# Biclustering Protein Complex Interactions with a Biclique Finding Algorithm

Chris Ding
Lawrence Berkeley Nat'l Lab
Berkeley, CA94720
chqding@lbl.gov

Ya Zhang
University of Kansas
Lawrence, KS 66045
yazhang@ittc.ku.edu

Tao Li
Florida International University
Miami, FL 33199
taoli@cs.fiu.edu

Stephen R. Holbrook
Lawrence Berkeley Nat'l Lab
Berkeley, CA 94720
SRHolbrook@lbl.gov

## Abstract

*Biclustering has many applications in text mining, web clickstream mining, and bioinformatics. When data entries are binary, the tightest biclusters become bicliques. We propose a flexible and highly efficient algorithm to compute bicliques. We first generalize the Motzkin-Straus formalism for computing the maximal clique from $L_1$ constraint to $L_p$ constraint, which enables us to provide a generalized Motzkin-Straus formalism for computing maximal-edge bicliques. By adjusting parameters, the algorithm can favor biclusters with more rows less columns, or vice verse, thus increasing the flexibility of the targeted biclusters. We then propose an algorithm to solve the generalized Motzkin-Straus optimization problem. The algorithm is provably convergent and has a computational complexity of $O(|E|)$ where $|E|$ is the number of edges. Using this algorithm, we bicluster the yeast protein complex interaction network. We find that biclustering protein complexes at the protein level does not clearly reflect the functional linkage among protein complexes in many cases, while biclustering at protein domain level can reveal many underlying linkages. We show several new biologically significant results.*

## 1 Introduction

Biclustering refers to finding sub-blocks in a rectangular data matrix, such as a word-document matrix, a web clickstream matrix (users vs websites), DNA microarray expression profiles, or the protein - protein complex interaction network. Biclustering can be viewed as simultaneous data clustering and feature selection, i.e., detection of significant clusters and the features that are uniquely associated with them. This is because not all features are relevant to certain clusters, for example, only a subset of websites are responsible for a subset of web users in the clickstream problem. Thus one needs to combine feature selection and clustering together. The definition of biclusters varies significantly[19], but the most intuitive notion of biclusters are *dense regions* in the rectangle data matrix.

In many applications, such as web clickstream, phylogenetic tree dataset, frequent itemset finding, data entries are binary. In these cases, biclustering becomes biclique finding, because the simplest and most dense sub-blocks are bicliques. Li et al [18] show there is a correspondence between biclique and frequent closed itemset.

At present, the most fruitful application of biclustering is in bioinformatics. In the following, we will focus on the protein interaction network problem as the target application. The algorithms developed for cliques and bicliques are completely general and can be applied to all bipartite graphs.

### Maximal-edge Biclique

A bipartite graph has two types of nodes, a set $R$ of $r$-nodes and a set $C$ of $c$-nodes. In the adjacency matrix of the bipartite graph, $r$-nodes are represented by the rows and $c$-nodes are represented by the columns. A biclique is a subset $(R_1, C_1)$ where $R_1 \subset R$ and $C_1 \subset C$ such that every r-node in $R_1$ is connected to every c-node in $C_1$.

Let $B$ be the adjacency matrix of the bipartite graph. $B$ is a rectangle matrix of $|R|$ rows ($r$-nodes) and $|C|$ columns ($c$-nodes). A biclique $Z_1 = (R_1, C_1)$ implies that the matrix sub-block $B_{R_1,C_1}$ is a block of all 1's. A biclique $Z_1$ is a maximal biclique if there exists no larger bicliques of which $Z_1$ is a subset. A biclique $(R_1, C_1)$ has $|R_1| + |C_1|$ nodes (the sum of the number of rows and columns of the subblock) and $|R_1| * |C_1|$ edges (the area of the subblock).

Thus, there are two types of maximum bicliques: (a) maximum edge bicliques where the number of edges is maximum/complete, and (b) maximum node bicliques where the number of node is maximum/complete. From the adjacency matrix point of view, the maximum node biclique selects the sub-block with the largest perimeter, while the maximum edge biclique selects the sub-block with the largest area (see Figure 2 for illustration). From the application point of view, the maximum edge bicliques are the interesting subgraphs that we wish to compute.

### Biological meaning of Bicliques/Biclusters

Biclustering is used to find a subblock in the 2D gene expression data (a rectangle matrix), where rows represent genes and columns represent either tissue samples or experimental conditions. The rationale is that a protein is only expressed in (or changes expression in) a set of tissues (phenotypes). Therefore, when clustering tissues, there is no need to include all genes. Another motivation is that a protein may participate in several functions, and thus a mutually exclusive partitioning of proteins (as occurring in standard clustering) is not appropriate.

### Clique and Biclique Finding

It is known that the maximum clique finding and the maximum edge biclique finding are NP-hard[10]. It is also known that the maximum clique problem is hard to approximate: there exists an $\epsilon > 0$ such that no polynomial time algorithm can approximate the size within a factor of $n^{1-\epsilon}$ [9, 4, 14, 5]). Clique and biclique detecting algorithms typically enumerate all clique/bicliques[6, 8, 20, 1, 18] making use of the fact that a subset of a clique/biclique is still a clique/biclique. These algorithms typically have high order complexity.

### Outline of the paper

In this paper, we use the algorithmic approach of the Motzkin-Straus theorem that relates clique finding to the optimization of a quadratic function with $L_1$ constraints (see §5).

Our main contributions are described in §5 - §6. We first generalize the Motzkin-Straus formalism to use $L_p$ constraints (see §5). We then generalize the Motzkin-Straus formalism with $L_p$ constraints to bipartite graphs and derive an iterative algorithm to compute the solution (see §6). We prove the correctness and convergence of the algorithm.

For presentation purpose, we first describe the new biclique finding algorithm in §2. We give an illustrative example to show how the algorithm works in §3 and show that the algorithm can effectively find large bicliques in random graphs in §4.

In §8, we apply the biclique algorithm to protein complex interactions and discuss the data, procedure, and results in detail. Our results in protein complex interaction biclustering show that biclustering protein complexes at the protein domain level is able to reveal many functional linkages among protein complexes which are not observed when biclustering the protein complex data at protein level.

