Book Chapter

UniBic: An Elementary Method Revolutionizing Biclustering

Zhenjia Wang[1†], Guojun Li[1†]*, Robert W Robinson[2] and Xiuzhen Huang[3]*

[1]School of Mathematics, Shandong University, P.R. China
[2]Department of Computer Science, University of Georgia, USA
[3]Department of Computer Science, Arkansas State University, USA

[†]Co-First Authors

***Corresponding Authors:** Guojun Li, School of Mathematics, Shandong University, Jinan, Shandong 250100, P.R. China

Xiuzhen Huang, Department of Computer Science, Arkansas State University, Jonesboro, AR72467, USA

Published **May 06, 2020**

This Book Chapter is a republication of an article published by Xiuzhen Huang, et al. at Scientific Reports in March 2016. (Wang, Z., Li, G., Robinson, R. et al. UniBic: Sequential row-based biclustering algorithm for analysis of gene expression data. Sci Rep 6, 23466 (2016). https://doi.org/10.1038/srep23466)

How to cite this book chapter: Zhenjia Wang, Guojun Li, Robert W Robinson, Xiuzhen Huang. UniBic: An Elementary Method Revolutionizing Biclustering. In: Mark W Ruddock, editor. Prime Archives in Algorithms. Hyderabad, India: Vide Leaf. 2020.

Acknowledgments: This work was partially supported by the grants from NSFC with codes 61432010, 61272016 and 31571354, and also partially supported by the National Institute of Health grants from the National Center for Research Resources (P20RR016460) and the National Institute of General Medical Sciences (P20GM103429), and the support from Arkansas Soybean Promotion Board.

Authors' Contributions: GL conceived and designed the study, ZW implemented the software, performed the analysis and evaluation, and helped GL writing the manuscript. RR and XH revised the manuscript. GL and XH oversaw the project.

Competing Interests: The authors declare no competing financial interests.

Abstract

Biclustering algorithms, which aim to provide an effective and efficient way to analyze gene expression data by finding a group of genes with trend-preserving expression patterns under certain conditions, have been widely developed since Morgan et al. pioneered a work about partitioning a data matrix into submatrices with approximately constant values. However, the identification of general trend-preserving biclusters, which are the most meaningful substructures hidden in gene expression data, remains a highly challenging problem. We found a revolutionary but elementary method by which biologically meaningful trend-preserving biclusters can be readily identified from noisy and complex large data. The basic idea is to apply the longest common subsequence (LCS) framework to selected pairs of rows in an index matrix derived from an input data matrix to locate a seed for each bicluster to be identified. We tested it on artificial and real datasets and compared its performance with currently competitive biclustering tools. We found that the new algorithm, named UniBic, overwhelmingly outperforms all previous biclustering algorithms in terms of commonly used

evaluation scenarios except BicSPAM, which was developed specifically for finding narrow biclusters. The program was written in ANSI C and the source code is available at: 2 http://sourceforge.net/projects/unibic/files/?source=navbar.

Keywords

Gene Expression Data; Microarray Data; Bicluster; Biclustering Algorithm

Introduction

Gene expression microarray data measures expression levels of transcribed mRNA and is arranged in a matrix in which genes correspond to rows and experimental conditions (samples) to columns. Each entry (a real number) represents the expression level of a gene under a specific condition. The need to analyze vast amounts of biological data, including gene expression data, has been driving the development of new data mining (especially biclustering) methods. At first, algorithms such as hierarchical clustering [1] and k-means [2] were investigated to identify sets of functionally related genes or conditions. These traditional clustering methods usually group genes which exhibit similar expression levels across all conditions by maximizing across-cluster variations or minimizing within-cluster variations. But genes may not co-express under all conditions. For instance, a cellular process may only affect a small set of genes under certain conditions, so that a subset of genes may be co-regulated or co-expressed under only a subset of experimental conditions. Biologically, genes which are co-regulated under a subset of experimental conditions exhibit expression patterns which are *trend-preserving*, but which may be quite different in values under those conditions. Here a gene expression pattern refers to the vector of expression values of the gene under the specific conditions. Two gene expression patterns are said to be trend-preserving if and only if their corresponding vectors are either order-preserving or order-reversing. Two vectors x and y are said to be order-preserving if and only if any two corresponding components have the same rank (with respect to the numerical value) in their respective vectors, and order-reversing if and only

if x and $-y$ (or equivalently $-x$ and y) are order-preserving. For the universal purpose, the entries in a row within a trend-preserving bicluster are allowed to be same. Consider the following example.

Example 1: A trend-preserving bicluster composed of three genes under seven conditions. The first and second rows are order-preserving, and the other two possibilities (first and third rows, second and third rows) are both order-reversing.

genes\conditions	c_1	c_2	c_3	c_4	c_5	c_6	c_7
g_1	5	3	3	-2	1	-9	8
g_2	8	6	6	-5	4	-12	11
g_3	-16	-10	-10	7	-5	28	-25

We call a bicluster order-preserving if every pair of rows is order-preserving. Obviously, any trend-preserving bicluster is either order-preserving, or else the disjoint union of two order-preserving biclusters. In example 1 these are $\{g_1, g_2\}$ and $\{g_3\}$.

