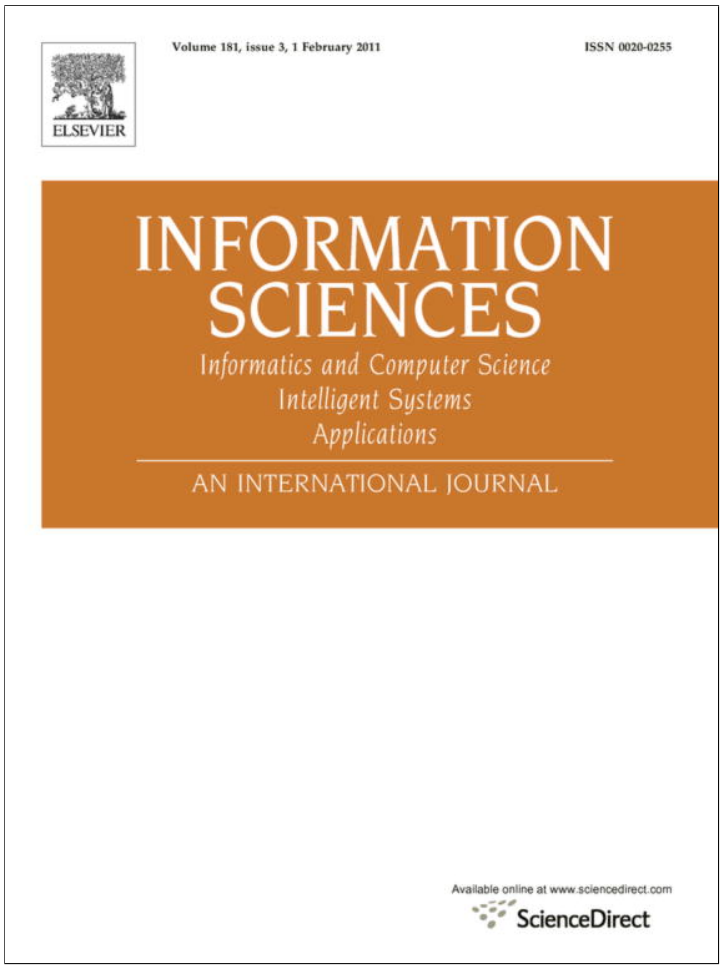


Information Sciences 181 (2011) 435–449

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Noise-robust algorithm for identifying functionally associated biclusters from gene expression data ☆

Jaegyoon Ahn [a], Youngmi Yoon [b], Sanghyun Park [a,*]

[a] Computer Science Department, Yonsei University, South Korea
[b] Division of Information Technology, Gachon University of Medicine and Science, South Korea

ARTICLE INFO

Article history:
Received 17 February 2010
Received in revised form 11 September 2010
Accepted 6 October 2010

Index Terms:
Knowledge discovery
Data mining
Biclustering
Gene expression data analysis
Microarray analysis

ABSTRACT

Biclusters are subsets of genes that exhibit similar behavior over a set of conditions. A biclustering algorithm is a useful tool for uncovering groups of genes involved in the same cellular processes and groups of conditions under which these processes take place. In this paper, we propose a polynomial time algorithm to identify functionally highly correlated biclusters. Our algorithm identifies (1) gene sets that simultaneously exhibit additive, multiplicative, and combined patterns and allow high levels of noise, (2) multiple, possibly overlapped, and diverse gene sets, (3) biclusters that simultaneously exhibit negatively and positively correlated gene sets, and (4) gene sets for which the functional association is very high. We validate the level of functional association in our method by using the GO database, protein–protein interactions and KEGG pathways.

1. Introduction

Finding sets of co-regulated genes can lead to identification of their functionality and eventually the genetic pathways involved. Clustering co-regulated genes from a microarray dataset is performed by examining the expression values of the genes under various conditions. Each condition, also called a sample, denotes the state to which the gene was exposed (e.g., temperature) or the stage of an arbitrary cellular processes. Note that not all samples are observed in a particular cellular process, nor do all genes in a microarray dataset participate in it. We can expect subsets of genes to be co-regulated under certain experimental conditions, but to behave almost independently under other conditions [7]. The data mining technique used to find a submatrix of a functionally coherent gene and sample set in a microarray is known as biclustering, as presented by Cheng and Church [9]. A set of genes in a bicluster exhibits several patterns of expression values. The data matrix in Fig. 1(a) exhibits no obvious pattern when plotted in Fig. 1(b). Biclusters that exhibit various patterns can be identified from the data matrix with various biclustering algorithms, as indicated in Fig. 1(c)–(e).

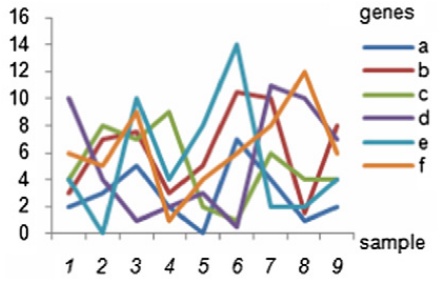
Many biclustering algorithms have been introduced [17], and most variants of the biclustering problem have been shown to be NP-hard [9], so all of these algorithms use heuristic methods or probabilistic approximations. Accordingly, these algorithms have various strengths and weaknesses and each identifies different patterns. Biclustering algorithms can be generally divided into two groups according to the type of patterns:

☆ This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2010-0008639).
* Corresponding author. Tel.: +82 2 2123 5714; fax: +82 2 365 2579.
E-mail addresses: ajk@cs.yonsei.ac.kr (J. Ahn), ymyoon@gachon.ac.kr (Y. Yoon), sanghyun@cs.yonsei.ac.kr (S. Park).

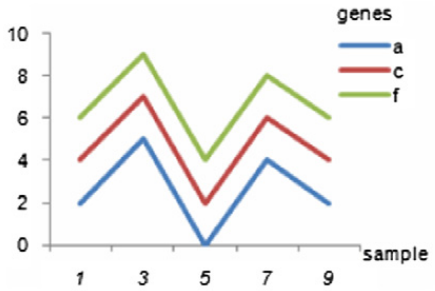
0020-0255/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.10.013

gene \ sample	1	2	3	4	5	6	7	8	9
a	2	3	5	2	0	7	4	1	2
b	3	7	7.5	3	5	10.5	10	1.5	8
c	4	8	7	9	2	1	6	4	4
d	10	4	1	2	3	0.5	11	10	7
e	4	0	10	4	8	14	2	2	4
f	6	5	9	1	4	6	8	12	6

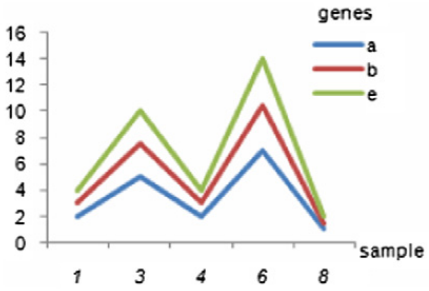
(a) Microarray data



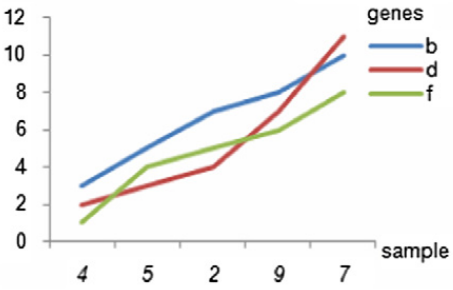
(b) Microarray data plotted



(c) Additive pattern



(d) Multiplicative pattern



(e) Order preserving pattern

Fig. 1. Gene expression patterns of a bicluster. (a) is microarray data of which cells represent gene expression values. (b) represents a plotted version of (a). (c), (d), and (e) display biclusters identified from microarray data (a).

1.1. Algorithms that find additive or multiplicative patterns

The δ-bicluster [9] is a submatrix for which MSR (mean squared residue) is small. When a high degree of noise is allowed in a δ-biclustering, it means that the δ-bicluster has a large MSR. However, a large MSR in this case leads to non-additive or non-multiplicative patterns. The δ-biclusters that have additive or multiplicative patterns should have small MSRs, therefore δ-biclustering cannot allow a high degree of noise. Moreover, δ-biclustering can easily miss overlapping clusters due to random value substitutions once a bicluster is identified. This is another weakness of δ-biclustering.

The p-Cluster [22] first scans the dataset to find all column-pair and row-pair maximal clusters called MDS (Maximum Dimension Set). Then it performs sequential pruning in turn using the row-pair MDS and the column-pair MDS. It then mines the final clusters based on a prefix tree. However, p-Cluster is not robust to noise either, because the number of clusters from the prefix tree is exponential to the allowed degree of noise and the size of the prefix. Those limitations make p-Cluster unpractical.

The Tri-Cluster [24] is the first algorithm that mines a 3 dimensional microarray dataset. It makes a DFS (Depth First Search) tree for which the nodes are genes which show the same range of fluctuation within a user specified threshold ε. If ε is too big, the DFS tree could grow too deep to complete the mining. However, Tri-Cluster with small ε does not allow a high degree of noise. Moreover, the time complexity is exponential to the number of samples, which makes Tri-Cluster difficult to use.