## 2 Biclique finding algorithm

In this section we outline the main biclique finding algorithm. In §3 - §4, we give an illustrative example and show the algorithm can find large bicliques in random graphs. Detailed analysis of the algorithm is explained in §5 - §6.

Let $B$ be the adjacency matrix of the bipartite graph (the input data matrix). $B$ is a rectangle matrix of $n = |R|$ rows and $m = |C|$ columns. Define the n-vector $\mathbf{x}$ on the r-nodes and the m-vector $\mathbf{x}$ on the c-nodes.

**Computing one biclique**.

Given initial $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)})$. [e.g., $\mathbf{x}^{(0)} = (1 \cdots 1)^T$ and $\mathbf{y}^{(0)} = (1 \cdots 1)^T$.] Set $\alpha = \beta = 1.1$. The iterative algorithm updates $\mathbf{x}, \mathbf{y}$ using:

$$x_i^{(t+1)} = \left( x_i^{(t)} \frac{(B\mathbf{y}^{(t)})_i}{[\mathbf{x}^{(t)}]^T B \mathbf{y}^{(t)}} \right)^{1/\alpha}, \qquad (1)$$

$$y_j^{(t+1)} = \left( y_j^{(t)} \frac{(B^T\mathbf{x}^{(t)})_j}{[\mathbf{x}^{(t)}]^T B \mathbf{y}^{(t)}} \right)^{1/\beta}. \qquad (2)$$

Let $\mathbf{x}^* = (x_1^*, \cdots, x_n^*)^T$ and $\mathbf{y}^* = (y_1^*, \cdots, y_m^*)^T$ be the converged solution. Let $R$ be the subset corresponding to nonzero elements in $\mathbf{x}$: $R = \{i \mid x_i^* > 0\}$. Let $C$ be the subset corresponding to nonzero elements in $\mathbf{y}$: $C = \{j \mid y_j^* > 0\}$. Then $(R, C)$ is a maximal biclique.

We can set $\alpha \neq \beta$ to favor different shapes of the computed maximal biclique. For example , if we set $\alpha > \beta$, this favors maximal bicliques which has larger row sizes. In general, setting $\alpha, \beta = 1.05 \sim 1.1$ gives very tight cliques (perfect cliques). Setting $\alpha, \beta = 1.2$ or larger gives loose cliques, i.e., cliques with some edges missing. This could be useful for some applications.

The algorithm is extremely simple to implement. The most time consuming part is matrix-vector multiplication which fit cache-based microprocessor architecture and thus runs very efficiently (in comparison to many graph algorithms which constantly encounter pointer-chasing and runs much less efficently due to cache miss).

**Computing multiple bicliques**.

Computing different bicliques are achieved by setting different initial starts: $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)})$. We have two approaches. (1) random starts. Setting $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)})$ to random

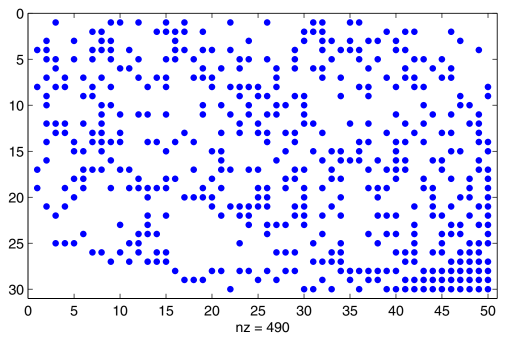

**Figure 1. The adjacency matrix $B$ for the bipartite graph. Each solid point at $(i, j)$ represent an edge exists between r-node $i$ and column-node $j$. The maximum biclique is at the lower-right corner.**

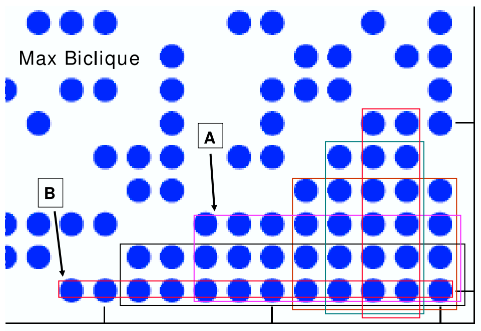

**Figure 2. Enlarged lower-right corner of the adjacency matrix in Fig.1. There are 6 maximal bicliques overlapping in this region as indicated by different rectangle boxes. The algorithm correctly picks up the the maximum-edge biclique $A$.**

uniform distributions. (2) Masking. After a clique $(R, C)$ is detected, we set the corresponding entries of $B$ to a small value.

## 3 An Illustrative Example

In all experiments/applications in this paper, we set $\beta = \alpha$ for simplicity.

The proposed algorithm is easily implemented. Here we report experiments on random graphs and bipartite graphs up to order of 1000. We first uses a bipartite graph of order 30x50 to illustrate some of concepts and features of the solution. In Figure 1, we show the adjacency matrix of the bipartite graph. The nonzero elements in $B$ represent edges.

In Figure 2, we show the maximum-edge biclique which is at the lower-right corner: $R_A = (28, 29, 30), C_A = (43, \cdots, 50)$ and $|R||C| = 24$. The algorithm correctly picks up this maximum-edge biclique. This is not a trivial task (even in this simple example), because there are 6 maximal bicliques overlapping in this region. We can easily verify that all other 5 maximal bicliques have $|R||C| < 24$.

We note in this region, the maximum-node biclique is clique $B$: $R_B = 30, C_B = (39, \cdots, 50)$. Thus $|R_B| + |C_B| = 13$. In comparison, the maximum-edge biclique $A$ has $|R_A| + |C_A| = 11$. Typically, maximum-node bicliques are long and narrow blocks, and thus not interesting from genomics point of view. Fortunately, our algorithm detects the more interesting maximum-edge bicliques.

In Figure 3, we show the the solution vector $\mathbf{x}^*$ at $\beta = 1.1$. The curve arise sharply at $R_1 = (28, 29, 30)$, indicating a maximal biclique there. The solution vector $\mathbf{y}^*$ at different $\beta = 1.1$ is shown in Figure 4. Clearly the curve arise sharply at $C_1 = (39, \cdots, 50)$, indicating a maximal biclique.

