JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 14, Number 7, 2007

Pp. 892–907
DOI: 10.1089/cmb.2007.0025

# Comparing Protein Interaction Networks via a Graph Match-and-Split Algorithm

MANIKANDAN NARAYANAN[1] and RICHARD M. KARP[1,2]

## ABSTRACT

**We present a method that compares the protein interaction networks of two species to detect functionally similar (conserved) protein modules between them. The method is based on an algorithm we developed to identify matching subgraphs between two graphs. Unlike previous network comparison methods, our algorithm has provable guarantees on correctness and efficiency. Our algorithm framework also admits quite general criteria that define when two subgraphs match and constitute a conserved module. We apply our method to pairwise comparisons of the yeast protein network with the human, fruit fly and nematode worm protein networks, using a lenient criterion based on connectedness and matching edges, coupled with a clustering heuristic. In evaluations of the detected conserved modules against reference yeast protein complexes, our method performs competitively with and sometimes better than two previous network comparison methods. Further, under some conditions (proper homolog and species selection), our method performs better than a popular single-species clustering method. Beyond these evaluations, we discuss the biology of a couple of conserved modules detected by our method. We demonstrate the utility of network comparison for transferring annotations from yeast proteins to human ones, and validate the predicted annotations. Supplemental text is available at *www.cs.berkeley.edu/∼nmani/M-and-S/supplement.pdf*.**

**Key words:** biological network comparison, functional modules, graph-matching algorithm, network alignment, protein-protein interactions.

## 1. INTRODUCTION

PROTEIN-PROTEIN INTERACTIONS are the basis of many processes that maintain cellular structure and function. Organizing the myriad protein interactions in a cell into functional modules of proteins is a natural step in exploring these cellular processes (Hartwell et al., 1999). The need for such organization is readily evident from a glance at the network of interactions observed in the yeast *Saccharomyces cerevisiae* (Fig. 1). Numerous such interactions are publicly available for other organisms too from high-throughput though noisy experiments or extensive literature scanning (Bork et al., 2004). Computational methods are valuable to make sense of this raw data and cope with its scale. These methods are similar in spirit to ones that organize raw genomic sequences into functional elements like genes, regulatory sequences, etc.

[1]Computer Science Division, University of California, Berkeley, California.
[2]International Computer Science Institute, Berkeley, California.

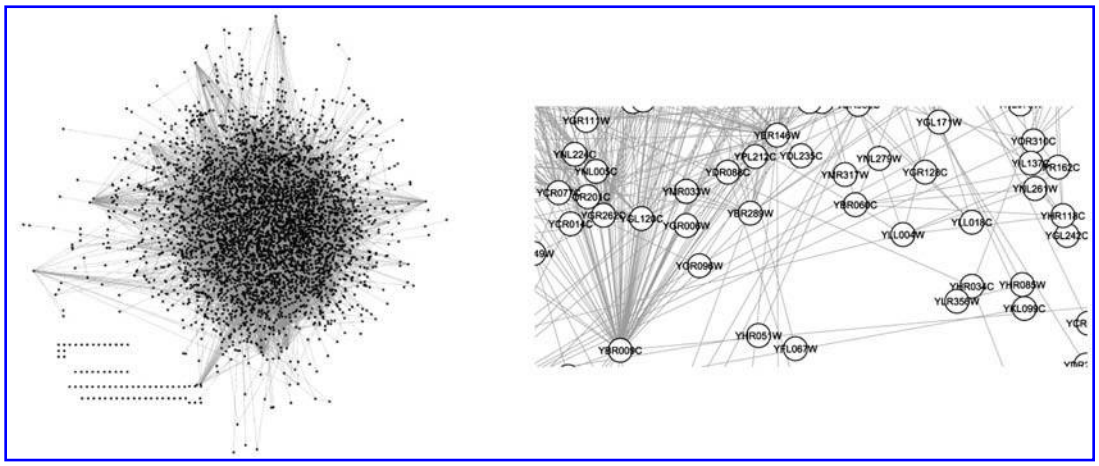

**FIG. 1.** Global view of the protein interaction network of *Saccharomyces cerevisiae* **(left)** comprising 14,319 interactions over 4389 proteins, and a zoomed view of some proteins **(right)**. A dot or circle represents a protein and a line joins two interacting proteins. The protein networks in this paper are drawn using the Cytoscape software (Shannon et al., 2003).