Finding a maximum subset of genes of trend-preserving expression patterns under a maximum subset of conditions is impossible using traditional clustering methods. Moreover, a single gene may participate in multiple pathways under different subsets of conditions, resulting in one function pattern under one subset of conditions and a different one under another, making the problem even more challenging. Biclustering methods have been proposed with the aim of overcoming these limitations in order to uncover the genetic relationships that are not apparent. Biclustering algorithms have been widely developed since the pioneering work of Morgan et al. [3] on partitioning a matrix into submatrices with approximately constant values. Cheng and Church [4] were the first to apply the biclustering idea to analyze gene expression data. Since then research on biclustering algorithm development in bioinformatics has focused on this application. Existing biclustering algorithms can be grouped into five categories in terms of the techniques on which they are based [5]:

(1) iterative row and column clustering combination: row clusters are combined with column clusters and vice versa, e.g.

Interrelated Two-Way Clustering [6] and Coupled Two-Way Clustering [7];

(2) divide and conquer: the problem is recursively broken down into checkerboard sub-problems, e.g. BiMax [8], Hartigan [9];

(3) greedy iterative search: locally optimal results are chosen in hopes that they might be globally optimal, e.g. Cheng and Church [4], the Flexible Overlapped biclustering algorithm [10], xMOTIFs [11];

(4) exhaustive bicluster enumeration: enumerating all the possible biclusters, e.g. SAMBA [12], OP-Cluster [13];

(5) distribution parameter identification: biclusters are assumed to follow a given statistical model and parameters are identified to fit in the best way, e.g. Spectral biclustering methods [14], Plaid [15] and Sheng et al [16].

Each of these biclustering algorithms is restricted to specific types of biclusters and datasets. In the assessment of twelve biclustering algorithms on twenty artificial datasets from various organism models [17], each algorithm performed well on one or a few datasets, but none performed well on all of them. With the availability of more and more microarray datasets, it has become important to develop a comprehensive biclustering algorithm to analyze gene expression data. In this article we present a revolutionary but elementary method for biclustering. Our method substantially overcomes the limitations of all prior biclustering algorithms, and enables discovery of the most biologically meaningful biclusters in gene expression datasets.

Biologically speaking, trend-preserving biclusters are the most meaningful local structures hidden in a data matrix. Trend-preserving biclusters are a generalization of all widely studied types of biclusters, including constant, shift, scale, and shift-scale biclusters. The latter two types of biclusters were ever considered computationally challenging to identify [18].

Ben-Dor et al [19] developed an algorithm (OPSM) to discover significant order-preserving biclusters based on statistical strategies. In their model, the rows of the input matrix are required to be permutations of some *m* positive integers, *1, 2, ..., m*, as well as to be pairwise different. The technique used in OPSM is essentially an exhaustive approach by iteratively growing each possible submatrix based on statistical evaluations. It has proved unsatisfactory to apply OPSM to the analysis of gene expression datasets [17]. Ever since many methods have been proposed to mine frequent sequential patterns as the extension of the OPSM approach, e.g. OPSM-RM [20] collects results from repeated experiments to cope with noise, GeBOPSM [21] proposes a generalized OPSM model by relaxing the requirement that each row has to be composed of different integers in OPSM, and POPSM [22] captures similar local correlations in probabilistic matrices with uncertain data. However in these models the optimal solutions may not be guaranteed as long patterns with few supports might be pruned in early stage and the requires of computational resource are explosive. Jiang et al [23] proposed a parallel partitioning and mining method based on Butterfly Network to extend and improve OPSM.

The biclustering algorithm QUBIC [24] we previously developed attempts to discover trend-preserving biclusters in gene expression data by granulating gene expression values into $r \geq 1$ ranks. However, it performs worse and worse as the number of ranks of gene expression values increases. Example 2 shows what is wrong with QUBIC for a bad case example.

Example 2: bad case for QUBIC
g_1: 2, 2, 4, 4, 4, 7, 8, 7, 8, 6
g_2: 4, 4, 8, 8, 8, 3, 2, 1, 1, 2

Obviously, expression patterns (2, 2, 4, 4, 4) of g_1 and (4, 4, 8, 8, 8) of g_2 under conditions 1, 2, 3, 4, 5 are order-preserving. After ranking as in QUBIC (for any $r > 0$), these two order preserved patterns could no longer be identified, which leads to an incorrect result. If $r = 1$ then all ranks are 1, so the whole 2×10

array is output as a bicluster, but it is not meaningful. If $r = 2$, the pattern becomes
2 2 2 2 2 1 1 1 1 1
1 1 1 1 1 2 2 2 2 2

which contains only the empty bicluster. Similarly, QUBIC recognizes only the empty bicluster in (g_1, g_2) for any $r > 1$. Our observation, which is very natural, leads to a **UNI**versal approach for discovering trend-preserving **BIC**lusters in gene expression data, which is based on an application of the longest common subsequence (LCS) algorithm [25] to a new matrix derived from the input data matrix. We tested it on both artificial and real datasets, and found that not only does it overwhelmingly outperform all the competitive biclustering algorithms, including OPSM [19], QUBIC [24], ISA [27], FABIA [28], and CPB [29], but also it is capable of accurately discovering the most general and meaningful trend-preserving biclusters hidden in large-scale gene expression datasets only if the hidden biclusters are not too narrow.

