Computing Protein Interaction Modules via Clique Finding based on Generalized Motzkin-Strauss Formalism

Chris Ding\textsuperscript{a}, Chunlin Wang\textsuperscript{b}, Qiaofeng Yang\textsuperscript{a}, Stephen Holbrook\textsuperscript{a}
\textsuperscript{a}Lawrence Berkeley National Laboratory, Berkeley, CA 94720
\textsuperscript{b}Stanford University, Stanford, CA 94305

July 4, 2007

Abstract

Predicting protein interaction modules from a network of protein interactions is a challenging task. In this paper, we propose to use a clique finding algorithm to compute cliques and quasi-cliques in the network. We first derive a new algorithm based on the generalized Motzkin-Strauss Formalism. We then prove the correctness and convergence of the algorithm. The algorithm is extremely simple to implement and has a computational complexity of $O(|E|)$ where $|E|$ is the number of interactions. We then apply the algorithm to three archaeal extremophilic organisms: \textit{Halobacterium NRC-1}, \textit{Pyrococcus furiosus}, and \textit{Sulfolobus solfataricus}; We have discovered a large number of protein interaction modules: 503 from \textit{Halobacterium}, 364 from \textit{P. furiosus}, and 155 from \textit{S. solfataricus}. By computing orthologs, we are able to determine a large number of conserved protein modules in the three organisms. Our results also confirm experimentally determined protein complexes.

1 Introduction

Although cells are composed of thousands of different types of molecules, the inner working of a cell can rarely be contributed to a single molecule. Different molecules act either collectively or synergistically to carry out cellular functions that are indispensable for cell survival and proliferation. The notion of functional "modules" as discrete functional entities arises in the need of understanding the functional organization of a cell [9]. Such functional modules could be a ribosome that synthesizes proteins or a signal transduction system that enables a cell to respond sensitively over a wide range of environmental stimuli. Functional modules can be insulated from, or connected to, each other and the structure of these functional modules can change over time. More importantly, the function of a module can be quantitatively regulated or switched qualitatively between distinct functions. Interactions or connections of multiple modules enables a cell to integrate information from its internal and external environment and react accordingly.

Therefore, identification of functional modules and how they interact with each other are critical steps in understanding how cells live and reproduce. In addition, the identification of modules conserved across several species can help us understand how modules are constructed and how connections between modules change during evolution to alter the behavior of cells and organisms.

In this paper, we focus on detecting protein interaction modules from protein interaction data, both experimental and computationally predicted. These data are integrated together as a network of protein interactions, and our task is to computationally predict protein interaction modules from this complex network.

The general idea is to define the interaction module as a highly connected subgraph/subnetwork in the network. The most densely connected subgraph is a clique, in which every protein interacts with every other protein. This definition is often too restrictive and "k-core" is used as the definition of a protein module [10, 1]. A k-core is a subgraph where each protein interacts with at least k other proteins in the subgraph. Thus, a k-core is a looser definition than a clique. A question is then how to choose the parameter k? Further work on protein module detection use a variety of other methods including the density-based method [2], the min-max spectral clustering [5, 6], orthlogs [16], network decomposition based methods [15, 12], and hyperclique [18].

Outline of the paper. In this paper, we propose using clique as the definition of a protein interaction module, due to its simplicity and intuitive appeal. This is possible due to a significant advance in computing maximal cliques as described here. We use the algorithmic approach of the Motzkin-Strauss theorem [13] that relates
clique finding to the optimization of a quadratic function with $L_1$ constraints (see §2). Our main contributions are the following: (1) We generalize the Motzkin-Strauss formalism to use $L_p$ constraints (see §3), and (2) We derive an iterative algorithm to compute the solution and prove the correctness and convergence of the algorithm (see §4).

The strength of the algorithm is that the formulation makes it easy to (1) compute pseudo cliques (i.e., densely connected subgraphs which are less dense than cliques, see §5), (2) compute cliques and pseudo cliques on weighted graphs (see §6).

These two features are particularly important for protein interaction module computation since protein modules are not necessarily cliques in the graph and protein interaction data are often weighted (with the edge weight being the probability of interaction).

In §7, we describe how to compute multiple pseudo cliques. In §8, we describe how to compute overlapping pseudo cliques. In §9, we describe how to compute transitive closure for weighted graphs. This is useful because protein interactions are sparse and thus cliques satisfying the rigorous definition in the interaction network are usually very small. We test the algorithm for graphs with cliques embedded in the graph (§10).