The reg-Cluster [23] mines additive and multiplicative co-regulation patterns by first defining d_{ij} as the difference in the gene expression values between conditions c_i and c_j and then finding the gene set for which the ratio of d_{01} to d_{ij} is within ε. Although the idea of mining additive and multiplicative patterns together is novel, there are also several problems with this. First, finding a proper ε is a very difficult task. If ε is too big, the gene set in a bicluster will have many false positive genes, while if ε is too small, the gene set will have many false negatives. Also, reg-Cluster constructs a DFS tree, in the same manner as Tri-Cluster, which leads to the same problems mentioned above.

Table 1
Summary of biclustering algorithms.

Algorithm	Features	Pros	Cons
δ-bicluster	Additive or Multiplicative patterns	Define Bicluster for the first time	Do not allow overlapping biclusters
p-Cluster	Additive or Multiplicative patterns	Generalize the subspace clustering	Exponential time complexity due to DFS tree
Tri-Cluster	Additive or Multiplicative patterns	Applicable to 3-D microarray including time axis	Exponential time complexity due to DFS tree
BiMine	Additive or Multiplicative patterns	Biologically sound evaluation function for biclusters	Exponential time complexity due to DFS tree
reg-Cluster	Additive and Multiplicative patterns	Biologically sound evaluation function for biclusters	Exponential time complexity due to DFS tree
RWB, PSObiclustering	Additive or Multiplicative patterns	Adopting heuristic search algorithms	Do not give global solutions
OPSM, OP-Cluster, KiWi	Order preserving patterns	Biologically meaningful results	Can miss significant patterns

The BiMine [5] mines sub-matrices for which the evaluation function called ASR (Average Correlation Value) is above a threshold. It constructs a DFS tree for which the node represents a submatrix. There are more genes in the submatrix and fewer samples which satisfy the biclustering condition as the tree grows. In principle, the time complexity of BiMine is exponential to the number of genes. However, its pruning method alleviates the computational cost. Moreover, it shows good GO (Gene Ontology) [4] validation results by adopting an appropriate evaluation function.

The number of nodes in a DFS tree grows exponentially as the level increases, which means that algorithms that construct DFS trees have a common problem: there are too many slightly differing biclusters. However, examining all those biclusters is not practical and, in most cases, we do not even know which particular bicluster effectively represents those similar biclusters.

Unlike the algorithms that construct a DFS tree, RWB [2] and PSObiclustering [18] use local optimization techniques to find submatrices that minimize MSR which is the same measure used in Cheng and Church [9]. PSObiclustering uses the particle swarm optimization technique, which is a population based evolutionary computation method. RWB uses a greedy technique enriched with a local search strategy to escape poor local minima. They showed that effective search was possible by adopting heuristic search methods.

Finally, a problem common to most of the previous algorithms is that a low level of noise is allowed, which makes it difficult to find all of the meaningful patterns. Thus, the available algorithms commonly find only strictly additive or multiplicative patterns.

1.2. Algorithms that find patterns by keeping an ordered sequence

The OPSM (Order Preserving Sub-Matrix) [7] defines a cluster as a submatrix of an original microarray matrix after performing column permutations separately for each row for which the gene expression values are arranged in a non-decreasing pattern (Fig. 1(e)). There are several different algorithms, such as OP-Cluster (Order Preserving Cluster) [16] and KiWi [10], which use the same basic definition of OPSM. Although OPSM shows good GO validation results [17], it can find only one bicluster at a time. Furthermore, OPSM-based algorithms can miss biologically significant patterns if they do not preserve the order [25].

Recently, a few algorithms that do not belong to the above categories have been developed [3,14]. The algorithms described above are summarized in Table 1.

In this paper, we propose a new model to identify biclusters with functionally highly correlated gene sets called RN (robust to noise)-cluster. The features of RN-Cluster are as follows: (1) RN-Cluster identifies gene sets that simultaneously exhibit additive, multiplicative, and combined patterns within user specified thresholds. This combination allows highly flexible patterns with specified levels of noise tolerance. Additionally, allowing a high level of noise does not require exponential time or space complexity in RN-Cluster, which means that RN-Cluster is robust to experimental noise. (2) RN-Cluster identifies multiple, possibly overlapping gene sets while guaranteeing gene-diversity in a bicluster by complying with a user specified similarity threshold. (3) RN-Cluster simultaneously identifies biclusters with negatively and positively correlated gene sets. (4) RN-Cluster identifies biclusters for which the functional association is very high, where the functional association is validated using the GO database, protein–protein interactions and KEGG (Kyoto Encyclopedia of Genes and Genomes) pathways [15].

2. Preliminaries

Before detailing the algorithm, we define our notation and introduce some preliminary concepts.

2.1. Notations

G	whole gene set in microarray. There are m genes
S	whole sample set in microarray. There are n samples
(O, T)	a submatrix of a microarray, where $O \subseteq G$, $T \subseteq S$
$g_0, g_1, \ldots$	genes in O
$s_0, s_1, \ldots$	samples in T
c_{ij}	expression value of gene g_i on sample s_j
d^k_{ab}	$c_{kb} - c_{ka}$, difference of expression values of sample s_a and sample s_b on gene g_k
t^i_{cd}	$d^i_{p(0)p(1)}/d^i_{cd}$, ratio of gene g_i, where $s_{p(0)}$ and $s_{p(1)}$ are the first and second samples in RN-Cluster, respectively
δ	user-specified maximum ratio threshold (> 1)
mg	user-specified minimum # of genes of an RN-Cluster
ms	user-specified minimum # of samples of an RN-Cluster
rt	user-specified similarity threshold of two RN-Clusters
$qnum$	user-specified # of priority queues to keep RN-Clusters
$qsize$	user-specified size of each priority queue

2.2. Preliminary concepts

Let $O = \{g_0, g_1, \ldots, g_{m-1}\}$ and $T = \{s_0, s_1, \ldots, s_{n-1}\}$. Let c_{ij} be the expression level of gene g_i in sample s_j. Let C be an m * n submatrix (O, T) of the dataset (G, S). We can write $C = (O, T) = \{c_{ij}\}$, $i \in [0, m-1]$ and $j \in [0, n-1]$.

Definition 1 (*RN-Cluster*). Let $C = (O, T)$ be a submatrix, where $g_h, g_i, g_j, g_k \in O$ and $T = \{\ldots, s_a, s_b, \ldots, s_c, s_d, \ldots\}$. C is an RN-cluster iff C satisfies the following properties:

1. $d^i_{ab} \neq 0,\ d^j_{ab} \neq 0,\ d^i_{cd} \neq 0$ and $d^j_{cd} \neq 0$
2. $\text{sign}(t^i_{cd}) = \text{sign}\left(t^j_{cd}\right)$, where sign (x) returns -1 if x is negative and +1 if x is positive
3. $|O| \geqslant mg \geqslant 2$ and $|T| \geqslant ms \geqslant 3$
4. $\left|t^j_{cd}\right|/\delta \leqslant |t^i_{cd}| \leqslant |t^k_{cd}| * \delta$, where $\left|t^j_{cd}\right|$ and $|t^k_{cd}|$ are maximum and minimum $|t^h_{cd}|$ values, respectively for all $g_h \in O$

The first property of Definition 1 implies that RN-Cluster does not contain genes for which the expression values are constant over a sample pair. Genes that have constant expression values over a set of samples do not exhibit any relationship. The second property implies that genes for which the t values have the same sign can be clustered together. There are 4 cases of tendency changes in each gene: (up, up), (up, down), (down, up) and (down, down), and the sign of each case is plus, minus, minus and plus, respectively. Genes for which the tendency is (up, up) and (down, down) can be clustered together, and (up, down) and (down, up) can be clustered together. Note that two genes for which the tendencies are (up, up) and (down, down) respectively can be said to be negatively co-regulated, as well as the case of (up, down) and (down, up).

The third property requires that RN-Cluster has more than mg genes and ms samples. In addition, RN-Cluster should have at least 2 genes and 3 samples. The fourth property ensures that the ratios of gene g_i, in a cluster are similar. So, the ratios are within a range of $\left|t^j_{cd}\right|/\delta$ and $|t^k_{cd}| * \delta$ where δ is a user-specified maximum ratio threshold.

Definition 2 (*p-RNC*). When RN-Cluster $C = (O, T)$ and $|T| = p$, we call C p-RNC, where p is number of samples involved in the bicluster. For example, if the number of samples $|T| = 4$, then C is 4-RNC. By the constraint that $|T| \geqslant 3$ in Definition 1, there is no 1-RNC or 2-RNC. However, exceptionally, we define 2-RNC which does not exhibit the properties of Definition 1. The 2-RNC is an $m \times 2$ submatrix of (G, S), where m is the number of genes in G as described.

Suppose there is a gene expression matrix with 10 genes and 6 samples, (G, S) as shown in Table 2. Let 2-RNC be the $G \times \{s_0, s_2\}$ submatrix, $ms = 3$, $mg = 3$, and $\delta = 2$. If we examine sample s_3, then $T = \{s_0, s_2, s_3\}$. The d^k_{02} and d^k_{23} for gene g_k are shown in Table 3. We can see that bicluster B_1 with gene set $O = \{g_0, g_2, g_4\}$ and sample set $T = \{s_0, s_2, s_3\}$ satisfies all the properties:

(1) d^k_{02} and d^k_{23} for k = 0, 2, 4 are not zero, (2) d^k_{02} and d^k_{23} for k = 0, 2, 4 have the same sign, (3) $|O| = 3 \geqslant 3$ and $|T| = 3 \geqslant 3$, and (4) the values $\max(|t^k_{23}|) = 9.14$ when k = 4, $\min(|t^k_{23}|) = 7.75$ when k = 2 and $|t^k_{23}| = 8$ when k = 0, thus 9.14/2(=4.58) < 8 < 7.75 * 2(=15.5). Because all the properties are satisfied, B_1 is a 3-RNC, and can be represented by the graph shown in Fig. 2a. Similarly, we can see that bicluster B_2 with gene set $O = \{g_3, g_7, g_9\}$ and sample set $T = \{s_0, s_2, s_3\}$ satisfies all of the above properties, and so B_2 is also a 3-RNC. B_2 is shown in Fig. 2-b. Bicluster B_3 with gene set $O = \{g_5, g_6, g_8\}$, shown in Fig. 2-c, is a 3-RNC as well. Genes of the biclusters in Fig. 2 show co-regulation or negative co-regulation on the set of conditions s_0, s_2 and s_3.

