NeMoFinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs

Jin Chen* Wynne Hsu Mong Li Lee
School of Computing
National University of Singapore
Singapore 119077
{chenjin, whsu, leeml}@comp.nus.edu.sg

See-Kiong Ng
Institute for Inforcomm Research
21 Heng Mui Keng Terrace
Singapore 119613
skng@i2r.a-star.edu.sg

ABSTRACT

Recent works in network analysis have revealed the existence of network motifs in biological networks such as the protein-protein interaction (PPI) networks. However, existing motif mining algorithms are not sufficiently scalable to find meso-scale network motifs. Also, there has been little or no work to systematically exploit the extracted network motifs for dissecting the vast interactomes.

We describe an efficient network motif discovery algorithm, NeMoFinder, that can mine meso-scale network motifs that are repeated and unique in large PPI networks. Using NeMoFinder, we successfully discovered, for the first time, up to size-12 network motifs in a large whole-genome *S. cerevisiae* (Yeast) PPI network. We also show that such network motifs can be systematically exploited for indexing the reliability of PPI data that were generated via highly erroneous high-throughput experimental methods.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: DATABASE MANAGEMENT—*Database Applications, Data Mining*

General Terms

Algorithms, Experimentation, Performance

Keywords

Network motif, Graph mining, Protein-protein interaction network

1. INTRODUCTION

Recent works in network analysis [15] have revealed that the topology of complex natural networks such as protein-protein interaction (PPI) networks are far from random. Many of these networks have been shown to exhibit such common global topological features as the "small world" and "scale free" properties. It turns out that in addition to these global topological characteristics, many local topological patterns can also be detected in the large complex natural networks. For example, Milo *et al.*[15] discovered various significant patterns of local connections that occurred more frequently in complex networks than in randomized networks. They called these recurring local topological substructures as "network motifs". While relatively less widely studied than the global topological features, such network motifs can lead to better understanding about various classes of complex networks, as some network motifs may be particular to specific classes of networks. For example, certain triad or tetrad motifs are specific topological patterns that are found to appear in biological networks rather than in other networks [15]. The presence of such network motifs also reveals the basic structural elements that underlie the hierarchical and modular architecture of such complex natural networks as PPI networks.

Researchers have only recently begun to employ network motifs in exploring the interactomes; for example, Saito *et al.* [17, 16] used manually derived network motifs to detect false positives in highly erroneous PPI networks, while Albert *et al.*[1] used them to predict PPIs. These pioneering works have achieved promising results even though the network motifs used in these works were rather limited—Saito *et al.* had used only 5 predefined network motifs of size 3 in their latest work on false positive detection [16], while Albert *et al.* had used only 4 predefined small network motifs for predicting interactions. This shows that the network motifs can provide a framework for the effective dissection of the complex PPI network based on the underlying structural principles.

As many of the relevant processes in biological networks have been shown to correspond to the meso-scale (5-25 genes or proteins) [19], it would be interesting to investigate if it is advantageous to use network motifs that are of equivalent sizes. However, existing network motif discovery algorithms [15, 9] are not applicable as they are mostly enumeration-based and limited to extracting smaller network motifs (up to size 8) for the following reasons:

1. The number of network motifs candidates increases exponentially with the motif size [8, 10].

*Contact author.

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

2. Interesting network motifs are typically repeated and unique [15], that is, the motifs occur repeatedly in the PPI network but not in the randomized networks. Such motifs do not have the downward closure property and *Apriori* algorithms are not applicable here.

3. The graph isomorphism problem, which is the essential technique to identify different network motifs, is an NP problem [4].

Such limitations impact the applicability of motif discovery approach for biological applications, as meso-scale network motifs are beyond the reach of existing exhaustive enumeration algorithms.

In this paper, we present an efficient graph mining algorithm called *NeMoFinder* to discover meso-scale repeated and unique network motifs in a large, genome-scale PPI network for biological applications. The proposed algorithm utilizes repeated trees to partition a network into a set of graphs. We introduce the notion of graph cousins to facilitate the candidate generation and frequency counting processes. Experiment results indicate that NeMoFinder is scalable and outperforms existing network motif discovery methods. We also use the network motifs that are mined from the real-life biological networks to detect false positives in the highly erroneous PPI network obtained from biological experimental methods. The experimental results demonstrate that the actual meso-scale network motifs extracted from the biological interaction networks can achieve better performance than using small, predefined ones for assessing the reliability of PPIs from conventional high-throughput experiments.

The rest of this paper is organized as follows. Section 2 introduces the basic concepts. Section 3 describes the related work in network motif algorithms. In Section 4, we describe the proposed NeMoFinder algorithm. Section 5 presents the comparative results of using NeMoFinder for discovering network motifs for *S. cerevisiae* PPI networks. In Section 6, we show how the extracted network motifs can be used to validate the interactions in a PPI network. Finally, we conclude in Section 7.

2. DEFINITIONS

In this work, we model a PPI network as an undirected graph $G = (V, E)$ where each vertex in V represents a unique protein, and each edge in E between two vertices v_A and v_B indicates that there is an interaction detected between the corresponding proteins A and B. We exclude self-loops from G here, as we are only interested to see the effectiveness of graph topology between proteins (see section 6).

By definition, a network motif is a frequently occurring subgraph pattern in a network (in this case, a large genome-wide PPI network such as the Yeast PPI network that consists of 4341 vertices and 10199 edges). The class of network motifs that we are interested in extracting from the interactomes are unique non-random subgraphs [15] that occur repeatedly in the underlying biological network.

Let f_g be the number of occurrences of a subgraph g in a PPI network G. We say that g is *repeated* if f_g is more than some user-specified value F.

Let $f_{g\_rand_i}$ be the frequency of g in a randomized network G_{rand_i}, for $1 \leq i \leq N$, where N is the number of the randomized networks. Let s_g be the number of times f_g is equal or greater than $f_{g\_rand_i}$, $1 \leq i \leq N$, over the total number of randomized networks N. We say that g is *unique* if its s_g is more than some user specified value S.

DEFINITION 2.1. **Network Motif.** *A network motif g in a PPI network G is a connected, unlabelled and undirected topological pattern of inter-connections that is* repeated *and* unique *in G.*

Note that it is common for proteins and their interactions in complex biological networks such as PPI networks to participate in multiple biological functional modules. It is therefore perfectly possible for multiple vertex- or edge-overlapping subgraphs to be simultaneously active at any time. Hence, during the subgraph counting process, we must consider patterns with arbitrary overlaps of vertices and edges. This results in a computationally more complex problem as the frequency of network motifs does not have the downward-closed property in this case.