In this work, we use the Prolinks database [4] which provides computationally determined protein interactions. (explained in detail in §11). We apply the clique algorithm to the interaction networks of 3 archae organisms: Halobacterium NRC-1, Pyrococcus furiosus, and Sulfolobus solfataricus. In §12, we discuss our extensive results on ”conserved protein complexes” and ”organism-specific” protein interaction modules.

### The clique finding algorithm

Set initial $x^{(0)} = (1 \cdots 1)^T$. Set $\beta = 1.1$. Let $A = \{0, 1\}^{n \times n}$ be the adjacency matrix of an unweighted undirected graph. Set $A_{ii} = 1, 1 \leq i \leq n$. The algorithm iteratively updates $x$ using:

$$
x_i^{(t+1)} = \left(\frac{x_i^{(t)}}{\lambda^*(A)x^{(t)}}\right)^{1/\beta},
$$

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

### Motzkin-Strauss Formalism for Clique Finding

It is known that maximum clique finding and maximal biclique finding are NP-hard. It is also known that the maximum clique problem is hard to approximate [3].

Motzkin & Strauss [13] transform computing maximal cliques to the optimization of a quadratic function:

$$
\max_x \ x^T A x \quad s.t. \quad \sum_{i=1}^{n} x_i = 1, \ x_i \geq 0,
$$

where $\text{diag}(A) = 0$.

**Theorem 2.1.** (Motzkin and Strauss). Let $G$ be an unweighted graph and $x^*$ the optimal solution for the problem of Eq.(2.2). 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 $x^*$ is called a characteristic vector of a subset $C$), $C$ is a maximal clique in $G$.

The Motzkin-Strauss theorem has been in a number of studies [14, 8]. The main drawback in the standard Motzkin-Strauss theorem is the requirement that $\text{diag}(A) = 0$ which makes it difficult to compute the maximum clique.

### 3 Generalized Motzkin-Strauss Formalism

#### 3.1 Motivation

Because $\text{diag}(A) = 0$, $A$ is indefinite, i.e., the eigenvalues of $A$ are evenly split on both sides of zero:

$$
\sum_{k=1}^{n} \lambda_k(A) = \text{Tr} A = 0.
$$

Thus there are many local optimal solutions, and each of them corresponds to a maximal clique. But the maximum clique corresponding to the global solution is difficult to find.

We improve this situation by setting

$$
\tilde{A} = A + I
$$

and solve the quadratic problem of $x^T \tilde{A} x$. This is an easier optimization problem because

$$
\sum_{k=1}^{n} \lambda_k(\tilde{A}) = n,
$$

i.e., there are more (stronger) eigenvectors with positive eigenvalues and less (weaker) eigenvectors with negative eigenvalues.

If $\tilde{A}$ is positive definite, there is a unique global solution to the quadratic optimization problem. In general, however, $\tilde{A}$ is not positive definite. Still, solving the quadratic optimization with $\tilde{A}$ is easier than with the original $A$. 

2
A straightforward generalization
\[
\max_{\mathbf{x}} \mathbf{x}^T \tilde{A} \mathbf{x} \quad s.t. \sum_{i=1}^{n} x_i = 1, \ x_i \geq 0,
\]
(3.6)
is not a valid generalization because in this formalism, the solution is not guaranteed to converge to a maximal clique.

This difficulty is resolved by modifying the feasibility constraint, the $L_1$ norm constraint. We generalize it to $L_p$-norm constraint. For a vector $\mathbf{x}$, $L_p$ norm is $\|\mathbf{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$. As long as $p \simeq 1$, the optimal solution vector is sparse, i.e., many elements in $\mathbf{x}^*$ are zero. Using the $L_1$ constraint to enforce sparsity is well-known in statistics as used in LASSO [17].

### 3.2 Generalized Motzkin-Strauss Formalism

We generalize the maximum clique problem to the following optimization problem:
\[
\max_{\mathbf{x}} \mathbf{x}^T \tilde{A} \mathbf{x} \quad s.t. \sum_{i=1}^{n} x_i^\beta = 1, \ x_i \geq 0,
\]
(3.7)
where $\beta \in [1, 2]$ is a parameter. We show the following:

1. The maximum clique is computed at $\beta = 1 + \epsilon$, $0 < \epsilon \ll 1$.
2. We derive an algorithm for $\beta \in [1, 2]$ and prove its convergence.
3. We demonstrate that as $\beta \to 1$, the sparsity of the solution increases steadily, showing the close relation between $L_1$ constraints and sparsity. At $\beta = 2$, the solution is given by the principal eigenvector of $A$.

