BIOINFORMATICS ORIGINAL PAPER Vol. 25 no. 14 2009, pages 1814–1821
doi:10.1093/bioinformatics/btp297

Systems biology

Structure discovery in PPI networks using pattern-based network decomposition

Philip Bachman[1] and Ying Liu[1,2,*]

[1]Department of Computer Science and [2]Department of Molecular Biology, University of Texas at Dallas, Richardson, TX 75083-0688

Received on December 1, 2008; revised on March 16, 2009; accepted on April 28, 2009

Advance Access publication May 15, 2009

Associate Editor: Trey Ideker

ABSTRACT

Motivation: The large, complex networks of interactions between proteins provide a lens through which one can examine the structure and function of biological systems. Previous analyses of these continually growing networks have primarily followed either of two approaches: large-scale statistical analysis of holistic network properties, or small-scale analysis of local topological features. Meanwhile, investigation of meso-scale network structure (above that of individual functional modules, while maintaining the significance of individual proteins) has been hindered by the computational complexity of structural search in networks. Examining protein–protein interaction (PPI) networks at the meso-scale may provide insights into the presence and form of relationships between individual protein complexes and functional modules.

Results: In this article, we present an efficient algorithm for performing sub-graph isomorphism queries on a network and show its computational advantage over previous methods. We also present a novel application of this form of topological search which permits analysis of a network's structure at a scale between that of individual functional modules and that of network-wide properties. This analysis provides support for the presence of hierarchical modularity in the PPI network of *Saccharomyces cerevisiae*.

Contact: ying.liu@utdallas.edu

1 INTRODUCTION

Recent advances in experimental techniques have led to a wealth of data describing the processes that produce life. Underlying many of these processes are complex networks of interacting proteins. New high-throughput techniques for detecting protein–protein interactions (PPIs) are quickly revealing the large sets of interactions between proteins in a range of model species (Aebersold and Mann, 2003; Fields, 2005). Both the size and the number of datasets describing these networks are growing rapidly.

As in the period ensuing development of high-throughput techniques for genome-scale sequencing, the current growth in interaction network data has led to a lively field of research: the development of computational tools and techniques for biological network analysis. Work is ongoing in extracting useful information from physical PPI networks, transcription networks (Guelzim *et al.*, 2002), metabolic networks (Jeong *et al.*, 2000; Lacroix *et al.*, 2006), gene expression data (Zhang *et al.*, 2005) and combination networks formed from superpositions of multiple network datasets (Yeger-Lotem *et al.*, 2004; Zhang *et al.*, 2005). Fortunately, previous research in related fields such as the analysis of social and complex networks (Albert *et al.*, 2000; Strogatz, 2001) provides those developing analytical techniques for application to biological networks with a solid theoretical foundation upon which to build.

In the study of PPI networks, previous work has primarily followed either of two approaches: large-scale statistically oriented study, or exact local topological analysis. Studies focusing on the large scale have been useful in uncovering aspects of high-level structure that seem to occur across all of the currently available PPI networks. Discoveries coming from this research include the particular degree distribution (Jeong *et al.*, 2000), clustering properties (Spirin and Mirny, 2003; Yook *et al.*, 2004) and possible hierarchical structure of the examined networks (Wuchty and Almaas, 2005; Yook *et al.*, 2004). Thus far, studies examining the networks on a smaller scale have focused on functional module detection (Bader and Hogue, 2003; Enright *et al.*, 2002; Spirin and Mirny, 2003; Ziv *et al.*, 2005), protein function prediction (see Sharan *et al.* 2007 for a review of network-centric methods) and motif discovery (Chen *et al.*, 2006; Milo *et al.*, 2002; Wuchty *et al.*, 2003; Yeger-Lotem *et al.*, 2004). Both module membership and protein function have been shown to be strongly influenced/expressed by the topology of the interactions surrounding a protein (Wuchty and Almaas, 2005; Yook *et al.*, 2004). More recently, a bridge has been provided between network-wide and local properties through the introduction of network similarity measures based on sub-graph frequencies (Pržulj *et al.*, 2004), including one called graphlet degree distribution, which generalizes degree of distribution using the distribution of small sub-graphs around each vertex (Pržulj, 2007).

Methods such as motif discovery and network comparison by sub-graph frequencies or graphlet degree of distributions require searching for exact structural patterns within PPI networks. Such searches can be performed either by sampling sub-graphs from a network or by enumerating the sub-graphs within a network that are isomorphic to some query graph. Sub-graph querying (Grochow and Kellis, 2007) and sub-graph sampling (Wernicke, 2006) have both been used for motif discovery, while sampling has also been applied to graphlet-based network comparison (Pržulj *et al.*, 2006). Unfortunately, both the sampling and sub-graph

*To whom correspondence should be addressed.

query-based approaches require solving computationally difficult problems. Sampling requires that *general* sampled graphs be given a canonical label (Babai and Luks, 1983) and sub-graph querying requires solving many instances of the sub-graph isomorphism problem. The computational complexity of both approaches limits the size of the structures for which precise searches are tractable. Yet, it may be desirable to search for exact appearances of meso-scale structures within PPI networks, as analysis at this scale is likely to provide information describing the existence and structure of higher order relations between groups of proteins. Such higher order relations may represent the interactions occurring between functional modules and/or complexes, as well as the way in which such interactions are arranged. These group-level interactions may provide evidence for or against meta-modular structure in a network, such as hierarchical modularity in PPI networks.

In this article, we present an algorithm based on the creation and use of certain constraint sets that permit rapid elimination of candidate vertices during the backtracking procedure, which forms the core of sub-graph isomorphism search. This algorithm offers a significant computational advantage over previous methods, such as those used in motif discovery and graphlet analysis. Further, to show that applications of topological search are yet to be exhausted, we present a novel method of using sub-graph queries for investigating the meso-scale structure of PPI networks. This method extracts regions of a network matching the topologies of query patterns designed to highlight potentially significant aspects of meso-scale network structure. We show that, using this method, a strong argument can be made for the existence of hierarchical modularity in PPI networks.

