Bioinformatics, 36(20), 2020, 5054–5060
doi: 10.1093/bioinformatics/btaa630
Advance Access Publication Date: 11 July 2020
Original Paper

Gene expression

RecBic: a fast and accurate algorithm recognizing trend-preserving biclusters

Xiangyu Liu[1,2,†], Di Li[1,2], Juntao Liu[2], Zhengchang Su[3,*] and Guojun Li[1,2,†,*]

[1]Research Center for Mathematics and Interdisciplinary Sciences and [2]School of Mathematics, Shandong University, Jinan 250100, China and [3]Department of Bioinformatics and Genomics, The University of North Carolina at Charlotte, Charlotte, NC 28223, USA

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first author and last author should be regarded as Joint Authors.

Associate Editor: Lenore Cowen

Received on July 19, 2019; revised on June 24, 2020; editorial decision on July 5, 2020; accepted on July 6, 2020

Abstract

Motivation: Biclustering has emerged as a powerful approach to identifying functional patterns in complex biological data. However, existing tools are limited by their accuracy and efficiency to recognize various kinds of complex biclusters submerged in ever large datasets. We introduce a novel fast and highly accurate algorithm RecBic to identify various forms of complex biclusters in gene expression datasets.

Results: We designed RecBic to identify various trend-preserving biclusters, particularly, those with narrow shapes, i.e. clusters where the number of genes is larger than the number of conditions/samples. Given a gene expression matrix, RecBic starts with a column seed, and grows it into a full-sized bicluster by simply repetitively comparing real numbers. When tested on simulated datasets in which the elements of implanted trend-preserving biclusters and those of the background matrix have the same distribution, RecBic was able to identify the implanted biclusters in a nearly perfect manner, outperforming all the compared salient tools in terms of accuracy and robustness to noise and overlaps between the clusters. Moreover, RecBic also showed superiority in identifying functionally related genes in real gene expression datasets.

Availability and implementation: Code, sample input data and usage instructions are available at the following websites. Code: https://github.com/holyzews/RecBic/tree/master/RecBic/. Data: http://doi.org/10.5281/zenodo.3842717.

Contact: guojunsdu@gmail.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Biclustering that identifies related entities (rows) under certain conditions (columns) in a data matrix is often desired in many fields of data analysis. For example, biologists would like to bicluster a gene expression matrix to identify functionally related genes (rows) under certain conditions (columns), measured using high-throughput methods such as DNA microarrays and more recently RNA-seq (Tanay *et al.*, 2002; Xie *et al.*, 2019). The goal of biclustering a gene expression matrix therefore is to find groups of genes with similar expression patterns under certain conditions (Cheng and Church, 2000). For instance, biclustering genes measured in cancerous tissues at different pathological stages may reveal genes in certain biochemical pathways dysregulated during the development of cancer (Dao *et al.*, 2010; Kluger, 2003). The biclustering problem can be traced back to Morgan and Sonquist (1963) and Hartigan (1972) who attempted to partition a numerical matrix into submatrices with entries being as similar as possible. Since Cheng and Church (2000) first introduced a biclustering algorithm for gene expression matrices in which the mean squared residue was used as a metric for approximately constant bicluster patterns, many biclustering algorithms (Bryan and Cunningham, 2008; Bryan *et al.*, 2006; Carmonasaez *et al.*, 2006; Kung *et al.*, 2006; Li *et al.*, 2006; Reiss *et al.*, 2006) have been developed based on the metric. However, it was soon recognized that the constant bicluster pattern metric is not sufficient to identify transcriptionally coregulated genes (Bendor *et al.*, 2003; Prelic *et al.*, 2006). Aguilar-Ruiz (2005) proposed more general bicluster metrics, including shifting, scaling and combinations, to characterize coregulated genes. However, identifying these types of biclusters are highly challenging. One of us has previously introduced QUBIC (Li *et al.*, 2009), a qualitative biclustering algorithm that solved the more general problem to some extent, but limitations still remain (see the counterexample to QUBIC in the Supplementary Materials).

More recently, it has been realized that most biologically relevant gene expression patterns tend to be trend-preserving. The expression patterns of two genes are said to be trend-preserving under certain conditions if and only if their expression vectors in the matrix are either order-preserving or order-reversing. Two vectors x and y are said to be order-preserving if and only if the corresponding

elements in their respective vectors have the same rank (with respect to the numerical value), and order-reversing if and only if x and $-y$ are order-preserving. A bicluster is said to be trend-preserving if and only if any pair of rows in the bicluster are trend-preserving. In general, the elements in a trend-preserving bicluster (even in the same row) can be the same. Genes in a trend-preserving bicluster are likely to be coregulated but may have quite different expression levels under the same conditions. Obviously, the widely studied biclusters with values being constant (Madeira and Oliveira, 2004), shifting (Aguilar-Ruiz, 2005), scaling (Aguilar-Ruiz (2005)) , shifting-scaling (Aguilar-Ruiz, 2005), or order-preserving (Bendor *et al.*, 2003) are all trend-preserving (see Supplementary Materials for details). Unfortunately, the problem of discovering trend-preserving biclusters is computationally intractable, as even the simplest case with binary values is NP-hard (Madeira and Oliveira, 2004). Wang *et al.* (2016) developed UniBic based on the longest common subsequence (LCS) algorithm (Cormen *et al.*, 2009) to identify trend-preserving biclusters using an index matrix derived from the original data matrix. Although UniBic outperforms QUBIC to some extent, it fails when the biclusters are relatively narrow (with very few columns but many rows), because its seed-locating procedure cannot guarantee a global optimum solution (see Supplementary Materials for a counterexample). Many other existing biclustering algorithms based on maximal row-sequential patterns also fail to identify narrow biclusters. Recently, Orzechowski *et al.*(2018) proposed a new biclustering algorithm EBIC using evolutionary computing to find narrow trend-preserving biclusters, but its performance degrades on broader ones.