Right after implementing our new biclustering algorithm UniBic, we stumbled on another two biclustering algorithms BicSPAM [26] and BicPAM [30] which are both developed by R. Henriques et al in 2014, where BicSPAM was also developed for the discovery of trend-preserving biclusters in a matrix. After a careful study of BicSPAM, we found that the biclusters implanted in its testing datasets are extremely narrow, e.g. of >200 rows and <8 columns, and that BicSPAM is specifically suitable for discovery of very narrow biclusters. Therefore, BicSPAM performs better than UniBic on the datasets with very narrow implanted biclusters as it was mentioned in BicSPAM paper that biclustering algorithms which are designed on the adoption of maximal sequential patterns may to some extent lose narrow biclusters. However, BicSPAM performs worse and worse as the columns of the to-be-identified biclusters increase in number. The comparison results from Supplementary Figure S1 show that UniBic overwhelmingly outperforms all the competitive ones, including BicSPAM, unless the to-be-identified biclusters are too narrow, while the BicSPAM

performs almost same as OPSM due to their similar exhaustive and statistical strategy in nature.

Results

Algorithm Validation

To evaluate the biclustering algorithm UniBic, we compared it with six currently popular biclustering algorithms, including OPSM [19], BicSPAM [26], QUBIC [24], ISA [27], FABIA [28] and CPB [29], on both synthetic and real datasets. Biclustering algorithms developed based on different methods tend to perform differently on various datasets, while some algorithms may perform better on one kind of datasets, and others may tend to be better on other kinds of datasets. In order to fairly evaluate these algorithms, we tested them on six different types of synthetic datasets and eight real datasets from GEO database [31] with the aid of BiBench framework [17].

Validation on synthetic data: As the biclusters to be discovered in synthetic data are supposed to be known, we compared identified biclusters with the genuine ones. Let b_1 and b_2 be two biclusters, their similarity is measured by Jaccard coefficient [32]:

$$s(b_1, b_2) = \frac{|b_1 \bigcap b_2|}{|b_1 \bigcup b_2|}$$

where $|b_1 \cap b_2|$ is the number of genes in their intersection, and $|b_1 \cup b_2|$ is the number of their union. For two sets of biclusters M_1 and M_2, the similarity score between them is calculated using the formula introduced in [8]:

$$S(M_1, M_2) = \frac{1}{|M_1|} \sum_{b_1 \in M_1} \max_{b_2 \in M_2} s(b_1, b_2)$$

which measures the average similarity of biclusters in M_1 with the biclusters in M_2. *Recovery score* is defined as $S(G, D)$, and

relevance score as $S(D, G)$, where G and D represent the sets of genuine biclusters and discovered biclusters, respectively.

Validation on GDS data: Evaluation of results on real data is different from on synthetic data as the genuine biclusters are not known. We validated biclusters by calculating the Gene Ontology enrichment (a statistically significant test which describes the probability of a gene set containing a certain number of particular GO terms) for genes in the discovered biclusters. This functional analysis was carried out using GOstats package [33]. A bicluster here is assumed to be enriched if it has at least one GO term with statistically significant *p*-value smaller than 0.05, where the *p*-value is adjusted by multiple test correction using the method from Benjamini and Hochberg [34] (a way to control the false discovery rate in large datasets to reduce the number of false positives).

Testing on Artificial Datasets

We first tested UniBic and the other six tools on artificial datasets. All the algorithms were executed with their optimized parameters, respectively. For the test on artificial datasets, we need not run the preprocessing steps (see Methods for more details) since the whole artificial data is supposed to be genuine. To skip over the preprocessing steps, UniBic simply ran with parameters $q = 0.5$ and $r = m$(# columns of the input data matrix).

We generated six different types of artificial datasets by replacing the selected biclusters with the generated biclusters, which are type I: trend-preserving biclusters; type II: column-constant biclusters; type III: row-constant biclusters; type IV: shift-scale biclusters; type V: shift biclusters; type VI: scale biclusters. Type I is biologically most meaningful bicluster which is obviously a generalization of the others, and is generated by randomly selecting a row within a selected submatrix as a base row of the entries same as the corresponding entries in the background matrix A, and then rearranging the entries in the other rows of the submatrix such that the rearranged submatrix is trend-preserving. Type II is generated by

randomly selecting a row within a selected submatrix, and copying it to other rows in this submatrix. Type III is generated by randomly selecting a column within a selected submatrix, and copying it to other columns in this submatrix. Type IV is generated by randomly selecting a row within a selected submatrix as a base row, and replacing the other rows of the submatrix by both shifting and scaling of the base row. Type V is generated as in the type IV but with scaling parameter 1. Type VI is generated as in the type IV but with shifting parameter 0.

Comparison on Six Types of Biclusters

To begin with, we generated at random three kinds of test matrices with scaling issues regarding size of test matrices, size/number of implanted biclusters: a) matrix of size 150*100 with three implanted biclusters of size 15*15; b) matrix of size 200*150 with four implanted biclusters of size 20*20; c) matrix of size 300*200 with five implanted biclusters of size 25*25. For each of the six types of biclusters, five datasets were generated for each kind of test matrix through repetition. The average relevance and recovery scores among all test matrices of each tool are shown in Figure 1.