This article is organized as follows. In Section 2, we give a brief description of the previously developed backtracking and *symmetry-breaking* (Grochow and Kellis, 2007) techniques used by our algorithm followed by a description of our algorithm. In Section 3, we show the relative efficiency of our algorithm. In Section 4, we describe our new method for pattern-based network decomposition and present results from its application to the PPI network of *Saccharomyces cerevisiae*. Throughout this article, we use the terms graph/network and vertex/node interchangeably due to their differing usage in the bioinformatics and computer science communities.

2 THE ALGORITHM

Our algorithm is designed to solve the following problem: given a query graph G_q, find all sub-graphs in a source graph G_s that are isomorphic to G_q. Our search algorithm comprises three components. The first is a basic backtracking search used in previous motif discovery algorithms, which is then enhanced by the second, a symmetry-breaking technique presented by (Grochow and Kellis, 2007).

To these, we add a third component, a constraint set for each vertex v in G_q that describes the set of all vertices in G_s that are potentially mappable onto v under some mapping of G_q onto an isomorphic sub-graph of G_s. These constraint sets permit quick elimination of candidate vertex pairs during the backtracking search, as many vertices in the source graph will be absent from a given constraint set. Generating these constraint sets requires a constraint database that is populated by performing an initial set of specific sub-graph queries. This database can be quickly populated using a modified version of the basic backtracking search. Our constraints are linked to the basic bactracking search, with symmetry-breaking, by the graph-theoretic concepts of canonical labels and automorphism orbits.

2.1 Graph-theoretic concepts

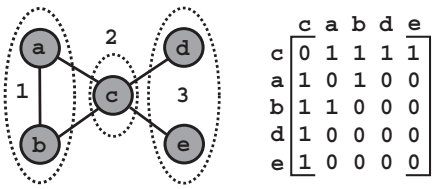
We define a graph G as $G=(V,E)$, where V is the set of vertices in the graph and E is the set of its edges. We define an ordering of a graph to be a specific ordering of its vertices. Given an ordering o, one can produce the adjacency matrix M_o that o induces by ordering the vertices of G as dictated by o and then generating an adjacency matrix that respects this ordering.

If two graphs G_i and G_j are isomorphic, then there exist orderings o_i of G_i and o_j of G_j such that $M_{o_i}=M_{o_j}$. Henceforth, if we say $G_i=G_j$, then we mean that G_i and G_j are isomorphic. A canonical labeling for a graph is given by any labeling function l such that, for any two graphs G_i and G_j:

$$l(G_i)=l(G_j) \Longleftrightarrow G_i=G_j.$$

One canonical label can be found by interpreting all possible orderings of a graph's vertices and selecting the ordering $o_{\max}$ such that $M_{o_{\max}}$, when interpreted as a binary number, is maximized (McKay, 1981). For example, we create this binary number by concatenating the rows of an adjacency matrix (Fig. 1). In this article, we refer to any consistent canonical labeling of some graph G as $l_C(G)$.

An automorphism of a graph G is any mapping of the graph's vertices onto each other induced by any two orderings o_i and o_j such that $M_{o_j}=M_{o_i}$. We refer to the set of all automorphisms of G as Aut(G). The automorphism orbits of G are the maximal sets of the vertices in G that are closed under all mappings in Aut(G). That is, an automorphism orbit A, of a graph G, contains all vertices in G, which are mapped onto each other by automorphisms in Aut(G). We refer to the automorphism orbit to which a vertex v belongs as AutOrb(v). We also define a canonical ID for each automorphism orbit of a graph G with respect to our definition of a canonical matrix. We generate the ID of each automorphism orbit by examining the canonical matrix for G and giving sequential IDs to the automorphism orbits of G based on their order of first appearance in the matrix. Though vertices from various automorphism groups may intermingle within the ordering corresponding to a canonical labeling, we look only at the order in which each automorphism group has a vertex first appear, which will be consistent across the possibly numerous canonical orderings/labelings given by a consistent canonical



	c	a	b	d	e
c	0	1	1	1	1
a	1	0	1	0	0
b	1	1	0	0	0
d	1	0	0	0	0
e	1	0	0	0	0

Fig. 1. On the left is a graph G and on the right is its canonical matrix form. The canonical matrix form was determined by finding the ordering of the vertices that gave the largest binary number when the rows of the matrix were concatenated. The corresponding number in this case is 01111101…10000. The automorphism groups of G have been circled and labeled as 1, 2 and 3. Under our ID assignment scheme, automorphism group 2 would receive the ID 0 as vertex c appears first in the canonical matrix ordering.

labeling function. For an example of automorphism groups and their ID assignments, see Figure 1.

2.2 Backtracking and symmetry-breaking

The basic backtracking algorithm for finding all sub-graphs in a source graph G_s that are isomorphic to a query graph G_q attempts, for each vertex v in G_s and each vertex u in G_q, to find all isomorphic mappings of G_q onto G_s such that u is mapped onto v. To do this, a mapping, I, of G_q onto G_s is initialized for each such (v,u) pair. Backtracking then recursively extends I using each pair of compatible vertices v' in G_s and u' in G_q such that v' and u' are both adjacent to vertices already in I. Here, compatible pairs of vertices are those that can extend the mapping I while maintaining the isomorphism of the partial mapping of G_q onto G_s induced by I. An isomorphism has been found once I maps all vertices in G_q onto compatible vertices in G_s.