We have the following:

**Theorem 3.1. (Generalized Motzkin-Strauss Theorem)** Using $\tilde{A}$ as the adjacency matrix, and setting $\beta = 1 + \epsilon$, $0 < \epsilon \ll 1$. Let $C = \{i \mid x_i^* > 0\}$ be the subset corresponding to nonzero elements of the solution. If nonzero elements have same values, $x_i^* = 1/|C|$ for $\forall i \in C$ is a maximal clique.

**Proof outline.** Since the nonzero elements of $\mathbf{x}$ have constant values, $\mathbf{x}^*$ must have the following form
\[
\mathbf{x}^* = (1/|C|^{1/\beta})(1 \cdots 1, 0 \cdots 0)^T,
\]
assuming, without loss of generality, we index the nodes of $C$ first. The objective becomes
\[
J = (\mathbf{x})^T \tilde{A} \mathbf{x} = |C|^{2-2/\beta}.
\]
Since $\beta > 1$, thus $2 - 2/\beta > 0$, and max $J$ is equivalent to max $|C|$.

Note that if we use $\beta = 1$, then $J = 1$ independent of $|C|$, i.e., we are not guaranteed to compute the maximal clique.

### 4 Algorithm for Generalized Motzkin-Strauss Formalism

The algorithm for solving for generalized Motzkin-Strauss theorem is given in Eq.(1.1). We analyze the basic properties of this algorithm. We prove the feasibility, correctness, and convergence of the algorithm.

#### 4.1 Feasibility

We show that from any initial $\mathbf{x}^{(0)}$, the iteration will lead to a feasible solution, i.e.,
\[
\sum_i [x_i^{(t+1)}]^\beta = \sum_i x_i^{(t)} (A\mathbf{x})^i_j / |\mathbf{x}^{(t)}|^\gamma A\mathbf{x}^{(t)} = 1.
\]

#### 4.2 Correctness

We show that the update rule satisfies the first order KKT optimality condition. To prove it, we form the Lagrangian function
\[
L = \mathbf{x}^T A \mathbf{x} - \lambda \sum_i x_i^\beta - 1)
\]
(4.8)
where $\lambda$ is the Lagrangian multiplier for enforcing the $L_p$ constraint. This leads to the KKT condition for the nonnegativity of $x_i$, that the optimal solution $\mathbf{x}^*$ must satisfy
\[
[2(A\mathbf{x})_i - \lambda x_i^\beta - 1)] x_i = 0.
\]
(4.9)
Summing over index $i$, we obtain the value for the Lagrangian multiplier $\lambda$.
\[
2\mathbf{x}^T A \mathbf{x} = \lambda \sum_i x_i^\beta = \lambda \beta.
\]
(4.10)
Clearly the updating rule
\[
[x_i^{(t+1)}]^\beta = x_i^{(t)} (A\mathbf{x})^i_j / (\lambda \beta / 2),
\]
(4.11)
at convergence satisfies the KKT condition Eq.(4.9). Substituting the value of $\lambda$ from Eq.(4.10), this becomes the update rule Eq.(1.1).

#### 4.3 Convergence

**Theorem 4.1. Under the updating rule of Eq.(1.1) the iteration converges to a fixed point.**

**Proof.** We first prove the following monotonicity property:
\[ L(x^{(0)}) \leq L(x^{(1)}) \leq \cdots \leq L(x^{(t)}) \leq \cdots. \]

1. We use the auxiliary function approach. A function \( G(x, x') \) is an auxiliary function of \( L(x) \) if \( G(x, x') \leq L(x); \) \( G(x, x) = L(x). \) Now, we define

\[ x^{(t+1)} = \arg \max_x G(x, x^{(t)}). \quad (4.12) \]

By construction, we have \( L(x^{(t)}) = G(x^{(t)}, x^{(t)}) \leq G(x^{(t+1)}, x^{(t)}) \leq L(x^{(t+1)}). \) Thus \( L(x^{(t)}) \) is monotonically increasing (non-decreasing) under the iterative updating rule of Eq.(1.1).

2. Now we show that the following function

\[ G(x, x') = \sum_{ij} x_i' A_{ij} x_j' (1 + \log \frac{x_i x_j}{x_i' x_j'}) - \lambda \left( \sum_i x_i^\beta - 1 \right) \]

is an auxiliary function of \( L(x) \). To see this, we note that \( \frac{x_i x_j}{x_i' x_j'} \geq \left( 1 + \log \frac{x_i x_j}{x_i' x_j'} \right), \) for \( x_i, x_j, x_i', x_j' > 0 \) using the inequality of \( z \geq 1 + \log(z) \) for any positive \( z \). Thus the first sum in \( L(x) \) is always greater than or equal to the first sum in \( G(x, x') \). The equality holds when \( x = x' \). The second term is identical in \( G(x, x') \) and \( L(x) \).