In this work, a protein (interaction) network refers to a graph whose nodes are the proteins of an organism and edges indicate physical interactions between proteins, and a protein module refers to a subset of proteins in this network along with the interactions between them. A functional module is then a protein module known or supposed to be involved in a common cellular process (e.g., a protein complex or a signaling pathway). Note however that concrete definition of a cellular process is a subject of continuing discussion in biology (Hartwell et al., 1999).

### *Previous work*

Several methods analyze a single organism's protein network to identify functional modules (as reviewed in Bork et al., 2004). A typical single-species method uses connectivity information to cluster a protein network into highly connected modules, for example, MCODE (Bader and Hogue, 2003). The method is effective if the modules it outputs align well with a set of reference or known functional modules, and if the output modules that don't match any known functional module suggest plausible and testable biological hypotheses.

A few recent methods compare protein networks from two or more species to identify functionally similar (conserved) protein modules between them (as reviewed in Sharan and Ideker, 2006). These pairwise or multiple network comparison methods improve over single-species methods by using information on conservation (cross-species similarity of protein sequences and interaction patterns) as well as connectivity of the networks. For instance amidst noisy data, conservation could reinforce evidence that some connected proteins participate in a common function. These comparative methods also enable transfer of functional annotations between organisms at the level of conserved modules, interactions and proteins, a key utility single-species methods cannot provide. Similarly, conserved modules could provide a basis for studies on the evolution of cellular structures and networks.

Current network comparison methods include NetworkBLAST (Sharan et al., 2005), MaWISh (Koyuturk et al., 2005), a Bayesian alignment method (Berg and Lassig, 2006), and Græmlin (Flannick et al., 2006). At a high level, these methods formulate a biologically inspired measure to score when a set of subgraphs from the input networks constitute a conserved module and use a greedy heuristic or some variant to search for all high-scoring similar subgraphs between the networks. Heuristics are necessary as their complicated scoring measures lead to intractable, that is, NP-hard (Garey and Johnson, 1979), search problems.

### *This work*

We present a pairwise network comparison method based on a graph-matching algorithm with provable guarantees. Our novel graph-matching algorithm and the guarantees on both its correctness and running

time makes this work markedly different from previous methods relying on search heuristics. Our search formulation is biologically meaningful and yields promising results in detecting functional modules and transferring functional annotations.

We formulate a conserved protein module as a pair of connected and locally matching subsets of proteins, one from each input network. A locally matching subset pair comprises protein pairs that have similar sequences and similar neighborhood or context in the networks. Search for such conserved modules is simpler than search formulations in previous methods and hence admits an efficient polynomial-time algorithm. The main operation of this algorithm is a recursive match and split of the proteins in the two input networks. We assess the statistical significance of the conserved modules found by the algorithm using similar paths statistics and reliabilities of noisy interactions.

Our polynomial-time search problem is a novel contribution in the broader field of graph-matching as well. Graph-matching refers to a class of problems that find similar subgraphs between two graphs (as reviewed in Schellewald, 2005) along with other related classes. Many graph-matching problems in literature are NP-hard (Garey and Johnson, 1979), permitting only heuristic or approximate solutions, due to a stringent global structural match that they require between the similar subgraphs, as opposed to the local match our problem requires. For instance, the maximum common subgraph problem requires an exact isomorphic match between subgraphs and is NP-hard (Garey and Johnson, 1979). Problems that require inexact match between graphs to deal with error-prone data are NP-hard too for a family of graph edit cost functions (Bunke, 1999). Some studies use a scoring function to measure how similar two subgraphs are. They find high-scoring subgraph pairs by finding heavy-weight subgraphs in a product graph of the input graphs (Sharan et al., 2005; Koyuturk et al., 2005), which is also NP-hard by reduction from the maximum weight induced subgraph (Koyuturk et al., 2005) or maximum clique problem (Garey and Johnson, 1979).