Unfortunately, this naïve method finds, for each sub-graph g in G_s, such that $g = G_q$, all mappings of G_q onto g. The number of such mappings is equal to the number of automorphisms in $\mathrm{Aut}(G_q)$, which grows factorially with the number of vertices in G_q. We can avoid these 'repeated' mappings by using symmetry-breaking constraints. The symmetry-breaking constraints ensure that only one distinct mapping of G_q onto each such g is found, thus reducing the number of found mappings by a factor equal to $|\mathrm{Aut}(G_q)|$. Briefly, the symmetry-breaking constraints rely on assigning distinct integer IDs to all vertices in G_q and G_s, and requiring that any mapping of G_q onto G_s induces numerical orderings of the vertex IDs that are consistent across the vertices in both G_q and G_s. Determining what this ordering should be, so that only one allowable mapping of G_q onto G_s exists for any sub-set of the vertices in G_s, is at the heart of symmetry-breaking constraint generation. Full explanation of the generation and use of symmetry-breaking constraints can be found in Grochow and Kellis (2007).

2.3 Generating constraint sets for the source graph

The database used for per-vertex constraint generation contains entries for each query graph G_q used to populate it. The entry for each G_q is a list of constraint sets, one for each automorphism orbit of G_q. The entry for G_q is referenced by $l_C(G_q)$ and the entries for its automorphism orbits are referenced by their respective canonical IDs. The constraint set for an automorphism orbit A of query graph G_q contains the name/label of each vertex v in the source graph G_s such that there exists a mapping of G_q onto some isomorphic sub-graph g of G_s that maps some member of A onto v.

The constraint sets for each query graph G_q can be generated by performing a slightly modified version of the basic backtracking search described above. This search is performed by checking each vertex v_s in G_s against a representative v_q from each automorphism orbit A of G_q. If an isomorphic mapping exists which maps v_q onto v_s, then v_s is added to the constraint set for A. This isomorphism also implies a mapping for each other vertex u_s from G_s onto which it maps some vertex u_q from G_q. Each such u_s can be added to the constraint set for $\mathrm{AutOrb}(u_q)$. This allows the skipping of any future checks for some v_s against some v_q where v_s is already in the constraint set for $\mathrm{AutOrb}(v_q)$. In practice, this allows quickly covering a graph with a number of mappings of G_q onto G_s that is much smaller than the total number of distinct appearances of G_q in G_s.

If sub-graphs with n vertices will be sampled while generating constraint sets for each vertex in a query graph, as discussed in the next section, then it is best to generate the constraint sets for all possible graphs of size n using the method described above. The database can be bootstrapped, starting with all connected graphs of some small size k, and then using the generated database to accelerate the generation of the database entries for all connected graphs of size $k+1$, and so on, until size n is reached.

In practice, this database can be generated quickly, as supported by our speed tests (Section 3). We used McKay's *geng* tool (McKay, 1998) for enumerating non-isomorphic, connected graphs of a given size. The database can be generated on the fly and stored in memory as a simple hash-map for smaller graph sizes. However, the number of graphs for sizes greater than seven makes this impractical. Thus, for larger graph sizes we stored this database in an on-disk hash-map, which also allows it to be used in later searches without regeneration.

2.4 Vertex constraints for a query sub-graph

Given a constraint database for a source graph G_s, a set of constraints for each vertex in any query graph G_q can be generated as follows.

1. For each vertex v in G_q, a sub-graph g from G_q having n vertices and including v is sampled, where n can be equal to any of the graph sizes that were used in generating the constraint database.
2. For each g, $l_C(g)$ is calculated and used to retrieve the vertex constraint sets for the automorphism orbits of g from the database.
3. For each vertex u in each sampled sub-graph g, the vertex constraint set for $\mathrm{AutOrb}_g(u)$ is added to a list of constraint sets associated with u. Note that each u is necessarily in G_q, as each g is a sub-graph of G_q.
4. The final constraint set for each vertex in G_q is generated by taking the intersection of all of the constraint sets that were associated with it during step 3.

Using these constraints during the backtracking search is straightforward. When checking for the compatibility of some vertex v from G_s with some vertex u from G_q, as previously described, one first checks if v is in the constraint set for u. If it is not, then the vertices are incompatible. If it is, then one continues the compatibility check as before. In Theorem 1, we show that the intersection of constraints performed in step 4 does not preclude any viable mappings of G_q onto G_s, as only source graph vertices fundamentally incompatible with a query graph vertex will fail to appear in one or more of the constraint sets associated with that vertex during step 3.

THEOREM 1. *Each vertex v in the source graph G_s that does not appear in the final constraint set for some vertex u in the query graph G_q cannot be mapped onto u by any isomorphic mapping of G_q onto G_s.*

PROOF. If a vertex v does not appear in the constraint set for u, then there exists a sub-graph g of G_q, with g sampled in step 1, in which u is a member of an automorphism orbit A such that v cannot be mapped onto *any* member of A. If v were mappable onto a member of A, then v would have appeared in the constraint set for

$\text{AutOrb}_g(u)$ that was associated with u in step 3. As u is a member of A, v cannot be mapped onto u under any isomorphic mapping of g onto G_s. As g is a sub-graph of G_q, this implies that v is not mappable onto u under any isomorphic mapping of G_q onto G_s.

2.5 Algorithm speed results

To show the computational benefits offered by our constraints, we measured the ratio between times required for constrained and unconstrained searches across a range of query graph sizes and edge densities. In these tests, the only additional constraints used during constrained search were those described in the previous section. Both constrained and unconstrained searches made use of symmetry-breaking and several additional heuristics during backtracking. The additional heuristics ensured that, at any point during backtracking, all mapped pairs of vertices (v,u), with v in the source graph and u in the query, were such that v had at least as many unmapped neighbors as u. These heuristics are equivalent to those used in the VF2 graph matching algorithm (Cordella *et al.*, 2001), as applied to undirected graphs.