To overcome the shortcomings of the existing methods, we developed RecBic aimed to identify trend-preserving biclusters, especially those with narrow shapes, based on the observation that there is a column permutation in a trend-preserving bicluster such that each permuted row is monotonic, i.e. either increasing or decreasing (not necessarily strictly). RecBic works by simply comparing the values in columns in a selected subdomain (see Section 2). When tested on real datasets as well as simulated ones in which both narrow and broad trend-preserving patterns were implanted, and values in background data and in implanted trend-preserving biclusters have the same distribution, RecBic outperformed all the salient tools compared in terms of accuracy and robustness. RecBic is very robust to noise and is only slightly affected by overlaps of clusters. In addition, RecBic is more parsimonious in memory usage and faster than EBIC (Orzechowski *et al.*, 2018). To our knowledge, RecBic is the first tool capable of finding various kinds of trend-preserving biclusters, especially, those with narrow shapes.

2 Materials and methods

To identify trend-preserving biclusters, RecBic first locates in columns potential seeds of biclusters, and then grows each of them into a full bicluster. Given a data matrix $A_{m \times n}$, obviously, each pair of its columns forms a trivial $m \times 2$ trend-preserving bicluster, thus a non-trivial seed should anchor at least three columns. Based this observation, RecBic first finds out all the most significant $h \times 3$ trend-preserving biclusters as the seeds, where h is called the height of the seeds. Clearly, the higher the height of a seed, the higher the probability of the seed being genuine. To grow a seed, RecBic iteratively finds the most significant biclusters of t columns by adding a new column to the current bicluster of t - 1 columns via repeatedly deciding if an element of the new column falls into a given interval of type $[a, b]$, $(-\infty, b]$ or $[a, +\infty)$, where a and b are the elements of two adjacent columns of the current bicluster (see Fig. 1 and Supplementary Fig. S6 for an overview of RecBic).

2.1 The RecBic algorithm

2.1.1 The subroutine of growing the current biclusters

Suppose *A(I, J)* is a current bicluster, where I is a vector of row indices, and $J = \{j_1, j_2, \ldots, j_{t-1}\}$ is an ordered vector of column indices such that each row in *A(I, J)* is monotonic (not necessarily strict). We find the most significant trend-preserving bicluster of t columns

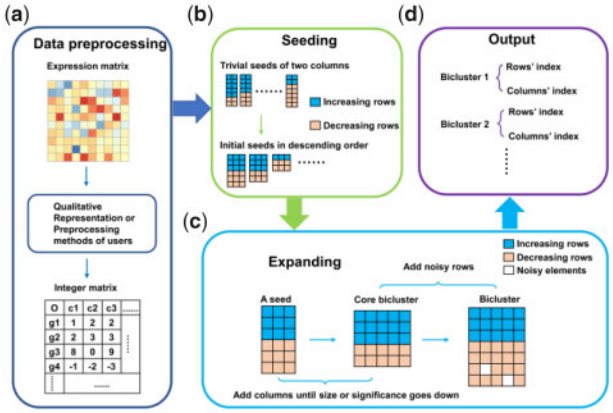

Fig. 1. Overview of RecBic algorithm

by adding a new column to *A(I, J)* as follows. For a new column j not in J, we form t-ordered vectors of column indices: $J_0 = (j, j_1, \ldots, j_{t-1})$, $J_1 = (j_1, j, j_2, \ldots, j_{t-1}), \ldots, J_{t-1} = (j_1, \ldots, j_{t-1}, j)$. Let $A(I_i, J_i)$, $i = 0, 1, \ldots, t-1$, be the submatrix obtained from $A(I, J_i)$ by deleting the rows violating monotonicity, where I_i is the remaining row indices. Let $l = argmax_i\{I_i : i = 0, 1, \ldots, t-1\}$, then $A(I_l, J_l)$ is the most significant trend-preserving bicluster of t columns determined by J_l. Greedily enumerating all j, we obtain the most significant trend-preserving bicluster of t columns based on *A(I, J)*.

2.1.2 The pseudocodes of the algorithm

Step 1 Data preprocessing: The input data is an expression matrix from microarray or RNA-seq experiments or a preprocessed integer matrix. We convert the typical expression matrix to an integer matrix by qualitative representation (Li *et al.*, 2009) (Fig.1a and see Supplementary Materials for details of qualitative representation).

Step 2.1 Partition of the columns: We partition the data matrix into k - 1 subsets by equally dividing the columns into k - 1 subsets. The integer k is determined using a previously described method (Wang et al., (2016)) (see Supplementary Materials for details).