## 2. METHODS

### 2.1. Conserved module premise

Our method compares protein networks from two species to find functionally similar or conserved protein modules between them. A conserved module is intuitively a pair of protein modules that share cross-species similarity at the node level (homology of corresponding protein sequences) and graph structure level (pattern of interactions). Our method's specific *premise* is that a conserved module is a pair of *connected* and *locally matching* subsets of proteins, one from each input graph. Two proteins locally match if their sequences and neighborhood in the network are similar, and a collection of such protein pairs is a locally matching subset pair. We support this premise in this section and formalize it in the next.

Our premise is biologically motivated. The premise's connectivity criterion makes it likely for a subset of proteins to be functionally homogeneous and its local matching criterion makes it likely for two protein subsets to be functionally similar. Moreover, these criteria are minimal requirements and hence sensitive in detecting reference functional modules. To illustrate, a functional module's counterparts in two different species needn't match exactly in their graph structure due to evolutionary divergence or errors in the interaction data, which makes a local match more sensitive than a stringent structural match like exact isomorphism. The two criteria yield good specificity too with some additional improvements detailed in Section 2.3.1. Note that a method's sensitivity denotes the fraction of reference modules it detects and specificity the fraction of modules output by it that match some reference module.

Our premise is also computationally attractive as it results in a search problem that admits provably good algorithms for many choices of the connectivity and local matching criteria. As mentioned in Section 1, our method's tractable search formulation contrasts it from previous network comparison methods and graph-matching problems.

### 2.2. Graph-matching engine

Finding conserved modules under the premise just outlined reduces to a graph-matching problem of finding similar subgraphs between two input graphs. We present this graph-matching problem variant and our polynomial-time algorithm for it in this section. We also discuss the algorithm's generality, which makes it relevant for application areas beyond protein networks.

