Chinese Journal of Electronics
Vol.21, No.4, Oct. 2012

A Graph-based Method to Mine Coexpression Clusters Across Multiple Datasets*

ZAN Xiangzhen[1], XIAO Biyu[1], MA Runnian[2], ZHANG Fengyue[3] and LIU Wenbin[1]

(1.*Department of Physics and Electronic Information Engineering, Wenzhou University, Wenzhou 325035, China*)

(2.*Telecommunication Institute, Air Force Engineering University, Xi'an 710077, China*)

(3.*Department of Biomedical Engineering, School of Life Science and Technology, Beijing Institute of Technology, Beijing 100081, China*)

Abstract — Mining coexpression clusters across multiple datasets is a major approach for identifying transcription modules in systems biology. The main difficulty of this problem lies in the fact that these subgraphs are buried among huge irrelevant connections. In this paper, we address this problem using a noise reduction strategy. It consists of three processes: (1) Coarse filtering; (2) Clustering potential subsets of graphs; (3) Refined filtering on those subsets. Using yeast as a model system, we demonstrate that most of the gene clusters derived from our method are enrichment clusters. That is they are likely to be functional homogenous entities or potential transcription modules.

Key words — Coexpression networks, Frequent dense vertex sets, Summary graph, Clustering.

I. Introduction

A key goal in systems biology is to characterize the underlying molecular mechanisms governing specific cellular behaviors and processes. Microarray technology has revolutionized the way of studying gene expression because of its capability measuring the activities of thousands of genes simultaneously. Transcription modules, which form the building block of genetic regulatory networks, are groups of genes regulated by the same transcription factor(s). The main approach to reconstruct transcription modules is identifying coexpression clusters, which are assumed likely to be controlled by the same transcription factors. However, it may result in a number of subtle and false modules because of (1) cellular processes are complex dynamical systems; (2) the transcription of genes in different modules may be perturbed simultaneously under a given condition; (3) the noisy nature of the high throughput technologies leads to a significant number of false positives or false negatives.

In order to overcome the three aforementioned problems, Zhou first systematically studied to extract the co-expression clusters across multiple datasets. In her seminar paper, she developed the concept of 2nd-order expression analysis where 1st-order expression analysis refers to the extraction of coexpression patterns from each microarray data set, and 2nd-order expression analysis refers to analyze their correlated occurrence across multiple data sets[1]. Since then, Yan *et al.* proposed a pattern-growth approach CloseCut, and a pattern-reduction approach SPLAT. Both aimed at identifying the exact recurrence of dense patterns in different graphs[2]. Hu *et al.* developed a platform, named CODENSE, to mine coherent dense subgraphs across multiple networks[3]. CODENSE utilizes a summary graph to integrate multiple data sets. However, such aggregation of multiple networks may produce tremendous false dense patterns. As the number of graphs increases, their summary graph may eventually saturate as a clique. Later, Yan *et al.* proposed to partition the input graphs into some potential subgroups according to their topological similarity[4]. Chen *et al.* recommended to first cluster edges with high correlated recurrence, and then mine frequent dense subgraphs in each edge cluster. As coexpression graphs often contain hundreds of thousands of edges, they adopted the min-hashing and locality-sensitive hashing techniques to obtain possible candidate clusters[5].

As the connection of frequent dense patterns may differ from network to network, Yan *et al.* relaxed the requirement of coherence and named these patterns across multiple graphs as Frequent dense vertexsets (FDVSs). The main difficulty of this problem lies in the fact that the FDVSs are generally surrounded among a huge amount of irrelative noise edges. In this paper, we solve the FDVSs problem through iteratively removing noise edges so that subgraphs of the FDVSs gradually come out.