Step 2.2 Generation of seeds: For each of the subsets of columns, we identify all the significant (highest) trend-preserving $h \times 3$ biclusters as the seeds based on each pair of columns in the subset using the subroutine described above. We sort the seeds in the descending order of their heights in a list L (Fig. 1b, Supplementary Fig. S6).

Step 3.1 Growth without noise: We start with the first seed in L as the current trend-preserving bicluster, and repeatedly grow it using the subroutine described above to obtain a core bicluster until the significance (size) of the next trend-preserving bicluster goes down (Fig. 1c).

Step 3.2 Growth with noise: We extend the core bicluster by adding as many rows as possible with a preset error rate α (Fig. 1c).

Step 4 Output: We output the resulting submatrix as a bicluster, and then remove from the list L, the seeds with their root column pairs in the discovered bicluster. Repeat step 3 until either L is empty or the prespecified number of biclusters has been obtained (Fig. 1d).

2.2 Performance comparison with existing algorithms

2.2.1 Collection of the existing algorithms with their running environments

We downloaded the R packages of ISA (Bergmann *et al.*, 2003) and FABIA (Hochreiter *et al.*, 2010), and the source code of EBIC (Orzechowski *et al.*, 2018), QUBIC2 (Xie *et al.*, 2019), UniBic (Wang *et al.*, 2016), QUBIC (Li *et al.*, 2009), CPB (Bozdağ *et al.*, 2009) and OPSM (Bendor *et al.*, 2003) from their respective websites. We ran ISA and FABIA in R version 3.4.1, and compiled CPB, EBIC, OPSM, QUBIC, QUBIC2 and UniBic on a Linux workstation. We choose these programs for the performance comparison with RecBic, as they represent the state-of-the-art of biclustering

algorithms. We ran the programs with the default settings, the parameters suggested in their publications (see Supplementary Materials), and 'number of biclusters' set to the number of implanted biclusters in the simulated datasets. We also tested EBIC with optimized parameter $n = 20\ 000$ (see Supplementary Materials). BicPAMS (Henriques *et al.*, 2017) was not included in the comparison as it was outperformed by EBIC (Orzechowski *et al.*, 2018) and UniBic (Wang *et al.*, 2016).

We evaluated RecBic by comparing it with the eight state-of-the-art tools mentioned above on various artificial datasets as well as real datasets. First, we compared RecBic with these tools on the simulated datasets for their capability of identifying both narrow and broad trend-preserving biclusters as well as six bicluster patterns. Second, we compared their capability of identifying overlapping biclusters on the simulated datasets. Third, we compared their robustness on datasets with different noise levels. Finally, we compared their capability of discovering functionally related genes using real gene expression datasets.

2.2.2 Evaluation criterion

Since there is no golden benchmark real dataset to validate the accuracy of a biclustering algorithm, it is a common practice to test it on simulated datasets using two measures, recovery score and relevance score, introduced by Prelić *et al.* (2006), based on the match scores between the predicted and actual biclusters. Concretely, the match score between two biclusters b_1 and b_2, is defined as the ratio between the number of genes in their intersection and the number of genes in their union (Jaccard coefficient), i.e. $ms(b_1, b_2) = |b_1 \cap b_2| / |b_1 \cup b_2|$, which measures the similarity between the two biclusters. For two sets of biclusters M_1 and M_2, the match score between them is defined as (Prelic *et al.*, 2006)

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

which measures the average similarity between biclusters in M_1 and M_2. Let G and D be the sets of genuine and detected biclusters, respectively, then we call $ms(G, D)$ and $ms(D, G)$ the recovery and relevance scores, respectively. Clustering Error (CE) (Horta and Campello, 2014; Orzechowski *et al.*, 2019; Padilha and Campello, 2017; Patrikainen and Meila, 2006) is used in Supplement Materials, as it offers more heavy punishments to incorrect assignments, which is more suitable for the task of simultaneous detection of multiple patterns.

2.2.3 Generation of simulated datasets

To generate simulated datasets, we first produced background matrices with the standard Gaussian distribution $N(0,1)$ with expectation 0 and standard deviation 1, and then implanted various trend-preserving biclusters in the background matrices. Specifically, to implant an $s \times t$ trend-preserving bicluster in a background matrix A, we randomly selected an $s \times t$ submatrix in A and a row in the submatrix, and then permutated other rows in the submatrix such that the permutated rows are all trend-preserving with the selected row, resulting in a trend-preserving bicluster. Clearly, the elements in the implanted bicluster have the same distribution as those in the background matrix. To test the robustness of the algorithm, we randomly perturbated a portion α (set to 0, 0.1, 0.2, 0.3) of elements of each implanted bicluster. We also generated datasets in which the implanted biclusters overlapped with one another at different levels. In addition, we downloaded from (Wang et al., (2016)) the datasets implanted with six different bicluster patterns (trend-preserving, column-constant, row-constant, shift, scale and shift-scale). Notice that the implanted trend-preserving biclusters except those downloaded from (Wang et al., (2016)) have the same distribution with the background matrix. We compared RecBic to all the salient algorithms on simulated datasets with the background matrix of 1000 rows and 200 columns. To mimic the size of real gene expression data, we also tested them on simulated datasets with background matrix of 20 000 rows and 250 columns as well as on datasets with biclusters differing from the background (see Supplementary Materials).