In addition, the uniqueness property of a network motif is also not downward-closed as a result of allowing vertex- and edge-overlap in the network motifs. When a motif g extends (or reduces) to its supergraph (or subgraph), the decrease (or increase) of f_g and f_{g_rand} is non-deterministic. This means that given a network motif g, we cannot directly infer whether the supergraphs and subgraphs of g are unique. In fact, even when we have found a non-unique motif, we still have to generate its supergraphs and check for their frequencies and uniqueness. This implies that determining the uniqueness value of a motif is also computationally expensive.

3. RELATED WORK

In terms of biological network motif mining, the pioneering work by Milo *et al.* employed an exhaustive search algorithm that counts all the subgraphs of a given number of vertices. As such, they could only discover small network motifs in the form of 3-vertex and 4-vertex subgraphs. Kashtan *et al.*[9] developed a more efficient sampling method to estimate the relative frequencies of subgraphs. Their method was useful for analyzing very large networks and for detecting high-order motifs since the runtime is independent of the network size. However, the sampling approach cannot be guaranteed to discover the complete set of network motifs. It also does not scale for large-size network motifs (the algorithm takes about 2 hours to find a size-8 motif in the network of transcriptional regulation of *E. coli* with 423 vertices and 519 edges).

On the other hand, the computationally savvy graph mining community has also been diligent in developing various algorithms to efficiently discover frequent subgraphs. The initial algorithms, notably the AGM [8] and FSG [10], were devised to find all the frequent subgraphs in a large graph database efficiently through the extension of the market basket analysis. The algorithms utilize the *Apriori* property to discover frequent subgraphs level by level. The gSpan [21] algorithm discovers frequent substructures by using a DFS-based canonical representation of graphs and enumerated the search space in a depth-first order. The FFSM [6] method improves the performance of gSpan by reducing

redundant subgraph candidates through a vertical search scheme with join and extension operations. Finally, the SPIN [7] algorithm overcomes the problem of cycles in graph by generating the frequent substructures hierarchically in two steps: starting from trees, and then extending the frequent trees to graphs.

All the above works have focused on mining subgraphs from a *collection* of graphs, and considered only the frequency but not the uniqueness property of subgraphs. Furthermore, in these works, the frequency of a subgraph is determined by the number of global graphs that the subgraph occurs in, regardless of whether the subgraph occurs many times within a particular graph. This is computationally easier than the network motif discovery problem where the frequency of a motif is determined by the number of occurrences, including vertex- and edge-sharing ones, within one large and complex graph.

Kuramochi *et al.*[11] designed two methods hSigGram and vSigGram to look for frequent subgraphs in a sparse graph. These methods first determine the number of edge-disjoint occurrences of a subgraph based on approximate and exact maximum independent set computations and then use it to prune infrequent subgraphs. However, the methods are not suitable for biological applications where a protein or an interaction can participate in multiple functional modules, in other words, the occurrences of a motif can overlap arbitrarily in a graph, which is a much more computationally challenging counting problem.

The FPF method by Schreiber *et al.*[18] extends hSigGram and vSigGram to find frequent subgraphs with arbitrary overlap. FPF uses the concepts of pattern tree and generating parent to prune redundant subgraph candidate generation. However, the method is expensive as it has to perform subgraph isomorphism test for all candidates. Furthermore, it is unable to prune the non-promising subgraphs as the frequency counting does not satisfy the downward closed property.

4. NEMOFINDER: NETWORK MOTIF DISCOVERY ALGORITHM

In this work, we propose a network motif discovery algorithm called NeMoFinder to discover repeated and unique meso-scale network motifs in a large PPI network (Algorithm 1).

The input to the algorithm is a PPI network G, a user defined frequency threshold F, a user defined uniqueness threshold S, and a user defined maximal network motif size K. The output of the algorithm is a set U of repeated and unique motifs from size 3 to size K. Note that a subgraph with k vertices is said to be a size-k subgraph. The proposed algorithm consists of three main steps. First, we find repeated subgraphs in the PPI network (Lines 4-15). Then we check the frequency of the repeated subgraphs in the randomized networks (Lines 16-21). Finally, we determine the uniqueness values of the the repeated subgraphs (Lines 22-28).

We illustrate the algorithm using the example graph G in Figure 1. Suppose we want to find all the motifs up to size 5 (*i.e.,* $K = 5$) from G. We let the frequency threshold $F = 2$, and the uniqueness threshold $S = 0.95$.

Algorithm 1 NeMoFinder

```
1: Input: G - PPI network;
          N - Number of randomized networks;
          K - Maximal network motif size;
          F - Frequency threshold;
          S - Uniqueness threshold;
2: Output: U - Repeated and unique network motif set;
3: D ← ∅;
4: for motif-size k from 3 to K do
5:    T ← FindRepeatedTrees(k);
6:    GD_k ← GraphPartition(G, T)
7:    D ← D ∪ T;
8:    D' ← T;
9:    i ← k;
10:   while D' ≠ ∅ and i ≤ k × (k − 1)/2 do
11:      D' ← FindRepeatedGraphs(k, i, D');
12:      D ← D ∪ D';
13:      i ← i + 1;
14:   end while
15: end for
16: for counter i from 1 to N do
17:   G_rand ← RandomizedNetworkGeneration();
18:   for each g ∈ D do
19:      GetRandFrequency(g, G_rand);
20:   end for
21: end for
22: U ← ∅;
23: for each g ∈ D do
24:   s ← GetUniqunessValue(g);
25:   if s ≥ S then
26:      U ← U ∪ {g};
27:   end if
28: end for
29: return U;
```

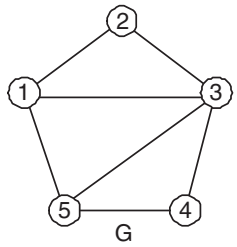

Figure 1: Example graph G.

Step 1. Discover Repeated Subgraphs.

The discovery of repeated size-k subgraphs in a PPI network, $2 < k \leq K$, involves the following three steps:

Step 1.1 Find Repeated Size-k Trees.

Algorithm NeMoFinder starts by finding the size-2 tree t_2 in G. Then the algorithm extends t_2 to a size-3 tree, size-4 trees, *etc.,* until size-K trees are obtained. Figure 2 shows all the size-2 to size-5 trees. Note that we have two size-4 trees (t_{4_1}, t_{4_2}) and three size-5 trees (t_{5_1}, t_{5_2}, t_{5_3}).