In Figure 4 we also show how the solution vector $\mathbf{y}^*$ varies at different $\beta = 1.1, 1.2, 1.3, 2$ . Starting at $\beta = 2$, $\mathbf{y}^*$ varies with no significant peak region. But at $\beta = 1.3, 1.2$, the solution becomes sharper and sharper, as expected. At $\beta = 1.1$ the solution is sharp enough to define the clique. At $\beta = 1.05$ the solution converges to the step function, the ideal solution. These can be seen from the enlarged detailed part at bottom panel.

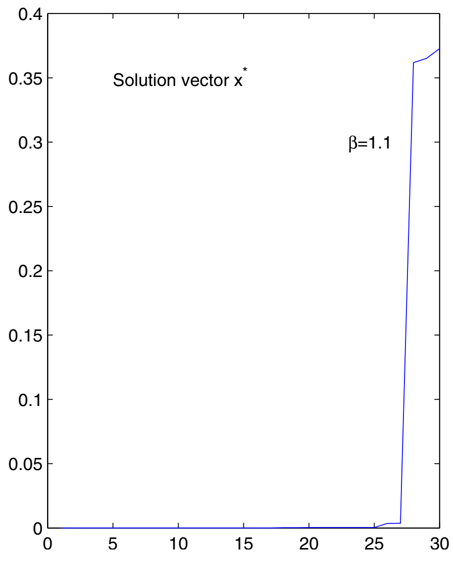

**Figure 3. Solution vector $x^*$ at $\beta = 1.1$.**

It is important to note that the maximum biclique $(R_1, C_1)$ overlaps with maximal biclique $(R_2, C_2)$. In Figures 1 and 2, the solution vectors converges to $(R_1, C_1)$ sharply, indicating the capability of the algorithm to identify the maximal cliques in overlapped situations.

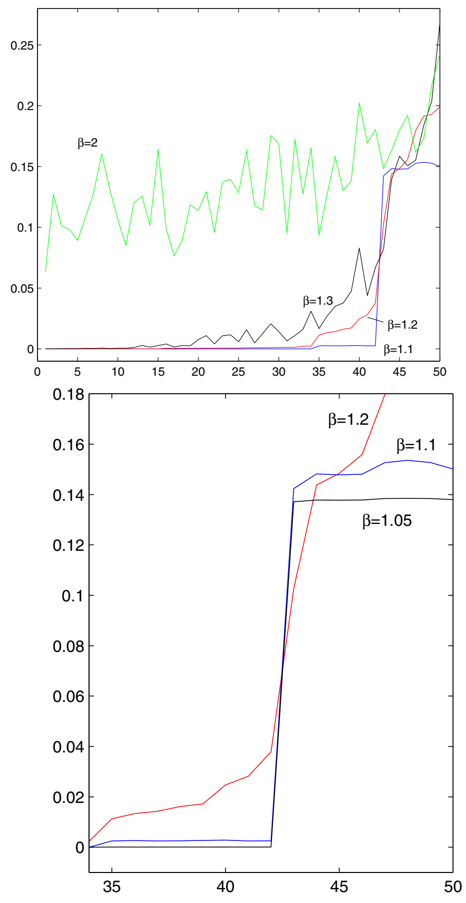

**Figure 4. Upper: Solution vector $y^*$ at $\beta = 2, 1.3, 1.2, 1.1$. Lower: Details of solution vector $y^*$ at $\beta = 1.2, 1.1, 1.05$.**

## 4 Detect Embedded Cliques

Next, we test the ability of the algorithm to detect maximum bicliques. We embed a known biclique into the standard random graphs (two nodes are joined with an edge with fixed probability $p = 0.3$). We vary the size of the embedded cliques, while fixing the rectangle matrix to be 500x1000. We set $\alpha = \beta = 1.05$. We tested on 20 bipartite graphs with randomly generated adjacency matrices. The embedded maximum bicliques are readily detected in all cases.

## 5 Motzkin-Straus formalism for clique finding

We approach the problem of detecting cliques through a theorem due to Motzkin and Straus [21] which relates maximal cliques to the optimization of a quadratic function. The use of the Motzkin and Straus theorem has been in a number of studies [22, 12].

Let $A$ be the adjacency matrix of an unweighted undirected graph. Computing the maximal cliques can be reformulated as the solution to the following optimization problem:

$$\max_{\mathbf{x}} \; \mathbf{x}^T A \mathbf{x} \quad s.t. \sum_{i=1}^{n} x_i = 1, \; x_i \geq 0, \tag{3}$$

where $A_{ii} = 0$.

**Theorem 1** [Motzkin and Straus]. Let $G$ be an unweighted graph. and $\mathbf{x}^*$ the optimal solution for the problem of Eq.(3). Let $C = \{i \mid x_i^* > 0\}$ be the subset corresponding to nonzero elements. If nonzero elements have the same values, $x_i^* = 1/|C|$ for $\forall i \in C$ (in this case $\mathbf{x}^*$ is called a characteristic vector of a subset $C$), $C$ is a maximal clique in $G$.

Note that the feasibility constraint is the $L_1$ norm. We generalize the Motzkin-Straus formulation in two directions. First we generalize the the feasibility constraint to use $L_p$ norm. For a vector $\mathbf{x}$, it is governed by

$$||\mathbf{x}||_p^p = \sum_{i=1}^{n} |x_i|^p.$$

As long as $p \simeq 1$, the optimal solution vector is sparse, i.e., many elements in $\mathbf{x}^*$ are zero. Using $L_1$ constraint to enforce sparsity is well-known in statistics as used in LASSO [23, 15].

Second, we generalize the Motzkin-Straus formulation to bipartite graph. This enables us to use it to compute bicliques.

## 6 Generalized Motzkin-Straus Formalism for bicliques

Let $G(B)$ be a bipartite graph with a set $R$ of $r$-nodes and a set $C$ of $c$-nodes. Let $B$ be the adjacency matrix of $G(B)$. $B$ is a rectangle matrix of $n = |R|$ rows ($r$-nodes) and $m = |C|$ columns ($c$-nodes).