Thus, our tests measure the relative reduction in search space exploration when our constraints are added to those presented in previous work. We also note that, as with any other algorithm that finds *all* instances of an arbitrary sub-graph G_q in a source graph G_s, our algorithm has a running time that is exponential in the size of G_q, as the number of instances of G_q in G_s is potentially exponential. Thus, even if we could find each instance of G_q in constant time (e.g. using a constant time algorithm/oracle for sub-graph isomorphism), the total time for counting *all* instances would remain exponential.

We measured time ratios between constrained and unconstrained searches for queries comprising from 8 to 19 vertices. At each query size, we independently measured the time ratios at four different query densities: 0.2, 0.4, 0.6 and 0.8. For each query size n, at each query density d, 100 connected Erdos–Renyi random graphs of size n and edge density d were generated, and the time ratio was measured as: $\text{time}_u/\text{time}_c$, where time_u and time_c were, respectively, the total times required for unconstrained and constrained processing of these queries. The measured time ratios are shown in Figure 2, which also includes the cumulative time ratios between unconstrained and constrained searches taken across all densities, for each query size. The cumulative time ratios spanned from a minimum ratio of 4.41 for query graphs with eight vertices to a ratio of 142.5 for query graphs with 17 vertices, while the ratios for larger graphs began to suffer due to an artificial upper bound that we placed on both the constrained and unconstrained search times. This bound was necessary due to the presence of pathological queries that required an impractical time to complete.

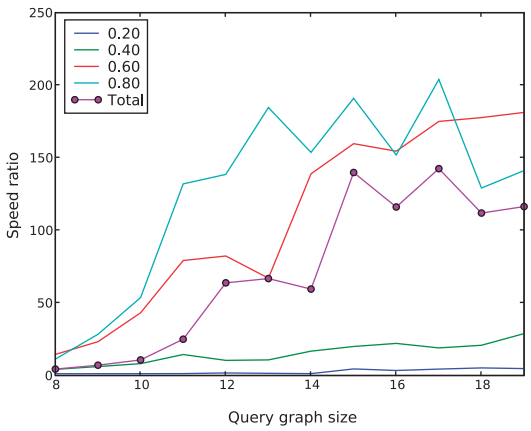


Fig. 2. This figure plots the ratio between the times required for unconstrained and constrained searches across sets of 100 randomly generated queries at each edge density and query size, with higher ratios representing greater benefits from using the constraints. The edge densities used were 0.2, 0.4, 0.6 and 0.8, with separate speed ratio plots given for each density. A plot of the cumulative speed ratio across all densities is also given (i.e. 'Total').

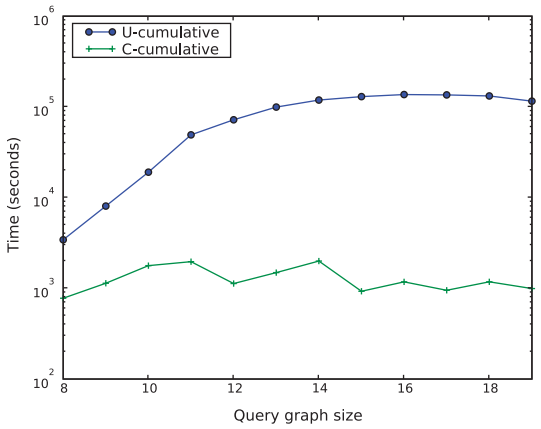


Fig. 3. This figure gives the cumulative constrained and unconstrained search times required for processing 400 queries at each query size from 8 to 19. At each query size, 100 random connected query graphs were processed for each of the edge densities: 0.2, 0.4, 0.6 and 0.8. The unconstrained search time appears to approach a limit as we stopped individual queries after 1000 s of computation, with this limit being achieved more frequently with increasing query size and density. The minimum and maximum cumulative constrained search times were 778 s and 1998 s, respectively, with the maximum occurring at a query size of 14, primarily due to a single pathological query, which forced the constrained search to time-out.

In Figure 3, we also provide a plot of the cumulative times for processing all 400 queries of various densities at each query size. In this plot, the cumulative time for unconstrained searches appears to plateau due to our imposed limit on search time, with the maximum possible cumulative time at each graph size being 400 000 s, as we set the individual query time-out to 1000 s. It was common for queries with 15 or more nodes and with edge densities of 0.6 or 0.8 to time-out during unconstrained search (e.g. 67 out of 100 queries with 18 nodes and an edge density of 0.8 timed-out). Only one constrained search out of the 3600 performed timed-out. Generally, the unconstrained search time was strongly affected by the edge-density of a query, with significantly more time required when processing denser queries. The effects of density were drastically reduced when using our constraints, thus explaining the differences in the time-ratio curves plotted in Figure 2, wherein the time-ratios for denser queries favor constrained search more significantly than the time-ratios for sparser queries.

It is also important to note the time required to fill the database used in constraint generation. For every distinct eight-node graph g, the completed database contained lists of all vertices in the source PPI network that were mappable onto each automorphism group

Table 1. CPU time for exhaustive search of all seven and eight vertex graphs

Graph size	Constrained time (s)	Unconstrained time (s)	Time saved (s)	Ratio
7	285	1811	1526	6.35
8	7258	44 619	37 361	6.14

of g. The database was bootstrapped as previously discussed, and the time required to do so was 4477 s. This is significantly <37 361 s that were saved by using the database during exhaustive search for all connected eight vertex graphs, as shown in Table 1. Such an exhaustive search might be used in calculating sub-graph densities for network comparison as in Pržulj *et al.* (2004), or in motif discovery. Additionally, the database produced in order to generate the constraints for each query graph is an inverse of the mapping used in Pržulj (2007) for comparing the vertices in a source network based on the automorphism groups of small sub-graphs to which they can be mapped during isomorphic mappings of those sub-graphs onto the source network.

