Identification of Active Biological Networks and Common Expression Conditions

Mio Seki and Jun Sese

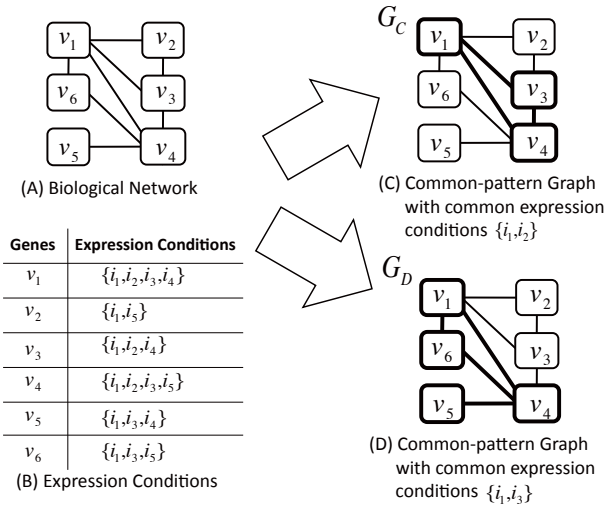
Abstract— **Biological networks such as protein-protein interaction networks and metabolic pathways are useful in understanding biological processes within living cells. The fact that some parts of the networks are activated under specific conditions, while other parts work continuously, prompts us to determine the parts and conditions. In this paper, we present a novel computational approach to simultaneously identify the locations and the conditions from gene expressions and protein interaction data. Our approach is based on the determination of large interaction networks whose genes share expression or repression conditions. We evaluate our method by using yeast protein-protein interactions and microarray expression data. The experimental results show that our method can extract networks associated with specific conditions; these networks are closely related to the Gene Ontology categories or KEGG pathways. Our method identifies the components in a proteasome complex along with their activating conditions and the relation between heat shock and cytoskeleton automatically.**

I. INTRODUCTION

Recent technological advance allows us to visualize large biological networks such as protein-protein interaction networks and metabolic pathways. Most studies on biological networks have focused on network topologies [1]. It is difficult to characterize biological networks that are active by using conventional techniques. Some networks are highly active under most stress conditions, while others are activated only under rare environmental conditions. A study on the different parts of biological networks will aid in elucidating the cellular machinery.

In a technique used to identify active networks, gene expressions are first clustered or biclustered [2] and then the members of the clusters are associated with a network. In this technique, the genes might be located far away from other members belonging to the same cluster on the network. The constrained clustering methods [3]–[5] yields gene clusters whose members are close to each other on the network. However, these methods cannot handle subnetworks that are activated in a part of the observed environment.

In this paper, we present a novel computational approach to simultaneously identify active biological networks and conditions from protein-protein interactions and gene expression data. Our approach is based on the identification of large interaction networks whose genes share expression or repression conditions. An example of the association genes with a network is shown in Figs. 1(A) and (B). In Fig. 1(A), 6 genes and 8 protein-protein interactions exist. For example, in the network, there is an interaction between genes v_1 and v_2. Fig. 1(B) represents the expression or repression conditions of genes observed by using microarrays. (For example, we regard a gene to be expressed when the gene expression level in a shocked sample is more than twice that in an unshocked or time-zero sample.) For the sake of simplicity, we explain only the expression conditions in the running example. In this example, gene v_1 is expressed under the following four conditions: i_1, i_2, i_3, and i_4.



Genes	Expression Conditions
v_1	$\{i_1,i_2,i_3,i_4\}$
v_2	$\{i_1,i_5\}$
v_3	$\{i_1,i_2,i_4\}$
v_4	$\{i_1,i_2,i_3,i_5\}$
v_5	$\{i_1,i_3,i_4\}$
v_6	$\{i_1,i_3,i_5\}$

Fig. 1. Examples of Common Pattern Graph

On the basis of the network and the expression conditions, we identify active subnetworks, whose members are connected to each other and share the expression conditions. Figs. 1(C) and (D) show two such networks, in which the connected subgraphs G_C and G_D, denoted by the bold lines share the expression conditions $\{i_1, i_2\}$ and $\{i_1, i_3\}$, respectively ($\{i_1, i_2\}$ implies "i_1 and i_2"). Thus, the genes in G_C share the expression conditions i_1 and i_2.