When a size-k tree t_k is formed, NeMoFinder counts its occurrences in G. If the occurrences of tree t_k is more than the user given threshold, then t_k is a repeated tree, and it is added to the set T_k.

In our example, the occurrences/frequencies of the various size trees are as follows: $f_{t_2} = 7$, $f_{t_3} = 13$, $f_{t_{4\_1}} = 6$, $f_{t_{4\_2}} =$

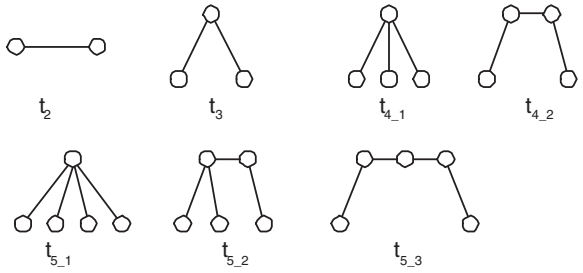

Figure 2: Size 2 to size 5 trees.

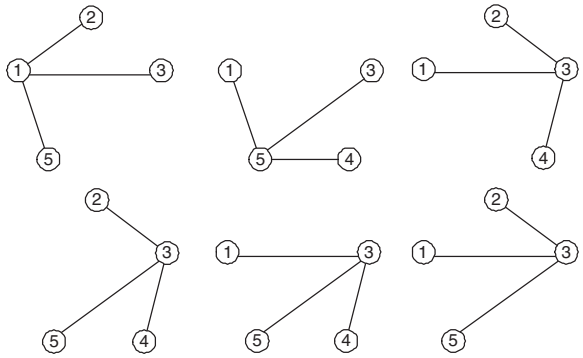

Figure 3: Occurrences of t_{4_1} in G.

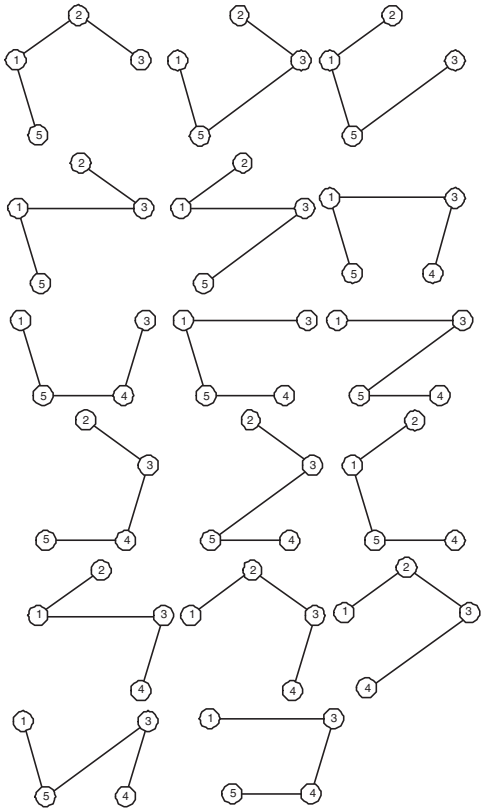

Figure 4: Occurrences of t_{4_2} in G.

17, $f_{t_{5\_1}} = 1$, $f_{t_{5\_2}} = 5$, $f_{t_{5\_3}} = 7$. All frequency values except for the frequency of t_{5_1} are more than the user given threshold of 2. Thus we have $T_2 = \{t_2\}$, $T_3 = \{t_3\}$, $T_4 = \{t_{4\_1}, t_{4\_2}\}$ and $T_5 = \{t_{5\_2}, t_{5\_3}\}$.

Step 1.2 Use Repeated Size-k Trees to Partition Graph.

Next, we use the size-k trees in T_k to partition the graph G into a set of graphs GD_k such that each graph G_{k_j} in GD_k embeds a size-k tree in T_k, $2 \leq k \leq K$ and $1 \leq j \leq |GD_k|$.

Consider the trees t_{4_1} and t_{4_2} in Figure 2. Figure 3 and 4 shows the occurrences of t_{4_1} and t_{4_2} in G. We use t_{4_1} and t_{4_2} to partition the PPI network G to obtain the set of graphs $GD_4 = \{G_{4\_1}, G_{4\_2}, G_{4\_3}, G_{4\_4}G_{4\_5}\}$ (Figure 5). Note that each graph in GD_4 embeds the tree t_{4_1} and/or t_{4_2}.

Step 1.3 Perform graph join operation to find repeated size-k graphs.

For each tree t in T_k, we generate size-k subgraphs with $k-1$ edges (the rules for generating the subgraphs are given in Section 4.1). Then we join t with each of these subgraphs to generate size-k subgraphs with k edges. The latter are added to the candidate set C_k.

Figure 6 shows the 4-vertex 3-edge subgraphs, $h_1, \ldots, h_5$, generated from the two size-4 trees t_{4_1} and t_{4_2} in T_4. We join t_{4_1} with h_1 and h_2, and join t_{4_1} with h_3, h_4 and h_5 to generate 4-vertex 4-edge subgraphs. Figure 7 shows the subgraphs obtained after joining t_{4_1} with h_1, and t_{4_2} with h_3. The non-redundant subgraphs g_{1_1} and g_{1_2} are added into the candidate set C_4.

For each subgraph $g \in C_k$, we check its occurrences in GD_k. If the occurrences of g is more than the threshold F, we add g to the set D_k. In our example, $f_{g_{1\_1}} = 2$ and $f_{g_{1\_2}} = 5$. Thus, g_{1_2} is a repeated subgraph and is added to the set of frequent subgraphs D.

Next, we use the repeated subgraphs obtained to generate all the possible k-vertex and k-edge subgraphs. These repeated subgraphs are joined with the newly generated subgraphs to get $(k+1)$-edge subgraphs. The repeated $(k+1)$-edge subgraphs are added to D. This process continues until a complete graph of $k*(k-1)/2$ edges is obtained, or no repeated subgraph can be found.

Figure 8 shows the 4-vertex and 4-edge subgraphs, h_6 and h_7, generated from the repeated subgraph g_{1_2}. We join g_{1_2} with h_6 and h_7 to get a 4-vertex and 5-edge subgraph g_2 (see Figure 9). Since the frequency of g_2 in GD_4 is not more than 2, it is not a repeated subgraph and the algorithm stops.