3 PATTERN-BASED NETWORK DECOMPOSITION

While the computational advantages of our algorithm are useful in existing applications of topological search to PPI network analysis, we feel that the potential applications of such search have yet to be fully explored. Thus, in this section, we present a novel approach to using sub-graph queries that allows a PPI network to be decomposed into sub-networks exhibiting specific structural patterns.

Specifically, we have developed a method intended for investigating the meso-scale interaction patterns both within and between densely interacting groups of proteins (e.g. functional modules and complexes) that are typically obscured by the inherent noise in high-throughput interaction datasets. These interaction patterns can be revealed by creating generalizations of their topological structure and then searching for all appearances of each generalization in the source network. These searches are performed using the algorithm presented in Section 2. The next sub-section describes the design and application of such pattern generalizations. A similar approach based on generalizing the topology of network motifs was presented and applied to transcription networks in Kashtan *et al.* (2004). Another method that relies on searching for specific topological structures in a PPI network was presented in Palla *et al.* (2005), in which network regions coverable by partially overlapping cliques of the same size were taken to be functional modules. Our method differs from Palla *et al.* (2005) in that it allows searches for arbitrary structures, including overlapping cliques, some of which may uncover relationships beyond co-membership in functional modules and complexes.

3.1 Generating and using query patterns

Given a specific form of inter-module interaction, deriving generalized query patterns is simple. As large near-cliques are likely to be completely coverable by multiple smaller cliques, a graph property that Grochow and Kellis referred to as *combinatorial clustering*, a query pattern may be created at the smallest size which adequately represents some desired structural features and allowed to 'expand' to cover larger sub-graphs, which still fit the interaction pattern that it was designed to represent. Two examples of such pattern expansion are shown in Figure 4, which shows how a dense module may be covered by a small clique query pattern and how a pair of interacting/overlapping modules may be covered by a small pair of overlapping cliques, such as those that Figure 5 refers to as a linear chain of 'interacting members' pattern. A pattern generalization created using this property of graphs can be used to *filter* the source network.

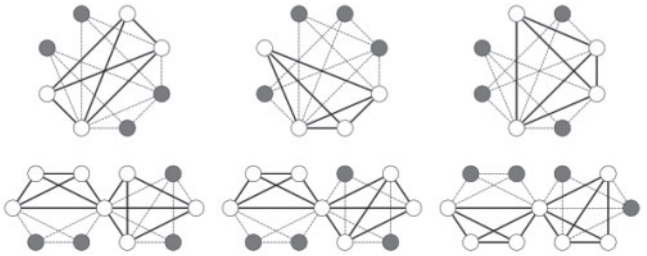


Fig. 4. The top row shows a four-node clique covering all edges and nodes in a dense eight-node graph. The bottom row shows overlapping four-node cliques expanding to fully cover overlapping dense six-node graphs. In this figure, the light colored nodes and the solid edges represent those covered by each simple pattern as it 'expands' to cover a larger graph containing *all* nodes and edges shown. The expansion process shown would result in both of the larger graphs being included, in their entirety, in the extracted networks (ENs) for their respective pattern generalizations, as the closure over all nodes and edges included in isomorphic mappings of the pattern generalizations onto the larger graphs includes all nodes and edges in the larger graphs.

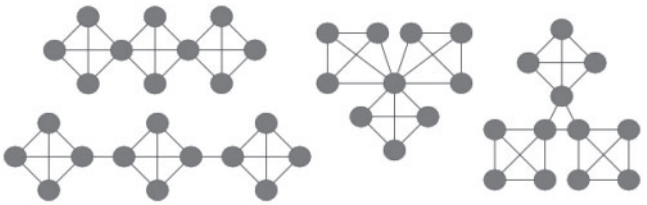


Fig. 5. Illustration of four sample pattern generalizations. The two left-most patterns are a linear chain of 'shared member' modules (top) and a linear chain of 'interacting member' modules (bottom). The two right-most paterns are a radial, hub-like arrangement of 'shared member' modules (left) and a radial, hub-like arrangement of 'interacting member' modules (right). These patterns were all significantly ($P << 0.01$) enriched in the source network compared with an ensemble of 100 random networks generated by edge-swapping, which preserves the source network's degree of distribution.

Filtering a source network using a generalized query pattern proceeds as follows. First, all instances of the query in the source network are found. Then, all edges and vertices from the source network that do not appear in some instance of the query are removed. Through this process, the source network is filtered, allowing direct inspection of all regions of the source network that have topologies matching the pattern that the query was designed to represent. We refer to a filtered network thusly acquired as an EN. A similar approach to easing visualization of network modules by using k-cores to decompose a PPI network was examined in Bader and Hogue (2002) and in Tong *et al.* (2001). Our approach here is more general, allowing the application of arbitrary filters.

Selecting appropriate query patterns is essential to revealing biologically relevant structures embedded in the source network. Several patterns that we applied as filters are shown in Figure 5. The two patterns that we call linear chains are meant to highlight two possible ways for communication to occur between functional modules in the PPI network. The 'shared member' chain represents communication between modules through a protein that participates in both of the modules. The 'interacting member' chain is intended to map onto modules that have a looser coupling, wherein the communication is via proteins specific to each module. As shown by the lower example in Figure 4, the proteins and interactions in the modules themselves will be well covered by the clique portions of the linear chain patterns. Variations on these patterns are also potentially useful. For example, patterns could be designed for finding modules which share multiple proteins instead of only one.