Table 2
10 × 6 microarray dataset.

gene\sample	s_0	s_1	s_2	s_3	s_4	s_5
YAL003W (g_0)	0.15	−0.07	−0.25	−0.3	−1.12	−0.67
YAL012W (g_1)	0.21	0.03	0.18	−0.27	−0.32	0.62
YAL014C (g_2)	−0.03	−0.07	0.28	0.32	−0.27	−0.36
YAL015C (g_3)	−0.25	0.58	0.77	0.28	0.32	0.65
YAL016W (g_4)	0.11	0.04	0.75	0.82	0.21	−0.2
YAL017W (g_5)	0.24	0.31	0.95	0.12	0.18	0.69
YAL021C (g_6)	−0.3	0.22	0.02	−0.64	0.06	−0.04
YAL022C (g_7)	−0.15	−0.25	0.18	0.06	−0.15	−0.17
YAL029C (g_8)	0	−0.74	−0.38	0.87	−0.34	0.12
YAL030W (g_9)	−0.15	0.2	0.31	0.15	0.04	−0.22

Table 3
Difference values and ratio of difference values for gene set O.

	g_0	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9
d^k_{02}	−0.4	−0.03	0.31	1.02	0.64	0.71	0.32	0.33	−0.38	0.46
d^k_{23}	−0.05	−0.45	0.04	−0.49	0.07	−0.83	−0.66	−0.12	1.25	−0.16
$\lvert t^k_{23}\rvert$	8	0.067	7.75	2.08	9.14	0.86	0.48	2.75	0.30	2.88

Definition 3 (*The similarity between two p-RNCs or two gene sets).* Consider two p-RNCs $C_1 = (O_1, T_1)$ and $C_2 = (O_2, T_2)$. When gene set $O = O_1 \cap O_2$, the similarity between C_1 and C_2 or the similarity between O_1 and O_2 is defined as max $(|O|/|O_1|, |O|/|O_2|)$, where max (a, b) returns a when $a \geqslant b$. We can say that two p-RNCs or two gene sets are similar if max $(|O|/|O_1|, |O|/|O_2|) \geqslant rt$.

Since we are interested in the diversity among gene sets rather than sample sets, we measure the similarity between the gene sets of two p-RNCs. For example, when p-RNC C_1 has $O_1 = \{g_0, g_2, g_3, g_5\}$ and C_2 has $O_2 = \{g_1, g_2, g_3, g_4, g_6\}$, the similarity between C_1 and C_2 or the similarity between $O_1$and O_2 is 0.5.

3. Algorithm

The RN-Cluster mines a set of genes that behave similarly on a set of samples. The RN-Cluster has two main steps: (1) obtain the initial 2-RNC set for which the samples are all possible sample pairs, and (2) for each p-RNC, find (p + 1)-RNC. We describe the details of each step.

3.1. Initial 2-RNC set

The set of two samples $\{(s_i, s_j)\}$, where i < j and the set of genes is $\{g_0, g_1, \ldots, g_{m-1}\}$, forms a 2-RNC. The number of 2-RNCs is $n(n\text{-}1)/2$, which counts all possible sample pairs.

For example, in Table 2, the possible sample sets of a 2-RNC are $\{s_0, s_1\}$, $\{s_0, s_2\}, \ldots, \{s_4, s_5\}$. Note that if $ms = 3$, then $\{s_0, s_5\}$, $\{s_1, s_5\}$, $\{s_2, s_5\}$, $\{s_3, s_5\}$ and $\{s_4, s_5\}$ cannot form a 2-RNC because they cannot grow to a 3 or higher-RNC (e.g., $G \times \{s_0, s_5, s_6\}$). Similarly, $\{s_0, s_4\}$, $\{s_1, s_4\}$, $\{s_2, s_4\}$ and $\{s_3, s_4\}$ cannot form a 2-RNC if $ms = 4$.

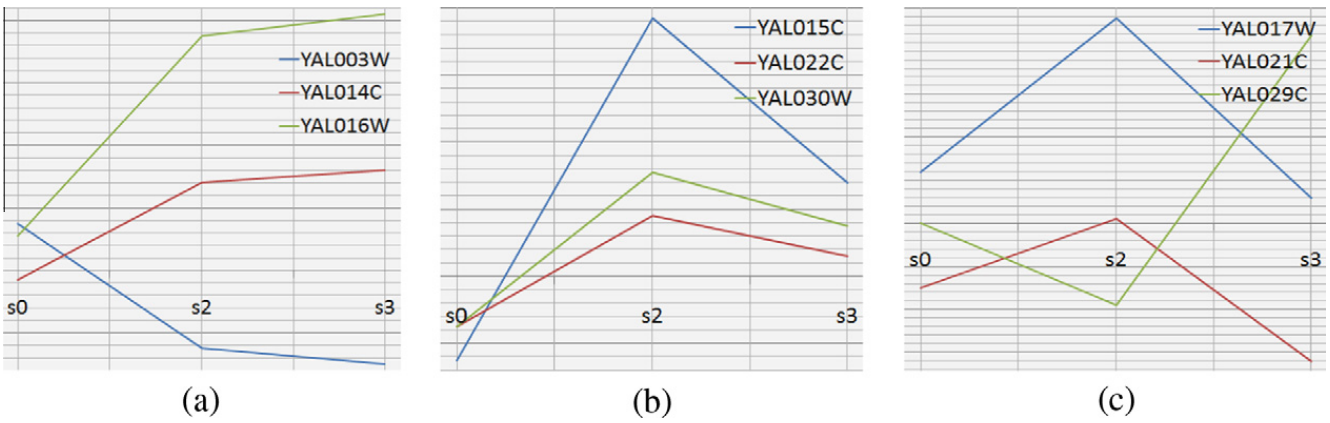


Fig. 2. Example of RN-Clusters.

3.2. Obtaining (p + 1)-RNCs from p-RNCs

For all 2-RNCs $C = (O, T)$, we make a 3-RNC by examining the sample s_i such that $last < i$, where s_{last} is the last sample in the sample set T. We can obtain 4-RNCs from 3-RNCs, 5-RNCs from 4-RNCs, etc. In other words, we perform a breadth first search to obtain the (p + 1)-RNCs from p-RNCs. This process is illustrated in Fig. 3, where each node stands for a p-RNC or (p + 1)-RNC, and the name of the node stands for the sample to be examined from its ancestor (p − 1)-RNC or p-RNC.

The entire algorithm is shown in Fig. A.1 in Appendix A. The examination process is composed of three parts: clustering, queuing, and eliminating duplicate biclusters. Details of each are described in the following subsections.

3.3. Clustering

The examination process first calculates t_{li}^k between s_{last} and s_i for all the genes g_k, where s_i is the sample we are examining and s_{last} is the last sample in the sample set T. Then, genes are clustered into two sets according to the sign of t_{li}^k: a set of positive values, *OP*, and a set of negative values, *ON*. If the sign of t_{li}^k is positive, g_k is included in *OP*; otherwise, g_k is included in *ON*.

Keeping t_{li}^k with the same sign in separate sets allows negatively correlated genes to be included in a bicluster.

The first table of Fig. 4 shows genes in *OP* and their t_{li}^k values. First, *OP* is sorted according to the t_{li}^k values in ascending order, and then we put each g_k into a bin according to the t_{li}^k values, as shown in the second table of Fig. 4. We can see that 16 bins ($bin_i \sim bin_{i+15}$) have g_k for which the t_{li}^k values are in the range of $t_{li}^0 * \delta^{i/8}$ and $t_{li}^0 * \delta^{(i+16)/8}$. This means that the maximal t_{li}^k cannot exceed the $(\text{minimal } t_{li}^k) * \delta^2$. Note that the number of bins was empirically set as 16, because we observed that using a finer (32, 64 or 128 bins) or coarser (4 or 8 bins) grain does not increase the number of resulting RN-Clusters, nor does it improve the quality of the RN-Clusters.

Let n_j be the number of the g_k in each of the 16 bins and let those 16 bins be labeled as B_j. For example, B_0 is the first 16 bins ($bin_0 \sim bin_{15}$) and has n_0 number of g_k for which the t_{li}^k values are in the range of $t_{li}^0 * \delta^0$ and $t_{li}^0 * \delta^2$. When we obtain the (B_j, n_j) pairs, we do not need to keep the (B_j, n_j) pair for which $n_j < mg$, the minimum number of genes. We sort (B_j, n_j) pairs according to n_j in descending order. The first B_j would have maximal n_j, and the gene set for the first B_j forms the gene set for the new (p + 1)-RNC for which the new sample set, T is $T \cup \{s_i\}$.

