in the format $Volume(numberOfRows \times numberOfColumns)$

of the *biclust* R package [1].

In the experiments, we use two real datasets *Yeast* [12] and *Lymphoma* [1]. The *Yeast* dataset consists of 2884 genes and 17 conditions. The *Lymphoma* dataset has 4026 genes and 96 conditions. These datasets are preprocessed by Cheng and Church[2]. The missing values of these datasets are processed as in [5]. The residue threshold of all algorithms on the *Yeast* and *Lymphoma* dataset are set to 300 and 1200 as in previous papers [5], [14]. Besides, we also use two synthetic datasets with dimension of 1200 × 40 to test the ability to discover overlapping biclusters of the algorithms. Each dataset has two main biclusters (after arranging row and column in a suitable order) as in Fig.5. The largest bicluster is made of the green and red part and the second largest bicluster consists of the blue and red part. We also denote the number of rows and columns of each part in the figure. The other values in the data matrix are filled by random values from 0 to 100. The residues of these biclusters are below 1. Therefore, the residue thresholds of all algorithms are set to 1 on these two synthetics datasets.

[1]http://cran.r-project.org/web/packages/biclust/

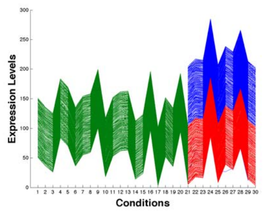
[2]http://arep.med.harvard.edu/biclustering

B. Identifying the overlapping biclusters

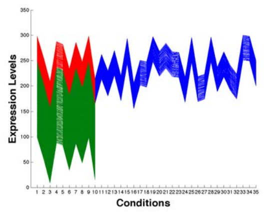
Two first columns in Fig.6 show the biclusters found by four algorithms on two synthetic datasets. We denote the first and second bicluster with the green and blue color and the overlap part with the red color. In other to plot the overlap part, we arrange the columns such that these overlapping columns are at the end of the first bicluster and at the beginning of the second bicluster. We plot the overlap part after plotting two biclusters, therefore the overlap part is always visual if any. For each algorithm, we only plot two biclusters with maximum overlap. When there is no overlap between biclusters, we plot the two largest biclusters of that algorithm. It can be seen that **CoreNode** has discovered different overlapping biclusters through the expansion steps. Besides, **FLOC** can only produce disjoint biclusters with a very small number of columns. Although **ImpPlaid** can also discover reasonably large biclusters on *Dataset 1*, these biclusters have no overlap. This comes from the fact that after subtracting the first layer corresponding to the first bicluster, the values of all cells in the overlap regions are also removed, i.e., in the remaining residue matrix, these values are almost the same as the background. Therefore, when adding the second layer, **ImpPlaid** cannot include the rows and columns in such overlap region. Similarly, on *Dataset 2*, **ImpPlaid** only produces a pair of biclusters with very few overlapping nodes. In contrast, **ChengChurch** produces very poor results in these two datasets with quite small biclusters. We also run **BICRELS** with the random masking process of the **ChengChurch** algorithm and observe that **BICRELS** can only discover the largest bicluster but cannot find the second largest one with the overlap part. For the sake of space, we do not plot the **BICRELS** biclusters.

Two last columns in Fig.6 show the biclusters found by four algorithms on two real datasets *Yeast* and *Lymphoma*. The figures show that **CoreNode** can find much more interesting non-redundant overlapping biclusters compared to those of the other algorithms. **FLOC** can only discover the overlapping biclusters on the *Yeast* dataset. However, the small bicluster is mostly included in the larger bicluster. In other words, the small bicluster is redundant when we already had the larger bicluster. As for **ImpPlaid**, on the *Yeast* dataset, **ImpPlaid** can produce a pair of small biclusters with a few overlapping rows and columns. However, it cannot find the overlapping ones on the *Lymphoma* dataset. **ChengChurch** can return biclusters with small overlap on the *Yeast* dataset and no overlap on the *Lymphoma* dataset. It is also noticeable that although the biclusters of **ChengChurch** share some common columns, **ChengChurch** cannot find at the same time common rows due to certain effects of the random masking process. Similarly, **BICRELS** also produces no overlapping biclusters on these datasets. In summary, on four datasets, **CoreNode** can produce non-redundant overlapping biclusters whereas the other algorithms either return disjoint or redundant biclusters.