At the end of Step 1, the algorithm outputs the set D which contains all the repeated trees and subgraphs from size-2 to size-K.

Step 2. Determine Subgraph Frequency in Randomized Networks.

Next, we use the Markov-chain algorithm [12] to generate randomized networks G_{rand_i} $(1 \leq i \leq N)$ by swapping randomly selected interactions, as was done in [15]. This ensures that the randomized networks have the same single-vertex characteristics as the PPI network, *i.e.,* each vertex in the randomized networks has the same number of neigh-

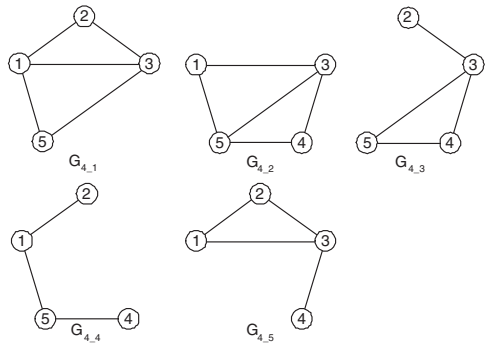

Figure 5: Set of graphs GD_4; each graph in GD_4 embeds t_{4_1} and/or t_{4_2}.

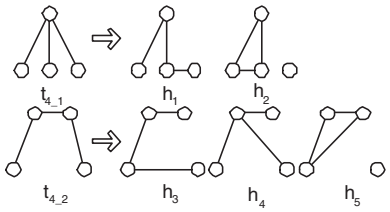

Figure 6: Generate 3-edge subgraphs from size-4 trees.

bors as the corresponding vertex in the PPI network. We check the frequency of the subgraphs in D in each of the randomized networks G_{rand_i} $(1 \leq i \leq N)$. The procedure is similar to Step 1.

Step 3. Compute Uniqueness of Subgraphs.

Finally, we compute the uniqueness value for each subgraph in D based on its frequencies in the input PPI network and the randomized networks.

NeMoFinder is scalable because the repeated trees naturally partitions the network into a set of graphs GD. Hence, the problem of counting the frequency of a size-k subgraph g in the network is reduced to the problem of finding the number of graphs in GD that contain the subgraph g, which is naturally downward closed.

In order to reduce the computational complexity, NeMoFinder adopts the idea in SPIN [7] to search for repeated trees and

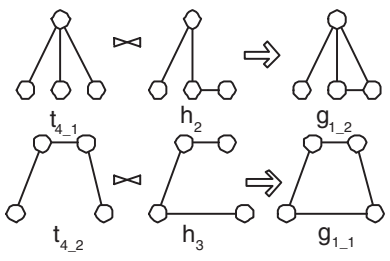

Figure 7: Examples of graph join operations for 3-edge subgraphs.

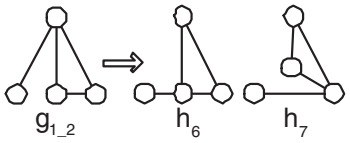

Figure 8: Generate 4-edge subgraphs from repeated 4-edge subgraphs of G.

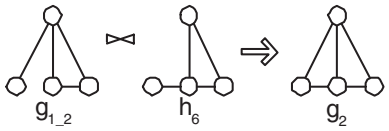

Figure 9: Examples of graph join operations for 4-edge subgraphs.

then extend them to subgraphs. However, NeMoFinder differs from SPIN in the following:

1. The notion of frequency in SPIN is different from our NeMoFinder. SPIN simply checks whether a subgraph occurs in a graph; it is not interested in counting how many times the subgraph occurs in the graph. In contrast, NeMoFinder considers occurrences of a subgraph in a network, including arbitrary overlaps.

2. SPIN uses equivalence classes to find maximal labelled frequent subgraphs in a set of graphs. In contrast, NeMoFinder is focused on discovering repeated unlabelled subgraphs from a single graph. Hence, our NeMoFinder is able to utilize the symmetry property of unlabelled trees to further reduce the number of candidate trees enumerated.

4.1 Candidate Generation using Graph Cousins

Finding repeated subgraphs involves generating candidate subgraphs and frequency counting (see Algorithm 2). The standard method to generate a subgraph candidate g_k from a tree t_k is to add a new edge to t_k and check whether the resulting graph is already in the candidate set C_k. However, C_k can become very large for meso-scale subgraphs, and checking whether a graph exists in C_k requires graph isomorphism test which is a NP problem.

Given that the network motifs are meso-scale, we use adjacency matrices to represent the subgraphs so as to facilitate the graph join operation to generate candidate subgraphs. A graph g with n vertices can be modelled using a $n \times n$ matrix M. An entry $m_{i,j}$ in an adjacency matrix is set to 1 if there is an edge from vertex i to j, and 0 otherwise. The code of M, denoted as $code(M)$, is a sequence formed by linking the lower triangular entries of M in the following order: $m_{1,1}m_{2,1}m_{2,2}...m_(i,j)...m_{n,1}m_{n,2}...m_{n,n}$ where $(0 < j \leq i \leq n)$.

We can transform any adjacency matrix into a unique representation called canonical adjacency matrix (CAM) [4]. Then two subgraphs that are isomorphic to each other have the same CAM, and vice versa. The canonical adjacency matrix (CAM) of a subgraph g, denoted as $CAM(g)$, is the adjacency matrix of g with the maximal code. The last edge

entry of $CAM(g)$ is the rightmost non-zero edge entry in $code(CAM(g))$. By removing the edge which corresponds to the last edge entry of $CAM(g)$, we obtain a subgraph of g. We call the adjacency matrix of such a subgraph as $subCAM(g)$ defined as follows:

DEFINITION 4.1. **subCAM of a graph.** *Let $CAM(g)$ be canonical adjacency matrix of a graph g. Then $subCAM(g)$ is a matrix obtained by setting the last edge entry in $CAM(g)$ to 0.*

Given two subgraphs g and h, if $subCAM(g) = subCAM(h)$, then we say that h is a *cousin* of g. There are three types of cousin relationship between g and h:

- **Type I: Direct Cousin** h is isomorphic to a subgraph g' which has the same number of vertices and edges as g, and $g \neq g'$;
- **Type II: Twin Cousin** h is isomorphic to subgraph g;
- **Type III: Distant Cousin** h is a disconnected subgraph.