The outline of this paper is organized as follows. In Section II, we introduce the problem formulation and some definitions needed. In Section III, we propose two key algorithms to eliminate irrelevant noise edges and then present the main framework of mining FDVSs. In Section IV, we take an enri-

*Manuscript Received Sept. 2011; Accepted Feb. 2012. This work is supported by the National Natural Science Foundation of China (No.60970065, No.30970666, No.61174162), Zhejiang Natural Science Foundation (No.R1110261, No.Y1080227), and Graduate Innovation Fund of Wenzhou University (No.3160601010951).

chment analysis of FDVSs obtained from microarray datasets of Saccharomyces cerevisiae and give an example of functional annotation. Finally, we conclude in Section V.

II. Problem Formulation

A microarray dataset is usually modeled as a simple unweighted and undirected graph, where each gene is represented by one node and two genes are connected with an edge if their expression profiles show high and significant correlation. A frequent densely connected subgraph across many graphs may correspond to a possible tight coexpression cluster. The FDVSs problem can be formulated as: Given m graphs with n common nodes, search all FDVSs which are densely connected in at least in k graphs ($0 < k \leq m$)). In the following, we present some definitions needed to illustrate our method.

Definition 1 (Graph set). A graph set $D = \{G_i = (V, E_i)\}$, $1 \leq i \leq m$, $E_i \subset V \times V$, where the graphs in this set share a common vertex set V.

Definition 2 (Graph density). Consider a graph $G(V, E)$, its density is defined as $\sigma = 2|E|/(|V|(|V|-1))$, where $|E|$ and $|V|$ represent the number of vertices and edges respectively.

Definition 3 (Edge clustering coefficient). Given a graph $G(V, E)$, the edge clustering coefficient of $uv \in E$ $(u, v \in V)$ is defined as Ref.[6]

$$EC_{uv} = \frac{z^{(3)}_{u,v}}{\max(d_u, d_v)}$$

where d_u and d_v represent the degree of vertices u and v respectively, $z^{(3)}_{u,v}$ is number of triangles pass through uv.

Definition 4 (Edge induced subgraph). Given a graph $G(V, E)$ and an edge $uv \in E$ $(u, v \in V)$, the subgraph induced by uv is defined as $g_{uv}(V', E')$, where $V' \subset V$ is the vertex set connected with u or v, $E' \subset E$ is the edge set between vertices in V'.

Definition 5 (Edge density coefficient). Given a graph $G = (V, E)$, the edge density coefficient of $uv \in E$ $(u, v \in V)$ is defined as

$$ED_{uv} = \frac{\sqrt{(d_u - 1)(d_v - 1)}}{(d_u + d_v - 2)/2}\sigma(g_{uv})$$

where d_u and d_v represent the degree of vertices u and v respectively, g_{uv} is the subgraph induced by edge uv.

Graph partitioning is a common way in complex networks analysis. Girvan and Newman proposed a global measure-edge betweenness, which counts the number of the shortest paths running through an edge[7]. Obviously, edges bridging dense communities tend to have higher edge betweeness score than those inside communities. However, its computation complexity is $O(|E||V|)$, which hinders its application to large scale networks. Radicchi introduced the edge clustering coefficient, which counts the number of triangles containing a given edge[7]. Recently, we proposed the edge density coefficient to measure the local density around an edge, and it is anti-correlated with the edge betweeness. This means an edge must be sparsely connected with other edges if this measure is very low[8].

Definition 6 (Frequent dense vertex set). Given a graph set $D = \{G_i = (V, E_i)\}$, a set of vertices $V' \subset V$ is a frequent dense vertex set if the density of its induced subgraphs $\delta(g_i(V'))$ is larger than or equal to δ at least in k graphs.

Definition 7 (Summary graph). Given a graph set $D = \{G_i = (V, E_i)\}$, its summary graph is a graph $S(V, \hat{E})$ where the average edge density coefficient of each edge $e \in \hat{E}$ is larger than a user-defined threshold.