3 Results

3.1 Test of the algorithm on artificial datasets

3.1.1 RecBic outperformed the existing tools for finding trend-preserving biclusters of different widths

We implanted in a 1000×200 background matrix three trend-preserving biclusters with six different widths of 100×5, 100×10, 100×20, 100×30, 100×40 and 100×120 (Section 2). We ran each tool four times on each of the six datasets and calculated the average recovery and relevance scores of the four runs on each dataset. As shown in Figure 2 a–f, RecBic almost reached the ceiling values for both relevance and recovery scores in all the six datasets, outperforming the other eight tools compared. EBIC, the runner-up, a genetic algorithm-based tool performed as well as RecBic on the first three datasets but its performance deteriorated as biclusters became broader, i.e. its relevance and recovery scores went down from (1.000, 1.000) to (0.994, 0.994), (0.880,0.875), (0.726,0.723), (0.770,0.770) and (0.710,0.710) as the number of columns of the implanted biclusters increased from 5 to 10, 20, 30, 40 and 120, respectively. When EBIC was tested using parameter $n = 20\ 000$ as used in the study by Orzechowski *et al.* (2018), its performance

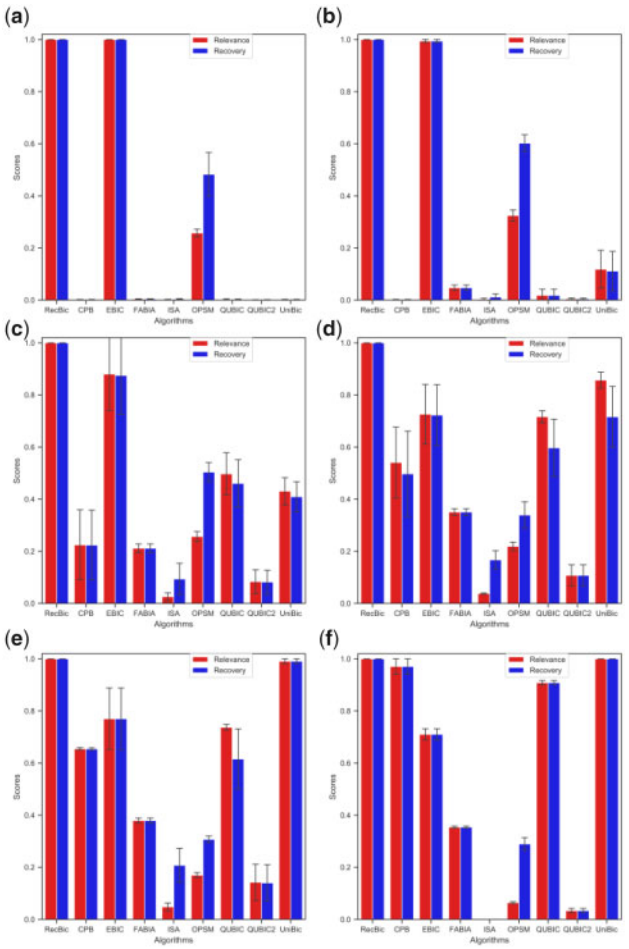

Fig. 2. Comparisons of the tools on the six simulated datasets implanted with trend-preserving biclusters of different widths. (**a**) Results on the dataset with three 100×5 implanted biclusters. (**b**) Results on the dataset with three 100×10 implanted biclusters. (**c**) Results on the dataset with three 100×20 implanted biclusters. (**d**) Results on the dataset with three 100×30 implanted biclusters. (**e**) Results on the dataset with three 100×40 implanted biclusters. (**f**) Results on the dataset with three 100×120 implanted biclusters

improved, but still was slightly worse than RecBic in terms of relevance and recovery scores, and substantially worse in terms of resources usage (see Supplementary Materials). All the other tools compared here favored broader biclusters because their recovery and relevance scores generally increased with the increase of the width of the implanted biclusters. There are two possible reasons why the other tools behaved so poorly compared with RecBic: (i) they were all developed based on row sequential pattern and thus unable to find narrow biclusters, which is also coincident with the fact that the broader biclusters are easier to be identified than the narrower ones; and (ii) they were designed to find biclusters with elements distributed differently from the background, but this is not the case for the simulated datasets. In this sense, RecBic is extraordinary for identifying both narrow and broader biclusters equally well. We also tested the tools on datasets in which implanted biclusters have their elements distributed differently from those in the background. It has been observed that most biclustering algorithms are only capable of working in the cases where prominent difference exists between the distributions of the elements of to-be-identified biclusters and those in the background matrix. It is worth stressing that RecBic performs equally well for identifying biclusters whose elements have the same distribution as those in the background matrix (see Supplementary Fig. S12).

3.1.2 RecBic outperformed the existing tools for finding biclusters with different patterns

We next compared the algorithms on 90 well-regarded earlier datasets (Wang *et al.*, 2016) implanted with six different bicluster patterns including trend-preserving, column-constant, row-constant, shift, scale and shift-scale. The datasets for each pattern have three different sizes and were implanted with different numbers of square biclusters: (i) three biclusters with a size of 15×15 were implanted in a matrix of size 150×100; (ii) four biclusters with a size of 20×20 were implanted in a matrix of size 200×150; and (iii) five biclusters with a size of 25×25 were implanted in a matrix of size 250×250. Each size comes with five datasets, so there are 90 datasets in total.