Figure 10 shows the adjacency matrices for the size-4 trees t_{4_1} and t_{4_2} and the generated subgraphs $h_1, \ldots, h_5$ in Figure 6. From the above definitions, we see that h_1 is a Type I direct cousin of t_{4_1} since it is isomorphic to t_{4_2}; h_2 is a Type III distant cousin of t_{4_1} since it is a disconnected subgraph; h_3 is a Type II twin cousin of t_{4_2} since it is isomorphic to t_{4_2}; h_4 is a Type I direct cousin of t_{4_2} since it is isomorphic to t_{4_1}; h_5 is a Type III direct cousin of t_{4_2} since it is a disconnected subgraph.

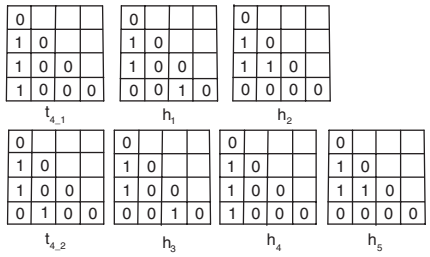

Figure 10: Adjacency matrices for the graphs in Figure 6.

We now show how the subgraph generation and frequency counting are efficiently carried out based on the cousins of a graph.

Given a repeated subgraph g of size k, we first find its set of cousins, H. Then we join g with each graph $h \in H$ to form new subgraphs of size k that have one more edge than g. Let $CAM(g)$ be CAM of g and $CAM(h)$ be CAM of h, then the adjacency matrix M of the new subgraph candidate is a $k \times k$ matrix and

$$m_{i,j} = \begin{cases} 1 & \text{if } CAM(g)_{i,j} = 1 \text{ or } CAM(h)_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Algorithm 3 gives the pseudo code for the candidate generation procedure.

The following theorem proves that the join operation generates the complete set of candidate subgraphs.

Algorithm 2 FindRepeatedGraphs(k, i, D')

```
1: Input: D' - Set of repeated subgraphs with k vertices and
   i − 1 edges;
2: Output: D'' - Set of repeated subgraphs with k vertices and
   i edges;
3: C ← CandidateGeneration(k, i, D');
4: D'' ← FrequencyCounting(k, i, C);
5: return D'';
```

Algorithm 3 CandidateGeneration(k, i, D')

```
1: Input: D' - Set of repeated subgraphs with k vertices and
   i − 1 edges;
2: Output: C - Set of candidates with k vertices and i edges;
3: C ← ∅;
4: for each g ∈ D' do
5:    H ← GetCousin(g);
6:    for each h ∈ H do
7:       g' ← join(g, h);
8:       C ← Can ∪ {g'};
9:    end for
10: end for
11: return C;
```

THEOREM 4.1. *Given all the subgraphs $g \in C_k$ which has k vertices and l edges ($l \geq k-1$), the join operation generates the complete set of subgraphs C'_k, where each $g \in C'_k$ has k vertices and $l+1$ edges.*

Proof: Let M be an adjacency matrix of a subgraph $g \in C'_k$ and e_1 be the last edge entry in M, such that matrix $M_1 = M - \{e_1\}$ is a CAM of a subgraph g_1. Let e_2 be the last edge entry in M_1. Since M_1 is a connected graph, its corresponding subgraph g_1 must be in C_k.

Let $M_2 = M_1 - \{e_2\} + \{e_1\}$ and M_2 be an adjacency matrix of a subgraph g_2, we have $g_1 \bowtie g_2 \Rightarrow g$. Based on the definition of graph cousins, if g_2 is isomorphic to g_1, g_2 is a Type II twin cousin of g_1; if g_2 is connected but not isomorphic to g_1, then g_2 is a Type I direct cousin of g_1; if g_2 is disconnected, g_2 is a Type III distant cousin of g_1.

Since the join operation joins g_1 with all its cousins, each $g \in C'_k$ is generated from C_k. □

4.2 Frequency Counting

A straightforward method to count the frequency of a size-k subgraph g in a graph G is to check all the graph in GD_k. However, this is an NP-complete subgraph isomorphism problem. Given that the discovery of network motifs requires checking the frequency of the candidate subgraphs in both the PPI network as well as the large number of randomized networks, it is critical for us to reduce the computational time of the frequency counting process. This can be achieved by leveraging the properties of the different types of cousins.

THEOREM 4.2. *Let L_x denote the set of graphs in GD_k such that each graph in L_x embeds x. Let h be a Type I direct cousin of a size-k subgraph g and g' be the subgraph obtained by joining g and h. Then we have $L_{g'} = L_g \cap L_h$, and the frequency of g' is given by $|L_g \cap L_h|$.*

Proof: Each graph in $L_{g'}$ must embed g and h since g' contains all the edges of both g and h. Thus, we have $L_{g'} \subseteq L_g \cap L_h$.

Algorithm 4 FrequencyCounting(k, i, C)

```
1: Input: GD_k - Set of graphs generated by partitioning G with
   size-k repeated trees;
     C - Set of subgraph candidates with k vertices and i edges;
     F - Frequency threshold;
2: Output: D'' - Set of repeated subgraphs with k vertices and
   i edges;
3: D'' ← ∅;
4: for each g' ∈ C do
5:    Get the join parameter of g': g and h;
6:    L_g ← set of graphs in GD_k embedding g;
7:    L_h ← set of graphs in GD_k embedding h;
8:    if f_g < F or f_h < F then
9:       f_g' ← 0;
10:   else if type of h = Type I direct cousin then
11:      f_g' ← |L_g ∩ L_h|
12:   else if type of h = Type III remote cousin then
13:      f_g' ← |L_g ∩ L_h|
14:   else if type of h = Type II twin cousin then
15:      f_g' ← CheckAllOccurances(g);
16:   end if
17:   if f_g' > F then
18:      D'' ← D'' ∪ {g'};
19:   end if
20: end for
21: return D'';
```

On the other hand, each graph in $L_g \cap L_h$ embeds both g and h. Hence, the graph must embed g', since each edge in g' is in either g or h. Thus, we have $L_{g'} \supseteq L_g \cap L_h$.