Summary graph was originally used to conserve all the relevant edges and eliminate some irrelevant ones at the same time. In this paper, we use the edge density coefficient to build summary graph . Obviously, it is more suitable than the frequency of edges used in previous papers[3−5], and can be applied to efficiently filter out irrelevant edges. As the deletion of an edge will affect its neighboring edges' density coefficients, we can iteratively eliminate irrelevant edges by returning the summary graph S to each graph $G_i \in D$ (see diction III.2)

Definition 8 (Edge support vector). Given a graph set $D = \{G_i = (V, E_i)\}$ and its summary graph $S(V, \hat{E})$, the support vector of an edge $e \in \hat{E}$, written as $\boldsymbol{v}_e$, is a binary vector of length m. The ith element of $\boldsymbol{v}_e$ indicates whether e appears in G_i or not.

The edge support vector actually indicates the occurrence of an edge across the graph set D, it can be used to find potential subsets of D containing at least one common FDVS. Because of the complexity and noise of biological networks, the support vectors of edges within the same FDVS may be very different. For example, the Hamming distance between support vectors 1111100000 and 1111101011 is 3. However, the most important information is they both appear in $G_1, \cdots, G_5$. In order to capture this feature, we define the similarity of two support vectors as

$$s(\boldsymbol{v}_e, \boldsymbol{v}_{e'}) = \frac{\langle \boldsymbol{v}_e, \boldsymbol{v}_{e'} \rangle}{\min(h(\boldsymbol{v}_e), h(\boldsymbol{v}_{e'}))}$$

where $\langle \cdot, \cdot \rangle$ and $h(\cdot, \cdot)$ denote the inner product and the Hamming weight of two vectors respectively.

III. Mining Frequent Dense Vertex Sets

1. Motivation

The difficulty of mining FDVSs mainly comes from the fact that they are buried among a huge amount of irrelevant edges. Here we define the irrelevant edges as those having no contribution to the formation of FDVSs. If we can effectively filter out most of those irrelevant or noise edges, the FDVSs will become easily detected. Then, the key issue is how to identify those noise edges. Intuitively, we have following observations:

(1) If an edge $e \in E_i$ is sparsely connected with its neighbor edges, then it has no contribution to the FDVSs.

(2) If an edge $e \in E_i$ is densely connected only in a few graphs, then it has no contribution to the FDVSs because of the frequency requirement.

(3) If an edge $e \in \hat{E}$ bridges two densely connected subgraphs in summary graph S, then it has no contribution to the FDVSs.

(4) If $e \notin \hat{E} \wedge e \in E_i (1 \leq i \leq m)$, then it has no contribution to the FDVSs.

(5) If $V' \subset V$ is a FDVS in a subset of graphs $SUB(D)$, then their connections in each graph of $D - SUB(D)$ have no contribution to this FDVS.

2. Algorithms

In order to address the five kind of noisy edges listed in above, we designed two algorithms: FILTER and GCLUSTER. FILTER contains 4 steps to single out those edges listed in above, and its workflow is illustrated in Fig.1.

Algorithm 1 FILTER

Input: Graph set $D = \{G_i = (V, E_i)\}$ $(1 \le i \le m)$,

Minimal density threshold δ,

Minimal frequent support k,

User defined parameter f, p, q;

Output: Summary graph $S(V, \hat{E})$ and the resulted graph dataset $D' = \{G_i = (V, E_i')\}$;

1. filter out edges with $ED_e < \delta/f$ in each graph G_i

2. build summary graph S with edges satisfying $\sum_i ED_e \ge pk\delta (1 \le i \le m)$;

3. filter out edges with $EC_e < q$ in summary graph S;

4. return S to G_i by only keeping edges satisfying $e \in \hat{E} \wedge e \in E_i'$;

5. repeat 1 to 4 until the summary graph S does not change any more.