3. Next, we show that the auxiliary function \( G(x, x') \) is a convex function and we obtain the global optimal solution. The first order derivative (the gradient) is

\[ \frac{\partial G(x, x')}{\partial x_i} = 2x_i' (A x_i') - \lambda \beta x_i^{\beta-1} = 0. \quad (4.13) \]

The second order derivatives (Hessian matrix) are

\[ \frac{\partial^2 G(x, x')}{\partial x_i \partial x_j} = -2 \frac{x_i' (A x_i')}{x_i^2} + \lambda \beta (\beta - 1) x_i^{\beta-2} \delta_{ij}. \]

Because \( \beta \geq 1 \), the Hessian is a diagonal matrix with negative quantities on the diagonals, i.e., it is a convex function. Thus there is a global optimum which is given by setting the gradient Eq.(4.13) to zero. According to Eq.(4.12), and setting \( x^{(t)} = x', x^{(t+1)} = x \), we recover the updating rule of Eq.(1.1).

5 Computing Quasi-cliques

In the protein interaction network, not all protein interaction modules are exact maximal cliques. We have checked that in a subgraph, if each node interacts with 80% of the other nodes, this subgraph is usually a functional module, i.e., every member of the subgraph possesses the same function. For this reason, we are interested in computing quasi-cliques.

It turns out that the above MS approach is particularly suitable for computing quasi-cliques. A quasi-clique is identified if in the converged solution, the nonzero elements do not have identical values. In practice, for all the real protein networks, most of the cliques computed via the MS formalism are quasi-cliques.

6 Computing Quasi-cliques on a Weighted Graph

Clique are only defined for unweighted graphs. For a weighted graph with edge weight \( W \), we may threshold the weights, i.e., setting \( A_{ij} = 1 \) if \( W_{ij} \geq h \); \( A_{ij} = 0 \) otherwise. We run the algorithm to compute cliques on \( A \). However, as we change the threshold \( h \), the results will change. Fortunately, the generalized clique algorithm do not require \( A \) being a matrix of \( \{0, 1\} \). Thus we can directly set \( A = W \) and compute the generalized cliques. This is a definition of the generalized cliques on weighted graphs and a practical method to compute them.

7 Computing Multiple Cliques

There are several approaches; we use a simple and efficient approach. After one maximal clique \( C \) is computed, we delete the clique, i.e., all nodes in \( C \) are deleted. Thus the remaining network is smaller and in the next run, the algorithm converges to a different clique. We repeat this procedure until all quasi-cliques are extracted.

8 Computing Overlapping Cliques

The cliques/quasi-cliques computed in this way are non-overlapping. However, a protein can participate in several cellular processes, i.e., overlapping protein functional modules may exist. We deal with overlapping cliques in the next section.

Figure 1 illustrates how the overlapping cliques behave using matrix notation. Here we describe an algorithm to compute overlapping cliques. From Figure 2, clique \( A \) is first computed. Clique \( B' \) is computed later. But \( B' \) is only a subset of \( B \). Our task here is to expand \( B' \) to the full \( B \). First, we compute the pairwise overlap between all pairs of cliques. Second, we expand \( B' \) until it can not be further expanded. In this way, we compute the overlapping cliques.

9 Transitive Closure for weighted graphs

Because protein interactions are typically very sparse, the cliques detected are very small. There are several
Figure 1: Illustration of non-overlapping and overlapping cliques.

Figure 2: For overlapping cliques, our algorithms find A first and B' second. as two non-overlapping cliques. In the algorithm for computing overlapping cliques, we (1) check that B’ is not a maximal clique and (2) expand B' into the maximal clique.

ways to resolve this problem.
(1) Compute quasi-cliques as explained in §5.
(2) Use k-core[11, 1], where each protein only interacts with a fraction of other proteins in the subgraph. This relaxes the strict definition of clique, but it introduces the new parameter k and raises the issue of how to choose k and whether the results be stable with respect to different choices of k.
(3) We resolve the sparsity issue by using the transitive closure of a weight graph. Details on how transitive closure is defined can be found in [7].

The idea is that on transitive closure, missing connection/interactions will be added and the graph becomes dense; therefore the detected cliques will be larger, thus closer to the true protein interaction modules.

10 Experiments: Synthetic Data
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 network to be 1000x1000. We set \( \alpha = \beta = 1.05 \). In 20 trials, random graphs, the embedded cliques, and presumably the maximum cliques, are always correctly detected. If we set \( \beta = 1 \), the algorithm fails to detect the maximum clique 13 times, because they often converge to local optimal solutions, not the global solution.