A maximal biclique is computed via the solution to the following optimization problem:

$$\max_{\mathbf{x} \in F_x^\alpha, \, \mathbf{y} \in F_y^\beta} \mathbf{x}^T B \mathbf{y} \tag{4}$$

where the feasible regions of $\mathbf{x}$ is defined by

$$F_x^\alpha = \{x \in \mathcal{R}^n | \sum_{i=1}^{n} x_i^\alpha = 1, \; x_i \geq 0\}. \tag{5}$$

where $\alpha = 1+\epsilon,\ 0<\epsilon \ll 1$, and the feasible regions of $\mathbf{y}$ is defined by

$$F_y^\beta = \{\mathbf{y} \in \mathcal{R}^m |\ \sum_{j=1}^m y_i^\beta = 1,\ y_j \ge 0\}. \quad (6)$$

Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution, $R_1 = \{i \mid x_i^* > 0\}$ be the subset of nonzero elements in $\mathbf{x}^*$, and $C_1 = \{j \mid y_j^* > 0\}$ be the subset of nonzero elements in $\mathbf{y}^*$.

**Theorem 2** [Generalized Motzkin-Straus Theorem for bipartite graph]. Let $\alpha = 1+\epsilon_1, 0 < \epsilon_1 \ll 1$. and $\beta = 1+\epsilon_2, 0<\epsilon_2 \ll 1$. For an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$, if nonzero elements of $\mathbf{x}^*$ have same values, and if nonzero elements of $\mathbf{y}^*$ have same values, then $(R_1, C_1)$ is a maximum edge biclique in $B$. The objective function has the optimal value $J = |R_1|^{1-1/\alpha}|C_1|^{1-1/\beta}$.

**Proof**. Assume subblock $B_{R_1,C_1}$ is a maximal biclique. $\mathbf{x}^*$ must have the form

$$\mathbf{x}^* = (1/|R_1|^{1/\alpha})(1\cdots 1, 0\cdots 0)^T,$$

if we index the $r$-nodes of $R_1$ first. $\mathbf{y}^*$ must have the form

$$\mathbf{y}^* = (1/|C_1|^{1/\beta})(1\cdots 1, 0\cdots 0)^T,$$

if we index the $c$-nodes of $C_1$ first. The objective becomes

$$J = (\mathbf{x}^*)^T B \mathbf{y}^* = |R_1|^{1-1/\alpha}|C_1|^{1-1/\beta}.$$

Because $1-1/\alpha > 0$ and $1-1/\beta > 0$, $|C_1|$ and $|R_1|$ are simultaneously maximized. ■

**Remark**. We can adjust $\alpha, \beta$ to favor different shapes of the computed maximal biclique. For example , if $\alpha > \beta$, then $1-1/\alpha > 1-1/\beta$. This favors maximal bicliques which has larger row sizes[1]. For example, if there exists two maximal cliques $A, B$ in the graph with same number of edges, i.e., $|R_A||C_A| = |R_B||C_B|$. But if $A$ has large row size: $|R_A| > |R_B|$, then $|R_A|^{1-1/\alpha}|C_A|^{1-1/\beta} > |R_B|^{1-1/\alpha}|C_B|^{1-1/\beta}$ . This implies $A$ will be computed first.

## Algorithm for computing biclique

We provide an iterative algorithm to compute the maximum biclique. We prove its feasibility, correctness, and convergence. Given initial $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)})$, the iterative algorithm updates $\mathbf{x}, \mathbf{y}$ using:

$$\left[x_i^{(t+1)}\right]^\alpha = x_i^{(t)} \frac{(B\mathbf{y}^{(t)})_i}{[\mathbf{x}^{(t)}]^T B \mathbf{y}^{(t)}}, \quad (7)$$

$$\left[y_j^{(t+1)}\right]^\beta = y_j^{(t)} \frac{(B^T\mathbf{x}^{(t)})_j}{[\mathbf{x}^{(t)}]^T B \mathbf{y}^{(t)}}, \quad (8)$$

[1]In DNA gene expressions, this means favoring biclusters with more genes instead of tissue samples. We may also set $\alpha < \beta$ to favor biclusters with more tissue samples than genes.

This is the same update rules of Eqs.(1,2).

**Feasibility**

Clearly, $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})$ remain in feasible region for all $t > 0$, because

$$\sum_i [x_i^{(t+1)}]^\alpha = \sum_i \frac{x_i^{(t)}(B\mathbf{y}^{(t)})_i}{[\mathbf{x}^{(t)}]^T B \mathbf{y}^{(t)}} = 1.$$

and

$$\sum_i [y_i^{(t+1)}]^\beta = \sum_i \frac{y_i^{(t)} B^T\mathbf{x}^{(t)})_i}{[\mathbf{x}^{(t)}]^T B \mathbf{y}^{(t)}} = 1.$$

**Correctness**

We solve the constraint optimization problem. The Lagrangian function incorporating constraint is

$$L_2 = \mathbf{x}^T B \mathbf{y} - \lambda(\sum_i x_i^\alpha - 1) - \zeta(\sum_j y_j^\beta - 1) \quad (9)$$

where the Lagrangian multiplier $\lambda$ is for enforcing $\mathbf{x}$ in $F_x^\beta$, and the Lagrangian multiplier $\zeta$ is for enforcing $\mathbf{y}$ in $F_y^\beta$. The nonnegativity of $\mathbf{x}$ is enforced by the complementarity condition, $\frac{\partial L}{\partial x_i} x_i = 0$, which leads to

$$\left[(B\mathbf{y}^*)_i - \lambda\alpha[x_i^*]^{\alpha-1}\right] x_i^* = 0. \quad (10)$$

Summing over index $i$, we obtain

$$\lambda = [\mathbf{x}^*]^T B \mathbf{y}^* / \alpha. \quad (11)$$

The nonnegativity of $\mathbf{y}$ is enforced by the complementarity condition, $\frac{\partial L}{\partial y_j} y_j = 0$, which leads to

$$\left[(B^T\mathbf{x}^*)_i - \zeta\beta[y_j^*]^{\beta-1}\right] y_j^* = 0. \quad (12)$$

Summing over index $j$, we obtain

$$\zeta = [\mathbf{x}^*]^T B \mathbf{y}^* / \beta. \quad (13)$$

We can easily verify that (a) the converged solution for $\mathbf{x}$ from Eq.(1) satisfies the KKT condition Eq.(10), using Eq.(11). (b) the converged solution for $\mathbf{y}$ from Eq.(2) satisfies the KKT condition Eq.(12). These ensure the correctness of the algorithm.