As shown in Figure 3a–f, RecBic substantially outperformed all the compared tools for both the relevance and recovery scores. Although EBIC, the runner-up, performed relatively well in identifying trend-preserving (Fig. 3a) and row-constant (Fig. 3c) biclusters, its performance decreased in finding biclusters with other patterns. In particular, the fitness function of EBIC is not suitable for scale biclusters (Fig. 3f). When tested with the parameter $n = 20\ 000$, EBIC's performance improved, but still was worse than RecBic, particularly, on the shift-scale and scale datasets (see Supplementary Materials). To mimic the size of human gene expression data, we also compared the tools on the background matrix of 20 000 rows and 250 columns. RecBic remained its superiority, although QUBIC2, a new version of QUBIC, and ISA showed some degree of improved performance (Supplementary Fig. S13).

3.1.3 RecBic outperformed the existing tools for finding overlapping biclusters

To compare the tools for identifying overlapping biclusters, we generated four datasets, each containing three 100×20 biclusters in a 1000×200 background matrix with different overlapping levels: 0×0 (no overlap), 30×3, 40×4 and 50×5 overlaps. We then ran all the tools four times on each of the datasets and calculated the average recovery and relevance scores for each tool on each dataset. As shown in Figure 4a–d, RecBic consistently outperformed the other tools by almost reaching the ceiling value 1 for both relevance and recovery scores. The performance of EBIC, QUBIC and OPSM deteriorated greatly, while that of RecBic, UniBic and CPB only decreased slightly as the overlapping level went up. However, ISA performed even better compared to itself for overlapping biclusters than no-overlapping ones, because it was designed especially for identifying overlapping biclusters (Bergmann *et al.*, 2003). We also compared the tools on simulated datasets of sizes mimicking real gene expression data. The performance of RecBic and FABIA only

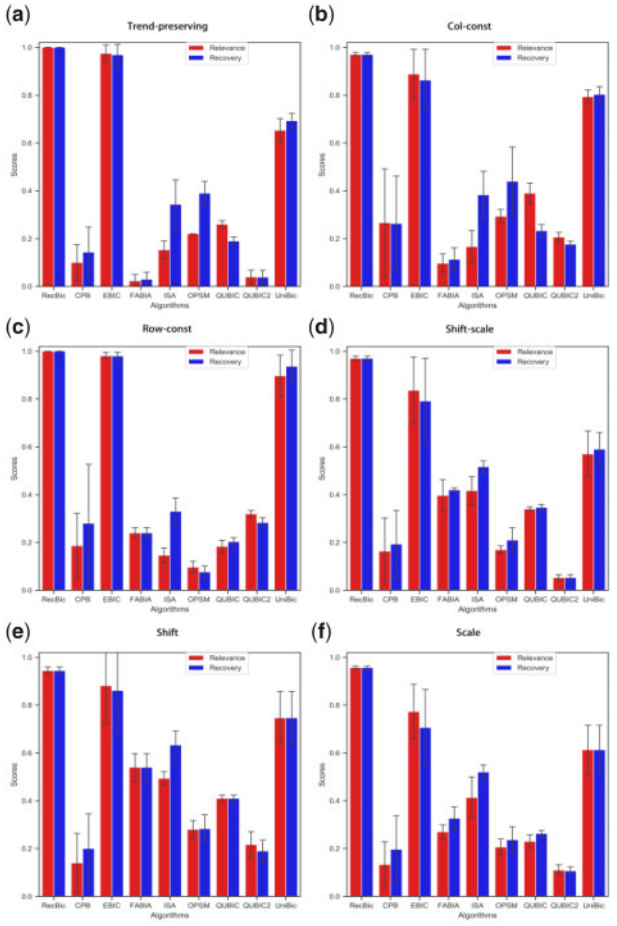

Fig. 3. Comparisons of the tools on the six datasets containing various bicluster patterns. (**a**) Results on the dataset implanted with trend-preserving biclusters. (**b**) Results on the dataset implanted with column constant biclusters. (**c**) Results on the dataset implanted with row constant biclusters. (**d**) Results on the dataset implanted with shift-scale biclusters. (**e**) Results on the dataset implanted with shift biclusters. (**f**) Results on the dataset implanted with scale biclusters

decreased slightly as the overlapping level went up, while that of EBIC and OPSM substantially deteriorated (Supplementary Fig. S16). We also tested EBIC with the parameter $n = 20\ 000$, slightly improving the performance, which still was worse than that of RecBic (see Supplementary Materials), since overlapping biclusters may lead EBIC to prematurely converge to a local optimum.

3.1.4 RecBic outperformed the existing tools for identifying noisy biclusters

The noise level is defined as the maximum of row noise levels in a bicluster, where a row noise level is the ratio between the number of elements whose removal will make the row monotonic and the number of all elements in the row. We tested the impact of noises on the performance of the tools on datasets containing three biclusters of size 100×20 with different noise levels (0, 0.1, 0.2 and 0.3) in a background matrix of size 1000×200 (see Section 2). As shown in Figure 5a–d, RecBic consistently achieved the highest relevance and recovery scores on all the datasets. EBIC, the runner-up, showed improved performance when tested with $n = 20\ 000$, but still underperformed RecBic as the noise level increased (see Supplementary Materials). Moreover, RecBic also outperformed the other tools on the datasets of size mimicking real gene expression data (Supplementary Fig. S15). Taken together, these results suggest that RecBic is more noise-tolerant than the other tools. Notice that, we compared RecBic with the competitors all under their default noise tolerant parameters.