11 Prolinks Database
Prolinks is a database of functionally related proteins derived from coevolution. It uses four genomic methods, namely phylogenetic profile, gene neighbors, Rosetta Stone and gene cluster to infer functionally related proteins based on the assumption that evolutionary pressure would force pairs of proteins that are functionally related to be both present or both absent within genomes, tend to be encoded close to each other across multiple genomes, fused into one protein in another organism, or belong to the same operon. Each method employs a probabilistic model to assign a P value to every pair of proteins, showing how likely they are to be functionally linked. See Figure 3.

Phylogenetic profile. The phylogenetic profile method uses co-occurrence or absence of pairs of non-homologous genes across multiple genomes to infer func-
tionally related proteins. The assumption is that co-evolving proteins tend to appear together. BLAST was used to search for homologs of the protein across multiple genomes. For each protein, a profile of zero and one was generated to represent the presence or the absence of its homolog across organisms. The hypergeometric distribution was used to compute the probability that a pair of proteins co-evolved together.

**Gene cluster.** The gene cluster method uses intergenic distance as a predictor of operons. In prokaryotes, a group of genes that carry out similar functions are often transcribed from a single functional unit, known as an operon. The assumption behind the gene cluster method is that the closer two genes are encoded on the same DNA strand, the more likely they belong to the same operon. The probability that two genes are separated by a certain distance was calculated using a Poisson distribution and the probability that two genes belong to the same operon was subsequently deduced from that.

**Gene neighbor.** The gene neighbor method uses conserved operons across multiple genomes to infer functionally related proteins. Homologs of the two genes were found across genomes first, then the likelihood that the two genes are components of a conserved operon was computed to obtain a P value for that pair of genes.

**Rosetta stone.** The Rosetta stone method detects gene-fusion events across multiple genomes. The assumption is that two proteins that are functionally linked in one organism occasionally can be found as one protein in the same or a second genome. To find such pairs of proteins, BLAST was used to identify non-homologous proteins that align to different portions of a third protein. Then the hypergeometric distribution was used to calculate the probability that the two proteins had fused.

### 12 Results on Archaea Organisms

The hyperthermophilic archaea, including *Pyrococcus furiosus* and *Sulfolobus solfataricus*, present many advantages for studying biologically important reversible or transient interactions. First, these single cell organisms have relatively simple genomes that have been sequenced in their entirety and annotated in detail. Second, unlike many single cell model organisms including yeast, the archaea *P. furiosus* and *S. solfataricus* have the additionally advantageous characteristic of hyperthermophilic growth. Protein complexes that are transient at hyperthermophilic temperatures (above 80 degrees C) will often be considerably more stable and long-lived at room temperature.

In this paper, we report the results on computing protein interaction modules of three archaean organisms: *P. furiosus*, *S. solfataricus*, and *Halobacterium NRC-1*. We use the Prolinks Protein Interaction Database.

#### 12.1 Summary of Results

Table 1 gives a summary of our clique finding results for each of the three archaean organisms of interest using three different algorithms (non-transitive closure, transitive closure and allowing overlapping cliques). Tables 2-4 gives a partial list of cliques identified in the 3 organisms.

**Table 1: Clique Summary**

<table>
<thead>
<tr>
<th>Organism</th>
<th>Non-Transitive Closure</th>
<th>Transitive Closure</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Pyrococcus furiosus</em></td>
<td>94 clicles from 420 proteins</td>
<td>482 clicles from 1741 proteins</td>
</tr>
<tr>
<td></td>
<td>average size of clicles is 4.47</td>
<td>average size is 3.61</td>
</tr>
<tr>
<td><em>Sulfolobus solfataricus</em></td>
<td>115 clicles from 537 proteins</td>
<td>187 clicles from 832 proteins</td>
</tr>
<tr>
<td></td>
<td>average clique size is 4.67</td>
<td>average clique size is 4.45</td>
</tr>
<tr>
<td></td>
<td>clique size distribution: 17:1, 16:1, 13:2, 11:1, 10:3, 8:3, 7:8, 6:8, 5:11, 4:27, 3:46</td>
<td></td>
</tr>
<tr>
<td><em>Halobacterium NRC-1</em></td>
<td>96 clicles from 429 proteins</td>
<td>523 clicles from 1902 proteins</td>
</tr>
<tr>
<td></td>
<td>average size of clicles is 4.47</td>
<td>average size of clicles is 3.63</td>
</tr>
<tr>
<td></td>
<td>clique size distribution : 22:1, 15:2, 12:1, 11:1, 10:2, 9:1, 8:1, 7:3, 6:7, 5:4, 4:3, 3:56</td>
<td></td>
</tr>
</tbody>
</table>

#### 12.2 Conserved Protein Complexes

One of our main goals is to find protein complexes conserved across multiple organisms. We give several examples in this section.