**Convergence**

**Theorem 3**. Under the update rule for $\mathbf{x}, \mathbf{y}$, the Lagrangian function $L_2$ of Eq.(9) is monotonically increasing (non-decreasing),

$$L_2(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}) \le L_2(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}) \le L_2(\mathbf{x}^{(2)}, \mathbf{y}^{(2)}) \le \cdots.$$

Since $J$ is bounded from above, the convergence of the iterative algorithm is thus established.

**Proof**. We use the auxiliary function approach. A function $G(\mathbf{x}, \mathbf{x}')$ is an auxiliary function of $L(\mathbf{x})$ if

$$G(\mathbf{x}, \mathbf{x}') \le L(\mathbf{x});\ G(\mathbf{x}, \mathbf{x}) = L(\mathbf{x}). \quad (14)$$

Now, we define

$$\mathbf{x}^{(t+1)} = \arg\max_{\mathbf{x}} G(\mathbf{x}, \mathbf{x}^{(t)}). \quad (15)$$

By construction, we have $L(\mathbf{x}^{(t)}) = G(\mathbf{x}^{(t)}, \mathbf{x}^{(t)}) \le G(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}) \le L(\mathbf{x}^{(t+1)})$. This proves the monotonicity of $L(\mathbf{x}^{(t)})$. under the iterative updating rule of Eq.(15).

We first note that

$$\mathbf{x}^T B \mathbf{y} = \sum_{ij} x_i B_{ij} y_j \ge \sum_{ij} x_i' S_{ij} y_j' (1 + \log \frac{x_i y_j}{x_i' y_j'})$$

because $\frac{x_i y_j}{x_i' y_j'} \ge \left(1 + \log \frac{x_i y_j}{x_i' y_j'}\right)$, for $x_i, y_j, x_i', y_j' > 0$ using the inequality of $z \ge 1 + \log(z)$ for any positive $z$. Therefore,

$$G(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{y}') =$$

$$\sum_{ij} x_i' S_{ij} y_j' (1 + \log \frac{x_i y_j}{x_i' y_j'}) - \lambda(\sum_i x_i^\alpha - 1) - \zeta(\sum_j y_j^\beta - 1)$$

is a auxiliary function of $L_2(\mathbf{x}, \mathbf{y})$, because $G(\cdot, \cdot)$ satisfies the condition Eq.(14): The first term in $L_2$ is always greater than or equal to the first term in $G$. The equality holds when $\mathbf{x} = \mathbf{x}', \mathbf{y} = \mathbf{y}'$. The second and third terms are identical in $G$ and $L_2$.

The maximum for Eq.(15) is obtained by setting the gradient to zero,

$$\frac{\partial G(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')}{\partial x_i} = \frac{x_i' (B\mathbf{y}')_i}{x_i} - \lambda \alpha x_i^{\alpha - 1} = 0. \quad (16)$$

and

$$\frac{\partial G(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')}{\partial y_i} = \frac{y_i' (B^T\mathbf{x}')_i}{y_i} - \zeta \beta y_i^{\beta - 1} = 0. \quad (17)$$

These two equations can be re-written as

$$[x_i]^\beta = x_i' \frac{(B[\mathbf{y}']^{(t)})_i}{\lambda \alpha},$$

$$[y_i]^\beta = y_i' \frac{(B^T[\mathbf{x}']^{(t)})_i}{\zeta \beta},$$

According to Eq.(15), the variable relations for $\mathbf{x}$ are $\mathbf{x}^{(t)} \leftarrow \mathbf{x}'$ and $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}$. Substituting the Lagrangian multiplier $\lambda$ value of Eq.(11), we recover the update rule Eq.(1). Similarly, the variable relations for $\mathbf{y}$ are $\mathbf{y}^{(t)} \leftarrow \mathbf{y}'$ and $\mathbf{y}^{(t+1)} \leftarrow \mathbf{y}$. Substituting the Lagrangian multiplier $\zeta$ value of Eq.(13), we recover the update rule Eq.(2).

To ensure the solution gives a maxima, we show that the second order derivatives (Hessian matrix) of $G$ are negative definite. We have

$$\frac{\partial^2 G(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')}{\partial x_i \partial x_j} = -[\frac{x_i' (B\mathbf{y}')_i}{x_i^2} + \lambda \alpha (\alpha - 1) x_i^{\alpha - 2}] \delta_{ij}.$$

$$\frac{\partial^2 G(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')}{\partial y_i \partial y_j} = -[\frac{y_i' (B^T\mathbf{x}')_i}{y_i^2} + \zeta \beta (\beta - 1) x_i^{\beta - 2}] \delta_{ij}.$$

$$\frac{\partial^2 G(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')}{\partial x_i \partial y_j} = 0.$$

Because $\alpha \ge 1$ and $\beta \ge 1$, this Hessian matrix is a diagonal matrix with negative quantities on the diagonals. Therefore $G(\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')$ is a concave function in $\mathbf{x}, \mathbf{y}$ and has a unique global maximum. This complete the proof of Theorem 3. ■

**Case** $\beta = 2$.

**Theorem 4**. For $\alpha = \beta = 2$, the optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ are given by the first singular vector pair of $B$. Let the singular value decomposition $B = \sum_{k=1} \sigma_k \mathbf{u}_k \mathbf{v}_k^T$. Then $\mathbf{x}^* = \mathbf{u}_1, \mathbf{y}^* = \mathbf{v}_1$.