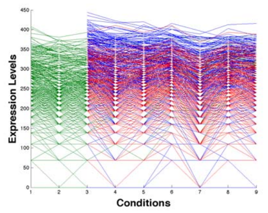
Table I shows the run-time comparison between five algorithms on four datasets. **CoreNode** is slower than **BICRELS**, **ChengChurch**. This comes from the fact that **CoreNode** aims at solving the overlapping biclustering problem which is much more complex the disjoint biclustering problem considered by **ChengChurch** and **BICRELS**. However, comparing with two other algorithms allowing the overlapping, **CoreNode** is much faster than **FLOC** and comparable to **ImpPlaid**.



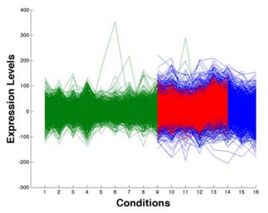
(a) *Dataset 1*, **CoreNode** biclusters: $6000(200\times30)$, $3960(396\times10)$



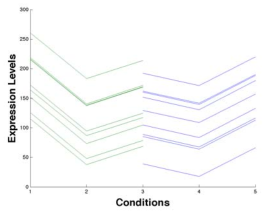
(b) *Dataset 2*, **CoreNode** biclusters: $4000(400\times10)$, $3500(100\times35)$



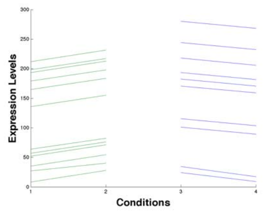
(c) *Yeast*, **CoreNode** biclusters: $4689(521\times9)$, $5880(840\times7)$



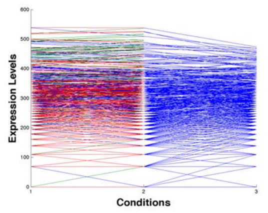
(d) *Lymphoma*, **CoreNode** biclusters: $13664(976\times14)$, $10832(1354\times8)$



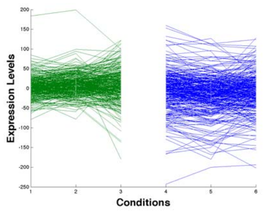
(e) *Dataset 1*, **FLOC** biclusters: $27(9\times3)$, $27(9\times3)$



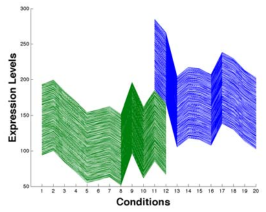
(f) *Dataset 2*, **FLOC** biclusters: $24(12\times2)$, $20(10\times2)$



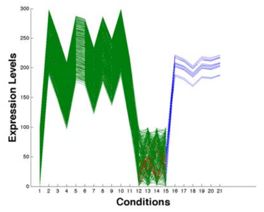
(g) *Yeast*, **FLOC** biclusters: $1500(750\times2)$, $2709(903\times3)$



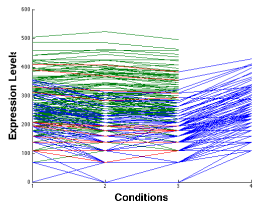
(h) *Lymphoma*, **FLOC** biclusters: $948(316\times3)$, $885(295\times3)$



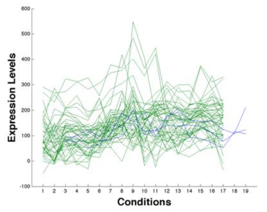
(i) *Dataset 1*, **ImpPlaid** biclusters: $2400(200\times12)$, $2000(200\times10)$



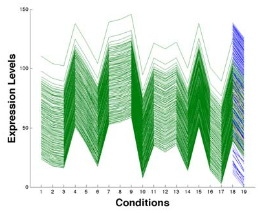
(j) *Dataset 2*, **ImpPlaid** biclusters: $3180(212\times15)$, $120(12\times10)$



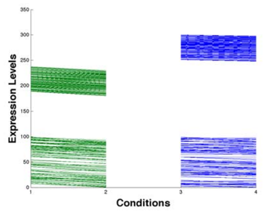
(k) *Yeast*, **ImpPlaid** biclusters: $555(185\times3)$, $512(128\times4)$



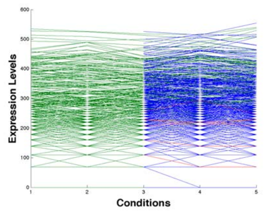
(l) *Lymphoma*, **ImpPlaid** biclusters: $816(48\times17)$, $51(3\times17)$