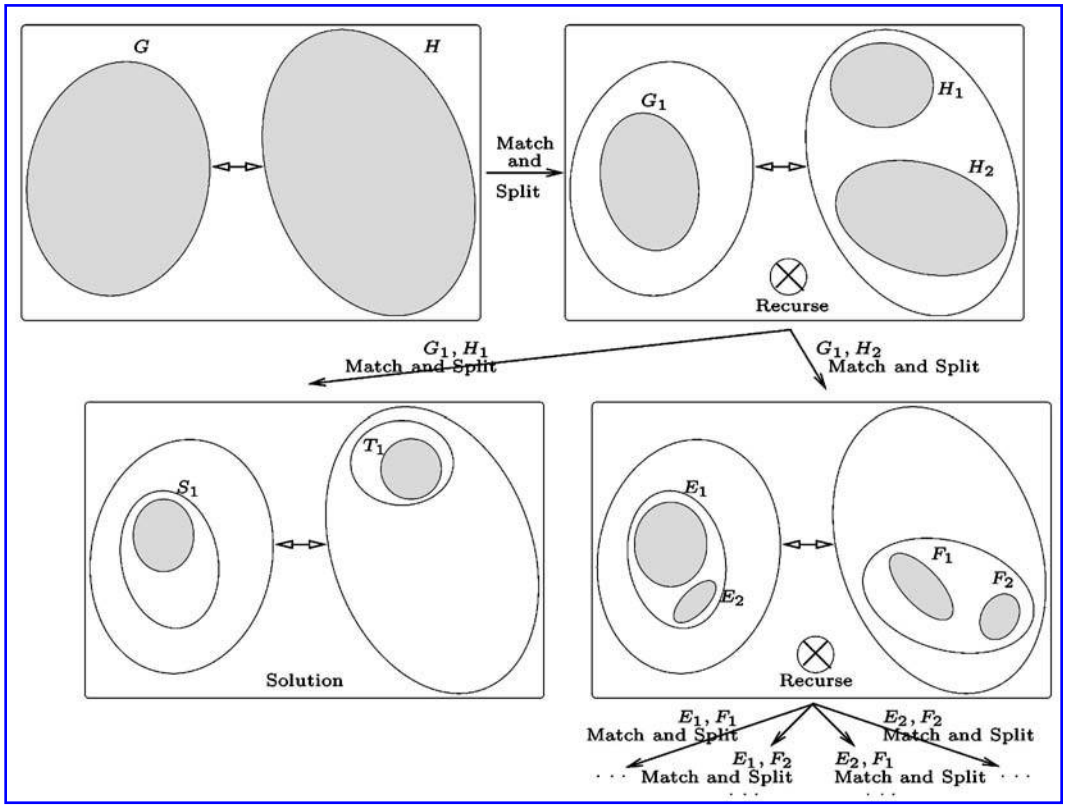


**FIG. 2.** Pictorial sketch of the main operations of our graph-matching algorithm. We show the input graphs $G, H$ and their subgraphs as ovals, hiding node and edge details. The algorithm focuses only on the shaded subgraph pairs at any point in its execution, and refines them recursively until all solutions (similar subgraph pairs) are found. A refinement involves doing a match and a split step to compute locally matching and connected node-sets between two shaded subgraphs (see text for details). The subgraph pair $S_1, T_1$ is a solution, and the algorithm might find more solutions as it recurses on the subgraph pairs $E_i, F_j$. The statistically significant solutions are finally output as conserved modules.

As a prelude, we sketch our graph-matching engine in Figure 2 and informally outline it now in the context of protein networks. Say we start with the yeast and human protein networks along with the homologous protein pairs between them. Our algorithm first computes *locally matching* proteins between the two networks, and safely discards the other proteins (i.e., any yeast protein with no human homolog or with some human homolog but with poor local match, and vice versa). The algorithm next splits the remaining yeast and human networks into *connected* sets of proteins. These connected subgraphs are locally matching with respect to the full input networks and contain amongst them the actual solutions with refined local matching relations. To find the solutions, our algorithm repeats the above match and split steps on each pair of the connected subgraphs recursively (Fig. 2). The ensuing text provides precise descriptions on matching general graphs.

*2.2.1. Problem statement.* We are given as input two graphs and a node similarity function $sim(.,.)$. The function $sim(u, v)$ is true whenever node $u$ is similar to node $v$ (e.g., based on sequence similarity of proteins) and false otherwise. This $sim(.,.)$ is a symmetric function defined over all pairs of nodes $u, v$, one from each input graph. The problem now is to list pairs of connected and locally matching subgraphs between the input graphs. In this work, a subgraph of a graph usually refers to an induced subgraph, which is a subset of nodes in the graph along with all edges between them.

We first build a *local-match*$_{S,T}(u, v)$ function for any subgraph pair $S, T$ of the input graphs using the $sim(.,.)$ function. This new function captures local or contextual match between the nodes $u$ of $S$ and $v$ of $T$ using similar local structures present around these nodes in $S$ and $T$. Two possible choices of similar

local structures are: *similar length-$p$ paths* for some small $p$ (say 2), and *$s$-similar neighborhoods* around nodes for some small $s$ (Fig. 3). In the former case, *local-match*$_{S,T}(u, v)$ is true whenever some length-$p$ path in $S$ containing $u$ is similar to some length-$p$ path in $T$ containing $v$ (two paths are similar if all their corresponding nodes are similar according to $sim(.,.)$). In the latter case, *local-match*$_{S,T}(u, v)$ is true whenever $sim(u, v)$ is true and $sim(u', v')$ is true for at least $s$ distinct neighbor pairs $u'$ of $u$ in $S$ and $v'$ of $v$ in $T$. The stringency of this criterion increases with $s$, with 1-similar neighborhoods being the least stringent.