However, it is difficult to identify such subnetworks because the number of subnetworks and expression conditions increases exponentially according to the size of a given network and the number of conditions, respectively. To overcome this difficulty, we enumerate graphs using depth-first search (DFS) method and introduce pruning techniques using expressed conditions size property. We apply our algorithm to yeast protein-protein interactions and gene expression profiling. The results reveal the automatic annotation of

This work was supported by KAKENHI (Grant-in-Aid for Scientific Research) on Priority Areas "Systems Genomics" from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

M. Seki and J. Sese are with Department of Computer Science, Ochanomizu University, 2-1-1 Otsuka, Bunkyo, Tokyo, 112-8610, Japan.{seki,sesejun}@sel.is.ocha.ac.jp

the protein interaction network along with the expression conditions. We confirm the annotation by comparing the results with the Gene Ontology (GO) [6] annotations and the KEGG pathways [7]. Our method identifies that the subcomponents of proteasome are activated under oxidative stress conditions, and confirms the relation between heat shock and cytoskeletons.

The rest of this paper is organized as follows. Section II presents the formal definition of our subnetwork identification problem and the DFS method along with the pruning technique. Then, we introduce a novel algorithm d-COPINE in Section III. Section IV presents the experimental results obtained using our algorithm in yeast protein-protein interactions along with the expression data. We conclude the paper in Section V.

II. Method

In this section, we present the formal definition of the problem and then introduce an effective enumeration technique.

A. Problem Definition

Let G be an undirected, unlabelled, unweighted graph whose each vertex has a set of items (an itemset). We designate this graph as the *common-expression graph (CE graph)*. Let $V(G)$, $E(G)$, $\mathcal{I}(G)$, and $I(v)$ respectively signify a set of vertices in G, a set of edges in G, a set of itemsets on vertices in G, and an itemset on $v \in V(G)$. For this description, $|G| = |E(G)|$ is the size of graph G. Figure 1(A) portrays an example of the CE graph. In this figure, $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E(G) = \{(v_1, v_2), (v_1, v_3), \ldots, (v_4, v_6)\}$. The size of G is eight. Each vertex in G has an itemset. For instance, $I(v_1) = \{i_1, i_2, i_3, i_4\}$ and $I(v_2) = \{i_1, i_5\}$.

Next, we define a graph whose vertices have common items.

Definition 1: (**CPG**) Let G' be a connected subgraph of the CE graph G. G' is also a CE graph. Define $I(G')$ as $\bigcap_{v \in V(G')} I(v)$. We designate $I(G')$ as a *common itemset of the graph G'*. When $I(G') \neq \phi$, we say that G' is a *common pattern graph (CPG)* with $I(G')$.

We present two examples of CPGs in Figures 1(B) and (C) as the bold lines, which have common itemsets $\{i_1, i_2\}$ and $\{i_1, i_3\}$, respectively. Note that the common itemset is determined only using the vertices, and not the edges of the graph. Therefore, we define $I(V(G)) = \bigcap_{v \in V(G')} I(v) = I(G)$.

The last equality is satisfied when $I(G_1) \subseteq I(G_2)$ or $I(G_1) \supseteq I(G_2)$. Only when the inclusion relation between the itemsets of two graphs exists, we consider that the graphs are connected. From this definition, G_B and G_C shown in Figures 1(B) and (C) are disconnected, although the two graphs share the vertices v_1 and v_4, and the edge (v_1, v_4).

If connected graphs exist, we are interested in the largest one. The following definitions of a closed CPG (CCPG) and a maximal CPG enable us to select the CPGs of interest. The terms, closed and maximal, are inspired from association rule mining researches.

Definition 2: (**CCPG and maximal CPG**) Given a CE graph G. Let G' be a CPG with I. When no edge $(v_1, v_2) \in G$, where $v_1 \in G'$ satisfies $I(V(G) \cup \{v_2\}) = I$, G' is called CCPG with I. We call CCPG whose size is the largest as the *maximal CPG*, and the N largest CCPGs as the N maximal CPGs.

In the running example, both G_B and G_C are CCPGs, and G_C is the maximal CPG because $|G_B| = 3$ and $|G_C| = 4$.

With this definition, we formalize the problem of finding the maximal CPG.

Definition 3: (**Finding the N maximal CPGs**) Given a CE graph G, a user-specified values N and θ. Compute the N maximal CPGs G' such that $|I(G')| \geq \theta$.

When $N = 10$, the problem of finding the largest CCPG demands the extraction of the 10 largest subgraphs that share a set of items. For simplicity, we discuss below only the case of $N = 1$. Then, we extend the discussion to handle the case of $N > 1$.