Figure 4 provides an example of the conserved oxidoreductase complex. The oxidoreductase complex is a
Figure 4: This oxidoreductase is a conserved complex across two organisms: *sulfolobus* and *pyrococcus*. Yellow lines indicate proteins interactions. Black lines indicates orthologs. In *sulfolobus*, our results indicates all 7 proteins form the oxidoreductase complex. The experimental results show there are 2 protein complexes, each with 3 proteins.

It is interesting that protein chromatographic multiple column fractionation experiments show that there are two protein complexes (see Figure 4). Our results indicate these two protein complexes are indeed parts of a single known protein complex. In addition, our results also indicate protein PF0971 is part of the protein complex.

Figure 5 provides another example for the ABC transporter complex. This complex is mainly composed of the ATP-Binding Cassette (ABC) Transporter superfamily of transmembrane (TM) proteins. These proteins bind ATP and use the energy to transport various molecules across all cell membranes. The complex is believed to exist for most organisms. We find this complex in all three archaeal organisms.

Figure 6 provides an example of the FGAM . (5′-phosphoribosylformylglycinamidine) complex which catalyzes the energy transfer reaction: FGAM+ATP → AIR+ADP. In this reaction, ATP is converted to ADP with an accompanying release of energy.

### Table 2: *Pyrococcus furiosus*

<table>
<thead>
<tr>
<th>Clique ID</th>
<th>Size</th>
<th>Number of Edges</th>
<th>Average Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clique 133</td>
<td>4</td>
<td>0/6</td>
<td>1.0</td>
</tr>
<tr>
<td>Clique 7</td>
<td>7</td>
<td>21/21</td>
<td>0.982</td>
</tr>
<tr>
<td>Clique 33</td>
<td>4</td>
<td>0/6</td>
<td>1.0</td>
</tr>
<tr>
<td>Clique 12</td>
<td>4</td>
<td>0/6</td>
<td>0.999</td>
</tr>
<tr>
<td>Clique 59</td>
<td>7</td>
<td>21/21</td>
<td>0.985</td>
</tr>
<tr>
<td>Clique 96</td>
<td>6</td>
<td>15/15</td>
<td>0.990</td>
</tr>
<tr>
<td>Clique 178</td>
<td>4</td>
<td>0/6</td>
<td>1.0</td>
</tr>
<tr>
<td>Clique 178</td>
<td>4</td>
<td>0/6</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### 12.3 Organism-Specific Protein Complexes

Although finding conserved protein complexes across multiple organisms is the main goal of this research, we are also interested in finding the protein modules which are specific to a particular organism, because they uniquely characterize the organism. There are many examples of organism specific complexes found. For example, in Table 2, Clique 133 consists of four proteins connected by six edges, all of which are annotated as methanol dehydrogenase regulators. This module is not found in either of the other two archaea we have examined.

### 12.4 Overlapping Complexes - Proteins Shared Between Complexes

Using the algorithm discussed in §7, we compute the overlapped protein modules. We present an example in Figure 7. To make the figure readable, we replicate shared proteins (a protein is linked to its replica with purple lines). This Figure shows...
Table 3: *Sulfolobus*