Next, we obtain another gene set by searching the next B_j in the sorted (B_j, n_j) pairs. Let the gene set of the B_j currently being searched be G and the gene sets that have already been formed into (p + 1)-RNC be *GSet*. G can be formed into a (p + 1)-RNC if the similarity as given by Definition 3 between G and every gene set of *GSet* is less than *rt*.

In Fig. 5, the sorted (B_j, n_j) pairs list is $(B_1, 7)$, $(B_2, 7)$, $(B_3, 6)$, $(B_4, 5)$, $(B_5, 5)$, $(B_6, 5)$, and $(B_7, 4)$. Therefore, the genes of B_1 form a (p + 1)-RNC. Let *rt* be 0.5. The genes of B_2 also form a (p + 1)-RNC, because there is no similarity with the first gene set. However, the genes of B_3 do not form a (p + 1)-RNC, because 5 genes (g_{11}, g_7, g_{12}, g_6 and g_0) overlap with the genes of B_1, i.e., the similarity between the gene sets of B_3 and B_1 is 5/6, which is greater than 0.5. Similarly, the genes in B_4 can form a (p + 1)-RNC. We can apply the same process to *ON*.

When generating (p + 1)-RNCs, if we cannot obtain any (p + 1)-RNCs from the *OP*s or *ON*s for all p-RNCs and all samples s_i where $s_i \in S - T$, then there are no valid (p + 1)-RNCs, and the entire process ends.

3.4. Queuing

Let the number of (p − 1)-RNCs be r. For each (p − 1)-RNC, there are at most n samples to examine, and for each examination, at most (m/mg) p-RNCs are generated, where m is the number of genes in a microarray, and mg is the user-specified minimum number of genes in an RN-Cluster. Thus there are $O(mnr)$ p-RNCs. We cannot keep all these p-RNCs due to memory limitations, and examining all of them is extremely time consuming. However, we observe that 1) we are only interested in distinguishing p-RNCs with bigger gene sets than the others, and 2) we only need to keep the p-RNCs which have a higher probability of growing to a p'-RNC, where $p' > p$.

To satisfy observation 1), we use a set of priority queues for which the priority measure is the size of the gene set, $|O|$, to keep the p-RNCs. The p-RNCs in these queues form the output biclusters. Fig. 6 compares two queuing strategies: a single priority queue and multiple priority queues. Our method uses latter strategy. Each node in the BFS tree of Fig. 6 denotes

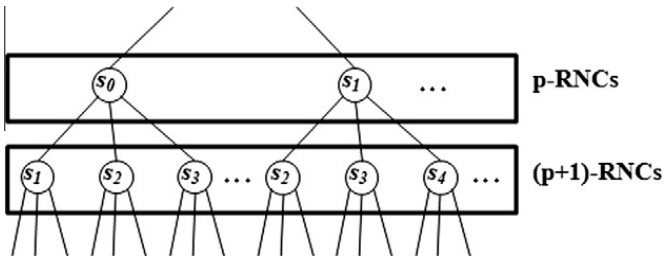


Fig. 3. Obtaining (p + 1)-RNCs from p-RNCs.

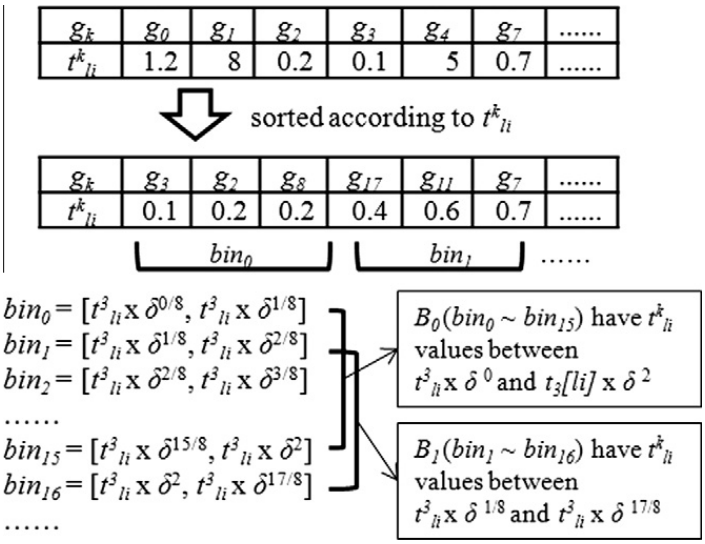


Fig. 4. Binning process.

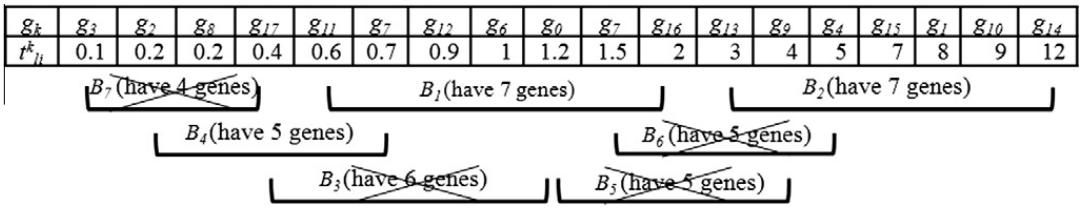


Fig. 5. Binning process when $\delta = 2$ and $rt = 0.5$.

the name of an RNC and the size of its gene set, $|O|$. The single priority strategy prunes (p + 1)-RNCs from p-RNCs B and C, meaning it does not guarantee gene variety. Thus, we keep multiple priority queues to guarantee this variety, and every (p + 1)-RNC in each priority queue can be a result.

In order to satisfy observation 2), we need to use another set of priority queues to keep p-RNCs for (p + 1)-RNCs. The queuing strategy is the same as above, but we heuristically set the priority measure to $|O| * (n - last)$, where *last* is the index of the last sample of T (for example, when $T = \{s_0, s_2, s_3\}$, *last* is 3). The benefits of using the priority measure $|O| * (n - last)$ are: (1) generally, a p-RNC with a bigger gene set grows to a (p + 1)-RNC with a bigger gene set, and 2) as *last* increases, the probability that the p-RNC grows to a larger-RNC decreases. For example, suppose that $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$ and there are two 3-RNCs, SB_1 and SB_2, for which the *Ts* are $\{s_0, s_1, s_2\}$ and $\{s_0, s_1, s_3\}$, respectively. SB_1 has more samples (s_3, s_4, s_5) to examine than SB_2 (s_4, s_5), which means that SB_1 can grow to three 4-RNCs with $T = \{s_0, s_1, s_2, s_3\}$, $T = \{s_0, s_1, s_2, s_4\}$, and $T = \{s_0, s_1, s_2, s_5\}$, while SB_2 can grow to only two 4-RNCs with $T = \{s_0, s_1, s_3, s_4\}$ and $T = \{s_0, s_1, s_3, s_5\}$. Furthermore, SB_1 can grow to a 6-RNC with $T = \{s_0, s_1, s_2, s_3, s_4, s_5\}$, while SB_2 cannot. Therefore, we can say that as *last* increases, the probability of the p-RNC growing decreases. Accordingly, the priority queues have p-RNCs which are more likely to grow to larger-RNCs.

After examining all p-RNCs, we are left with two sets of (p + 1)-RNCs: one for the output, and the other for the candidates of next (p + 1)-RNCs.

3.5. Elimination of duplicated RNCs

After obtaining two sets of priority queues, we can eliminate duplicate RNCs in order to obtain the distinguished (p + 1)-RNCs. We can do this to candidate RNCs, result RNCs, or both. First, each set of priority queues is merged into one priority queue, maintaining the priority measure. Then, we prepare a bit string of size |G| and set all bits to false. Let the gene set from the (p + 1)-RNC from the merged priority queue be $G_i = \{g_j | 0 \leqslant j < |G|\}$. We set the j'th bit of the bit string to true if the j'th bit is false and G_i contains g_j. We define *ratio* = |count of the bit that was newly set to true|/$|G_i|$. If *ratio* $\geqslant rt$, then G_i is said to be distinguished. Otherwise, the bit string is restored to the previous state.

There are five gene sets in Fig. 7. For G_0, *ratio* is 1 because the bit string is initially set to false. Because *ratio* = 1 > rt = 0.6, G_0 is distinguished. G_1 has genes g_1 and g_3, for which the bits are already set to true; therefore, *ratio* of G_1 = 2/4 < rt = 0.6. Note that for G_1 in Fig. 7, only the first and sixth bits of the string are marked as true. Because G_1 is not distinguished, the bit string is restored. G_2, G_3 and G_4 can be similarly tested, and by this means we find that G_4 is not distinguished.