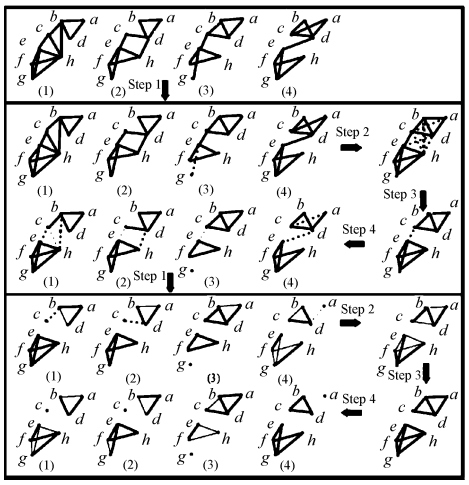

Fig. 1. Schematic illustration of the workflow of FILTER, dashed lines represent those to be deleted by previous step

Fig.1. shows a cartoon of four graphs processed by FILTER. Compared with the original graph set D, the resulted graph set D' contains less noise edges. However, we can't extract those FDVSs directly from the resulted summary graph S at this time. For example, the dense subgraph formed by vertices $\{a, b, c, d\}$ in S actually represents two FDVSs $\{a, b, d\}$ and $\{b, c, d\}$. As a FDVS only appears in a subset of graphs, it can be easily identified if we know in which subsets it appears. For $m = 20$ and $k = 6$, there are 38760 possible subsets of graphs. Obviously, it is impractical to test each of them.

GCLUSTER is designed to cluster the potential subsets of D including at least one FDVS based on those edges in the final summary graph S. Concerning the frequency requirement, we take vectors with Hamming weight k or $k + 1$ as the possible seeds to cluster. It is because we don't know the precise subsets containing at least one FDVS. GCLUSTER consists of two processes: (1) project all vectors to seed vectors; (2) select a seed vector to cluster.

Algorithm 2 GCLUSTER

Input: Summary graph $S(V, \hat{E})$, Graph number m, Minimal frequent support k,

Minimal Hamming distance threshold τ;

Output: Cluster center set C;

1. assign the support vector of each edge $e \in \hat{E}$ a weight $w(\boldsymbol{v}_e) = 1$, and puts those with Hamming weight k or $k + 1$ to set A and others to set B;

2. merge the same vectors in set A and B by adding their weight and leave only one vector;

3. for each edge $\boldsymbol{v}_e \in$ B do

find a subset SUB(A)$\subset$A where each vector having the maximal similarity score $s(\boldsymbol{v}_e, \boldsymbol{v}_{e'})$ with $\boldsymbol{v}_e$;

update the weight of $\boldsymbol{v}_{e'} \in$SUB(A) by $w(\boldsymbol{v}_{e'}) = w(\boldsymbol{v}_{e'}) + w(\boldsymbol{v}_e) * w(\boldsymbol{v}_{e'}) / \sum w(\boldsymbol{v}_{e'})$;

remove $\boldsymbol{v}_e$ from B;

4. move those vectors in A with Hamming weight k to set B;

5. repeat step 3;

6. A=A$\cup$B

7. sort vectors $\boldsymbol{v} \in$A in a decreasing order according to their weight $w(\boldsymbol{v})$;

8. do{

set T=NULL

move the first vector from A to T;

for each $\boldsymbol{v} \in$A do

if $\sum_{t \in T} h(\boldsymbol{v}, t)/|\mathrm{T}| < \tau$, then move $\boldsymbol{v}$ from A to T;

decide a cluster center c based on vectors in T and $w(c) = \sum_{t \in T} w(t)$;

add c to set C;

} while (A!=NUKK)

Not only does the projecting significantly reduce the clustering space, but also makes GCULSTER working like a soft clustering. In step 7, the majority rule is used to determine a cluster center based on the vectors in T. Specifically, we just count the weight of 1s and 0s in each bit. If the weight of 1s is larger than that of 0s, then it is 1 and vice versa. The weight of those centers can be understood as the number of edges they may represent. The larger it is, the more FDVSs they may contain. Fig.2 illustrates the workflow of GCLUSTER based on the final graph set in Fig.1. GCLUSTER projects those vectors to seed vectors with Hamming weight 2 or 3, and finally results in three cluster centers 1110, 1101 and 0011, which correspond to the subset of graphs containing $\{a, b, d\}$, $\{e, f, g, h\}$ and $\{b, c, d\}$ respectively.