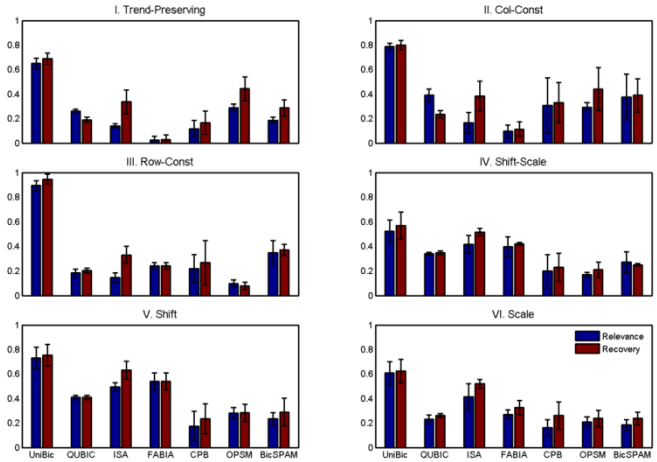

Figure 1: Error bar added, the relevance and recovery scores of the seven algorithms on six types of biclusters.

On type I test matrix with trend-preserving biclusters implanted, UniBic overwhelmingly outperformed all other competitive algorithms with average relevance score 0.65 against the second highest relevance score 0.29 of OPSM, and with recovery score 0.69 against the second highest recovery score 0.44 of OPSM. The results implied that UniBic could discover trend-preserving biclusters in data array more readily than any compared tools. Similar situations occurred respectively on type II and type III test matrices, with more obvious advantages. On the other test matrices, the advantage of UniBic over others was comparatively less significant, but still outperformed all the others. Therefore, comparison results demonstrate that not only does UniBic perform better but also more stable than any compared tools.

OPSM got its best performances on type I and type II test matrices, in which the values are strict order-preserving with relatively large value gaps between bigger and smaller values in each row of implanted biclusters. While on the type III test matrices, where the entries in each row of implanted biclusters are consist with a constant value, its performance became rather poor. BicSPAM performed slightly better than OPSM on most types of test matrices, and as it allows equal values in matrices, it got good performance on type III test matrices. QUBIC performed well on type IV and V test matrices with large q values because it is suitable for datasets with biclusters which can be granulated to be separated from the background data. ISA and FABIA both showed their best performances on type V test matrices, as they were designed to perform well on datasets generated from data distribution with large variances, where the mean values of implanted biclusters are significant differ from the mean values of the test matrices, and their performances on type IV and type VI test matrices were also better than on other test matrices. CPB showed the least stable performance in repetitive experiments compared with other tools as it starts with a randomly selected set of columns, and this may lead to quite unstable performance for each time.

The comparison results shown in Figure 1 demonstrate that the UniBic does overwhelmingly outperform all the compared tools on the datasets with implanted biclusters of nearly square shape.

Our original intention in development of biclustering algorithms is to seek those biclusters of (nearly) square shape just like most computational scientists did. However, the narrow biclusters, with huge number of rows but only a few columns, which are usually more important to biologists, tend to be lost for the algorithms designed based on the adoption of maximal sequential row patterns. To evaluate the capability of finding narrow biclusters, we further tested them on the datasets with narrow biclusters implanted. Comparison results from Supplementary Figure S1 show that the UniBic keeps performing better until the implanted biclusters become too narrow, and that it is almost independent of the number of rows. When the implanted biclusters become too narrow, e.g. with less than 8 columns but with more than hundreds of rows, the algorithm BicSPAM is more capable of returning accurate results as it is specially designed to identify this kind of narrow biclusters.

Comparison on overlapping biclusters

Then we tested the seven tools on artificial datasets with overlapping biclusters. The overlapping biclusters were generated by replacing the selected biclusters with trend-preserving biclusters. Four kinds of artificial matrices were generated with three implanted biclusters overlapped of size 0*0, 3*3, 6*6 and 9*9 respectively, where the matrices are of size 200*150 and biclusters are of size 20*20. Values in each of the three selected biclusters were shifted with 2, 4 and 6 to ensure that they were still trend-preserved while overlapped. Repeating the procedure five times, we obtained five artificial matrices for each overlapping size.

The relevance and recovery scores of seven algorithms on each kind of the test matrices with overlapping biclusters are shown in Figure 3. The results showed that for most algorithms, their performances went down as the degree of overlap increased. OPSM's scores did not change much as its initial scores were low. ISA and FABIA showed robust performances with high scores. Our UniBic found nearly all the implanted biclusters when the biclusters were not overlapped. Although our

performance was affected when the biclusters are overlapped, it still found most of the implanted biclusters, and the result did not change much when the overlapping size increased. This indicates that the UniBic is more capable of finding overlapping biclusters than other compared tools.