If gene sets are not distinguished like G_1 and G_4 in Fig. 7, we calculate the similarity score between each gene set and every (p + 1)-RNC previously proven to be distinguished, according to Definition 3. If the similarity score is less then rt, then

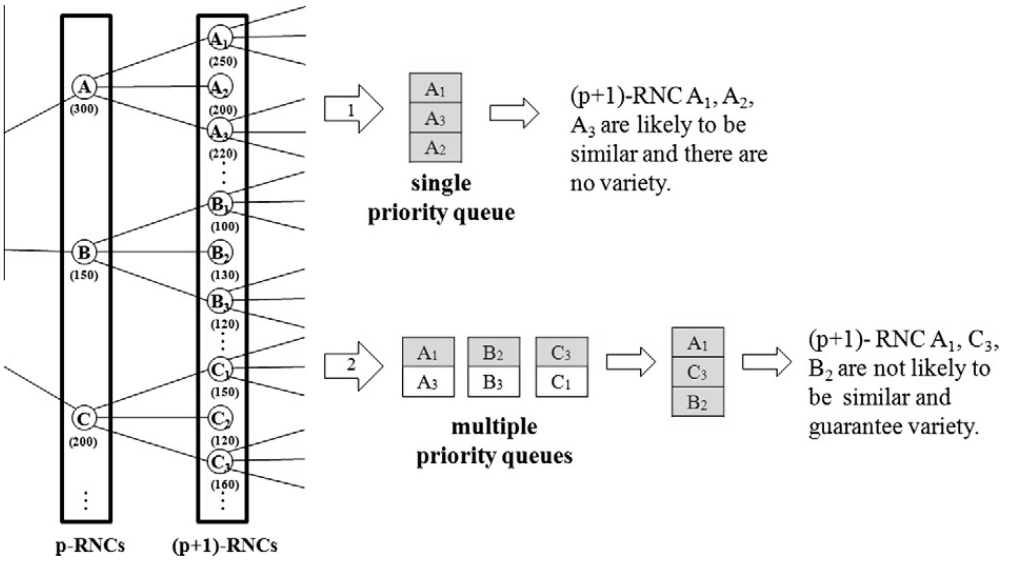


Fig. 6. Queuing strategies.

G can be consider to be distinguished. For example, if *rt* is 0.6, the gene set of every RN-Cluster is guaranteed to be overlapped with the gene sets of other RN-Clusters by at most 60%.

3.6. Analysis of time complexity

The time taken by the whole RN-Clustering process is same as the time for obtaining the set of p-RNCs, where $2 \leqslant p < n$. The time cost to obtain the initial 2-RNCs is $O(n^2)$ because there are $n(n-1)/2$ sample pairs. Let the time cost to obtain the $(p+1)$-RNCs from one p-RNC be t_p, where $p > 2$. Obtaining all $(p+1)$-RNCs from all p-RNCs takes $t_p * qsize * qnum$, where $qsize * qnum$ is the number of all p-RNCs that are in the priority queue. For each p-RNC, we examine at most n samples and, as we have already seen, examining each sample mainly involves the clustering process. There are at most m bins, and the most time-consuming process in clustering is sorting the bins. Therefore, clustering takes $O(mlogm)$, and $t_p = O(nmlogm)$.

Since obtaining all $(p+1)$-RNCs from all p-RNCs occurs at most n times and *qsize and qnum* are just user specified parameters, the whole RN-Clustering process takes approximately $O(n^2mlogm)$. Note that the time taken to cluster 2-RNCs is ignored because it is relatively small compared with the time taken by the whole RN-Clustering process.

4. Experimental results

As experimental environments, we used a Windows XP operating system on an AMD Athlon 64 X2 Dual, 2.81 GHz, 1.93 GB RAM machine. We have implemented our algorithm using the C++ language with STL (Standard Template Library). We used both synthetic and real microarray datasets. The synthetic microarray datasets were generated by varying the size of the samples and genes. Sub-matrices with additive and multiplicative patterns are hidden in each synthetic microarray. We use two real microarray datasets. The Gasch et al. [11] yeast dataset consists of 2993 genes over 173 samples exposed to various kinds of stress, and the Tavazoie et al. [20] yeast dataset consists of 2884 genes over 17 samples at two complete cell cycles. In all experiments, we used the following parameters: $ms = 4$, $mg = 5$, $\delta = 1.8$ and $rt = 0.7$ unless otherwise specified.

4.1. Setting optimal parameter values

This section describes the optimization process for six user-specified parameters for RN-Cluster: δ, *mg*, *ms*, *rt*, *qnum* and *qsize*.

	0	1	2	3	4	5	6	7	8	9	ratio
$G_0 : g_0, g_2, g_3, g_5, g_8$	t		t	t		t			t		1
~~$G_1 : g_0, g_1, g_3, g_6$~~		t					t				~~1/2~~
$G_2 : g_0, g_4, g_6, g_7$					t		t	t			3/4
$G_3 : g_1, g_6, g_9$		t								t	2/3
~~$G_4 : g_2, g_8, g_9$~~											~~0~~

Fig. 7. Elimination of duplicated RNCs when $rt = 0.6$.

Generally, parameter *mg* (the minimal number of genes) is initialized with a value of two. The parameter *ms* (the minimal number of samples) is set to the number of samples that are involved in the same cellular function or are exposed to the same condition. The number of samples can be easily obtained by reviewing the description of the samples of the microarray data.

Parameter *rt* refers to the degree of overlap between biclusters. As *rt* increases, more RN-Clusters are generated, because a larger portion of overlapping genes among RN-Clusters is allowed. The *rt* can be easily set by users in possession of domain knowledge, with values ranging from 0.2 to 0.8.

As defined in Section 2, *qsize* is the size of each priority queue and *qnum* is the total number of priority queues. Let $k = qnum * qsize$. Then, k is the total number of p-RNCs that exist before the duplicate RNCs are eliminated. Generally, we can expect a larger k to lead to less pruning, thus preventing local optima. That means that a bigger k has an opportunity to produce p-RNCs which smaller k cannot produce.

The experimental results using the Gasch yeast dataset in Table 4 reveal that *qnum* does not affect the number of RN-Clusters identified if the value is bigger than 100, however a larger *qsize* produces more RN-Clusters. Thus, it is better to choose a larger *qsize* than a larger *qnum* when $qsize * qnum$ is the same, and it is appropriate to set *qsize* to as big as possible, as long as *qnum* is greater than 100. We can set *qnum* as 100 and set *qsize* as big as the system memory permits, referring to Fig. 8(a), which shows the memory usage as a function of *qsize*, using the Gasch yeast dataset.

Finally, δ represents the behavior differences that genes in the same RN-Cluster can tolerate. A bigger δ should be used to obtain lenient results on microarrays with high levels of noise. Since δ should be varied according to the characteristics of the microarray, applying various δ to the microarray data is required. Fig. 8-(b) shows the time costs associated with varying δ in the Gasch yeast dataset. The graph is almost in log form, which means that even though δ is big enough, the time cost for RN-Clustering won't be increased proportionally. It is important for a biclustering algorithm to allow high levels of noise while remaining practical, which is why we call our algorithm RN-Clustering.

4.2. Computational costs of RN-Clustering

We generated synthetic data with various sizes and checked whether RN-Cluster is practical on a large microarray dataset. Fig. 8-(c) and (d) show the time cost as a function of the number of genes and samples, respectively. The graphs approximately fit with the time complexity analyzed in the previous section. Because microarrays generally hold no more than 100,000 genes (humans have about 22,000 genes) and 200 samples, RN-Clustering is practical to use.

Secondly, we measured the time costs by varying *qnum* and *qsize* on the Gasch yeast dataset, with the results shown in Fig. 8-(e) and (f), respectively. The graphs show linear relationships, which matches our earlier analysis of the time complexity of RN-Cluster.

4.3. Functional association analysis

We validated the results through the GO (Gene Ontology) database using FuncAssociate [8]. The GO database is a collection of GO terms, which imply biological processes, cellular components and molecular functions of gene products. Genes are functionally annotated by biological or computational experiments. Every gene belongs to at least one GO term, and a set of genes in one GO term is said to be functionally related. As a larger portion of genes in a bicluster belong to one GO term, genes in the bicluster can be said to be more functionally related. Because almost all the genes of yeast are functionally annotated, we can map the clustered genes onto the GO database and confirm that the genes are functionally related.

We compared the GO validation results with those of OPSM [7], SAMBA [19], δ-biclustering [9], and BiMine [5]. We used the toolbox of BicAT [6] for the first three algorithms, and the executable file of BiMine which was provided by its authors. Fig. 9 shows the proportion of biclusters significantly enriched by GO terms with significance level $\alpha = 1xe - 15$ for the Gasch yeast dataset and $\alpha = 0.0005$ for the Tavazoie yeast dataset. The results indicate that almost all RN-Clusters are functionally related.