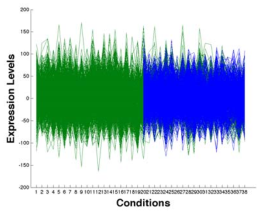
(m) *Dataset 1*, **ChengChurch** biclusters: $2470(130\times19)$, $286(143\times2)$



(n) *Dataset 2*, **ChengChurch** biclusters: $406(203\times2)$, $398(199\times2)$



(o) *Yeast*, **ChengChurch** biclusters: $2540(508\times5)$, $2058(686\times3)$



(p) *Lymphoma*, **ChengChurch** biclusters: $39026(1027\times38)$, $10697(563\times19)$

Fig. 6: Biclusters found by four algorithms on four datasets. Each bicluster's information is presented in the format $Volume(numberOfRows \times numberOfColumns)$.

TABLE I: Run-time Comparison (in seconds) of five algorithms on four datasets.

Dataset / Algorithm	*Dataset 1*	*Dataset 2*	*Yeast*	*Lymphoma*
CoreNode	22.88	25.19	112.88	718.76
FLOC	2833.59	3462.77	16428.79	5891.46
BICRELS	13.01	13.11	33.04	108.34
ChengChurch	7.44	9.31	0.40	1.56
ImpPlaid	36.89	41.96	44.49	62.93

C. Analyzing the overlapping biclusters

To test whether the overlapping genes actually belong to meaningful gene ontology terms, we perform gene ontology analysis on the two first overlapping biclusters discovered by **CoreNode** (in Fig.6c) on the *Yeast* dataset. We use the *GO::Term Finder* software [3] [4] to find significant gene clusters on the gene sets of two biclusters. The statistical significance for functional category enrichment called *p-value* is measured by using a cumulative hypergeometric distribution to compute the chance probability of observing the number of genes from a particular gene ontology category within each cluster. In detail, the chance probability of obtaining at least k genes from a specific functional category within a cluster of size n is:

$$p = 1 - \sum_{i=0}^{k-1} \frac{\binom{f}{i}\binom{g-f}{n-i}}{\binom{g}{n}} \tag{9}$$

where f is the total number of genes within that functional category and g is the total number of genes within the genome [12]. Often, a test with *p-value* ≤ 0.05 is considered as having statistical significance.

We search for pairs of gene clusters with largest overlap where one cluster in the pair belonging to the first bicluster and the other in the second bicluster. We also remove pairs with identical functions or having parent-child relation. Table II shows 10 such pairs of gene clusters. The common genes can be classified in the biological process having the properties of both clusters or in another different process. For example, searching for gene ontology terms of the common genes in the first pair (the *cellular process* and the *metabolic process*), we find a group of $167/169$ overlapping genes belonging to the *cellular metabolic process* with *p-value*$= 3.64 \times 10^{-37}$, i.e., the chance probability of obtaining at least 167 genes from the *cellular metabolic process* within a cluster of size 169 is 3.64×10^{-37}. Similarly, $149/150$ common genes of the second pair (the *single organism process* and the *cellular process*) are clustered in the *single-organism cellular process* with *p-value* $= 2.84 \times 10^{-50}$. Testing on the common genes of the other pairs, we also see that most common genes are grouped into significant gene ontology terms.

D. Comparison on the largest bicluster

In this section, we compare the largest biclusters produced by five algorithms. In some cases, some algorithms like **FLOC** can be stuck in local minima and return biclusters with very small volumes whereas their residues are much lower than the threshold. Therefore, for a clearer comparison, we set different

[3]http://go.princeton.edu/cgi-bin/GOTermFinder

TABLE II: Gene Clusters of two biclusters discovered by **CoreNode** on the *Yeast* dataset. Each line corresponding to a pair of two gene clusters in two biclusters.