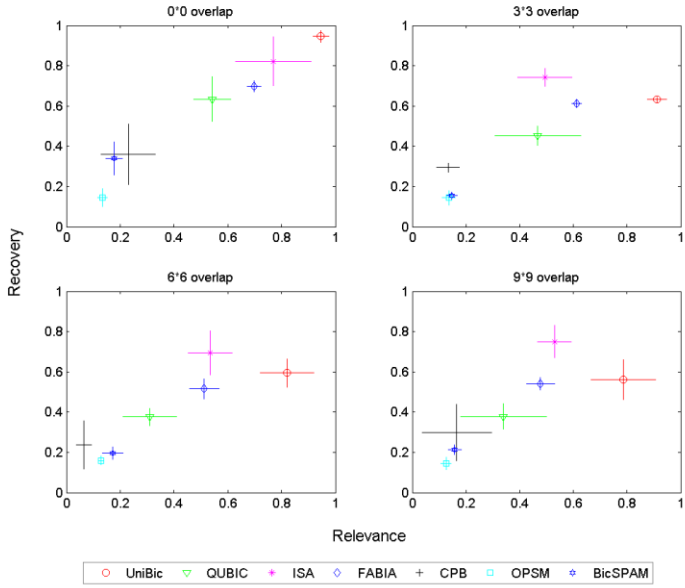

Figure 3: Error bar added, the relevance and recovery scores of the seven algorithms on artificial matrices with overlapping biclusters.

Testing on Real Datasets

We further tested the seven tools on the eight gene expression datasets GDS181, GDS589, GDS1406, GDS1451, GDS1490, GDS2520, GDS3715, GDS3716 from the GEO database [31]. The detailed description of these datasets is summarized in Table 2.

Considering the inactive entries and noise interference, we first preprocessed all the datasets (see Methods for more details). Up- and down-regulated values were separated from the background data with parameter q to be $15/m$ and r to be 15. For other algorithms that are required to be run on array data without

missing values, the PCA imputation [35] was carried on the expression datasets, and all the algorithms were run with their parameters optimized. The GO enrichment was evaluated for each bicluster discovered by each tool, with significant *p*-value 0.05. Since different algorithms are based on different theoretic models, and their best performances with respectively optimized parameters may lead to different number of output biclusters, we assessed their performances by the proportion of GO enriched biclusters. All the statistics are shown in Table 3. UniBic outputted 62 enriched biclusters from 151 discovered ones, and reached the highest average enrichment level of 41.1% of these eight datasets. FABIA showed 22 enriched biclusters from 80 discovered ones, reached the lowest average enrichment level. ISA discovered the most biclusters of 217, but with a comparative lower average enrichment level of 32.7%. OPSM found the second most biclusters, but still with a comparative lower average enrichment level of 29.5%. QUBIC and CPB both had relatively higher average enrichment levels. We ran BicSPAM on the same real datasets, but we did not get final results because it was always out of memory.

Table 2: Description of GDS datasets.

Dataset	Genes	Samples	Description
GDS181	12626	84	Large-scale analysis of the human Transcriptome
GDS589	8799	122	Multiple normal tissue gene expression across strains
GDS1406	12488	87	Brain regions of various inbred strains
GDS1451	8799	94	Toxicants effect on liver: pooled and individual sample comparison
GDS1490	12488	150	Neural tissue profiling
GDS2520	12625	44	Head and neck squamous cell carcinoma
GDS3715	12626	110	Insulin effect on skeletal muscle
GDS3716	22283	42	Breast cancer: histologically normal breast epithelium

Table 3: The results of GO enrichment analysis on eight GDS datasets.

Algorithm	Found	Enriched
UniBic	151	62(41.1%)
OPSM	163	48(29.5%)
QUBIC	91	34(37.4%)
ISA	217	71(32.7%)
FABIA	80	22(27.5%)
CPB	96	34(35.4%)

Utilization of Computing Resource

Biclustering has been well known to be computationally intractable, and therefore it is highly challenging to develop an effective and efficient heuristic algorithm in order to meet the needs of analyzing large data matrices. According to the principle of algorithmic design, Unibic takes $O(nm)$ time to process data separation and granulation, and $O(nqm\log(qm))$ to create the index matrix, and $O(q^2n^2m^2/k)$ to locate seeds of to-be-identified biclusters. The computational complexity of bicluster extension is up-bounded by $O(Sq^2m^2n)$, where S is the number of selected seeds. Thus the overall running time of UniBic is up-bounded by $\max\{O(q^2m^2n^2/k), O(Sq^2m^2n)\}$ from which we see that the running time of UniBic is independent of size of the biclusters to be identified, and even almost independent of columns of input matrix because qm approaches a constant value.

To compare the computing resources usage for different algorithms. We ran the seven tools on the test matrices with fixed number of 50 columns, and calculated the individual running time distributions of the seven tools with their respective default parameters against the number of rows. The algorithms are tested on these large test sets on a desktop computer (2.66 GHz Inter Core, 2 Duo CPU, and 4 GB memory). Figure 4 displays the comparison results among the seven individual running time distributions against the number of rows.

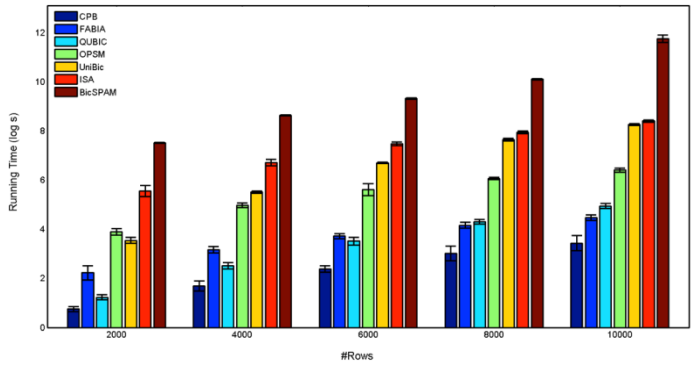

Figure 4: Error bar added, comparison of the distributions of running time of the seven tools against the number of rows on the matrices of 50 columns. The *y*-axis is in $\log_2$ scale.