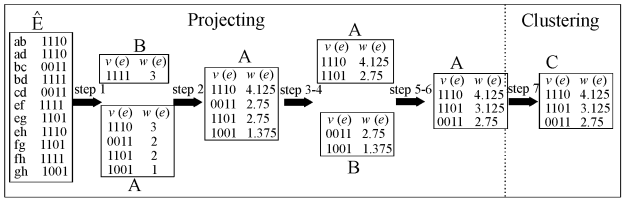

Fig. 2. Schematic illustration of the workflow of GCLUSTER by graphs in Fig.1

3. The main framework

Based on the above two algorithms, the mining process for FDVSs can be formulated as follows:

(1) Invoke FILTER to perform a coarse filtering of irrelevant edges on graph set D;

(2) Invoke GCLUSTER to find potential cluster centers C based on the resulted graph set D and the corresponding summary graph S;

(3) Build a graph set SUB(D) based on each cluster center $c \in$ C, and invoke FILTER to perform a refined filtering of irrelevant edges.

(4) Detect dense subgraphs in the resulted summary graphs and output their vertex sets.

(5) Merge identical vertex sets and separate some large vertex sets.

Concerning the user-defined parameters in FILTER and GCLUSTER, Our simulation results show that parameter f can be set from 4 to 10. The parameter p can be set from 0.1 to 0.2 for coarse filtering and from 0.8 to 0.9 for refined filtering. It is because the FDVSs appear in most graphs of SUB(D) in the latter case. The parameter q can be set as a constant 0.334. The Hamming distance threshold is set as $\tau = 2$, that means the Hamming distance between any two vectors in T is no more than 2.

If the frequency of a FDVS is larger than k, then it can be extracted from not only one SUB(D). For example, $\{a, b, d\}$ can be obtained both from cluster centers 1110 and 1101 in Fig.2. Therefore, there exist some identical FDVSs from different center $c \in$ C. In this paper, two FDVSs are identical if they differ less than 2 vertices and we can merge them as one. On the other hand, a large FDVS may contain a small one. This situation can be addressed by checking their occurrence in D. If they are highly correlated then they should be merged as one; otherwise the large FDVS should be broke into two small FDVSs.

Finally, the overlapping problem between FDVSs can be easily addressed by our method if their occurrences are not high related. For example, $\{a, b, d\}$ and $\{b, c, d\}$ in Fig.1 share a common pair of vertex $\{b, d\}$ and form a dense subgraph in summary graph S, they can be easily distinguished from two cluster centers 1110 and 0011 respectively.

IV. Experimental Study

We use 10 datasets of Saccharomyces cerevisiae from Stanford microarry dataset (http://smd.stanford.edu) and the NCBI Gene expression omnibus (http://www.ncbi.nlm.nih.gov/geo/). According to the experimental conditions, they are partitioned into 20 datasets with at least 7 experiment data (see Table 1 for detailed information.). The similarity between two genes in one dataset is measured by Pearson's correlation between their expression profiles. We transform the Pearson's correlation (denoted as r) into another quantity, $\sqrt{(n-2)r^2/(1-r^2)}$, and model it as a t-distribution with $n-2$ degrees of freedom, where n is the number of data points used. Two genes are connected if the Pearson's correlation of their expression patterns is significant at $a = 0.001$ level. Finally, we build 20 co-expression networks comprising 5672 common genes.

Table 1. The sources of microarray files

	Experimental conditions	Data points	References
1	Alpha factor release	18	Spellman *et al.* (1998)
2	Elutriation	14	
3	DNA damage (MMS) response	17	Gasch *et al.* (2001)
4	Gamma radiation	17	
5	Mock irradiation	8	
6	Diamide	8	Gasch *et al.* (2000)
7	Heat shock	22	
8	Nitrogen depletion	9	
9	Nutrition limitation	10	
10	Sorbitol effects	6	
11	Steady state	8	
12	Cell cycle alpha factor	13	Zhu *et al.* (2000)
13	Fkh1_2_alpha_fator	13	
14	Nutrition	8	Sudarsanam *et al.* (2000)
15	Sporulation	7	Chu *et al.* (1998)
16	Diauxic_shift_timecourse	7	DeRisi *et al.* (1997)
17	Signaling and circuitry of multiple MARK pathways	56	Roberts *et al.* (2000)
18	Glucose pulse on galactose chemostat	26	Ronen *et al.* (2006)
19	Calcineurin/Crzlp signaling pathway for Ca	24	Yoshimoto *et al.* (2002)
20	Calcineurin/Crzlp signaling pathway for Na	8	