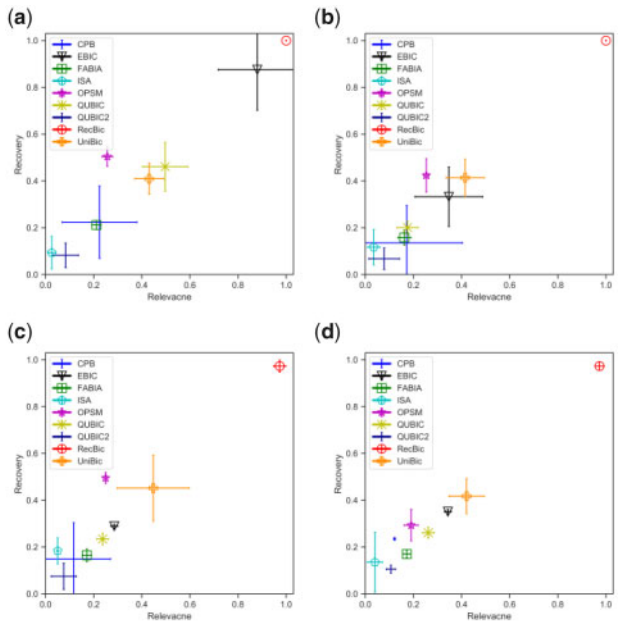

Fig. 4. Comparisons of the tools on four datasets implanted with three biclusters of size 100×20 with different overlapping levels in a 1000×200 background matrix with error bars. (**a**) Results on the dataset with non-overlapped biclusters. (**b**) Results on the dataset with the biclusters overlapping by 30×3. (**c**) Results on the dataset with the biclusters overlapping by 40×4. (**d**) Results on the dataset with the biclusters overlapping by 50×5

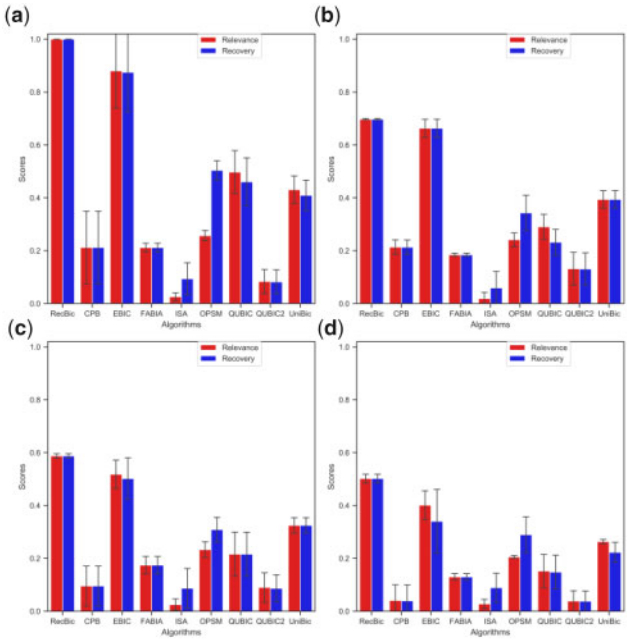

Fig. 5. Comparisons of the tools on datasets with different noise levels. (**a**) Results on the dataset without noise. (**b**) Results on the dataset with a noise level of 0.1. (**c**) Results on the dataset with a noise level of 0.2. (**d**) Results on the dataset with a noise level of 0.3

3.2 RecBic outperformed the existing tools for identifying functionally related genes in real datasets

We further compared RecBic with the other tools on eight real gene expression datasets from the GEO database that had been used in a previous study (Eren *et al.*, 2013): GDS181, GDS589, GDS1406, GDS1451, GDS 1490, GDS2520, GDS3715 and GDS3716. Each dataset contains more than 12 000 genes and dozens of conditions/samples (Table 1). As some datasets have missing values for some genes, we used the PCA method (Stacklies *et al.*, 2007) to estimate them. Since the genuine biclusters in the datasets are unknown, we evaluated each identified bicluster by each tool using the GO terms enrichment method (Eren *et al.*, 2013), with a significant P-value $<$ 0.05 (multiple hypotheses-corrected). Since the tools may output different numbers of biclusters, we assessed their performances by the proportion of GO terms enriched biclusters (Eren *et al.*, 2013). We ran RecBic with $\alpha = 0.1$ or 0.2, and set the maximal number of output biclusters to be 100 in each dataset. Results of other algorithms were quoted from Orzechowski *et al.* (2018). As summarized in Table 2, RecBic founds 558 (69.8%) GO term-enriched biclusters out of 800 output biclusters with the noise level $\alpha = 0.2$, substantially outperforming the other tools. We also ran all the algorithms with their parameters described in Supplementary Table S1, and showed their respective biclusters identified on GDS datasets as well as their enriched GO terms identified on GDS datasets (see Supplementary Materials for details).