Discussion

Since the first biclustering strategy was pioneered by Morgan et al. [3] in 1963, researchers have attempted to develop an effective and efficient algorithm capable of discovering trend-preserving biclusters. There have been various biclustering algorithms which have been playing important roles in the analysis of gene expression data, but the identification of general trend-preserving biclusters remains a challenging problem. Our main contribution in this article is the finding of the feasible way to make the intractable problem routinely tractable. Intuitively, also mentioned in [19], the key of discovery of the biclusters in a data matrix should lie in the prediction of a seed for each significant (trend-preserving) bicluster hidden in the data matrix to be analyzed. It has been considered to be rather challenging [19] even in the special case where input matrix consists of n distinct permutations of $(1, 2, ..., m)$. The UniBic captures the essence of how to locate a seed of each to-be-identified bicluster hidden in a background matrix by finding a longest common subsequence between two rows of the index matrix derived from the input matrix. This provides a transformation from the problem of discovering trend-preserving biclusters in a background matrix to a simple problem of finding longest common subsequence between two rows of the index matrix

derived from the background matrix. This transformation seems to be routine, but it does revolutionize the traditional biclustering approaches. Methodologically, UniBic takes an essential step towards the identification of the most general and meaningful biclusters hidden in a noisy and complex data matrix. The results on both synthetic and real data sets demonstrate that UniBic is more promising in discovery of functionally correlated expression patterns in gene expression data, and proves to be a powerful biclustering analysis tool when dealing with large microarray data.

Methods

In this section, we present our innovative biclustering algorithm, which is capable of discovering all the significant trend-preserving biclusters hidden in a data matrix. The basic idea behind the algorithm comes from the following observations: 1) there exists a column permutation of an order-preserving bicluster such that the entries of each permuted row within the bicluster are increasingly (not necessarily strictly) arranged, and 2) the key to biclustering is the accurate prediction of the columns of each to-be-identified bicluster. Motivated by these two observations, we designed a novel algorithm by applying the LCS algorithm to selected pairs of rows of an index matrix derived from the input data matrix.

The foundation of the algorithm is the fact that if two rows of the input matrix A belong to a significant order-preserving bicluster, then the corresponding two rows of the index matrix Y will contain a significant common subsequence with a high probability, and vice versa. This elementary observation leads to a novel method to identify a seed for each potential trend-preserving bicluster. To achieve this goal, we could calculate all the significant common subsequences by applying the LCS algorithm to each pair of rows of Y. Instead, we identify a number k such that every significant order-preserving bicluster B must contain at least k+1 rows. Now assume that B is such a bicluster, if we equally partition the set of rows of A into k subsets of rows, then there must be at least two rows of B falling into one of these k subsets, and the two rows are sufficient to

locate a seed for *B*. Therefore, applying the LCS algorithm to each pair of rows in each of the *k* subsets of *Y* would be sufficient to anchor a seed for each significant order-preserving bicluster of more than *k* rows. This process identifies a seed for each potential bicluster hidden in the data matrix. The algorithm follows the steps below in order:

Algorithm UniBic
Step 1: Index matrix creation:

Let $Y=\{y_{ij}\}$ be the index matrix derived from input matrix $A=\{a_{ij}\}$ by setting:
$y_{ij}=r$ if and only if a_{ir} is the j'th smallest entry in row i (*)

where ties are broken based on the rule that the smaller column index has higher priority to be ranked (see Supplementary Methods for more details).

Step 2: Index matrix partition:

We calculate an integer *k* based on the significance (default set to 0.05) of the to-be-identified trend-preserving biclusters using the techniques developed in [19]. We then equally partition *Y* into *k* subsets of rows (see Supplementary Methods for more details).

Step 3: Application of LCS:

Apply the LCS algorithm to each pair of rows of each of the *k* subsets of *Y* to find all the significant longest common subsequences. For each pair of rows having a significant longest common subsequence, one such subsequence is chosen as a seed to which steps 4, 5, and 6 are to be applied. They are listed in decreasing order in length with the longest one at the front.

Step 4: Strict order-preserving bicluster development:

We start with a longest seed at the front of the seed list obtained from step 3. The LCS algorithm is then repeatedly applied to

find a $3 \times C$ order-preserving submatrix of A, where two of the rows are from the seed and the value of C is as large as possible. We continue to add rows one at a time in a greedy fashion until the order-preserving submatrix has more rows than columns, at which point the submatrix from the previous stage is passed on to step 5.