**Proof**. Restricting to $\alpha = \beta = 2$. In this case, from Eqs.(11,13), $\lambda = \zeta$. Eqs.(10,12), become

$$x_i^* (B\mathbf{y}^*)_i = 2\lambda [x_i^*]^2, \ \ y_i^* (B^T\mathbf{x}^*)_i = 2\zeta [y_i^*]^2$$

Clearly, setting $\mathbf{x}^* = \mathbf{u}_1, \mathbf{y}^* = \mathbf{v}_1$ and $\lambda = \zeta = \sigma_1/2$ satisfy these KKT conditions. Furthermore, according to Perron Theorem, the elements of $\mathbf{u}_1, \mathbf{v}_1$ must be nonnegative since $B$ is nonnegative. ■

## 7 Generalized cliques on weighted graphs

Cliques are only defined for unweighted graphs. For a weighted graph with edge weight $W$, we may threshold the weights with a threshold $h$: setting $A_{ij} = 1$ if $W_{ij} \ge h$; $A_{ij} = 0$ otherwise. We run the algorithm to compute cliques on $A(h)$.

We may compute cliques for several threshold $h$'s: $h_1 > h_2 > \cdots > h_k$. The thresholded graphs $A(h)$ clearly satisfy $A(h_1) \subset A(h_2) \subset \cdots \subset A(h_k)$. Let $C(h)$ represents the set of cliques on graph $A(h)$. Then we have $C(h_1) \subset C(h_2) \subset \cdots \subset C(h_k)$.

A natural question then arise: which $h$ we should set? Fortunately, using Motzkin-Straus Formalism, the edge weights in $A$ are not required to be either 0 or 1. Thus we can directly set $A = W$ and run the algorithm; and the nonzero entries correspond to the *generalized* cliques. This provides a definition of the generalized or pseudo-cliques on weighted graphs. We can similarly define generalized bicliques on weighted bipartite graphs.

## 8 Biclustering the Protein Complex - Protein Domain Network

Proteins interact with other proteins in many cellular processes, including metabolism, endocrine and exocrine, signaling, synthesis and transport. A protein complex is formed by two or more proteins that physically interact with each other, and is often called molecular machines of life. A systematic identification, characterization and understanding of these molecular machines will provide an essential knowledge base, and link proteome dynamics and architecture to cellular function and phenotype.

A number of experimental approaches provide genomic scale protein interaction data. The two-hybrid genetic screen [17, 24] yields binary interaction data. High throughput mass spectrometry methods [11, 16] combine tagged proteins and protein-complex purification schemes with mass spectrometric measurements to yield physiologically relevant data on intact multi-protein complexes.

However, these high-throughput experiments are often associated with large false-positive rates (30-50%). Protein interaction data obtained in two independent two-hybrid screening experiments [17] and [24] overlap by less than 4%. The overlap between two mass spectrometry methods [11, 16] is also around 5%. Therefore, using computational methods to predict protein interaction modules from these data is an important way to extract additional information.

The nature of protein complexes can be easily expressed as a bipartite graph, where $p$-nodes refer to protein complexes, and $r$-nodes refer to proteins. We are interested in discovering densely connected subgraphs in the bipartite graph which are likely candidates for protein interaction modules. In graphs, the most densely connected objects are cliques. Thus we wish to detect bicliques in the bipartite graph.

Previous analysis [16, 2, 7], of protein complex data were largely performed on the protein level, i.e., discovering protein functional modules etc. on the network of protein complex - protein. However, comparison of complexes at protein level is not able to reflect the function linkage among protein complexes in many cases. For example, the protein composition of the cytoplasmic ribosomal large subunit and the mitochondrial ribosomal large subunit both function for protein synthesis. When comparing the protein composition of the two complexes, no single protein is found to present in both complexes. When we compare the two protein complex at domains level, 14 domains are found to share between the two complexes Because different proteins with similar activities are likely to contain similar/same domains, there are multiple instances of the domains dispersed in the data sets, while there might be only a few copies of each protein. Analyzing complexes at the domain level has the advantages of providing an enriched functional information[26, 25]. This also eliminates the need to model internal complex-protein and protein-domain connectivity. The domain-complex association demonstrates connections between the biological processes and what the domains facilitate.

| Complex | #Protein | #Domain |
|---|---|---|
| Cytoplasmic RLS | 81 | 50 |
| Mitochondrial RLS | 44 | 29 |
| Shared unit | 0 | 14 |

**Table 1. Protein domain composition of cytoplasmic and mitochondrial ribosomal large subunits.**

### 8.1 Bipartite Graph Model

The relationship between protein complexes and domains is naturally modeled with a bipartite graph. A bipartite graph has two types of nodes, and edges connecting nodes of different types. Let $G(C, D, E)$ be the bipartite graph representing protein complexes and their constitutional proteins, with C-type nodes representing protein complex $c_i$ and D-type nodes representing protein domain $d_k$. The adjacency matrix $E$ is

$$E_{ik} = \begin{cases} 1 & \text{if protein complex } c_k \text{ contain domain } d_i \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

This simple graph model assign binary weights to the edges, i.e. 1 if a domain is present in the protein, 0 otherwise.

### 8.2 Data sources

Protein complex data were obtained from MIPS Comprehensive Yeast Genome Database [13]. The main part of the complex data consists of manually annotated protein complexes derived from literatures. But it also includes several hundred of complexes identified by high throughput experiments. The latter set of complex data are biased by bait selection and purification method. These complexes may only represent a population of different biological complexes with which the bait protein is associated. Therefore, for this task, we only use the protein complexes identified from literature. As a result, 267 complexes were used and 1237 unique proteins are contained in the set of complexes.

The domain definitions of the yeast proteins are according to Pfam [3], which contains hidden Markov model based profiles (HMM-profiles) of many common protein domains based on multiple sequence alignments. The Pfam database contains two parts: one is the curated part called Pfam-A and the other is automatically generated supplement called

Pfam-B which represents small families taken from the PRODOM database that do not overlap with Pfam-A. Only Pfam-A families are used here. In total, 709 Pfam domains are defined on the set of complexes. Proteins without defined domains are disregarded.