Unfortunately, it is difficult to compute the N largest subgraphs because the number of subgraphs increase exponentially with the size of a graph. In particular, if the size of CPG is large, we might generate all the subgraphs of the maximal CPG even when we can avoid the generation of uninformative subgraphs. To overcome this difficulty, we introduce a DFS itemset tree and pruning techniques on the tree.

B. Enumeration of Common Pattern Graphs

For enumeration of subgraphs, two different approaches exist: April-like generations (breadth-first search) [8]–[10] and a pattern-growth approach (DFS) [11], [12]. We design a novel algorithm COPINE using the pattern-growth approach. In this section, we introduce a DFS itemset tree, which enables us to prune subgraphs whose common itemset sizes are less than θ.

In this subsection, we introduce the DFS itemset tree and the pruning technique using the DFS itemset tree. When performing the DFS in a graph, we construct a DFS itemset tree, whose node consists of a vertex of G and an itemset associated with the graph described by a path from the root to the node on the DFS tree. The tree is based on a DFS tree [13] containing all the subgraphs of G according to the DFS lexicographic order [11]. The DFS itemset tree does not contain edges because the common itemset of G can be calculated from $V(G)$ without $E(G)$. Because a large graph is suitable in finding the largest CCPG, we consider all the edges between vertices in G.

Definition 4: **(DFS itemset tree)** We use DFS to enumerate all the subgraphs in a given CE graph G. Let T be a DFS itemset tree and $n_1, n_2 \in T$ be associated with the CE graphs $G_1, G_2 \subset G$, respectively. In addition, n_2 is a child of n_1 on T if G_2 can be generated by adding a new vertex v to G_1. (at least one edge (v, v'), where $v' \in G_1$, is required.) In the DFS itemset tree, the node n_1 contains v and $I(G_1)$.

The root of T is *null*, and the children of the root represent single-vertex graphs.

Next, we examine the construction of the DFS itemset tree in Figure 2 from the graph in Figure 1(A). The starting node can be chosen randomly. We decide the order, designated by the symbol $\prec$, between vertices in advance. In the running example, we treat $v_i \prec v_j$ when $i < j$. We traverse the graph starting from the smallest vertex v_1. We add a node n_1 containing $v_1$to the DFS itemset tree T as the child of the root. The node n_1 is shown in Figure 2. This node is also associated with $I(v_1) = \{i_1, i_2, i_3, i_4\}$. Then, we add n_2 containing v_2, which is the smallest vertex connected to v_1. n_2 contains $I(v_1) \cap I(v_2) = \{i_1\}$.

In this tree, the path from the root to a node represents the order of visited vertices in the DFS itemset tree. For example, in Figure 2, the path from the root to n_2 is $\langle v_1, v_2 \rangle$ and indicates that the edges are visited in the order of v_1 following by v_2. Define $G(P)$ as a CE graph represented by a path P, and $I(P)$ as $I(G(P))$. Let P be $\langle v_1, v_3, v_4 \rangle$. $E(G(P)) = \{(v_1, v_3), (v_3, v_4)\}$, $V(G(P)) = \{v_1, v_3, v_4\}$ and $I(P) = I(v_1) \cap I(v_3) \cap I(v_4) = \{i_1, i_2\}$.

Property 1: Let path P_1 be $\langle v_1^1, v_2^1, \ldots, v_d^1 \rangle$. Let us add v_{d+1}^1 to P. If an edge $(v_j^1, v_{d+1}^1)(j = 1, 2, \ldots, d)$ exists, any edge $(v_j^1, v_k^1)(k = 1, 2, \ldots, d)$ satisfies $v_j^1 \prec v_{d+1}^1$.

Property 2: Let the paths $P_1 = \langle v_1^1, \ldots, v_{i-1}^1, v_i^1, \ldots \rangle$ and $P_2 = \langle v_1^1, \ldots, v_{i-1}^1, v_i^2, \ldots \rangle$. If P_1 is traversed before P_2, $v_i^1 \prec v_i^2$.

These properties are the specific cases of the DFS lexicographic order introduced in gSpan [11]. These properties allow us to traverse all the subgraphs by using a unique path. For example, we can generate graph containing v_1, v_3, and v_4 by using $\langle v_1, v_3, v_4 \rangle$ or $\langle v_1, v_4, v_3 \rangle$. However, with Property 1 and the existence of the edge (v_3, v_4), we construct only $\langle v_1, v_3, v_4 \rangle$.