4.4. Effects of noise in RN-Clusters

There are mainly two sources of noise in microarray data generation. One source of noise arises from preparation of the cDNA of the microarray, and the other arises from hybridization [21]. Because the elimination of noise in microarray

Table 4
No. of RN-Clusters as a function of *qsize* and *qnum*.

qsize	10	50	100	200	500	1000
Total No. of RNCs	64	279	502	854	1700	3263
No. of 15-RNCs	6	25	48	84	185	315
qnum	10	50	100	200	500	1000
Total No. of RNCs	352	499	502	463	430	426
No. of 15-RNCs	23	43	48	52	52	21

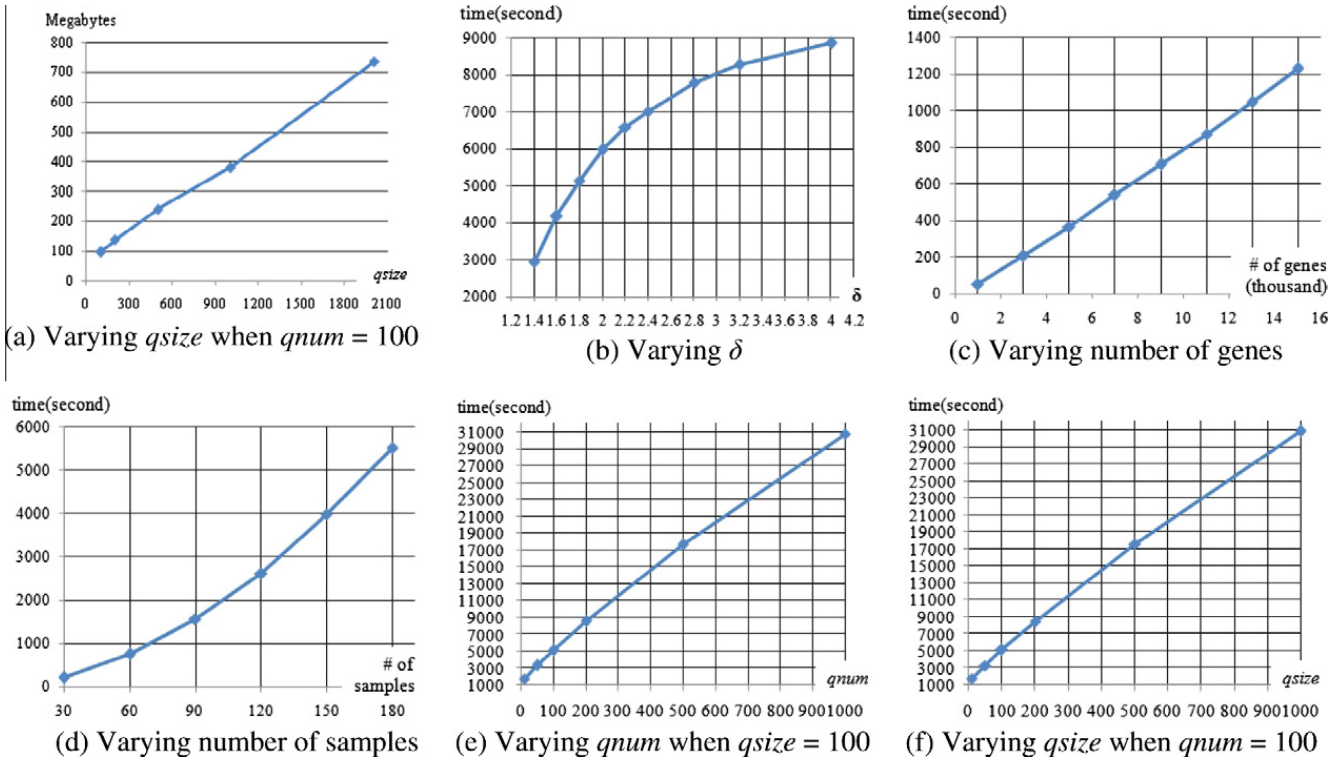


Fig. 8. Time and memory cost as a function of the size of microarray and parameters *qsize*, *qnum* and δ.

experiments is inevitable, robustness to noise is a very important feature of the algorithms that deals with microarray data. We have observed that the robustness to noise of RN-Clustering is superior to those of OPSM, SAMBA, δ-biclustering, and BiMine. We used the Gasch yeast dataset and parameters *ms* = 15, *mg* = 20, δ = 2.0, *rt* = 0.7, *qsize* = 100 and *qnum* = 20. For the noise effect, each c_{ij} (the expression value of each gene g_i on each sample s_j) was transformed into *ns*, where *ns* exhibits a normal distribution with $\mu = c_{ij}$ and σ = *noise level*. We set *noise level* as 0.0, 0.1, 0.2, 0.3, 0.4, 0.5 and measure the proportion of biclusters enriched with significance level α = 1xe − 15. A *noise level* above 0.6 was not considered because a high level of noise makes a microarray totally different from the original one. The results are shown in Fig. 10.

When the *noise level* is increased from 0 to 0.5, the proportion of biclusters is decreased from 100% to 87.32% in RN-Clustering, while the proportion of biclusters is decreased from 64.29% to 42.62% in OPSM. The proportion of biclusters is decreased from 85.45% to 50.90% in SAMBA. We can confirm that the decreasing rate of RN-Clustering is much slower than

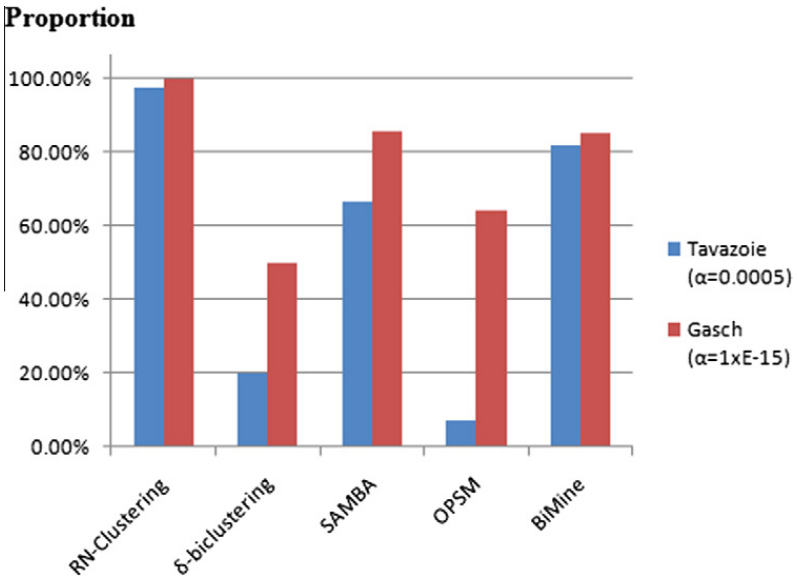


Fig. 9. Proportion of biclusters GO-enriched using two real yeast datasets.

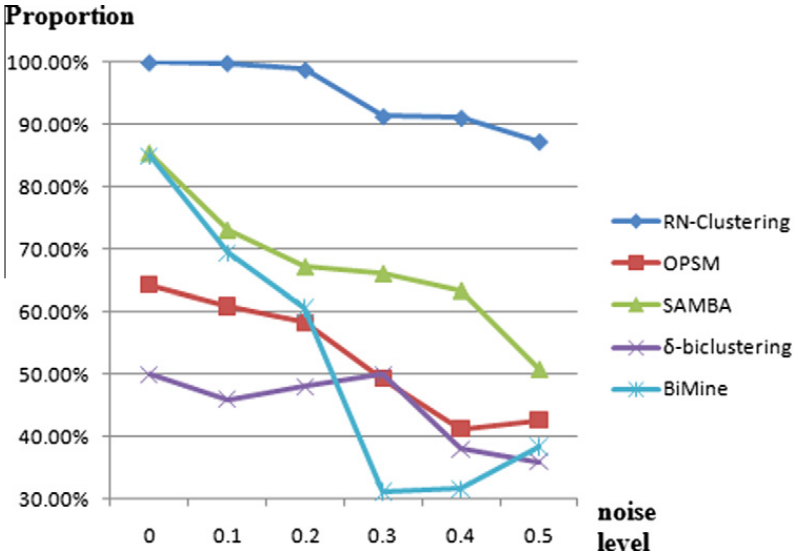


Fig. 10. Proportion of biclusters enriched with significance level $\alpha = 1\text{xe} - 15$, increasing *noise level*.