3.3 Complexity estimation

Biclustering has been proved to be NP-hard even for the special case where the background matrix is binary (Madeira and Oliveira, 2004). A brute force algorithm to locate a trend-preserving bicluster of size s × t in a data matrix $A_{m,n}$ would have to evaluate the significance for each of $C_m^s C_n^t$ submatrices for all $s \in (1, m]$ and $t \in (1, n]$, this is of course computationally forbidden. Therefore, a heuristic approach has been widely adopted to identify biclusters in a data matrix in an accurate and efficient way. Having demonstrated the superior accuracy of RecBic relative to the existing state-of-the-art tools, we now estimate its computational complexity. Given an $A_{m \times n}$ data matrix, where n is the number of columns and m the number of rows, let t be the largest number of columns of the biclusters to be identified, o the number of biclusters to be output (a parameter prespecified by the user), and k the smallest number of columns of the biclusters to be identified. RecBic takes $O(n^3m/k)$ comparisons to identify all the seeds with three columns from k submatrices, and $O(t^2m)$ comparisons to grow a seed into a bicluster. Therefore, it takes $O(n^3m/k) + O(ot^2m) = O(max(n^3/k, ot^2)m)$ comparisons in total. It is worth emphasizing that RecBic can be easily parallelized as identifying seeds in each submatrices is independent. Figure 6 shows the running times of the tools on a laptop (with a CPU core-i7 6700hq, GTX970M and RAM 16 GB) on five datasets with background matrices of 100 columns and varying number (5000–25 000) of rows, and each was implanted with a 100×10 bicluster. Clearly, RecBic is faster than EBIC and UniBic, two most accurate algorithms so far, when the number of rows was more than 10 000. Consistent with the above analysis of time complexity, the running time of RecBic scaled linearly to the number of rows of the data matrix, indicating that RecBic is more favored to identify trend-preserving biclusters in data matrix with a great number of rows and a relatively small number of columns. Therefore, RecBic is a practical algorithm as gene expression data matrices usually consist of a large number of rows (genes) and only a few of columns (conditions/samples). Despite this, we further tested the tools on simulated datasets of with a large number of columns, found that RecBic still substantially outperform the other tools with an acceptable running time until the number of columns reaches 2000 (Supplementary Fig. S18).

4 Discussion

Since Cheng and Church's seminal work (Cheng and Church, 2000), biclustering has been widely used in analyses of gene expression data, as it provides flexibility to identify co-expressed genes under some but not necessarily all conditions/samples, which the traditional clustering methods lack. However, most of the existing biclustering algorithms were designed to identify a special kind of biclusters. In this study, we developed a novel bicluster algorithm RecBic to discover trend-preserving biclusters, be it narrow or broad, in any type of data matrices. RecBic first identifies globally optimized

Table 1. Description of the GDS datasets

Dataset	Genes	Samples	Organism	Description
GDS181	12 626	84	*Homo sapiens*	Large-scale analysis of human Transcriptome
GDS589	8799	122	*Rattus norvegicus*	Multiple normal tissue gene expressions across strains
GDS1406	12 488	87	*Mus musculus*	Brain regions of various inbred strains
GDS1451	8799	94	*Rattus norvegicus*	Toxicants effect on liver: pooled and individual sample comparison
GDS1490	12 488	150	*Mus musculus*	Neural tissue profiling
GDS2520	12 625	44	*Homo sapiens*	Head and neck squamous cell carcinoma
GDS3715	12 626	110	*Homo sapiens*	Insulin effect on skeletal muscle
GDS3716	22 283	42	*Homo sapiens*	Breast cancer: histologically normal breast epithelium

Table 2. Performance of the tools on the eight GDS datasets

Algorithm	Found	GO term enrichment
RecBic ($\alpha = 0.2$)	800	558 (69.8%)
EBIC, 0.75	589	323 (54.8%)
RecBic ($\alpha = 0.1$)	800	433 (54.1%)
EBIC, 0.5	145	76 (52.4%)
UniBic	151	62 (41.1%)
OPSM	163	48 (29.5%)
QUBIC2	800	182 (22.8%)
QUBIC	91	34 (37.4%)
ISA	217	71 (32.7%)
FABIA	80	22 (27.5%)
CPB	96	34 (35.4%)

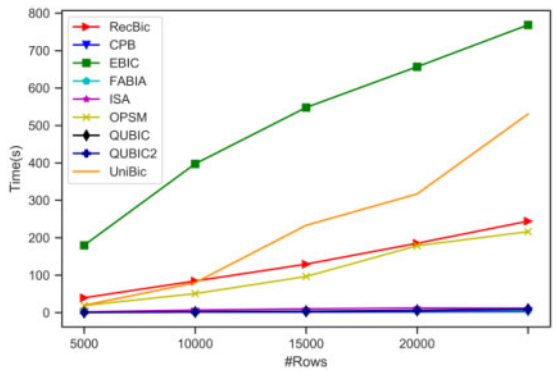

Fig. 6. Comparisons of distributions of running times against the number of rows in data matrices of 100 columns. Numbers on *x*-axis represent the rows of the background matrices and those on *y*-axis the running times of the algorithms

seeds, and then grows them into full-sized biclusters using a greedy strategy. It grows an optimum seed by greedily adding a new column to the current bicluster by repeatedly deciding if a given numerical element belongs to a given closed interval. Mathematically, RecBic globally optimizes the solutions in each iteration. To our best knowledge, RecBic is the first to identify a bicluster with a global optimum in each iteration. Comparing RecBic with eight existing state-of-the-art biclustering algorithms, we demonstrated that RecBic substantially and consistently outperformed all of them in identifying various kinds of biclusters in simulated datasets as well as functionally enriched biclusters in real gene expression datasets. RecBic can be a useful tool for analyzing gene expression data and elucidating transcriptional regulation networks.