Although the ordering of vertices obviates the generation of duplicate subgraphs, visiting all subgraphs still imparts a high cost. The following property enables us to prune the subtree whose common itemset size is less than the user-specified threshold θ.

Property 3: Given a CE graph G. Let G_p and G_c be CE graphs, $G_p, G_c \subset G$, and $V(G_p) \subset V(G_c)$. Then, $I(G_p) \supseteq I(G_c)$.

Proof: Let $V' = V(G_c) - V(G_p)$. For any $v \in V'$ $I(V(G_p) \cup \{v\}) = I(G_p) \cap I(v) \subseteq I(G_p)$. Therefore, $I(G_p) \supseteq I(G_p) \cap \bigcap_{v \in V'} I(v) = I(G_c)$. Consequently, $I(G_p) \supseteq I(G_c)$. ■

From Property 3, the sizes of itemsets have a monotonic property of the DFS itemset tree: if the common itemset size in a node n of the DFS itemset tree is less than θ, the common itemset size in any descendant node n' of n is less than θ.

In the running example, let $\theta = 2$. The itemset on n_1 in Figure 2 is $\{i_1\}$ and its size is $1 < \theta$. Therefore, the common itemset size on any descendant of n_1 is less than 2. On the other hand, we must consider the children of n_2 because its common itemset size is $2 \geq \theta$.

Even if we can prune subgraphs by using the above properties, we might traverse the same edge many times. In the running example, v_4 in Figure 1 might be used in the different paths $\langle v_1, v_3, v_4 \rangle$, $\langle v_1, v_4, v_6 \rangle$, $\langle v_3, v_4, v_1 \rangle$, etc. To avoid the duplication of traversal, we introduce the following property.

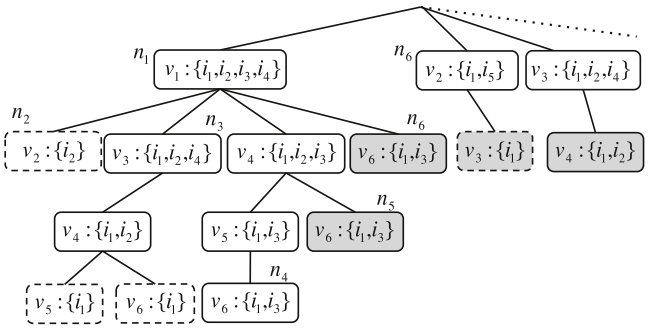


Fig. 2. DFS Itemset Tree (dashed and gray nodes are pruned by Properties 3 and Theorem 1, respectively)

Theorem 1: Let n_1 and n_2 be the nodes of the DFS itemset tree and n_1 be generated before n_2. If both n_1 and n_2 contain v and $I(n_1) \supseteq I(n_2)$, no CCPG exists in a descendant of n_2.

Proof: Omitted ■

This property implies that if we visit an already visited node v and the common itemset of the current path is the same or a subset of a previously visited itemset on v, we can prune the subtree rooted by the current node in the DFS itemset tree. Therefore, this property enables us to avoid unnecessary subgraph exploration.

In the running example, presume that we visit n_5 in Figure 2. $I(\langle root \rightarrow n_5 \rangle) = \{i_1, i_3\}$ and $\{i_1, i_3\}$ is generated at v_6. From Theorem 1, we can prune the subtree rooted by n_5. The subtree rooted by n_6 also can be pruned. This pruning reduces the search space considerably. The gray boxes in Figure 2 are the nodes pruned by the Theorem 1.

C. Disconnected Networks

The enumerated CCPGs might include overlapped graphs in which most of the edges and conditions are overlapped. The overlapping of networks is often caused by the replacement between highly correlated items. For example, suppose that a CCPG is associated with $\{i_1, i_2, i_3\}$, and i_3 and i_4 are highly related sample. In this situation, the CCPG would be similar to CCPG associated with $\{i_1, i_2, i_4\}$. To select disconnected networks from the computed CCPGs, we introduce a heuristic.