Therefore, we have $L_{g'} = L_g \cap L_h$ and the frequency of g' is given by $|L_g \cap L_h|$. □

Let us consider t_{4_1} and h_2 in Figure 7. We have $L_{t_{4\_1}} = \{G_{4\_1}, G_{4\_2}, G_{4\_3}, G_{4\_5}\}$ and $L_{h_2} = \{G_{4\_1}, G_{4\_2}, G_{4\_3}, G_{4\_4}, G_{4\_5}\}$ (see Figure 5). Then, for subgraph g_{1_2} which is generated by joining t_{4_1} and h_2, the graphs in GD_4 that embed g_{1_2} are $L_{g_{1\_2}} = L_{t_{4\_1}} \cap L_{h_2} = \{G_{4\_1}, G_{4\_2}, G_{4\_3}, G_{4\_5}\}$. Hence, the frequency value of g_{1_2} is 4.

Similarly, we can prove that if h is a Type III distinct cousin of a size-k subgraph g, the frequency of g' (the subgraph obtained by joining g and h) is also given by $|L_g \cap L_h|$.

However, if h is a Type II twin cousin of a size-k subgraph g, then h is isomorphic to g. In order to determine the frequency of the subgraph obtained by joining g and h, we have to check all the graphs in GD_k that embeds g. This frequency counting involves the NP-complete subgraph isomorphism test. Hence, given that the same subgraph can be generated by joining g with its various types of cousins, we choose to join g with its Type I or Type III cousin whenever possible to avoid the subgraph isomorphism test. Algorithm 4 gives the pseudo-codes for the frequency counting process.

For the complexity analysis of NeMoFinder, please refer to our technical report TRC6/06 (June 2006) [2].

5. PERFORMANCE STUDY

We have implemented our NeMoFinder algorithm in C++ and carried out experiments to compare NeMoFinder with existing network motif discovery algorithms such as the enumeration method [15], sampling method [9], and FPF [18].

We use two real-life datasets, the Uetz dataset and the original MIPS CYGD dataset. The Uetz dataset [20] contains 957 PPIs and 1004 proteins of *S. cerevisiae* and can be downloaded from the BRITE website. The MIPS CYGD dataset [14] is the whole-genome PPI network of *S. cerevisiae* from the Munich Information Center for Protein Sequences. After removing redundancy and orphan links, this dataset contains 10199 PPIs involving 4341 proteins that have been detected with high-throughout genome-wide biological experimental methods.

First, we evaluate the runtime of the four network motif discovery methods (enumeration, sampling, FPF, NeMoFinder) in finding network motifs of varying sizes in the Uetz dataset. We set the frequency threshold to 50, the uniqueness threshold to 0.95, and the number of randomized networks to 100. Figure 11 shows that NeMoFinder consistently gives the best performance, with 20- to 100-fold speed up. We also observe that only NeMoFinder manages to find all the motifs within a reasonable amount of time.

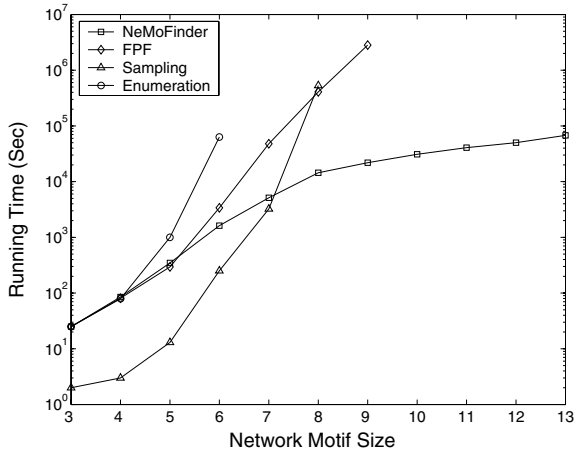

Figure 11: Comparison of computational times to find network motifs of varying sizes in Uetz PPI network.

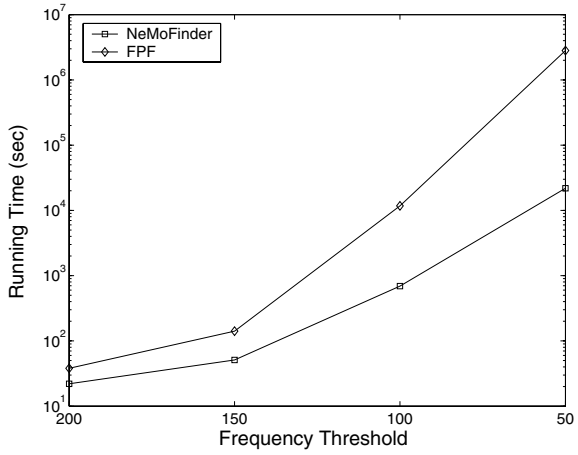

Figure 12: Comparison of computational times to find network motifs in Uetz PPI network under varying frequency thresholds.

Next, we evaluate the performance of NeMoFinder under varying frequency thresholds. We set the uniqueness thresh-

old to 0.95, the number of randomized networks to 100, and the maximal size of network motif to 9. The enumeration method and sampling method have been excluded from this experiment because they could not scale up to size-9 motifs. Figure 12 indicate that NeMoFinder is able to achieve up to 100-fold speedup over FPF.

We also compare the maximal motif size and the total number of identified motifs by the four algorithms to find network motifs of varying sizes in the MIPS dataset, which is much larger than the Uetz dataset. We set the frequency threshold to 50, the uniqueness threshold to 0.95, the number of randomized networks is set to 1000. Figure 13 shows that NeMoFinder was able to extract network motifs up to size 12, while the maximum sizes of the motifs discovered by FPF, sampling method and enumeration method are 9, 8 and 5 respectively. In addition, NeMoFinder was able to find a total of 11140 motifs, while FPF, sampling method and the enumeration method discovered only 1112, 848 and 21 network motifs respectively. The limited number of network motifs found by FPF, sampling and enumeration methods was due to the limitation of the motif size that these algorithms could handle.

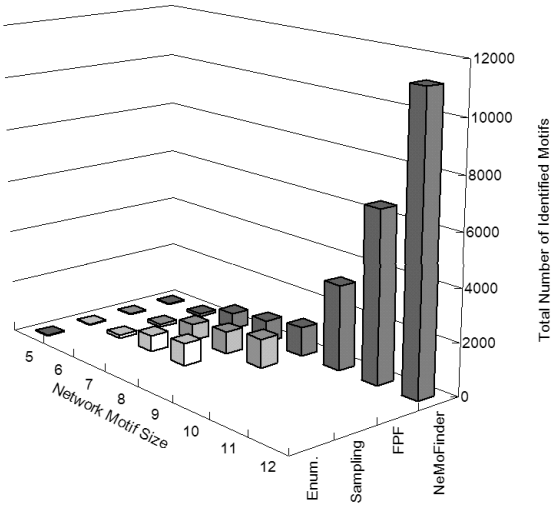

Figure 13: Comparison in size and number of network motifs that can be found by four algorithms in MIPS PPI network.