1. The distribution of edges

We first study the performance of FILTER on the 20 coexpression graphs. At beginning, there are about 820967 edges whose frequencies are larger than 3 across these graphs. After processed by FILTER with $\delta = 0.6$ and $k = 6$, only 348222 edges remain. The coarse filtering process actually eliminates more than half of the original edges which are irrelevant edges. Fig.3 shows the distribution of edge numbers before and after filtered versus the Hamming weight of support vectors. The number of edges roughly obeys a power law distribution before filtered and this means the frequency of most edges is relative small, of which about 90% of edges are less than 8. It changes to a biased bell shape after a coarse filtering. Concerning the

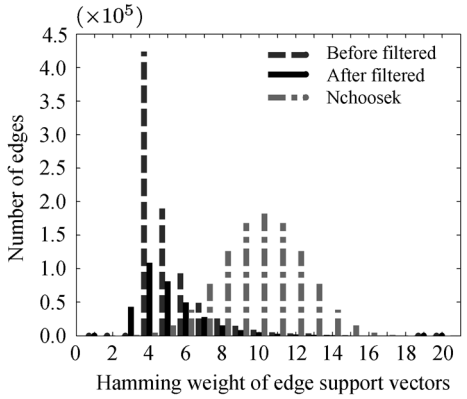

Fig. 3. The distribution of edges in graph set D before and after invoking FILTER for coarse filtering with $k = 6$, $\delta = 0.6$

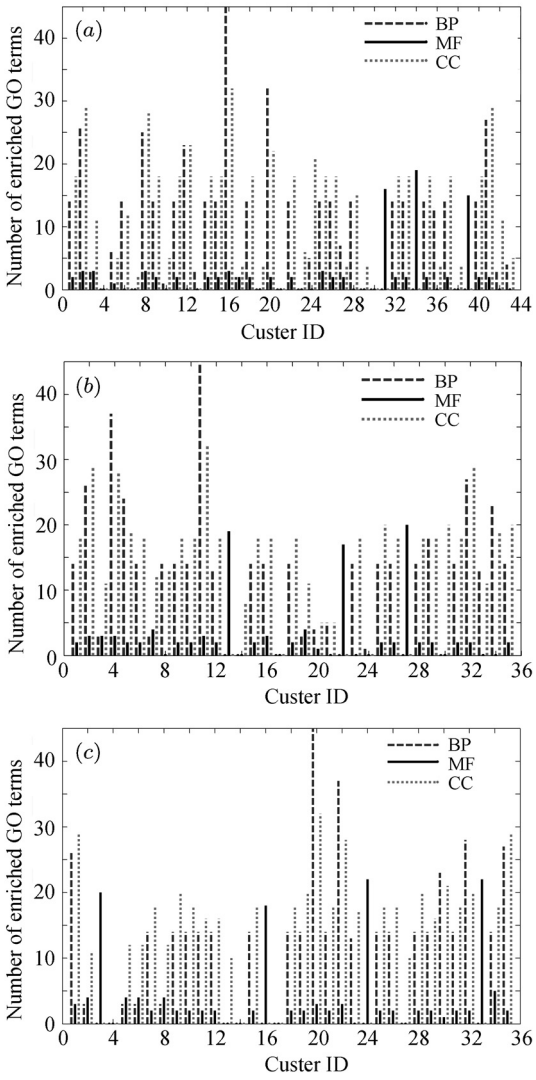

Fig. 4. The number of GO terms significantly shared by obtained clusters appearing in at least 6 graphs. Plots (a), (b) and (c) correspond to $\delta = 0.6$, 0.7, 0.8 respectively

FDVSs appearing in at least $k = 6$ graphs, GCLUSTER project all other edge vectors to those with Hamming weight 6 or 7. As they are about 22% of the total edges, using them as seed vectors to project is reasonable. Fig.3 also presents the distribution of combinatorial numbers for choosing k from $n = 20$. Having a further look at Hamming weight 3, 4 and 5, we can see that the number of edges is extraordinarily larger than the corresponding combinatorial number. This demonstrates there are huge amount of edges sharing the same support vectors at each Hamming weight. And the number of vectors to be projected and then clustered by GCLUSTER will be relatively small.