Let $overlap(G_1, G_2)$ be defined as

$$\frac{|E(G_1) \cap E(G_2)|}{\min\{|E(G_1)|, |E(G_2)|\}} \times \frac{|I(G_1) \cap I(G_2)|}{\min\{|I(G_1)|, |I(G_2)|\}},$$

where $|E(G_1)|$ and $|I(G_1)|$ are the number of edges in G_1 and itemset size of $I(G_1)$, respectively. When G_1 and G_2 have no common edges or samples, $overlap(G_1, G_2)$ is zero. On the other hand, when G_1 is included in G_2, $overlap(G_1, G_2)$ is close to 1. We define a new parameter, the overlap ratio r. We identify that network G_1 and G_2 is

Algorithm 1 d-COPINE

Require: CE-graph $G = (V(G), E(G), \mathcal{I}(G))$, user-specified threshold θ and, t

1: $V \leftarrow V(G)$
2: remove nodes from V whose itemset size is less than θ.
3: $T \leftarrow$ *null* // DFS itemset tree
4: $\mathcal{N}, \mathcal{C} \leftarrow$ *null* // candidate CCPGs and disconnected CCPGs
5: // generate subgraphs from every nodes in $V(G)$
6: **for** each $v \in V$ **do**
7: $V' \leftarrow V$
8: $\mathcal{N} \leftarrow \mathcal{N} \cup$COPINE-VISIT ($V'$, v, $root$)
9: $V \leftarrow V - v$
10: **end for**
11: $\mathcal{N} \leftarrow$ sort $\mathcal{N}$ by size in descending order
12: **for** each $N \in \mathcal{N}$ **do**
13: **if** $\exists C \in \mathcal{C}$ satisfying $overlap(N, C) > r$ **then**
14: next;
15: **end if**
16: $\mathcal{C} \leftarrow \mathcal{C} \cup \{N\}$
17: **end for**

Algorithm 2 COPINE-VISIT(V, v, n)

1: $C \leftarrow$ all v's neighbors in V
2: **if** C is *null* **then**
3: add a graph associated with n to $\mathcal{L}$
4: return
5: **end if**
6: **for** each $c \in C$ **do**
7: next if c does not satisfy DFS lexicographic order
8: $I \leftarrow$ generate an intersection of $I(c)$ and $I(n)$
9: // Pruning using itemset size (Property 3)
10: **if** $|I| < \theta$ **then**
11: next
12: **end if**
13: // Pruning with Theorem 1
14: **if** $I(c)$ is already traversed on c **then**
15: next
16: **end if**
17: $n' \leftarrow$ generate new node containing c and add it to n.
18: $V \leftarrow V - c$
19: $n' \leftarrow$ COPINE-VISIT(V, c, n')
20: **end for**
21: **if** n has no child or no child has the same itemset of n **then**
22: add a graph associated with n to $\mathcal{L}$
23: **end if**

overlapped when $overlap(G_1, G_2) > r$. Using this heuristics, we can identify important networks easily.

III. The d-COPINE Algorithm

Next, we formulate our algorithm called disconnected COmmon Pattern Itemset NEtwork identification (*d-COPINE*), which is based on the DFS on a given graph and uses DFS itemset tree.

Algorithm 1 sets the pseudo-code of the framework. Lines 1–3 initialize the variables. From line 4 to 9, we shrink the vertex set V by removing the vertex after all the subgraphs including the vertex have been searched. Lines 10–16 presents the identification of disconnected CCPGs.

COPINE-VISIT generates the DFS itemset tree recursively. Lines 6–20 are the main generation step of the DFS itemset tree. We can prune the subgraph whose itemset size is less than θ at line 10 on the basis of Property 3. At line 14, we check the subtree on the basis of Theorem 1.

IV. Experiments

We have investigated the effectiveness and usefulness of the d-COPINE approach using protein-protein interaction networks [14]–[16] consisting of 7,564 interactions and gene expressions including 6,152 genes under 173 types of stress conditions [17]. We convert the numerical expressions into Boolean expressions by using thresholds. We denotes E as the threshold. In all experiments, we have set the overlap ratio threshold $r = 0.1$.

To check the expression networks, we use three different expression thresholds, $E = 1.0, 1.5, 2.0$. When the value of the expression is more than E, we consider that the expression is over-expressed. To detect the repression networks, we use three different thresholds, $E = -1.75, -2.0, -2.25$. When the value of the expression is less than E, we consider that the expression is repressed. We compute CCPGs having more than 20 edges for three different minimum condition (itemset) size, $\theta = 3, 5$ and 7. Table I lists the result including the number of disconnected networks, edges, and genes for every parameter. For example, when we compute CCPGs with $E = 1.0$ and $\theta = 3$, this table shows us that the result consists of 20 disconnected networks containing 411 genes and 559 interactions. This table shows us that d-COPINE can extract disconnected networks from a real biological dataset. In contrast with positive threshold of E, when E is negative, average numbers of genes and edges in the networks are large. The genes in the large networks are repressed under heat shock conditions and highly associated with prune/pyrimidine metabolisms (data not shown).