<table>
<thead>
<tr>
<th>Clique 24</th>
<th>Size=3, number of edges = 3/3, average weight = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO2099</td>
<td>Cobalamine biosynthesis precorrin-1 methylase, putative (cbiF)</td>
</tr>
<tr>
<td>SSO2435</td>
<td>Uroporphyrinogen III methylase (cobA)</td>
</tr>
<tr>
<td>SSO2908</td>
<td>Uroporphyrin-III C-methyltransferase (siroheme synthase) (cygG)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clique 107</th>
<th>Size=7, number of edges = 21/21, average weight = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO1338</td>
<td>Permease, sugar transport protein, putative</td>
</tr>
<tr>
<td>SSO1539</td>
<td>Transport protein</td>
</tr>
<tr>
<td>SSO2039</td>
<td>Metabolite transport protein</td>
</tr>
<tr>
<td>SSO2528</td>
<td>Metabolite transport protein</td>
</tr>
<tr>
<td>SSO2852</td>
<td>Proline/betaine transporter, putative</td>
</tr>
<tr>
<td>SSO2938</td>
<td>Permease (proline/betaine)</td>
</tr>
<tr>
<td>SSO3121</td>
<td>Metabolite transport protein</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clique 135</th>
<th>Size=4, number of edges = 6/6, average weight = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO1457</td>
<td>NAD specific glutamate dehydrogenase (gdhA-1)</td>
</tr>
<tr>
<td>SSO1907</td>
<td>NAD specific glutamate dehydrogenase (gdhA-2)</td>
</tr>
<tr>
<td>SSO1930</td>
<td>NAD specific glutamate dehydrogenase (gdhA-3)</td>
</tr>
<tr>
<td>SSO2044</td>
<td>NAD specific glutamate dehydrogenase (gdhA-4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clique 3</th>
<th>Size=7, number of edges = 21/21, average weight = 0.901</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO1153</td>
<td>Acyl-CoA dehydrogenase (acd-1)</td>
</tr>
<tr>
<td>SSO1598</td>
<td>Partial transposase ISC1225</td>
</tr>
<tr>
<td>SSO2060</td>
<td>Acyl-CoA dehydrogenase (acd-3)</td>
</tr>
<tr>
<td>SSO2511</td>
<td>Acyl-CoA dehydrogenase (acd-4)</td>
</tr>
<tr>
<td>SSO2761</td>
<td>Acyl-CoA dehydrogenase (acd-5)</td>
</tr>
<tr>
<td>SSO2877</td>
<td>Acyl-CoA dehydrogenase (acd-6)</td>
</tr>
<tr>
<td>SSO3145</td>
<td>Acyl-CoA dehydrogenase (acd-7)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clique 6</th>
<th>Size=6, number of edges = 15/15, average weight = 0.943</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO0593</td>
<td>ATP phosphoribosyltransferase</td>
</tr>
<tr>
<td>SSO0596</td>
<td>Imidazoleglycerol-phosphate dehydratase (hisB)</td>
</tr>
<tr>
<td>SSO0597</td>
<td>Imidazoleglycerol-phosphate synthase, cyclase subunit (hisF)</td>
</tr>
<tr>
<td>SSO0599</td>
<td>Histidinol dehydrogenase (HDH) (hisD)</td>
</tr>
<tr>
<td>SSO0600</td>
<td>Amidotransferase hisH (hisH)</td>
</tr>
<tr>
<td>SSO0627</td>
<td>Phosphoribosyl-AMP cyclohydrolase (hisI)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clique 92</th>
<th>Size=10, number of edges = 45/45, average weight = 0.924</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO1209</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-1)</td>
</tr>
<tr>
<td>SSO2091</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-2)</td>
</tr>
<tr>
<td>SSO2150</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-3)</td>
</tr>
<tr>
<td>SSO2433</td>
<td>Carbon monoxide dehydrogenase, small chain (cutC-1)</td>
</tr>
<tr>
<td>SSO2637</td>
<td>Carbon monoxide dehydrogenase, small chain. Amino-end fragment (cutC-2)</td>
</tr>
<tr>
<td>SSO2639</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-4)</td>
</tr>
<tr>
<td>SSO2760</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-5)</td>
</tr>
<tr>
<td>SSO2942</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-6)</td>
</tr>
<tr>
<td>SSO3009</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-7)</td>
</tr>
<tr>
<td>SSO3239</td>
<td>Carbon monoxide dehydrogenase, large chain (cutA-8)</td>
</tr>
</tbody>
</table>

various complexes from the three archaea involved in amino acid biosynthesis. Proteins are shown to be shared among pathways for different amino acids.

12.5 Metabolic Pathway The histidine metabolic pathway is an ancient pathway that evolved before the diversification of bacteria, archaea and eucarya. Histidine is a precursor to several other amino acids and shares precursors in common with purines. It is interesting that one of the protein modules we discovered contains nine of the ten proteins in the histidine biosynthetic pathway.

![Figure 5: ABC transporter complex is conserved across 3 organisms. Yellow lines indicate proteins interactions. Black lines indicates orthologs.](image)

13 Summary

In this paper we describe how to use an efficient and versatile clique finding algorithm to compute cliques and quasi-cliques in a graph. We vigorously prove the correctness and convergence of the algorithm. We then apply the algorithm to three archaea organisms: *Halobacterium NRC-1*, *Pyrococcus furiosus*, and *Sulfolobus solfataricus*. We identified a large number of protein interaction modules. From there, using orthologs information, we discovered a large number of protein complexes conserved across the three organisms.