Step 5: Extension to an approximately trend-preserving bicluster:

From the strict order-preserving bicluster obtained in step 4, we extend it by first repeatedly adding new columns one at a time with an error rate $r \leq 0.3$ until none is available. Up to now, the bicluster obtained is order-preserved. To identify a significant trend-preserving bicluster, we have to get those remaining original rows and their negative ones involved in the row extension process by repeatedly adding new rows (original or negative) one at a time with an error rate ≤ 0.15 until none is available. The row extension would be achieved by applying the LCS algorithm between the common (consensus) sequence of the column extended order-preserving bicluster and the corresponding index row in Y or its reverse row when we consider negative rows to be added (see Supplementary Methods for more details). Then remove from the current seed list those with two corresponding rows belonging to discovered biclusters. Repeat step 4 for the next potential trend-preserving bicluster until the list is exhausted.

Step 6: Output as many trend-preserving biclusters as the user needs:

We calculate the significance value for those trend-preserving biclusters obtained in step 5. Those with p-value less than 0.05 are decreasingly ordered in their significance. Then UniBic outputs first o trend-preserving biclusters, where o is a parameter which can be pre-specified by users with a default set to 100.

Example 3 illustrates how to locate an initial seed of a trend-preserving bicluster in the input matrix A.

Example 3: Illustration of locating an initial seed.

a. Input matrix A:with entries of two rows and eight columns.

row\column	$a_{\cdot 1}$	$a_{\cdot 2}$	$a_{\cdot 3}$	$a_{\cdot 4}$	$a_{\cdot 5}$	$a_{\cdot 6}$	$a_{\cdot 7}$	$a_{\cdot 8}$
i	15	3	10	11	12	10	1	7
j	10	2	6	11	8	6	16	9

b. The index matrix Y of A: with entries being obtained based on (*, refer to Step 1 of Algorithm UniBic).

row\column	$y_{\cdot 1}$	$y_{\cdot 2}$	$y_{\cdot 3}$	$y_{\cdot 4}$	$y_{\cdot 5}$	$y_{\cdot 6}$	$y_{\cdot 7}$	$y_{\cdot 8}$
i	7	**2**	8	**3**	**6**	4	**5**	**1**
j	**2**	**3**	**6**	**5**	8	**1**	4	7

c. Initial seed: obtained by the longest common subsequence (2, 3, 6, 5, 1) through applying the LCS algorithm between rows i and j in Y.

row\column	$a_{\cdot 2}$	$a_{\cdot 3}$	$a_{\cdot 6}$	$a_{\cdot 5}$	$a_{\cdot 1}$
i	3	10	10	12	15
j	2	6	6	8	10

Data Preprocessing

When the algorithm UniBic is applied to real data matrices, especially gene expression data, it is better to preprocess the input data to alleviate the adverse impacts to data entries since corresponding genes are not activated under all conditions and there is noise interference from data approximation.

Data Separation

The values of interest are usually hidden in a massive data matrix to be analyzed. Of interest in gene expression microarray matrix are those entries representing genes up- or down-regulated under corresponding conditions, which are usually only a small portion of the whole data matrix. Biologically, up-(down-) regulated expression values tend to be comparatively bigger (smaller). Those middle values which represent genes

being inactive under corresponding conditions are comparatively less important in the analysis of gene expression data. Therefore, it is helpful to distinguish the values of interest from others in gene expression microarray data matrices. To do so, we chose a percentage parameter q with the default value set to $15/m$ (the value q=0.5 is specially provided for data without separation preprocessing), where m is the number of columns of the input matrix, and we select entries with values significantly away from the median value in each row of input matrix A as up- (down-) regulated values as follows:

1. entries in each row i of A are increasing ordered: a_{i1}, …a_{is}, …a_{il}, …a_{it}, …a_{im}, where $s=qm$, $l=m/2$ and $t=(1-q)m$, $d=\min\{a_{il} - a_{is}, a_{it} - a_{il}\}$.

2. for the values bigger than $a_{il} + d$, they are treated as up-regulated values, and values smaller than $a_{il} - d$ are treated as down-regulated values.

3. set all the other entries in A to be zero, and denote by A' the resultant matrix.

Data Granulation

Data array, e.g. gene expression microarray data, generated from wet laboratory is inevitably approximated, leading the algorithms, including UniBic, to be affected adversely to some extent. To avoid suffering from this approximation, we further preprocessed the input data by equally partitioning all the up-regulated decreasingly ordered entries in each row of A' into r (a parameter which may be pre-specified by user) intervals, then we set all the entries belonging to the i'th interval to be the integer i, while the down-regulated increasingly ordered entries in each row of A' were also separated into r intervals and entries belonging to the i'th interval were set to be the integer $-i$, then we get a new integer matrix denoted by A".

Obviously, the trend-preserving biclusters in A" equivalently correspond to those in A. Therefore, we may apply the UniBic on A" to discover all the significant trend-preserving biclusters

hidden in *A*. This approach has been experimentally proved to be helpful in reducing adverse impacts on performance.