We next investigate whether d-COPINE can be used to annotate protein-protein networks correctly. To annotate time- or temperature-dependent network, we use $\theta = 3$ and $E = 1.0$ since groups of time-course samples contain 5 or 10 types of conditions and approximately 5% of the observed genes and conditions are over expressed when $E = 1.0$.

To evaluate the performance of d-COPINE, we compare our network with the GO [6] and KEGG [7] database. We use second level groups from the root of each GO category (GO terms are classified into three categories and forms directed acyclic graph structure). We use the GO terms and KEGG pathways associated with less than 500 genes.

Table II lists the details of the 5 largest disconnected CCPGs, which includes the number of edges, number of genes, conditions of common patterns, GO terms or KEGG pathways most associated with the network in terms of binomial tests, and p-value of the binomial test. Visualization of the networks are shown in Fig. 3. Upper center of the figure shows all the protein-protein interactions. 5 largest disconnected CCPGs in Table II are magnified and colored. Red, green, orange, blue, purple networks are related to 1st to 5th largest CCPGs, respectively. The figure include five expression level graphs. Each of the graph associated with one of the five largest CCPGs. Colors in the expression level graphs are related to the colors in 5 largest CCPGs. In the

TABLE I

NUMBER OF CLOSED COMMON PATTERN NETWORK WITH YEAST STRESS EXPRESSIONS

Threshold		$\theta = 3$			$\theta = 5$			$\theta = 7$		
E	avg. size	# of nets	# of genes	# of edges	# of nets	# of genes	# of edges	# of nets	# of genes	# of edges
1.0	10.1	20	411	559	13	173	236	5	59	80
1.5	4.78	2	41	41	0	0	0	0	0	0
2.0	2.65	0	0	0	0	0	0	0	0	0
−1.75	4.21	17	587	819	9	285	406	11	227	301
−2.0	3.03	14	338	466	9	198	268	9	148	181
−2.25	2.23	11	232	307	7	119	154	6	86	105

TABLE II

5 LARGEST DISCONNECTED CLOSED COMMON PATTERN GRAPHS ($E = 1.0, \theta = 3$)

Rank	# of edges	# of genes	Common Expression Conditions	GO/KEGG	p-values
1	66	15	1.5 mM diamide (30 min),1.5 mM diamide (50 min), 1.5 mM diamide (60 min)	Proteasome (KEGG)	9.07e-20
2	58	55	Nitrogen Depletion 1 d,Nitrogen Depletion 3 d, Nitrogen Depletion 5 d	Autophagy (GO)	1.07e-08
3	50	48	Heat Shock 17 to 37, 20 min., Heat Shock 21 to 37, 20 min., Heat Shock 25 to 37, 20 min.	structural constituent of Cytoskeleton (GO)	6.53e-03
4	46	11	DBY7286 + 0.3 mM H2O2 (20 min), DBYmsn2msn4 (good strain) + 0.32 mM H2O2, DBYyap1- + 0.3 mM H2O2 (20 min)	Proteasome (KEGG)	1.35e-16
5	45	13	DBY7286 + 0.3 mM H2O2 (20 min), DBYmsn2msn4 (good strain) + 0.32 mM H2O2, DBYyap1- + 0.3 mM H2O2 (20 min)	Proteasome (KEGG)	1.35e-16