6. A MOTIF APPLICATION: PPI VALIDATION

Previous works in biological network motifs have focused mostly on motif discovery; there has been little or no work in showing how the network motifs can be systematically exploited. In this section, we describe how we can exploit the extracted network motifs in PPI validation. Our results show that the inclusion of the larger meso-scale network motifs indeed leads to better results.

Technological developments in high-throughput PPI detection methods such as *yeast-two-hybrid* and *protein chips* have enabled biologists to experimentally detect PPIs at the whole genome level for many organisms. For example, currently more than 15000 PPIs have already been detected and deposited in biological databases for *S. cerevisiae*. The abundant number of PPIs enables scientists to analyze these organisms at the genome level. However, a significant proportion of the PPI networks obtained from high throughput biological experiments has been found to contain false positives. Recent surveys have revealed that the reliability of the popular high-throughput *yeast-two-hybrid* assay can be as low as 50% [13]. These errors in the experimental data may lead to spurious discoveries that can be potentially costly, e.g., wrong drug targets for diseases.

In a first attempt to validate detected interaction candidates, Saito *et al.* [17] develop an interaction reliability index called *interaction generality* (IG1). The IG1 measure is based on the notion that the neighbors of the interacting partners are likely to also interact with each other. In other words, it uses a very simple network motif to detect false positives by noting that interactions between partners that are involved in lone star-like motifs are probably spurious. Positive results from the various experiments conducted by Saito *et al.* suggest that the use of even such a seemingly primitive network motif in dissecting genome-wide PPI networks is helpful in increasing the reliability of currently erroneous experimental interaction data. In fact, the authors subsequently developed a second reliability index called IG2 [16] that was based on 5 possible network motifs that involved a third protein C together with the candidate interacting protein pair (A, B). Their experimental results showed that IG2 outperformed IG1, suggesting the advantage of using more sophisticated network motifs in dissecting the experimentally derived PPI networks.

In this section, we investigate whether using the actual network motifs can indeed give better performance than using the simple, predefined ones such as those employed in IG1 and IG2.

6.1 Motif Strength

We have seen how NeMoFinder is able to discover a much more comprehensive set of network motifs as compared to the other methods (Section 5). For it to be useful in practice, it is important that the set of network motifs can provide sufficient coverage of the vast interactome. We found that 96% of PPIs in the MIPS dataset was indeed covered by at least one network motif discovered by NeMoFinder.

First, we rank the network motifs in terms of their contribution to the PPI network with respect to their individual sizes, frequencies and uniqueness. For simplicity, we assume that the motifs are independent here. We define the strength $MS^k(g)$ for each motif g as:

DEFINITION 6.1. **MotifStrength.** *The strength of a size-k motif g, denoted as $MS^k(g)$, is the frequency value of the motif times its uniqueness value over max_k, where max_k is the maximal value of $s(g) \times f(g)$ of all size-k motifs.*

$$MS^k(g) = \frac{s(g) \times f(g)}{max_k} \tag{2}$$

6.2 Evaluation based on motif strength

Having defined the MotifStrength, we score each interaction in the PPI network by combining the strengths of the network motifs that contain the interaction (edge).

DEFINITION 6.2. **Reliability Index of PPI** *The reliability index of a PPI (A, B), denoted as $I(A, B)$, is the sum of the MotifStrength of all the motifs that contain the edge (A, B).*

$$I(A,B) = \sum_{k=2}^{K} \sum_{i=0}^{n} MS^k(g_i) \times k \quad (3)$$

where $g_i, 1 \leq i \leq n$ are the motifs where edge (A, B) occurs and k is the size of g_i.

We apply our method, as well as IG1 and IG2, on the MIPS CYGD dataset described in Section 5 to compute reliability indices for the 10199 *S. cerevisiae* PPIs in the dataset. We then compare the quality of the various reliability indices in the following three different aspects:

1. **Function Homogeneity.** The cellular functions of the protein partners in a genuine biological interactions are likely to be similar. As such, we would expect an interactome that has been sorted with a good reliability index to exhibit a high degree of functional homogeneity in the interactions with high reliability scores. We use the Comprehensive *S. cerevisiae* Genome Database (dated 2005-06-20) at MIPS [14] as the ground truth for the proteins' functional annotations. Out of the 4341 proteins in the MIPS CYGD interaction dataset, 3150 proteins have functional annotations and 4743 interactions involve the annotated proteins.

2. **Localization Coherence.** With the exception of the proteins involved in cellular pathways such as the signalling pathway, the cellular localizations of the protein partners in a genuine biological interactions are expected to be the same. As such, a better reliability index would exhibit a higher degree of cellular colocalization amongst the protein partners in the sorted interactions. We use the cellular localization annotations of the *S. cerevisiae* proteins in the MIPS database as the basis for comparison in our experiment.

3. **Gene Expression Correlation.** Studies have shown that the average correlation coefficient of gene expression profiles that corresponds to interacting protein pairs is significantly higher than those that correspond to random pairs [5]. As such, we can also use the degree of gene expression correlation to evaluate the relative quality of the PPI reliability indices. For gene expression correlation analysis, we downloaded the *S. cerevisiae* gene expression dataset from Eisen's Lab [3]. The dataset comprises expression vectors from 80 experiments on 6221 genes.

Figure 14 shows that as the reliability index value is increased, the proportion of interacting pairs with common cellular functions also increases, indicating an increase in the number of true positives in the filtered interaction data. The reliability indices generated using the NeMoFinder's network motifs show significant increases (from 61% to 87% and 81%) than those using IG1 and IG2 (from 61% to only 68% and 73% respectively).

Figure 15 shows the relative performance in terms of cellular localization coherence. Using the reliability indices computed by the network motifs, the proportion of interacting pairs with common cellular localization increases from

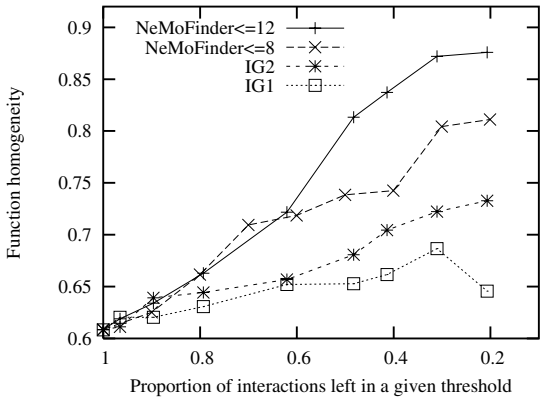

Figure 14: Proportion of interacting proteins with common cellular functional roles increases at different rates under different interaction reliability measures.