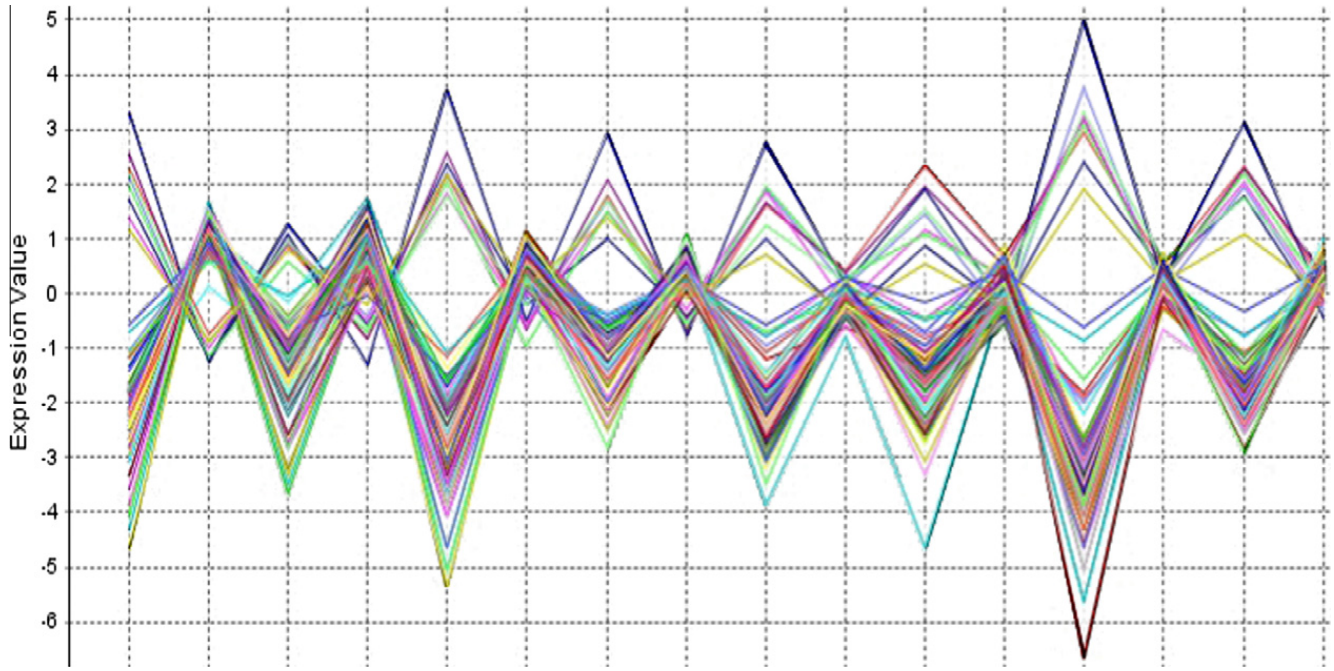
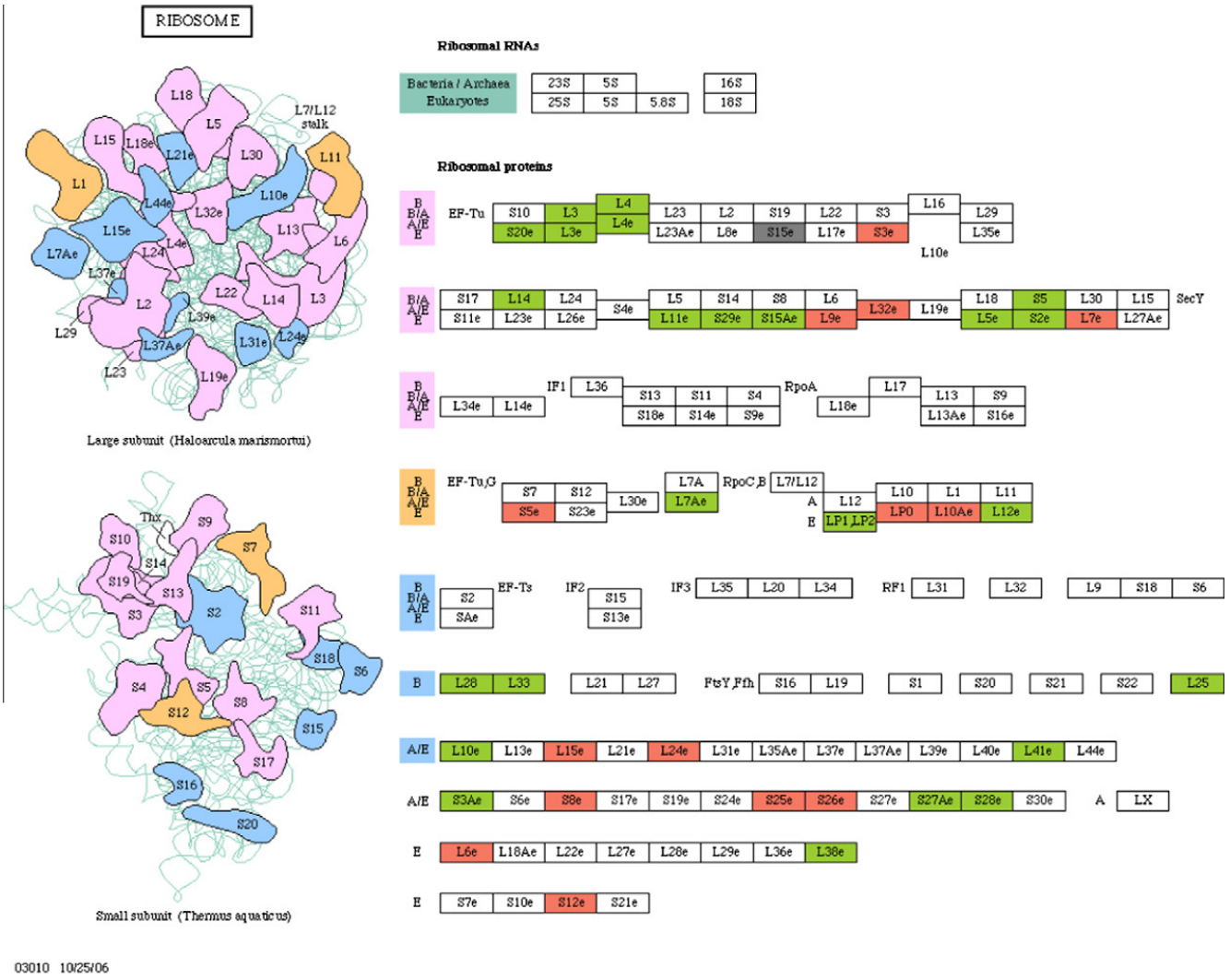


Fig. 11. Gene curve graphs of 61 × 16 RN-Cluster.

Table 5
k, *c*, *p*, and *p*-values.

k	7	8	9	10
c	809	466	586	426
p	0.0913	0.0304	0.0446	0.018
p-value	2.12e−15	1.62e−94	3.97e−85	8.51e−151
k	11	12	13	14
c	405	318	463	371
p	0.022	0.0148	0.0187	0.0136
p-value	1.66e−108	2.04e−104	3.48e−170	4.30e−152
k	15	16	17	18
c	426	333	272	158
p	0.0134	0.0092	0.0075	0.0048
p-value	2.68e−197	5.01e−173	6.74e−143	4.18e−79



(a) Ribosomal Proteins

(b) RNA polymerase process

Fig. 12. Genes mapped onto KEGG pathways.

those of OPSM and SAMBA. The δ-biclustering shows some fluctuations in decreasing rate, because δ-biclustering replaces the data in the submatrix of the previously found bicluster with random values, and the replacement could reduce the effect of noise.

RN-Clustering is robust to noise because the algorithm identifies lenient additive-patterns and lenient multiplicative-patterns in the same RN-Cluster, as shown in Fig. 11. The graph in Fig. 11 shows the gene expression values for the samples in an RN-Cluster using the Gasch yeast dataset with 16 samples and 61 genes. We can notice the additive and multiplicative patterns, which mean that RN-Clustering allows a high level of noise in the microarray data. We can also notice the positive and negative correlations.

4.5. Validation through protein pathways

In addition to validation through GO, we validated the results through protein–protein interactions, and KEGG pathways. Although these data reveal other aspects of the underlying biological system, one can expect that genes that participate in the same pathway respectively form a protein complex also show similar expression patterns as discussed in Prelic et al. [17]. We show that the computed RN-Clusters reflect this correspondence.

4.5.1. Validation through protein–protein interactions

We performed experiments using a set of yeast protein–protein interactions, which are pairs of proteins that activate or inhibit each other. Because proteins are the results of gene expression, we expect functionally related genes to form protein–protein interactions. Using hierarchical clustering, it has been shown that there are significant relationships between protein–protein interactions and gene sets [13]. Hierarchical clustering can also be used to cluster functionally related gene sets, but the functional significance of hierarchical clustering is very low when compared to other biclustering algorithms [17].

We use biologically validated protein–protein interactions from Gavin et al. [12]. Let c be the count of protein–protein interactions for which two proteins are in the same p-RNC, and let p be the probability that two randomly chosen proteins are both in the same p-RNC. We can calculate the p-value to reject the H_0 that protein interactions and the set of p-RNCs are not relevant. This is done using the following formula, based on the binomial distribution:

$$P(i > c) = \sum_{i=c}^{I} p^i (1-p)^{I-i} \left[\frac{I!}{I!(I-i)!} \right], \quad p = \frac{c'}{100,000},$$

where I is the total number of protein–protein interactions and c' is the number of protein–protein interactions for which the two proteins were both in the same p-RNC when they were randomly chosen 100,000 times.

The experiments were performed on a set of protein–protein interactions with socio-affinity score [12] > 5 (I = 4417). RN-Clustering was run with the following parameters: *qsize* = 500, *qnum* = 50, and *rt* = 0.6. Table 5 shows the c, p, and p-values (=$P(i > c)$) applied to the gene clusters of k-RNCs. Because (k + 1)-RNCs are subsets of k-RNCs, we separated the RN-Clusters according to the number of samples. The p-values are extremely low, which means there are significant relationships between RN-Clusters and protein–protein interactions. Note that all the p-values in Table 5 were 0 when we included protein–protein interactions with socio-affinity score < 5.

4.5.2. Validation through KEGG pathways

Analysis of the real yeast data set was carried out in order to enrich the KEGG pathways in the 16-RNC in Fig. 11. The KEGG pathways describe the roles of genes, and provide more biologically meaningful results. We mapped the gene sets of the 16-RNC onto the KEGG pathways using Babelomics, which is a suite of web-tools for functional annotation and analysis of groups of genes in high-throughput experiments [1]. The results are shown in Fig. 12(a) and (b). The genes found in the 16-RNC are colored green and red. The color red represents up-regulated genes and the color green represents down-regulated genes.

Fig. 12-(a) and (b) shows the genes (proteins) that constitute the ribosome and are involved in the RNA-polymerase process, respectively. The p-values, which mean the significance of the gene set over the ribosome and the RNA-polymerase process, were also calculated by Babelomics. The calculated p-values were 2.82e − 14, and 1.95e − 02 for the ribosome and the RNA polymerase process, respectively. From these results, we can conclude that the gene sets in the RN-Cluster are functionally associated.

5. Conclusion

It is significant to identify the genes that are involved in the same cellular processes or functions. In this paper, we proposed a novel biclustering algorithm, RN-Clustering, applied the algorithm to yeast microarray data, and acquired a large number of gene clusters that are proven to be functionally associated by using the well-constructed Gene Ontology database.