We are ready to state the problem. Given two input graphs $G$, $H$, a node similarity function $sim(.,.)$, and a *local-match*$_{S,T}(.,.)$ function computable for any two subgraphs $S, T$, the problem is to find all maximal induced subgraph pairs $S \subseteq G$, $T \subseteq H$ that satisfy two criteria:

*Connectivity:* $S, T$ are each connected, and
*Local Matching:* Each node $u$ in $S$ locally matches at least one node $v$ in $T$ according to the *local-match*$_{S,T}(u, v)$ function, and vice versa.

Any subgraph pair that satisfy the above two criteria is called a solution (Fig. 3), and maximality requires that of two solutions $S, T$ and $S', T'$ with $S' \subseteq S, T' \subseteq T$, we only output the maximal one $S, T$ to avoid redundancy.

*2.2.2. Algorithm and guarantees.* We present a simple and efficient algorithm for the above problem for any *monotone* local matching criterion. A monotone criterion is one where any two nodes that locally match remain so even after adding more nodes to the subgraphs under consideration (i.e., if *local-match*$_{S,T}(u, v)$ is true, then *local-match*$_{S',T'}(u, v)$ is also true for any $S' \supseteq S, T' \supseteq T$). The similar length-$p$ paths or $s$-similar neighborhoods criterion from last section are monotone. A useful property of monotonicity is that the maximal solutions are only quadratic in number, since it lets us merge any two solutions $S, T$ and $S', T'$ with a common node pair $u, v$ (i.e., $u \in S \cap S', v \in T \cap T'$) into one solution $S \cup S', T \cup T'$.

Our algorithm presented below (Algorithm 1) matches and splits the nodes of G and H into smaller components, and then recurses on each of the component pairs. For induced subgraphs $S \subseteq G, T \subseteq H$, we let $lm(S, T)$ denote all nodes $u$ in $S$ for which *local-match*$_{S,T}(u, v)$ is true for some node $v$ in $T$.

**Algorithm 1**

*Match-and-Split*$(G, H)$:
*[Match]* Compute induced subgraph:
    $G'$ of $G$ over the locally matching nodes $lm(G, H)$, and
    $H'$ of $H$ over the locally matching nodes $lm(H, G)$.
*[Split]* Find connected components:
    $G_1, \ldots, G_c$ of $G'$, and
    $H_1, \ldots, H_d$ of $H'$.
*[Recurse]*
**if** $(c = 1, d = 1$ and $G' = G, H' = H)$
    Output the maximal solution $G, H$. [base case]
**else**
    **for** $i = 1$ to $c$, $j = 1$ to $d$
        *Match-and-Split*$(G_i, H_j)$. [recursive case]

**Correctness.** Consider any solution $S, T$. Each recursive call retains all locally matching nodes and processes all pairs of resulting components. Hence in at least one path of the recursion call tree, all nodes in $S, T$ remain locally matching due to monotonicity and connected as part of a bigger subgraph pair. This retained unsplit $S, T$ is finally output as part of a solution. Further, the output solutions are maximal because no node pair is common to any two output solutions (as shown in Section 2.2.3).