2. Enrichment analysis of FDVSs

Applying the framework in Section III.3, we obtained 43, 35 and 35 FDVSs when δ is 0.6, 0.7, and 0.8 respectively and k is 6. To quantify the functional homogeneity, we submitted those clusters to GOEAST (http://omicslab.genetics.ac.cn /GOEAST/)[9]. For each cluster, GOEAST returns a list of GO terms in three categories: Biological process (BP), Molecular function (MF) and Cellular component (CC). In this paper, an enrichment cluster is defined as if more than 80% of its genes significantly share at least one GO term with $a = 0.001$ level. Biologically, enrichment clusters tend to be regulated by the same transcription factor(s), and thus form a transcription module. Fig.4 shows the number of GO terms significantly shared by each cluster. Obviously, most of the clusters are enrichment clusters. The average numbers of GO terms shared by each cluster are 25.14, 30.77 and 30.83 when δ is 0.6, 0.7, and 0.8 respectively. This means the increasing of threshold δ can improve the functional homogeneity of clusters.

3. Functional annotation

Fig.5 presents one cluster containing seven genes YGR118W, YPR043W, YPL079W, YOR167C, YJL136C, YLR388W, YNL303W across the 20 graphs. They are densely connected in 11 graphs. When submitted to GOEAST, it returns three directed graphs corresponding to GO terms significantly shared by six genes except for YNL303W in blue color, which has no functional annotation now yet. Fig.6 shows the relation of the significantly shared GO terms in three catalogs: Biological process (BP), Cellular component (CC) and Molecular function (MF). Apart from MF, the terms in BP and CC are densely related, and the two leaf nodes GO0006412 (translation) and GO-0022626 (cytosolic ribosome) have a depth 6 and 9. This indicates they share high functional homogeneity. The degree of gene YNL303W in the 11 dense subgraphs is 5, 6, 5, 6, 6, 3, 5, 6, 6, 4, and 6 respectively. This indicates it is highly related with other six genes. Thus, we can speculate that gene YNL303W has same functions with other six genes.

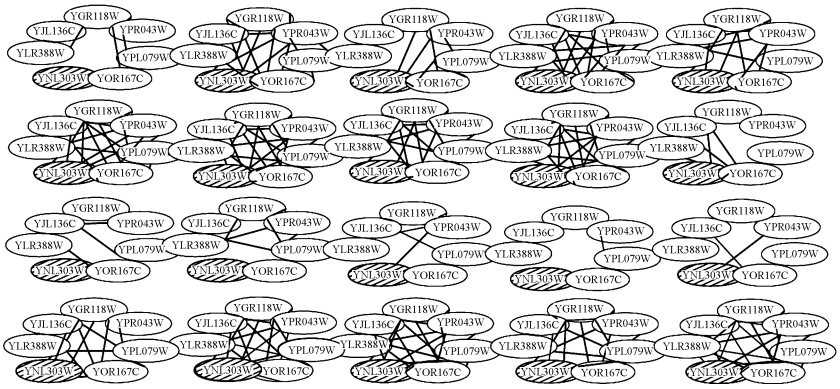

Fig. 5. A potential transcription module which is tightly coexpressed in 11 out of 20 datasets

V. Conclusions

We developed a new framework to mining frequent dense vertex sets in multiple coexpression networks. The main idea is to filter out irrelevant edges and identify FDVSs in some potential subsets of networks. In order to achieve this aim, we designed two algorithms: FILTER and GCLUSTER. Combined with the summary approach, FILTER just

uses the edge density coefficient and the edge clustering coefficient to detect and delete irrelevant edges iteratively. Through a projecting technique, GCLUSTER can achieve a simple fuzzy clustering. Our method is scalable in the number and size of graphs to be mined. And it is also extendable to weighted and directed