Funding

This work was supported by National Science Foundation of China [11931008, 61771009 to G.L.], and National Science Foundation [DBI1661332 to Z.S.]. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Conflict of Interest: none declared.

References

Aguilar-Ruiz,J.S. (2005) Shifting and scaling patterns from gene expression data. *Bioinformatics*, **21**, 3840–3845.
Bendor,A. *et al.* (2003) Discovering local structure in gene expression data: the order-preserving submatrix problem. *J. Comput. Biol.*, **10**, 373–384.
Bergmann,S. *et al.* (2003) Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys. Rev. E*, **67**, 031902.
Bozdağ,D. *et al.* (2009) A biclustering method to discover co-regulated genes using diverse gene expression datasets. *J. Bioinf. Comput. Biol.*, Springer. **5462**, 151–163.
Bryan,K. and Cunningham,P. (2008) Extending bicluster analysis to annotate unclassified ORFs and predict novel functional modules using expression data. *BMC Genomics*, **9**, S20–15.
Bryan,K. *et al.* (2006) Application of simulated annealing to the biclustering of gene expression data. *Int. Conf. IEEE Eng. Med. Biol. Soc.*, **10**, 519–525.
Carmonasaez,P. *et al.* (2006) Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinformatics*, **7**, 78–78.
Church,G,M. (2000) Biclustering of Expression Data Intell Syst Mol Biol, **8**, 93.
Cormen,T.H. *et al.* (2009) *Introduction to algorithms*. MIT Press.
Dao,P. *et al.* (2010) Inferring cancer subnetwork markers using density-constrained biclustering. *Bioinformatics*, **26**, i625–i631.
Eren,K. *et al.* (2013) A comparative analysis of biclustering algorithms for gene expression data. *Brief. Bioinf.*, **14**, 279–292.
Hartigan,J.A. (1972) Direct clustering of a data matrix. *J. Am. Stat. Assoc.*, **67**, 123–129.
Henriques,R.T. *et al.* (2017) BicPAMS: software for biological data analysis with pattern-based biclustering. *BMC Bioinformatics*, **18**, 82.
Hochreiter,S. *et al.* (2010) FABIA: factor analysis for bicluster acquisition. *Bioinformatics*, **26**, 1520–1527.
Horta,D. and Campello,R.J. (2014) Similarity measures for comparing biclusterings. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **11**, 942–954.
Kluger,Y. (2003) Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res.*, **13**, 703–716.
Kung,S. *et al.* (2006) Symmetric and asymmetric multi-modality biclustering analysis for microarray data matrix. *J. Bioinf. Comput. Biol.*, **4**, 275–298.
Li,G. *et al.* (2009) QUBIC: a qualitative biclustering algorithm for analyses of gene expression data. *Nucleic Acids Res.*, **37**, e101–e101.
Li,H. *et al.* (2006) A general framework for biclustering gene expression data. *J. Bioinf. Comput. Biol.*, **04**, 911–993.
Madeira,S.C. and Oliveira,A.L. (2004) Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, **1**, 24–45.
Morgan,J.N. and Sonquist,J.A. (1963) Problems in the analysis of survey data, and a proposal. *J. Am. Stat. Assoc.*, **58**, 415–434.
Orzechowski,P. *et al.* (2019) Scalable biclustering—the future of big data exploration? *GigaScience*, **8**, giz078.
Orzechowski,P. *et al.* (2018) EBIC: an evolutionary-based parallel biclustering algorithm for pattern discovery. *Bioinformatics*, **34**, 3719–3726.
Padilha,V.A. and Campello,R.J. (2017) A systematic comparative evaluation of biclustering techniques. *BMC Bioinformatics*, **18**, 55.
Patrikainen,A. and Meila,M. (2006) Comparing subspace clusterings. *IEEE Trans. Knowl. Data Eng.*, **18**, 902–916.

Prelic,A. *et al.* (2006) A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, **22**, 1122–1129.

Reiss,D.J. *et al.* (2006) Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks. *BMC Bioinformatics*, **7**, 280.

Stacklies,W. *et al.* (2007) pcaMethods—a bioconductor package providing PCA methods for incomplete data. *Bioinformatics*, **23**, 1164–1167.

Tanay,A. *et al.* (2002) Discovering statistically significant biclusters in gene expression data. *Intell. Syst. Mol. Biol.*, **18**, 136–144.

Wang,Z. *et al.* (2016) UniBic: sequential row-based biclustering algorithm for analysis of gene expression data. *Sci. Rep.*, **6**, 23466–23466.

Xie,J. *et al.* (2019) QUBIC2: a novel and robust biclustering algorithm for analyses and interpretation of large-scale RNA-Seq data. *Bioinformatics*, **36**, 1143–1149.