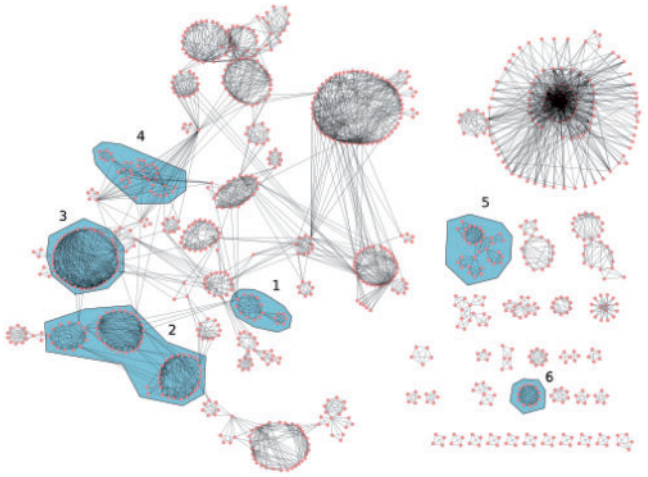


Fig. 6. (**a**) Two overlapping modules taken from the EN that was generated by using a four-node clique as a filter. Both modules 1 and 2 show strong enrichments for MIPS categories 11.02.03 (i.e. mRNA synthesis) and 11.02.03.01 (i.e. general transcription activities). For the former category, modules 1 and 2 show respective enrichment *P*-values of 10^{-11} and 10^{-4}. For the latter category, the modules show respective enrichments of 10^{-15} and 10^{-5}. However, for category 11.02.03.01.04 (i.e. transcription elongation), a sub-category of 11.02.03.01, module 2 shows an enrichment of 10^{-9}, with module 1 showing no enrichment. Module 1 shows an enrichment of 10^{-12} for category 16.03.01 (i.e. RNA binding), a category for which module 2 is not enriched. (**b**) A series of overlapping modules that appeared as shown when the source network was filtered using the pattern that Figure 5 referred to as a linear chain of shared members. These modules were hidden in the largest connected component of the EN shown in Figure 7, thus showing the utility of more selective filter patterns. Both groups 1 and 2 in this figure show enrichment *P*-values $< 10^{-7}$ in MIPS categories 10.03.01.01.01 and 11.02.03.04. These categories represent the G1 phase of mitotic cell division and transcription control, respectively. Though they share function, these modules also have specializations. Module 1 shows an enrichment of 10^{-10} in MIPS category 01.03.16.01 (i.e. RNA degradation), a category for which no proteins exclusively in module 2 are annotated. Module 2 shows strong enrichments in MIPS categories 14.07.04 (i.e. modification by acetylation, deacetylation) and 42.10.03 (i.e. organization of chromosome structure), with those values being 10^{-11} and 10^{-8}, respectively. Only one protein in module 1 and not in module 2 is annotated for either of these categories. This connected component stood alone in the EN from which it was taken.

3.2 An application of pattern-based network decomposition

We applied this technique using several different pattern generalizations. These searches provided support for hierarchical modularity in the Yeast PPI network. All of the results in this section were generated using a recent version of the Core PPI network for Yeast downloaded from DIP (Salwinski *et al.*, 2004). This network describes 17 000 interactions between 5000 proteins. Some pattern generalizations that we used are shown in Figure 5. All of the generalizations that we searched for appeared as significantly enriched motifs in the source networks with respect to an ensemble of 100 random networks with the same degree of distribution as the source network, indicating potential biological significance. During our analysis, we occasionally had to determine boundaries between overlapping groups of proteins (e.g. Fig. 6). We determined these boundaries heuristically, by looking for 'bottlenecks' between groups of densely interacting proteins. These bottlenecks exists wherever the overlap between two groups of proteins is much smaller than the size of either group.

The most immediate results and the largest EN both came from using a four-node clique as the query pattern (Fig. 7). This EN contained all proteins and all PPIs in the source network that participated in any four-node clique. Some immediately interesting observations can be made just from the size of the EN. The EN contained 74.8% of all nodes in the source PPI network that have degree greater than two (i.e. all nodes possibly involved in a clique of size four). It also contained 67.1% of all edges in the source network and 81.9% of all edges between nodes with degree greater than two (i.e. all edges possibly involved in a clique of size four). These numbers are significantly higher than would be expected in a random network having the same degree sequence. In 1000 random networks created to share the source network's degree of distribution, four-node cliques covered 2.6% of the nodes and 2.5% of the edges on average.

The EN consisted of one large component having 587 nodes and 2401 edges, one smaller component having 119 nodes and 792 edges, and a number of still smaller components. These components likely represent the most evolutionarily conserved core of the Yeast PPI network. It was shown by Wuchty *et al.* (2003) that proteins participating in four-node cliques have an evolutionary conservation rate that is over 400 times higher than that which would be expected. The same work also observed that these cliques display a remarkable level of functional homogeneity, in agreement with our observation of strong functional enrichment among modules in the EN.

Examination of both the components in the EN and their constituent proteins' functional annotations supported the hierarchical modularity of the source network. While the functional modules detected by clustering methods such as MCODE (Bader and Hogue, 2003) tend to be small and highly specialized, it becomes evident that these small modules are linked together into larger, functionally related groups of modules when pattern-based decomposition is applied. This hierarchical structure was shown by adjacent small modules showing both shared and distinct functional enrichments, calculated using the hyper-geometric mean and the annotations in the MIPS CYGD (Mewes and *et al*, 2002). Figure 6a gives an example of such a relationship between adjacent modules,