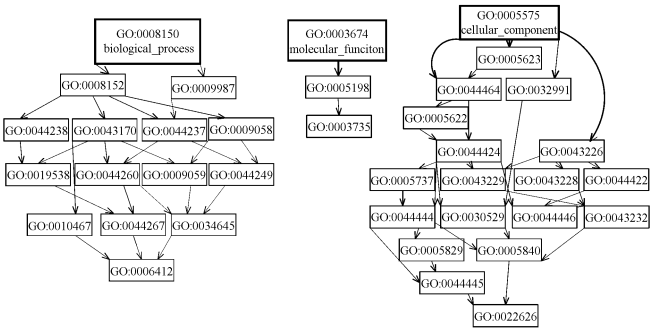

Fig. 6. The relation between GO terms significantly shared by six genes in Fig.5 in three catalogs: BP, CC and MF

graphs. We demonstrated its application in identifying coexpression clusters across 20 microarray dataset of yeast. Most of the identified gene clusters are enrichment clusters which significantly share a number of GO terms. Therefore, the discovered clusters can be used to predict functions of unknown genes, construct transcription modules and infer potential biological mechanisms.

References

[1] X.J. Zhou *et al*, "Functional annotation and network reconstruction through cross-platform integration of microarray data", *Nature Biotechnology*, Vol.23, No.2, pp.238–243, 2003.

[2] X. Yan, X. Zhou and J. Han, "Mining closed relational graphs with connectivity constraints", *Proc. 2005 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases*, pp.324–333, 2005.

[3] H. Hu, X. Yan *et al.*, "Mining coherent dense subgraphs across massive biological networks for functional discovery", *BMC Bioinformatics*, Vol.21, pp.213–221, 2005.

[4] X. Yan, M. Mehan, Y. Huang, M.S. Waterman, P.S. Yu, X.J. Zhou, "A graph based approach to systematically reconstruct human transcriptional regulatory modules", *Bioinformatics*, Vol.23, No.13, pp.577–586, 2007.

[5] L. Chen, S.M. Wang, R.S. Chen, "A method to detect gene coexpression clusters from multiple microarrays", *Progress in Biochemistry and Biophysic*, Vol.35, No.8, pp.914–920, 2008.

[6] F. Radicchi, C. Castellano, F. Cecconi *et al.*, "Defining and identifying communities in networks", *PNAS*, Vol.101, No.9, pp.2658–2663, 2004.

[7] M. Girvan, M.E.J. Newman, "Community structure in social and biological networks", *PNAS*, Vol.99, No.12, pp.7821–7826, 2002.

[8] H. Zhang, X. Zan *et al.*, "Detecting dense subgraphs in complex networks based on edge density coefficient", *IEEE Fifth International Conference Bio-Inspired Computing: Theories and Applications* (*BIC-TA*), pp.51–53, 2010.

[9] Q. Zheng, X.J. Wang, "GOEAST: a web-based software toolkit for gene ontology enrichment analysis", *Nucleic Acids Res.*, Vol.36, pp.358–363, 2008.

ZAN Xiangzhen was born in 1986. He received the B.S. degree and is assiduously study the M.S. degree in the Department of Physics and Electronic Information Engineering, Wenzhou University, China. His research interest is bioinformatics and pattern recognition.

LIU Wenbin was born in 1969. He is a professor of the Department of Physics and Electronic Information Engineering, Wenzhou University, China. He received Ph.D. degree from the Department of Control Science and Engineering, Huazhong University of Science and Technology in 2004. Then he worked as a post Ph.D. researcher at the same group for two years. In 2007, he visited the Institute of Sytems Biology for one year. His major interests include computational biology, data mining, pattern recognition DNA computing and evolutionary algorithms. (Email: wbliu6910@126.com)