## 8.3 Result

We apply the biclique finding algorithm to the adjacent matrix $E$. Significant bicliques obtained are listed in Table 3. Complete results are available upon request.

The bicliques in the domain-complexes bipartite graph define modules in protein complexes, as domains in a biclique are highly affiliated with each other.

To see the importance of biclustering protein complex interactions at domain level, we define a score $\rho$ to measure the ratio of the shared domains that are attributed to shared proteins.

$$\rho = \frac{\#D - \#D_p}{\#D},$$

where $\#D$ is the number of domains in the biclique pattern, $\#D_p$ is the number of domains resulted from shared proteins among the set of complexes in the biclique pattern. The ratio $\rho$ ranges from 0 to 1, with $\rho$=1 indicating no protein is shared among the set of complexes but they do share the set domains. About half of the bicliques discovered have $\rho$=1, suggesting/confirming the importance of analyzing the protein complexes at the domain level.

## 8.4 Biological Examples

The bicliques represent groups of functionally-related domains as well as functionally-related protein complexes. For example, RNA polymerase I, RNA polymerase II, and RNA polymerase III are all DNA-dependent RNA polymerases responsible for the polymerization of ribonucleotides into a sequence complementary to the template DNA. These three forms of RNA polymerases transcribe different sets of genes and synthesize different RNAs. RNA polymerase I is located in the nucleoli and it synthesizes precursors of most ribosomal RNAs. RNA polymerase II occurs in the nucleoplasm and it synthesizes mRNA precursors. RNA polymerase III also occurs in the nucleoplasm and it synthesizes the precursors of 5S ribosomal RNA, the tRNAs, and a variety of other small nuclear and cytosolic RNAs. One discovered biclique contains the three RNA polymerases and a set of 19 domains, all of which are annotated as DNA-directed RNA polymerase activity (GO:0003899) by gene ontology. These 19 domains consist of a core set of domains necessary for RNA synthesis.

Another instance is the biclique of two types of ribosomal large subunit and 14 domains. As discussed before, the two types of ribosomal large subunits share no proteins. The domain level analysis allows us to functionally link the cytoplasmic ribosomal large subunit and mitochondrial ribosomal large subunit based on their 14 shared domains. The set of 14 shared domains are mostly annotated by gene ontology for protein biosynthesis (GO:0006412). Similarly, we found 11 shared domains (annotated as protein biosynthesis) in the cytoplasmic ribosomal small subunit and mitochondrial ribosomal small subunit. This allows us to functionally link the two types of ribosomal small subunit although they share no common proteins.

The replication complexes, DNA polymerase alpha (I) - primase complex, DNA polymerase epsilon (II), and DNA polymerase delta (III) are linked in a biclique together with their shared domains DNA_pol_E_B, DNA_pol_B_exo, and DNA_pol_B. These complexes all involves DNA synthesis and these three domains all have DNA-directed DNA polymerase activity (GO:0003887).

The above examples show that the biclique finding method is able to simultaneously group domains as well as protein complexes. This grouping clearly indicates functional linkage among the set of protein complexes and domains.

## 9 Summary

In this paper, we propose a new algorithm for computing bicliques in bipartite graphs. The algorithm is based on generalized Motzkin-Straus formalism for cliques. It has the flexibility to favor different shapes of targeted bicliques. The algorithm can be easily implemented; it can effectively and efficiently compute maximal bicliques.

We also show that biclique finding is similar to biclustering of a bipartite graph. We provide a detailed case study on finding bicliques on the protein complex interactions. We show that biclustering at the protein domain level provides many more functional linkages than biclustering at the protein level.

## Acknowledgement

This work is supported in part by U.S. Department of Energy, Office of Science, through a MAGGIE project in the GTL Program, under contract DE-AC02-05CH11231.

## References

[1] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. Hammer, and B. Simeone. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145:11–21, 2004.

[2] G. D. Bader and C. W. V. Hogue. Analyzing yeast protein-protein interaction data obtained from different sources. *Nature Biotechnology*, 20:991–997, 2002.

[3] A. Bateman, L. Coin, R. Durbin, et al. The Pfam protein families database. *Nucleic Acids Res. (Database Issue)*, 33:D138–D141, 2004.

[4] M. Bellare, M. Goldreich, and M. Sudan. Free bits, pcps and non-approximability — towards tight results. *Proc. IEEE Symp. on Foundations of Computer Science*, pages 422–431, 1995.

[5] I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 4. Kluwer Academic Publishers, Boston, MA, 1999.

[6] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16:575–577, 1973.

[7] C. Ding, X. He, R. Meraz, and S. Holbrook. A unified representation for multi-protein complex data for modeling protein interaction networks. *Proteins: Structure, Function, and Bioinformatics*, 57:99–108, 2004.

[8] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information Processing Letters*, 51:207–211, 1994.

[9] U. Feige, S. Goldwasser, L. Lov'asz, M. Safra, and M. Szegedy. Approximating clique is almost np-complete. *Proc. IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991.

[10] M. Garey and D. Johnson. *Computers and Intractability: A guide to the theory of NP Completeness*. Freeman, 1979.

[11] A.-C. Gavin, M. Bosche, R. Krause, et al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147, 2002.

[12] L. Gibbons, D. Hearn, P. Pardalos, and M. Ramana. Continuous characterizations of the maximum clique problem. *Mathematics of Operations Research*, 22:754–768, 1997.

[13] U. Guldener, M. Munsterkotter, G. Kastenmuller, et al. Cygd: the comprehensive yeast genome database. *Nucleic Acids Res. (Database Issue)*, 33:D364–D368, 2005.

[14] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[15] T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning*. Springer Verlag, 2001.

[16] Y. Ho, A. Gruhler, A. Heilbut, et al. Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, 415:180–193, 2002.

[17] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two hybrid analysis to explore the yeast protein interactome. *Proc. Natl. Acad. Sci.*, 98(8):4569–4574, 2001.

[18] J. Li, H. Li, D. Soh, and L. Wong. A correspondence between maximal complete bipartite subgraphs and closed patterns. *9th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD), Porto, Portugal, October 2005*, pages 146–156, 2005.