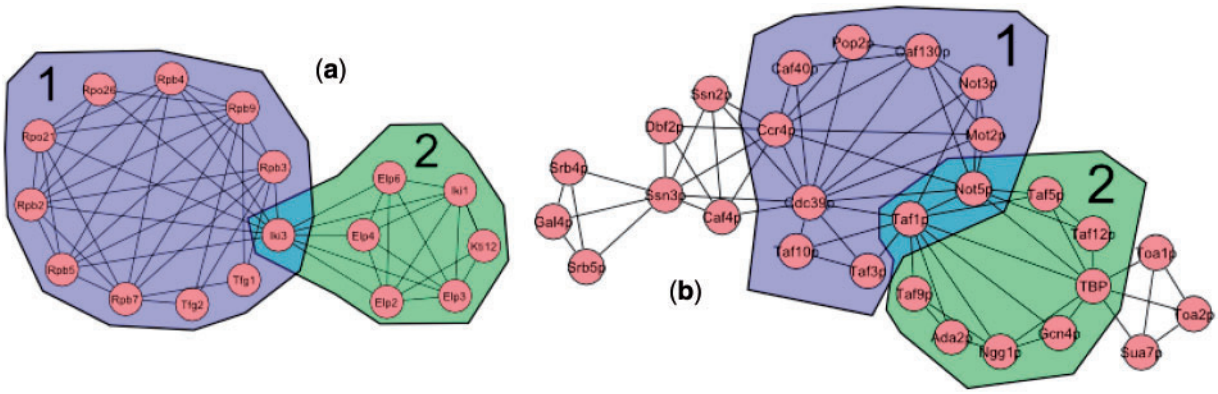


Fig. 7. The network extracted using a four-node clique as a query pattern. Group 1 contains the modules described in Figure 6a. Group 2 contains the overlapping modules from Figure 6b. Group 3 represents the majority of proteins in the complex with PPI ID 15740, the proteasome complex. Group 4 contains many of the nuclear pore (NUP) proteins. This apparently nebulous group was separated into its own component when a more complex query pattern involving overlapping cliques was used. The largest, densest portion of group 5 contains many SEC proteins implicated in vesicular fusion and the small cluster at approximately 4 o'clock contains all COG proteins listed by MIPS for vesicular recycling. This group thus diplays shared high-level function, with differences at lower levels between adjacent modules. Group 6 has a functional enrichment of 10^{-24} in MIPS category 10.03.01.0 (M Phase). It covers all proteins in the complex with PPI ID 16672 which was listed at the time this paper was written as supported only by 'confirmational text mining'.

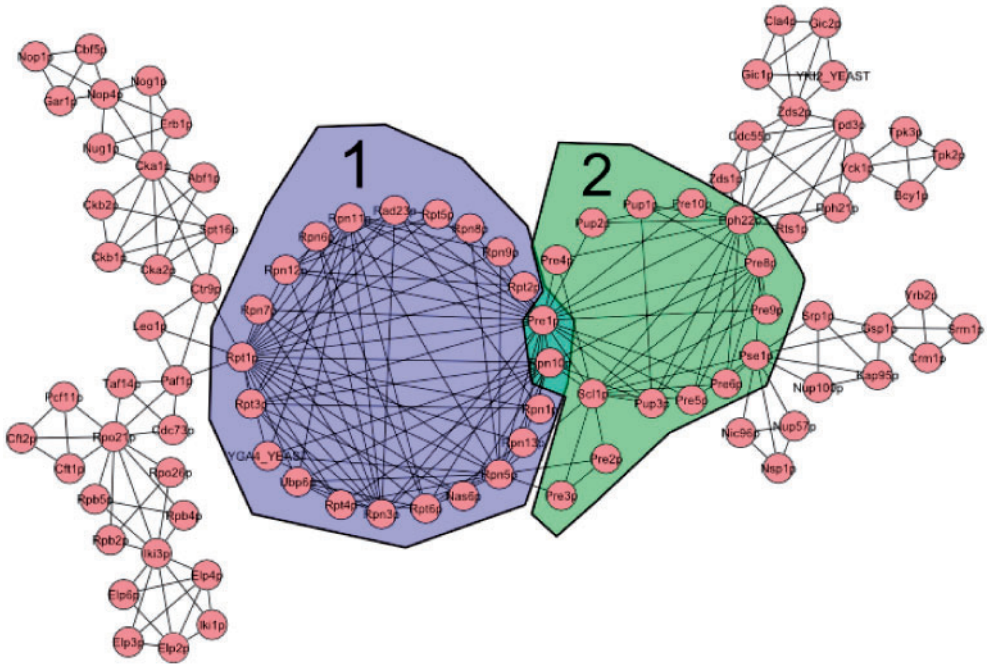


Fig. 8. This figure shows a pair of protein complexes that are revealed when the linear chain of shared members pattern is applied to the source network. Together, groups 1 and 2 represent group 3 in Figure 7, which comprises the majority of proteins in the proteasome complex. The separation created between groups 1 and 2, through filtering with a more specific pattern, is supported by their MIPS complex annotation, with group 1 representing most of MIPS complex 360.10.20 (19/22S proteasome) and group 2 representing most of MIPS complex 360.10.10 (20S proteasome). The connected component in this figure stood alone in the EN from which it was taken.

as observed in the EN. Similar evidence for hierarchical modularity was previously presented by Rives and Galitski (2003).

When we used more selective query patterns to filter the network, relationships that were hidden in the EN just described became visible. One such set of proteins and interactions was revealed when we filtered the source network using the pattern that Figure 5 refers to as a linear chain of 'shared member' modules. This subnetwork is described in Figure 6b. It contains two adjacent modules which both appear to participate in the G1 phase of mitotic cell division and transcription control. However, one of these modules

shows specialization for RNA degradation while the other shows specialization for modification by acetylation, deacetylation and organization of chromosome structure.

Using the same linear chain of 'shared member' pattern, another relationship hidden in the less selective EN was revealed, as shown in Figure 8. This involves the large protein group labeled 3 in Figure 7. In the earlier EN, this group contained PRE, PUP, RPN and RPT proteins which represent the majority of proteins in the proteasome complex. In the more selective EN, this group was split into two sub-components, one containing the PRE and PUP proteins and one containing the RPN and RPT proteins. This division is significant as these groups are identified in the MIPS complex catalog as representing two separate complexes, with the former representing complex 360.10.10 (i.e. 20S proteasome) and the latter representing complex 360.10.20 (i.e. 19/22S proteasome). These results, which were not apparent in the less selective EN, both show a pattern of shared and differing function that supports hierarchical modularity.