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


---

**Table 4: Halobacterium NRC-1**

<table>
<thead>
<tr>
<th>Clique</th>
<th>Size = 12, number of edges = 66/66, average weight = 0.980</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNG1769G</td>
<td>Htr5 transducer</td>
</tr>
<tr>
<td>VNG0355G</td>
<td>Htr14 transducer</td>
</tr>
<tr>
<td>VNG0783G</td>
<td>Htr6 transducer</td>
</tr>
<tr>
<td>VNG1856G</td>
<td>Htr3 transducer</td>
</tr>
<tr>
<td>VNG0614G</td>
<td>Htr16 transducer</td>
</tr>
<tr>
<td>VNG1523G</td>
<td>Htr8 transducer</td>
</tr>
<tr>
<td>VNG1759G</td>
<td>Htr7 transducer</td>
</tr>
<tr>
<td>VNG806G</td>
<td>Htr4 transducer</td>
</tr>
<tr>
<td>VNG1765G</td>
<td>Htr2 transducer</td>
</tr>
<tr>
<td>VNG1013G</td>
<td>Htr13 transducer</td>
</tr>
<tr>
<td>VNG1442G</td>
<td>Htr12 transducer</td>
</tr>
<tr>
<td>VNG0971G</td>
<td>Chemotaxis protein</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clique</th>
<th>Size = 6, number of edges = 15/15, average weight = 0.998</th>
</tr>
</thead>
<tbody>
<tr>
<td>VNG0371G</td>
<td>Acyl-CoA dehydrogenase</td>
</tr>
<tr>
<td>VNG1191GM</td>
<td>Acyl-CoA dehydrogenase</td>
</tr>
<tr>
<td>VNG1482G</td>
<td>Acyl-CoA dehydrogenase</td>
</tr>
<tr>
<td>VNG0679G</td>
<td>Acyl-CoA dehydrogenase</td>
</tr>
<tr>
<td>VNG2499G</td>
<td>Glutaryl-CoA dehydrogenase</td>
</tr>
<tr>
<td>VNG0775G</td>
<td>acyl-CoA dehydrogenase</td>
</tr>
</tbody>
</table>

---

**Table 4:** Halobacterium NRC-1

- **Clique 6**
  - Size=12, number of edges = 66/66, average weight = 0.980
  - VNG1769G: Htr5 transducer
  - VNG0355G: Htr14 transducer
  - VNG0783G: Htr6 transducer
  - VNG1856G: Htr3 transducer
  - VNG0614G: Htr16 transducer
  - VNG1523G: Htr8 transducer
  - VNG1759G: Htr7 transducer
  - VNG806G: Htr4 transducer
  - VNG1765G: Htr2 transducer
  - VNG1013G: Htr13 transducer
  - VNG1442G: Htr12 transducer
  - VNG0971G: Chemotaxis protein

- **Clique 8**
  - Size=6, number of edges = 15/15, average weight = 0.998
  - VNG0371G: Acyl-CoA dehydrogenase
  - VNG1191GM: Acyl-CoA dehydrogenase
  - VNG1482G: Acyl-CoA dehydrogenase
  - VNG0679G: Acyl-CoA dehydrogenase
  - VNG2499G: Glutaryl-CoA dehydrogenase
  - VNG0775G: acyl-CoA dehydrogenase

- **Clique 12**
  - Size=6, number of edges = 15/15, average weight = 0.994
  - VNG0882G: Imidazoleglycerol-phosphate synthase
  - VNG2087G: Imidazoleglycerol phosphate synthase subunit HisH
  - VNG1444G: Histidinol dehydrogenase
  - VNG2295G: Imidazoleglycerol-phosphate dehydratase
  - VNG2294G: Phosphoribosylformimin-5-aminoimidazole carbamoyl-ride isomerase
  - VNG2247G: ATP phosphoribosyltransferase

- **Clique 40**
  - Size=4, number of edges = 6/6, average weight = 0.985
  - VNG0086GM: Molybdopterin biosynthesis enzyme
  - VNG0085G: Molybdenum cofactor biosynthesis protein
  - VNG0090G: Molybdenum cofactor biosynthesis protein
  - VNG1273G: Molybdenum cofactor biosynthesis protein C

---

Figure 6: FGAM is conversed across 3 organisms. Yellow lines indicate proteins interactions. Black lines indicates orthologs.
Figure 7: Shared proteins in different protein complexes for tryptophan biosynthesis, histidine biosynthesis, anthranilate biosynthesis, etc. Purple lines connect the same protein occurring at different protein modules. Black lines indicate orthologs. Blue lines are the original interactions.

Figure 8: Histidine metabolism pathway (adopted from KEGG http://www.genome.jp/kegg/). Blues circles indicate proteins found in a protein module.