85.3% to 94.0% and 91.7% for the NeMoFinder network motifs, again outperforming IG1 and IG2 (from 85.3% to 87.0% and 90.1%).

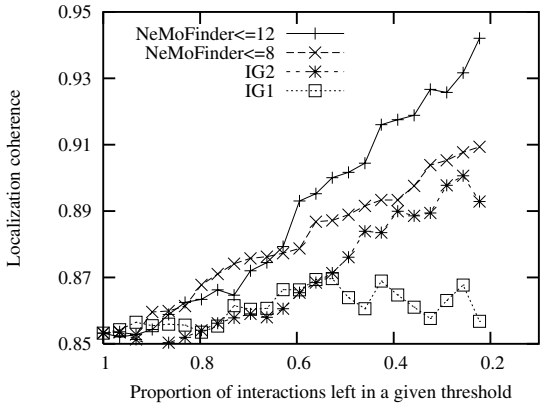

Figure 15: Proportion of interacting proteins with common cellular localizations increases at different rates under different interaction reliability measures.

The results based on gene expression correlation, shown in Figure 16, exhibit a similar trend. Again, the increase in the average gene expression correlation between the protein partners in the sorted PPIs is much more significant when using reliability indices computed with NeMoFinder's network motifs (from 26.4% to 33.5% and 30.8%) than those generated by using IG1 and IG2 (from 26.4% to 27.6% and 29%).

These results show that the PPI reliability indices computed using the NeMoFinder network motifs are more reliable than those computed using IG1 and IG2, demonstrating the positive effect of using a more comprehensive set of actual network motifs against a small number of simple, predefined motifs. Additionally, we also compared the performance of using motifs of different sizes. In all three evaluation experiments, the reliability indexes computed using NeMoFinder network motifs of sizes up to 12 consistently

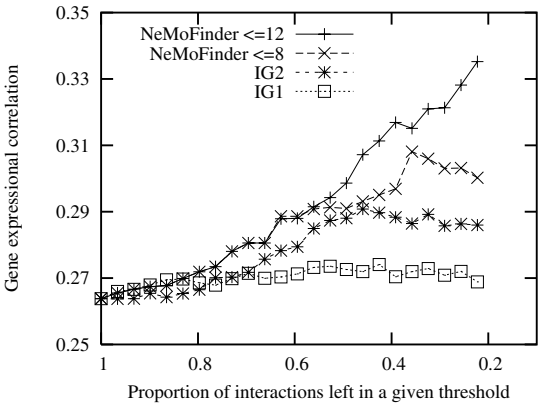

Figure 16: Overall correlation of gene expression for interacting proteins increases at different rates under different interaction reliability measures.

show superior performance over that computed with motifs of sizes only up to 8. This indicates that it is advantageous to include the larger motifs, justifying the need for discovering meso-scale network motifs.

7. CONCLUSIONS

Existing network motif discovery algorithms are limited to extracting smaller network motifs and cannot be employed to mine meso-scale level network motifs in large biological networks. In this paper, we have presented an efficient network motif discovery algorithm called NeMoFinder to discover larger-sized repeated and unique network motifs in PPI networks. The algorithm utilizes repeated trees to partition a network into a set of graphs. We have introduced the notion of graph cousins for efficient candidate generation and frequency counting. We use NeMoFinder to successfully extract, for the first time, up to size-12 network motifs from the whole *S. cerevisiae* PPI network. The network motifs discovered by NeMoFinder provided a good coverage of the PPIs in the vast interactome.

In this work, we also showed an example of how the network motifs can be systematically applied in the validation of the PPIs in an interactome. Our results confirmed that employing the larger actual network motifs derived from biological networks instead of predefined small-sized network motifs can indeed achieve better results. Future work will include directed network motif discovery and network motif labelling.

8. REFERENCES

[1] I. Albert and R. Albert. Conserved network motifs allow protein-protein interaction prediction. *Bioinformatics*, 20(18):3346–3352, 2004.

[2] J. Chen, W. Hsu, ML. Lee, and SK. Ng. Discovering and exploiting meso-scale network motifs in protein interactomes. Technical Report TRC6/06, National University of Singapore, 2006.

[3] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci. USA*, 95:14863–14868, 1998.

[4] S. Fortin. The graph isomorphism problem. *Technical Report TR96-20, Department of Computing Science, University of Alberta*, 1996.

[5] A. Grigoriev. A relationship between gene expression and protein interactions on the proteome scale. *Nucleic Acids Res*, 29(17):3513–3519, 2001.

[6] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. *ICDM*, pages 549–552, 2003.

[7] J. Huan, W. Wang, J. Prins, and J. Yang. Spin: Mining maximal frequent subgraphs from graph databases. *SIGKDD*, 2004.

[8] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph. *PKDD*, pages 13–23, 2000.

[9] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[10] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *TKDE*, 2004.

[11] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. *In SIAM International Conference on Data Mining*, 2004.

[12] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296(5569):910–913, 2002.

[13] C.V. Mering, R. Krause, B. Snel, et al. Comparative assessment of largescale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

[14] H.W. Mewes, D. Frishman, U. Guldener, et al. Mips: a database for genomes and protein sequences. *Nucleic Acids Res*, 30(1):31–34, 2002.

[15] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[16] R. Saito, H. Suzuki, and Y. Hayashizaki. Construction of reliable protein-protein interaction networks with a new interaction generality measure. *Bioinformatics*, 19:756–763, 2002.

[17] R. Saito, H. Suzuki, and Y. Hayashizaki. Interaction generality, a measurement to assess the reliability of a protein-protein interaction. *Nucleic Acids Res*, 30:1163–1168, 2002.

[18] F. Schreiber and H. Schwobbermeyer. Frequency concepts and pattern detection for the analysis of motifs in networks. *Transactions on Computational Systems Biology*, 3(LNBI 3737):89–104, 2005.

[19] V. Spirin and L.A. Mirny. Protein complexes and functional modules in molecular networks. *PNAS*, 100(21):12123–12128, 2003.

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

[21] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. *ICDM*, 2002.