[19] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM. Trans. on Computational Biology and Bioinformatics*, 1:25 – 45, 2004.

[20] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. *Proc Scandinavian Workshop on Algorithm Theory*, pages 260–272, 2004.

[21] T. Motzkin and E. Straus. Maxima for graphs and a new proof of a theorem of turan. *Canad. J. Math.*, 17:533–540, 1965.

[22] M. Pelillo. Relaxation labeling networks for the maximum clique problem. *Journal of Artificial Neural Networks*, 2(4):313–328, 1995.

[23] R. Tibshirani. Regression shrinkage and selection via the LASSO. *J. Royal. Statist. Soc B.*, 58:267–288, 1996.

[24] P. Uetz., L. Cagney, G. Mansfield, et al. A comprehensive analysis of protein-protein interactions in *saccharomyces cerevisiae*. *Nature*, 403(6770):623–627, 2000.

[25] Y. Zhang, J.-M. Chandonia, C. Ding, and S. Holbrook. Comparative mapping of sequence-based and structure-based protein. *BMC Bioinformatics*, page 77, 2005.

[26] Y. Zhang, H. Zha, C. Chu, and X. Ji. Protein interaction inference as a max-sat problem. *Proc. of IEEE CVPR 2005 Workshop on Computer Vision Methods for Bioinformatics*, 2005.

**Table 2. Computed bicliques from the domain-complex bipartite graph.**

| Domain | Complex | $\rho$ |
|---|---|---|
| **RNA_pol_Rpb1_1; RNA_pol_Rpb1_2; RNA_pol_Rpb1_3; RNA_pol_Rpb1_4; RNA_pol_Rpb1_5; RNA_pol_L; RNA_pol_A_bac; RNA_pol_Rpb2_1; RNA_pol_Rpb2_2; RNA_pol_Rpb2_3; RNA_pol_Rpb2_5; RNA_pol_Rpb2_6; RNA_pol_Rpb2_7;** RNA_pol_Rpb5_N; RNA_pol_Rpb5_C; DNA_RNApol_7kD; RNA_pol_N; RNA_pol_Rpb8; RNA_pol_Rpb6 | RNA polymerase I (510.10); RNA polymerase II (510.40.10); RNA polymerase III (510.120) | 0.68 |
| **Ribosomal_L14; Ribosomal_L1; Ribosomal_L5; Ribosomal_L5_C; Ribosomal_L23; Ribosomal_L11_N; Ribosomal_L11; KOW; Ribosomal_L2; Ribosomal_L2_C; Ribosomal_L6; L15; Ribosomal_L3; Ribosomal_L13** | cytoplasmic ribosomal large subunit (500.40.10); mitochondrial ribosomal large subunit (500.60.10) | 1 |
| **Ribosomal_S17; Ribosomal_S9; S4; Ribosomal_S5; Ribosomal_S5_C; Ribosomal_S14; Ribosomal_S10; Ribosomal_S15; Ribosomal_S2; Ribosomal_S7; Ribosomal_S19** | mitochondrial ribosomal small subunit (500.60.20); cytoplasmic ribosomal small subunit (500.40.20) | 1 |
| **TFIID-18kDa;** TFIID_90kDa; TFIID_20kDa; TFIID_30kDa; TAF; DUF1546; TFIID-31kDa | SAGA complex (230.20.20); SAGA complex (510.190.10.20.10); TAFIIs (510.70.20) | 0.14 |
| Actin; **Bromodomain; SNF2_N; Helicase_C; SNF5; Myb_DNA-binding; SWIRM; YEATS** | SWI/SNF transcription activator complex(510.190.50); RSC complex (Remodel the structure of chromatin) (400) | 0.88 |
| **Transket_pyr; Pyr_redox_2; Pyr_redox; Biotin_lipoyl; 2-oxoacid_dh; E1_dh; GIDA Pyr_redox_dim** | Pyruvate dehydrogenase (390); 2-oxoglutarate dehydrogenase (20) | 1 |
| **Adaptin_N; Adap_comp_sub; Clat_adaptor_s** | AP-2 complex (260.20.20); AP-3 complex (260.20.30); AP-1 complex (260.20.10) | 1 |
| **DNA_pol_E_B; DNA_pol_B_exo; DNA_pol_B** | Replication complex (410.35); DNA polymerase alpha (I) - primase complex (410.40.60); DNA polymerase epsilon (II) (410.40.100); DNA polymerase delta (III) (410.40.90) | 1 |
| **Pyr_redox_2; DAO; FAD_binding_2; Succ_DH_flav_C** | other respiration chain complexes (420.60); Succinate dehydrogenase complex (complex II) (420.20) | 0 |
| MutS_I; MutS_II; MutS_III; MutS_IV; MutS_V | MSH2/MSH3 complex (510.180.50.10); MSH2/MSH6 complex (510.180.50.20) | 0 |
| **GTP_EFTU; GTP_EFTU_D2** | eEF1(500.20.10); eRF3(500.30.30); eEF2 (500.20.20); eIF2 (500.10.20) | 1 |
| **XPG_N; XPG_I** | NEF3 complex (510.180.10.30); Exonucleases (410.40.120) | 1 |
| RNase_P_Rpp14; RNase_P_p30; RNase_P_pop3; POP1; POPLD | RNase P (440.14.10); RNase MRP (440.12.20) | 0 |
| **WD40; Helicase_C; DEAD; RRM_1** | CCR4 complex (510.190.110); mRNA splicing (440.30.10); rRNA splicing (440.30.20) | 1 |
| HATPase_c; DNA_gyraseB; DNA_topoisoIV | Synaptonemal complex (SC) (490); Topoisomerases (410.40.140) | 0 |
| **ATP-synt_ab_N; ATP-synt_ab; ATP-synt_ab_C; ATP-synt_C** | F0/F1 ATP synthase (complex V) (420.50); H+-transporting ATPase, vacuolar (220) | 1 |
| Prenyltrans; PPTA | Farnesyltransferase (FTase) (180.10); Geranylgeranyltransferase I (GGTase I) (180.20); Geranylgeranyltransferase II (GGTase II) (180.30) | 0 |