Pair	Gene Clusters in Bicluster 1 Function (Size), p-value	Gene Clusters in Bicluster 2 Function (Size), p-value	Overlapping Cluster Function (Size / Total), p-value
1	cellular process (415), p-value $= 4.02 \times 10^{-10}$	metabolic process (536), p-value $= 2.87 \times 10^{-08}$	cellular metabolic process (167/169), p-value $= 3.64 \times 10^{-37}$
2	single organism process (283), p-value $= 4.25 \times 10^{-09}$	cellular process (664), p-value $= 2.45 \times 10^{-15}$	single-organism cellular process (149/150), p-value $= 2.84 \times 10^{-50}$
3	localization (149), p-value $= 1.88 \times 10^{-09}$	single organism cellular process (414), p-value $= 1.29 \times 10^{-07}$	cellular localization (59/75), p-value $= 1.87 \times 10^{-38}$
4	establishment of localization (139), p-value $= 4.14 \times 10^{-09}$	cellular localization (135), p-value $= 2.05 \times 10^{-06}$	establishment of localization in cell (51/55), p-value $= 6.08 \times 10^{-44}$
5	transport (135), p-value $= 7.77 \times 10^{-09}$	macromolecule localization (128), p-value $= 3.05 \times 10^{-06}$	organic substance transport (49/51), p-value $= 1.38 \times 10^{-41}$
6	cellular protein modification process (82), p-value $= 1.56 \times 10^{-07}$	cellular metabolic process (517), p-value $= 4.80 \times 10^{-09}$	cellular protein metabolic process (51/51), p-value $= 2.58 \times 10^{-31}$
7	protein modification process (82), p-value $= 1.56 \times 10^{-07}$	macromolecule metabolic process (417), p-value $= 5.70 \times 10^{-09}$	cellular protein modification process (51/51), p-value $= 4.28 \times 10^{-51}$
8	phosphorus metabolic process (90), p-value $= 2.61 \times 10^{-06}$	organic substance metabolic process (512), p-value $= 7.05 \times 10^{-09}$	organophosphate metabolic process (27/46), p-value $= 4.39 \times 10^{-18}$
9	cell communication (58), p-value $= 2.33 \times 10^{-07}$	biological regulation (242), p-value $= 2.88 \times 10^{-07}$	regulation of cellular process 26/26, p-value $= 1.43 \times 10^{-16}$
10	organic substance transport (97), p-value $= 3.76 \times 10^{-08}$	primary metabolic process (498), p-value $= 1.43 \times 10^{-08}$	macromolecule localization (20/22), p-value $= 3.63 \times 10^{-15}$

residue thresholds in our algorithm to produce biclusters with similar residues as in those of the other algorithms. The overlap constraint is not necessary in comparison as the largest bicluster with residue below a threshold is found with no overlap constraint i.e., when the given bicluster set is empty ($\boldsymbol{B} = \emptyset$). Besides, we cannot set the overlap constraint for the other algorithms either.

Table III shows the performance comparison of all algorithms on the largest biclusters. It can be seen that **CoreNode** produces much larger biclusters with smaller residues compared to those of the other algorithms. It is easy to see that the biclusters of the other algorithms have MSRs much lower than the setting threshold, e.g. on the *Yeast* dataset, with the residue threshold 300, **FLOC** and **ChengChurch** can only find largest solutions with MSR of 183.17 and 237.33, respectively. These algorithms may converge prematurely by grouping, at the early phases, heterogeneous nodes and have no mechanism to resolve this premature convergence problem later. **BICRELS** and **CoreNode** instead can overcome this issue by replacing unsuitable nodes in a bicluster with more suitable nodes outside it during the $replaceNodes$ procedure.

TABLE III: Performance comparison of five algorithms on the largest biclusters.

(a) On the *Yeast* dataset

Algorithm	Max Volume ($\|I\| \times \|J\|$)	MSR
CoreNode ($\delta = 300$)	**16968** (1414 × 12)	299.99
FLOC ($\delta = 300$)	2709 (903 × 3)	183.17
CoreNode ($\delta = 183.10$)	8910 (1485 × 6)	182.97
BICRELS ($\delta = 300$)	16577 (1507 × 11)	299.93
CoreNode ($\delta = 299.90$)	16956 (1413 × 12)	299.83
ChengChurch ($\delta = 300$)	12012 (1001 × 12)	237.33
CoreNode ($\delta = 237.30$)	12490 (1249 × 10)	237.26
ImpPlaid	564 (141 × 4)	270.37
CoreNode ($\delta = 270.30$)	14772 (1231 × 12)	270.15

(b) On the *Lymphoma* dataset

Algorithm	Max Volume ($\|I\| \times \|J\|$)	MSR
CoreNode ($\delta = 1200$)	**45878** (1582 × 29)	1199.65
FLOC ($\delta = 1200$)	948 (316 × 3)	460.31
CoreNode ($\delta = 460.30$)	13796 (3449 × 4)	460.03
BICRELS ($\delta = 1200$)	43907 (1909 × 23)	1199.50
CoreNode ($\delta = 1199.50$)	45663 (1473 × 31)	1199.41
ChengChurch ($\delta = 1200$)	39026 (1027 × 38)	1101.52
CoreNode ($\delta = 1101.50$)	40185 (2115 × 19)	1101.08
ImpPlaid	816 (48 × 17)	2323.56
CoreNode ($\delta = 2323.50$)	113435 (2315 × 49)	2323.02

TABLE IV: The coverage percentage of 10 biclusters of fives algorithms over the full matrix.