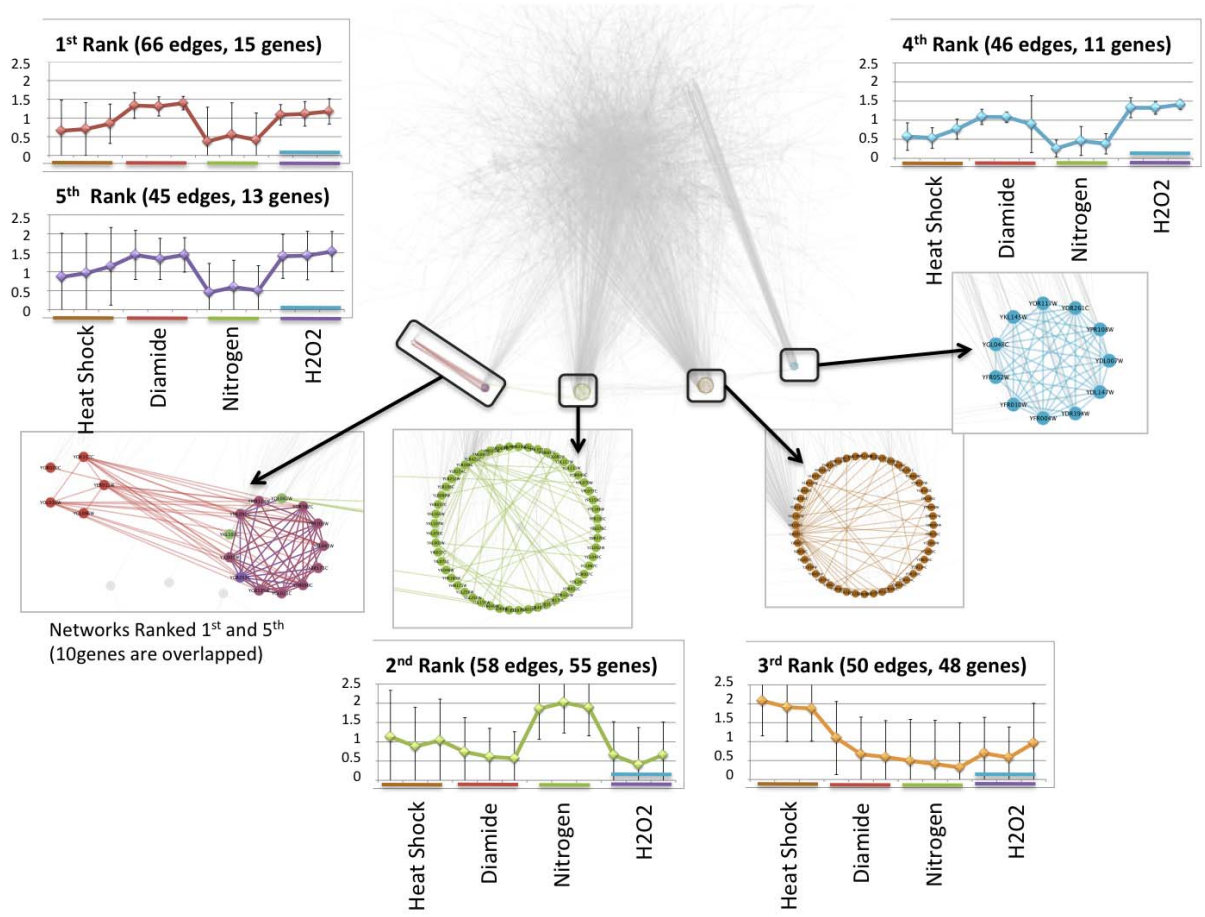


Fig. 3. Visualization of the 5 Largest Disconnected CCPGs in Table II and Expression Levels of Related Genes

graph, we display average values and standard deviations of gene expression levels. We only show 12 from 173

conditions, which are associated with the 5 largest CCPGs. For example, this figure indicates that genes in 2nd rank network colored in green are significantly up-regulated under three nitrogen conditions.

CCPG ranked 1st containing 15 genes and 66 edges is related to three diamide conditions. From this result, we infer that the network is closely associated with a high density of diamide (a sulfhydryl oxidizing agent). Note that we do not make groups of conditions in advance. The closely related conditions are selected by our method automatically. The GO or KEGG category most closely associated with the network is the proteasome complex in KEGG. All the 15 genes are the components of the proteasome protein complex. Proteasome is one of the well-known complexes responding to oxidative stress. Therefore, the conditions associated with the network detected by d-COPINE and the known functions of genes are in perfect harmony.

In network ranked 3rd, since all the common expression conditions of the network are heat shocks from low to high temperature, we can infer that the network would be associated with heat shock stress. The network is associated with the structural constituent of the cytoskeleton in the GO category. In [18] and [19], it has been reported that there is an association between heat-shock proteins and the maintenance of actin cytoskeletons. Therefore, the relation between the network and the GO term can be correct. Graph clustering methods are difficult to find this functional module network because the network has low density. Network ranked 2nd also has low density. From these result, d-COPINE has possibility to reveal unknown networks.