References

1. RR Sokal. A statistical method for evaluating systematic relationships. Univ Kans Sci Bull. 1958; 38: 1409-1438.
2. JA Hartigan, MA Wong. Algorithm AS 136: A K-Means Clustering Algorithm, Journal of the Royal Statistical Society. Series C (Applied Statistics). 1979; 28: 100-108.
3. JN Morgan, JA Sonquist. Problems in the analysis of survey data, and a proposal. Journal of the American statistical association. 1963; 58: 415-434.
4. Y Cheng, GM Church. Biclustering of Expression Data, in Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology. 2000; 93-103.
5. SC Madeira, AL Oliveira. Biclustering algorithms for biological data analysis: a survey, IEEE/ACM transactions on computational biology and bioinformatics. 2004; 1: 24-45.
6. Haixun Wang, Wei Wang, Jiong Yang, Philip S Yu. Clustering by pattern similarity in large data sets. In Proceedings of the 2002 ACM SIGMOD international conference on Management of data. 2002; 394-405.
7. Gaddy Getz, Erel Levine, Eytan Domany. Coupled two-way clustering analysis of gene microarray data. Proceedings of the National Academy of Sciences. 2000; 97: 12079-12084.
8. Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, et al., A systematic comparison and evaluation of biclustering methods for gene expression data. Bioinformatics. 2006; 22: 1122-1129.
9. JA Hartigan. Direct clustering of a data matrix. Journal of the american statistical association. 1972; 67: 123-129.
10. J Yang. δ-clusters: Capturing subspace correlation in a large data set, in Data Engineering, 2002. Proceedings. 18th International Conference on. 2002; 517-528.
11. T Murali, S Kasif. Extracting conserved gene expression motifs from gene expression data. In Pacific Symposium on Biocomputing. 2003; 77-88.
12. Amos Tanay, Roded Sharan, Ron Shamir. Discovering

statistically significant biclusters in gene expression data. Bioinformatics. 2002; 18: S136-S144.
13. Jinze Liu, Jiong Yang, Wei Wang.Biclustering in gene expression data by tendency, in Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE. 2004; 182-193.
14. Kluger Y, Basri R, Chang JT, Gerstein M. Spectral biclustering of microarray data: coclustering genes and conditions. Genome research. 2003; 13: 703-716.
15. L Lazzeroni, A Owen. Plaid models for gene expression data. Statistica sinica. 2002; 12: 61-86.
16. Sheng Q, Moreau Y, De Moor B. Biclustering microarray data by Gibbs sampling. Bioinformatics. 2003; 19: ii196-ii205.
17. Kemal Eren, Mehmet Deveci, Onur Küçüktunç, Ümit V Çatalyürek. A comparative analysis of biclustering algorithms for gene expression data. Briefings in bioinformatics. 2013; 14: 279-292.
18. JS Aguilar-Ruiz. Shifting and scaling patterns from gene expression data. Bioinformatics. 2005; 21: 3840-3845.
19. Amir Ben-Dor, Benny Chor, Richard Karp, Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. Journal of computational biology. 2003; 10: 373-384.
20. Chun Kit Chui , Ben Kao, Kevin Yip, Saudan Lee. Mining order-preserving submatrices from data with repeated measurements, in Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. 2008; 133-142.
21. Qiong Fang, Wilfred Ng, Jianlin Feng, Yuliang Li. Mining bucket order-preserving submatrices in gene expression data, Knowledge and Data Engineering. IEEE Transactions on. 2012; 24: 2218-2231.
22. Qiong Fang, Wilfred Ng, Jianlin Feng, Yuliang Li. Mining order-preserving submatrices from probabilistic matrices. ACM Transactions on Database Systems (TODS. 2014; 39: 6.
23. Tao Jiang, Zhanhuai Li, Qun Chen, Zhong Wang, Wei Pan, et al. Parallel Partitioning and Mining Gene Expression Data with Butterfly Network. In Database and Expert Systems Applications. 2013; 129-144.

24. Guojun Li, Qin Ma, Haibao Tang, Andrew H Paterson, Ying Xu. QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. Nucleic acids research. 2009; 37: e101-e101.
25. Wikipedia contributors. 28 September 2014 06:44 UTC). Longest common subsequence problem. Available Online at: http://en.wikipedia.org/w/index.php?title=Longest_common_subsequence_problem&oldid=627149016
26. R Henriques, SC Madeira, BicSPAM: flexible biclustering using sequential patterns. BMC bioinformatics. 2014; 15: 130.
27. Sven Bergmann, Jan Ihmels, Naama Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. Physical review E. 2003; 67: 031902.
28. Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, et al. FABIA: factor analysis for bicluster acquisition. Bioinformatics. 2010; 26: 1520-1527.
29. Doruk Bozdağ, Jeffrey D Parvin, Umit V Catalyurek. A biclustering method to discover co-regulated genes using diverse gene expression datasets, in Bioinformatics and Computational Biology. Berlin: Springer. 2009; 151-163.
30. R Henriques, SC Madeira. BicPAM: Pattern-based biclustering for biomedical data analysis. Algorithms for Molecular Biology. 2014; 9: 27.
31. Edgar R, Domrachev M, Lash AE. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic acids research. 2002; 30: 207-210.
32. Wikipedia contributors. 19 January 2015 06:29 UTC). Jaccard index. Available Online at: http://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=634979038
33. S Falcon, R Gentleman. Using GOstats to test gene lists for GO term association. Bioinformatics. 2007; 23: 257-258.
34. Y Hochberg, Y Benjamini. More powerful procedures for multiple significance testing. Statistics in medicine. 1990; 9: 811-818.
35. Stacklies W, Redestig H, Scholz M, Walther D, Selbig J. pcaMethods—a bioconductor package providing PCA methods for incomplete data. Bioinformatics. 2007; 23: 1164-1167.