4 CONCLUSION

Analysis of the complex networks formed by the molecular interactions underlying cellular processes is an important area of research. Many of the interesting related problems are computationally hard, and require sophisticated algorithms to keep up with growing datasets. In this article, we have presented an algorithm for one such problem, sub-graph isomorphism, that is more efficient than previous algorithms.

Existing analyses of PPI networks have primarily focused on either one/two body network-wide properties such as degree of distributions and clustering coefficients, or local features such as motifs and functional modules. We have shown that, while our algorithm can benefit existing approaches to network analysis, the sub-graph queries that it performs have potential for further use. In concert with suitable query patterns that exploit some simple properties of graphs, query-based graph search can be used to examine network structure at a scale that reveals relationships within and between groups of interacting proteins. In the Yeast PPI network, these relationships support a hierarchical modularity. Insight into the processes by which such aspects of network structure might evolve is essential to our understanding of the functioning and evolution of biological systems as a whole.

Conflict of Interest: none declared.

REFERENCES

Aebersold,R. and Mann,M. (2003) Mass spectrometry-based proteomics. *Nature*, **422**, 198–207.

Albert,R. *et al*. (2000) Error and attack tolerance of complex networks. *Nature*, **406**, 378–381.

Babai,L. and Luks,E.M. (1983) Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. Boston, MA, USA, pp. 171–183.

Bader,G.D. and Hogue,C.W. (2002) Analyzing yeast protein-protein interaction data obtained from different sources. *Nat. Biotechnol.*, **20**, 991–997.

Bader,G.D. and Hogue,C.W. (2003) An automated method for finding molecular complexes in large protein networks. *BMC Bioinformatics*, **4**, 2.

Chen,J. *et al*. (2006) Nemofinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Datamining*. Philadelphia, PA, USA, pp. 106–115.

Cordella,L.P. *et al*. (2001) An improved algorithm for matching large graphs. In *Proceedings of the 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen.* Ischia, Italy, pp. 149–159.

Enright,A.J. *et al*. (2002) An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, **30**, 1575–1584.

Fields,S. (2005) High-throughput two-hybrid analysis: the promise and the peril. *FEBS J*., **272**, 5391–5399.

Grochow,J.A. and Kellis,M. (2007) Network motif discovery using subgraph enumeration and symmetry-breaking. In *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology, RECOMB 2007*. Oakland, CA, USA, pp. 92–106.

Guelzim,N. *et al*. (2002) Topological and causal structure of the yeast transcriptional regulatory network. *Nat. Genet.*, **31**, 60–63.

Jeong,H. *et al*. (2000) The large-scale organization of metabolic networks. *Nature*, **407**, 651–654.

Kashtan,N. *et al*. (2004) Topological generalizations of network motifs. *Phys. Rev. E*, **70**, 031909.

Lacroix,V. *et al*. (2006) Motif search in graphs: Application to metabolic networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **3**, 360–368.

McKay,B.D. (1981) Practical graph isomorphism. *Congressus Numerantium*, **30**, 45–87.

McKay,B.D. (1998) Isomorph-free exhaustive generation. *J. Algorithms*, **26**, 306–324.

Mewes,H.W. *et al* (2002) Mips: a database for genomes and protein sequences. *Nucleic Acids Res.*, **30**, 31–34.

Milo,R. *et al*. (2002) Network motifs: Simple building blocks of complex networks. *Science*, **298**, 824–827.

Palla,G. *et al*. (2005) Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, **435**, 814.

Pržulj,N. (2007) Biological network comparison using graphlet degree distribution. *Bioinformatics*, **23**, e177–e183.

Pržulj,N. *et al*. (2004). Modeling interactome: scale-free or geometric. *Bioinformatics*, **20**, 3508–3515.

Pržulj,N. *et al*. (2006) Efficient estimation of graphlet frequency distributions in protein-protein interaction networks. *Bioinformatics*, **22**, 974–980.

Rives,A.W. and Galitski,T. (2003) Modular organization of cellular networks. *PNAS*, **100**, 1128–1133.

Salwinski,L. *et al*. (2004) The database of interacting proteins: 2004 update. *Nucleic Acids Res.*, **32**, 449–451.

Sharan,R. *et al*. (2007) Network-based prediction of protein function. *Mol. Syst. Biol.*, **3**, 88.

Spirin,V. and Mirny,L.A. (2003) Protein complexes and functional modules in molecular networks. *PNAS*, **100**, 12123–12128.

Strogatz,S. (2001) Exploring complex networks. *Nature*, **410**, 268–276.

Tong,A. *et al.* (2001) A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules. *Sciencexpress*, **10**.

Wernicke,S. (2006) Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol.*, **3**, 347–359.

Wuchty,S. and Almaas,E. (2005) Peeling the yeast protein network. *Proteomics*, **5**, 444–449.

Wuchty,S. *et al.* (2003) Evolutionary conservation of motif constituents in the yeast protein interactino network. *Nat. Genet.*, **35**, 176–179.

Yeger-Lotem,E. *et al.* (2004) Network motifs in integrated cellular networks of transcription-regulation and protein–protein interaction. *PNAS*, **101**, 5934–5939.

Yook,S. *et al.* (2004) Functional and topological characterization of protein interaction networks. *Proteomics*, **4**, 928–942.

Zhang,L. *et al.* (2005) Motifs, themes and thematic maps of an integrated saccharomyces cerevisiae interaction network. *J. Biol.*, **4**, 6.

Ziv,E. *et al.* (2005) Information-theoretic approach to network modularity. *Phys. Rev. E*, **71**, 046117.