Note that the common expression conditions of the networks ranked 4th and 5th are the same, all of which are observed under oxidative stress conditions. Because d-COPINE eliminates connected networks, these two networks are disconnected. (The networks are blue and purple networks in Fig. 3.) We examined in detail the result in the Saccharomyces Genome Database annotation [20]. seven out of 11 genes in ranked 4th network are the components of 19S subunit of 26S proteasome. Ten out of 12 genes in the network ranked 5th are components of 20S subunit of 26S proteasome. Therefore, the two networks are totally different, but genes in these networks are expressed under the oxidative stress conditions.

The networks ranked 1st and 5th share 10 genes. The gene PRE6/YOL038W is present in the network ranked 1st, but not in the network ranked 5th; it relocates from cytosol to the mitochondrial surface due to a kind of oxidative stress. The d-COPINE result might reveal the the type of stress.

V. Concluding Remarks

We have presented a novel computational algorithm d-COPINE, which simultaneously identifies the location and status of networks. from gene expressions and protein interaction data. Our approach is based on the detection of large interaction networks whose genes share expression or repression conditions. We have evaluated our method by using yeast protein-protein interactions and microarray expression data. The experimental results show that d-COPINE can extract networks associated with specific conditions; these networks are closely related to the GO categories or KEGG pathways. d-COPINE has identified the components of a proteasome complex along with its activating conditions and the relation between heat shock and cytoskeleton automatically. The incorporation of other protein networks and expression profiles in GEO [21] can be used to annotate which parts are activated in cells and when the parts are activated.

References

[1] A. L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[2] Y. Cheng and G. M. Church, "Biclustering of expression data," in *ISMB*, 2000, pp. 93–103.

[3] W. Pan, "Incorporating gene functions as priors in model-based clustering of microarray gene expression data," *Bioinformatics*, vol. 22, pp. 795–801, 2006.

[4] M. Shiga, I. Takigawa, and H. Mamitsuka, "Annotating gene function by combining expression data with a modular gene network," vol. 23, pp. i468–i478, 2007.

[5] E. Zeng, C. Yang, T. Li, and G. Narasimhan, "On the effectiveness of constraints sets in clustering genes," in *BIBE*, 2007.

[6] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler *et al.*, "Gene ontology: tool for the unification of biology. the gene ontology consortium." *Nat Genet*, vol. 25, no. 1, pp. 25–29, May 2000.

[7] M. Kanehisa and S. Goto, "KEGG: Kyoto Encyclopedia of Genes and Genomes," *Nucleic Acids Research*, vol. 28, no. 1, pp. 27–30, 2000.

[8] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *PKDD '00*, 2000.

[9] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *ICDM*, 2001, pp. 313–320.

[10] S. Nijssen and J. N. Kok, "A quickstart in frequent structure mining can make a difference," in *KDD '04*, 2004, pp. 647–652.

[11] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *ICDM '02*, 2002, p. 721.

[12] J. Huan, W. Wang, J. Prins, and J. Yang, "Spin: mining maximal frequent subgraphs from graph databases," in *KDD '04*, 2004, pp. 581–586.

[13] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[14] T. Ito *et al.*, "Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins." *Proc Natl Acad Sci*, vol. 97, no. 3, pp. 1143–1147, 2000.

[15] P. Uetz *et al.*, "A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae," *Nature*, vol. 403, no. 6770, pp. 623–627, 2000.

[16] N. J. Krogan *et al.*, "Global landscape of protein complexes in the yeast saccharomyces cerevisiae," *Nature*, vol. 440, pp. 637–643, 2006.

[17] A. P. Gasch *et al.*, "Genomic expression programs in the response of yeast cells to environmental changes," *Mol. Biol. Cell*, vol. 11, no. 12, pp. 4241–4257, 2000.

[18] J. Gu, M. Emerman, and S. Sandmeyer, "Small heat shock protein suppression of vpr-induced cytoskeletal defects in budding yeast," *Mol. Cell. Biol.*, vol. 17, pp. 4033–4042, 1997.

[19] B. G. Leicht, H. Biessmann, K. B. Palter, and J. J. Bonner, "Small heat shock proteins of drosophila associate with the cytoskeleton," *Proc. Natl. Acad. Sci.*, vol. 83, pp. 90–94, 1986.

[20] J. Cherry, C. Adler, C. Ball *et al.*, "Sgd: Saccharomyces genome database," *Nucleic Acids Res*, vol. 26, pp. 73–79, 1998.

[21] T. Barrett, T. Suzek, D. Troup *et al.*, "Ncbi geo: mining millions of expression profiles–database and tools," *Nucleic Acids Res*, vol. 33, pp. D562–D566, 2005.