(a) On the *Yeast* dataset

Algorithm	Coverage Percentage
CoreNode	75.43%
FLOC	17.06%
BICRELS	61.44%
ChengChurch	56.59%
ImpPlaid	5.63%

(b) On the *Lymphoma* dataset

Algorithm	Coverage Percentage
CoreNode	37.56%
FLOC	0.77%
BICRELS	33.29%
ChengChurch	17.96%
ImpPlaid	0.22%

E. Comparison on the coverage of K biclusters

We compare the coverage percentage of 10 first biclusters generated by the five algorithms in Table IV. The residue thresholds of all algorithms are set to the same values (300 and 1200 on the *Yeast* and *Lymphoma* dataset, respectively). When computing the coverage, we only count the overlap part once, so the overlap constraint does not affect the coverage. The coverage of **FLOC** is computed as the mean coverage of 10 run times. It can be seen that **CoreNode** covers a significantly larger areas compared to those of the other algorithms on two real datasets. For example, on the *Yeast* dataset, **CoreNode** explores more than 60%, 15%, 20% and 70% of the data matrix compared to **FLOC**, **BICRELS** and **ChengChurch**, **ImpPlaid** respectively. In other words, not only can **CoreNode** produce large and overlapping biclusters, but it can also discover the regions that are omitted by the other algorithms.

VI. Conclusion

We proposed an algorithm, called **CoreNode**, for detecting non-redundant overlapping biclusters on gene expression data. In contrast to previous approaches, our algorithm allows users to control the overlap limit between biclusters. Various experimental results confirm that our algorithm can discover different and overlapping biclusters. Besides, under the same constraints, **CoreNode** produces much larger and higher-quality biclusters compared to those of the other state-of-the-art algorithms. In the future, we plan to extend and generalize **CoreNode** to analyze other types of data in different fields like collaborative recommendations, text clustering and so on.

References

[1] A.A. Alizadeh, M.B. Eisen, R.E. Davis, C. Ma, I.S. Lossos, A. Rosenwald, J.C. Boldrick, H. Sabet, T. Tran, X. Yu, et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[2] P. Baldi and G.W. Hatfield. *DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modelling*. Cambridge University Press, 2002.

[3] Roberto Battiti, Mauro Brunato, and Franco Mascia. *Reactive Search and Intelligent Optimization*. Operations research/Computer Science Interfaces. Springer Verlag, 2008. ISBN: 978-0-387-09623-0.

[4] E. I. Boyle, S. Weng, J. Gollub, H. Jin, D. Botstein, J M. Cherry, and G. Sherlock. Go:: Termfinder? open source software for accessing gene ontology information and finding significantly enriched gene ontology terms associated with a list of genes. *Bioinformatics*, 20(18):3710–3715, 2004.

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

[6] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274. ACM, 2001.

[7] D. Gondek and T. Hofmann. Non-redundant clustering with conditional ensembles. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 70–77. ACM, 2005.

[8] S. Günnemann, E. Müller, I. Färber, and T. Seidl. Detection of orthogonal concepts in subspaces of high dimensional data. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1317–1326. ACM, 2009.

[9] H.P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1, 2009.

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

[11] S.C. Madeira and A.L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[12] S. Tavazoie, J.D. Hughes, M.J. Campbell, R.J. Cho, and G.M. Church. Systematic determination of genetic network architecture. *Nature genetics*, 22:281–285, 1999.

[13] D.T. Truong and R. Battiti. A flexible cluster-oriented alternative clustering algorithm for choosing from the Pareto front of solutions. *Machine Learning*, pages 1–35, 2013.

[14] D.T. Truong, R. Battiti, and M. Brunato. A repeated local search algorithm for biclustering of gene expression data. In E. Hancock and M. Pelillo, editors, *Proceedings of the 2nd International Workshop on Similarity-Based Pattern Analysis and Recognition (SIMBAD 2013)*, number 7953 in LNCS, pages 281–296. Springer Verlag, 2013.

[15] H. Turner, T. Bailey, and W. Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational statistics & data analysis*, 48(2):235–254, 2005.

[16] J. Yang, H. Wang, W. Wang, and P. Yu. Enhanced biclustering on expression data. In *Third IEEE Symposium on Bioinformatics and Bioengineering*, pages 321–327. IEEE, 2003.