Functional associations of RN-Clusters were also supported by yeast protein–protein interactions. This is meaningful because these protein–protein interactions construct the protein network, which shows the coordinated action of multiple gene products. We found that the 16-RNC in Fig. 11 almost completely describes the RNA polymerase process (Fig. 12-a), and it contains many genes which are translated into proteins that organize the ribosome in which the RNA polymerase process takes place (Fig. 12-b). These results indicate that RN-Clustering could be further used to construct more accurate protein networks for eukaryote species in addition to yeast.

The high level of functional association of RN-Clusters arises from the features of RN-Clustering: 1) robustness to experimental noise, 2) guaranteed diversity as a result of unique ranging, tree forming, and queuing algorithms.

The rapid increase in the amount of large-scale gene expression data allows us to integrate many microarray datasets and identify biclusters. In the future, we plan to extend and apply RN-Clustering to integrated microarray datasets and to the identification of genetic regulation of specific biological pathways under a variety of conditions.

Appendix A

See Fig. A.1.

Input : D: a microarray dataset G x S matrix (*m* genes and *n* samples)
parameters: *mg, ms, δ, rt, qnum, qsize*
Output : p-RNC set (p ≥ *ms*)
Method :

```
 1: Make two sets of priority queues, one for candidate RNCs and the other
    for result RNCS. Each set has qnum priority queues, and the size of each
    priority queue is qsize
// Making a 2-RNC set
 2: for i = 0 to n-ms+1 do
 3:    for j = i+1 to n-ms+2 do
 4:       make 2-RNC with T={s_i, s_j} and O=G and put it into temporary
          storage
// Making a p-RNC set
 5: p ← 2
 6: while (true)
 7:    foreach p-RNC (O, T) = {c_ij} in temporary storage do
 8:       last ← last index of T
 9:       foreach sample s_i such that last < i do
10:          sample_set ← T U {s_i}
11:          foreach gene g_k ∈ O do
12:             d^k_01 ← c_k1 − c_k0
13:             d^k_1i ← c_ki − c_k1
14:             t_k ← |d^k_01 / d^k_1i|
15:             if sign(d^k_01) = sign(d^k_1i) then X_0 = X_0 U {g_k}
16:             else then X_1 = X_1 U {g_k}
17:          for i = 0 to 1 do
18:             sort X_i according to t_k
19:             make bins according to Sec.3.(c) and binning genes into bins
20:             find consecutive 16 bins that have maximal number of genes
                  and put those genes into gene_set_0
21:             j ← 0;
22:             While there are consecutive 16 bins to be examined
23:                j ← j +1
24:                find another consecutive 16 bins according to Sec.3.(c) and
                   put those genes into gene_set_j
25:             foreach gene_set_j do
26:                if the size of gene_set_j ≥ mgs then
27:                   make (p+1)-RNC with gene_set_j and sample_set
28:                   keep (p+1)-RNC into one of candidate priority queue
29:                   if the size of sample_set ≥ ms then
30:                      keep (p+1)-RNC into one of result priority queue
31:    if there are no (p+1)-RNCs in candidate priority queues then exit
32:    if (p+1) ≥ ms then
33:       Elimination of duplicated (p+1)-RNCs with rt for both sets of
          priority queues, according to Sec.3.(e)
34:       Output all (p+1)-RNCs in result priority queues
35:    clear temporary storage
36:    copy all (p+1)-RNCs of candidate priority queues into temporary
        storage
37:    clear candidate and result priority queues
38:    p ← p+1
```

Fig. A.1. RN-Clustering Algorithm. Lines 2–4 describe how initial clusters (2-RNCs) are made. Each 2-RNC is composed of gene set *O* and sample set *T* which has 2 samples. The remaining lines describe how to generate (p + 1)-RNCs from p-RNCs, starting from p = 2. Lines 7 to 30 explain the clustering and queuing processes, which were described in sub-section C and D of Section 3. Lines 32 to 34 explain the elimination process, which was described in sub-section E of Section 3. Lines 35 to 37 initialize the data structures for the next (p + 1)-RNCs. The realization of our algorithm is available by e-mail contact to the corresponding author (mailto: sanghyun@cs.yonsei.ac.kr).

References

[1] F. Al-Shahrour, P. Minguez, J.M. Vaquerizas, L. Conde, J. Dopazo, Babelomics: a suite of web-tools for functional annotation and analysis of group of genes in high-throughput experiments, Nucleic Acid Research 33 (2005) W460–W464.
[2] F. Angiulli, E. Cesario, C. Pizzuti, Random walk biclustering for microarray data, Information Sciences 178 (2008) 1479–1497.
[3] V.N.M. Aradhya, F. Masulli, S. Rovetta, A novel approach for biclustering gene expression data using modular singular value decomposition, Lecture Notes in Computer Science 6160 (2010) 254–265.
[4] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, G. Sherlock, Gene ontology: tool for the unification of biology, Nature Genetics 25 (2000) 25–29.
[5] W. Ayadi, M. Elloumi, J. Hao, A biclustering algorithm based on a bicluster enumeration tree: application to DNA microarray data, BioData Mining 2 (2009) 9.
[6] S. Barkow, S. Bleuler, A. Prelic, P. Zimmermann, E. Zitzler, BicAT: a biclustering analysis toolbox, Bioinformatics 22 (10) (2006) 1282–1283.
[7] A. Ben-Dor, B. Chor, R. Karp, Z. Yakhini, Discovering local structure in gene expression data: the order–preserving submatrix problem, in: Proceedings of the Sixth International Conference on Computational Biology, 2002, pp. 49–57.
[8] G.F. Berriz, O.D. King, B. Bryant, C. Sander, F.P. Roth, Characterizing gene sets with FuncAssociate, Bioinformatics 19 (18) (2003) 2502–2504.
[9] Y. Cheng, G.M. Church, Biclustering of expression data, in: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, La Jolla, California, 2000, pp. 93–103.
[10] B.J. Gao, O.L. Griffith, M. Ester, S.J.M. Jones, Discovering significant OPSM subspace clusters in massive gene expression data, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, Philadelphia, 2006, pp. 922–928.
[11] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, P.O. Brown, Genomic expression programs in the response of yeast cells to environmental changes, Molecular Biology of the Cell 11 (2000) 4241–4257.
[12] A. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L.J. Jensen, S. Bastuck, B.D.-mpelfeld, A. Edelmann, M. Heurtier, V. Hoffman, C. Hoefert, K. Klein, M. Hudak, A. Michon, M. Schelder, M. Schirle, M. Remor, T. Rudi, S. Hooper, A. Bauer, T. Bouwmeester, G. Casari, G. Drewes, G. Neubauer, J.M. Rick, B. Kuster, P. Bork, R.B. Russell, G. Superti-Furga, Proteome survey reveals modularity of the yeast cell machinery, Nature 430 (2006) 631–636.
[13] H. Ge, Z. Liu, G.M. Church, M. Vidal, Correlation between transcriptome and interactome mapping data from *Saccharomyces cerevisiae*, Nature Genetics 29 (2001) 482–486.
[14] N. Gupta, S. Aggarwal, Using mutual information for biclustering gene expression data, Pattern Recognition 43 (2010) 2692–2697.
[15] M. Kanehisa, S. Goto, M. Hattori, F.K. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, M. Hirakawa, From genomics to chemical genomics: new developments in KEGG, Nucleic Acid Research 34 (2006) D354–D357.
[16] J. Liu, W. Wang, Op-cluster: Clustering by tendency in high dimensional space, in: Proceedings of the IEEE International Conference on Data Mining, 2003, pp. 187–194.
[17] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. Bhlmann, W. Gruissem, L. Hennig, L. Thiele, E. Zitzler, A systematic comparison and evaluation of biclustering methods for gene expression data, Bioinformatics 22 (9) (2006) 1122–1129.
[18] D. Shyama and S.M. Idicula, "Biclustering gene expression data using KMeans-binary PSO hybrid", in: Proceedings of the International Symposium on Biocomputing, No. 43, 2010.
[19] A. Tanay, R. Sharan, R. Shamir, Discovering Statistically Significant Biclusters in Gene Expression Data, Bioinformatics 18 (2002) 136–144.
[20] S. Tavazoie, J.D. Hughes, M.J. Campbell, R.J. Cho, G.M. Church, Systematic determination of genetic network architecture, Nature Genetics 22 (1999) 281–285.
[21] Y. Tu, G. Stolovitzky, U. Klein, Quantitative noise analysis for gene expression microarray experiments, PNAS 99 (22) (2002) 14031–14036.
[22] H. Wang, W. Wang, J. Yang, P.S. Yu, Clustering by pattern similarity in large data sets, in: Proceedings of the 2002 ACM SIGMOD international conference on Management of data, Madison, Wisconsin, 2002, pp. 394–405.
[23] X. Xu, Y. Lu, A.K.H. Tung, W. Wang, Mining shifting-and-scaling co-regulation patterns on gene expression profiles, in; Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE2006), 2006, pp. 89–99.
[24] L. Zhao, M.J. Zaki, triCluster: An effective algorithm for mining coherent clusters in 3D microarray data, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2005, pp. 694–705.
[25] Y. Zhao, G. Wang, Y. Yin, G. Yu, Mining Positive and negative co-regulation patterns from microarray data, in: Proceedings on the Sixth IEEE Symposium on Bioinformatics and BioEngineering, 2006, pp. 86–93.