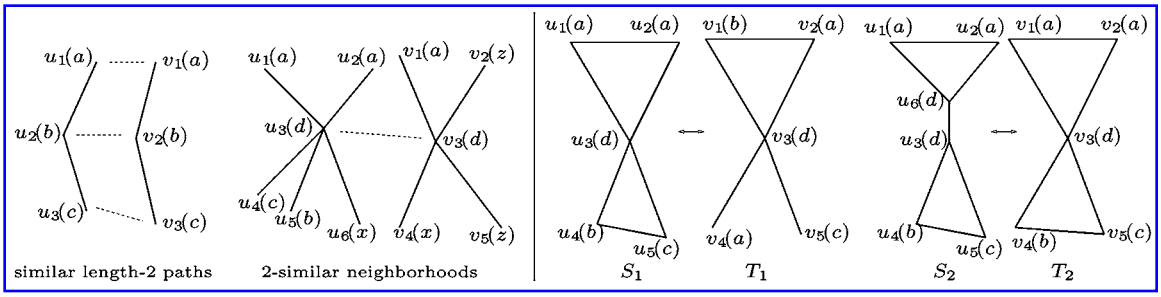


**FIG. 3.** Illustration of two similar local structures **(left)** and two solutions **(right)**. Assume $sim(u_i, v_j)$ is true whenever $u_i, v_j$ have the same label shown inside brackets. The dotted lines (left) show some of the locally matching node pairs. The subgraph pair $S_1, T_1$ is a solution when the local matching criterion is based on similar length-$p$ paths ($p = 1$ or 2) or 1-similar neighborhoods, and $S_2, T_2$ is a solution when the criterion is based also on 2-similar neighborhoods.

**Running time.** Let $n_F, m_F$ denote the number of nodes, edges respectively in a graph $F$. The algorithm runs in time $O(n_G n_H + (n_G + n_H)m_G m_H)$ on the graphs $G, H$ when the local matching criterion is similar length-1 paths. The algorithm is efficient in practice too as the locally matching proteins between two graphs in our experiments reduces drastically as the recursion depth increases.

*2.2.3. Analysis of running time.* The running time bound mentioned above follows from bounds proved here, which hold for more general monotone local matching criteria. Let $n_F, m_F$ be as defined above, and $f(G, H)$ be a function bounding the process time (of the match and split steps) on the graphs $G, H$.

**General bounds.** A simple bound on the running time is $O(n_G n_H f(G, H))$. This running time, which holds for a general $f(.,.)$, is polynomial-time if $f(.,.)$ is asymptotically bounded by a polynomial. To derive this bound, we show the number of recursion calls made by the algorithm is at most $2n_G n_H$ (see next subsection) and bound the process time at each call by a loose $f(G, H)$ (as we may assume $f(,,.)$ is a non-decreasing function).

A better running time bound of $O(n_G n_H + (n_G + n_H)f(G, H))$ applies for a broad class of $f(.,.)$ that satisfies the condition: $\sum_{i=1}^{c}\sum_{j=1}^{d}(f(G_i, H_j) - f_0) \leq (f(G, H) - f_0)$ for any $G, H$, where $f_0$ is the constant term in $f(G, H)$ (technically $f(G, H) \doteq f(n_G, m_G, n_H, m_H)$ and $f_0 \doteq f(0, 0, 0, 0)$). The two terms in this bound are: $O(n_G n_H f_0)$ for the $f_0$ time spent at each of the $O(n_G n_H)$ recursion calls made, and $O((n_G + n_H)(f(G, H) - f_0))$ for the $(f(G, H) - f_0)$ remaining process time at each of the $O(n_G + n_H)$ *levels* (assuming the condition; see next subsection for details and proof).

The condition above holds for many reasonable functions including any polynomial $f(.,.)$ whose monomials each contain the factors $n_G$ or $m_G$, and $n_H$ or $m_H$. For such polynomials, the condition follows readily from the connected components of a graph being node-disjoint and edge-disjoint. To illustrate, let the local matching criterion be similar length-1 paths. Then $f(G, H) = k\, m_G m_H$ for some constant $k$ as the match step simply considers all edge pairs in $G, H$ and the split step runs a linear-time connected components procedure. This $f(.,.)$ satisfies $\sum_{i=1}^{c}\sum_{j=1}^{d} k\, m_{G_i} m_{H_j} = k \sum_i m_{G_i} \sum_j m_{H_j} \leq k\, m_G m_H$, where the crucial last step is from the disjointness of the $G_i$'s and the $H_j$'s. Other criteria like similar length-$p$ paths or $s$-similar neighborhoods also yield a polynomial $f(.,.)$ satisfying the condition.

**Completing the proof.** We analyze the algorithm's recursion structure to complete the proof of the two running time bounds outlined above. In the recursion call tree of the algorithm on the input graphs $G, H$, the number of leaves (and hence output solutions) is at most a quadratic $n_G n_H$, internal nodes also at most $n_G n_H$ and levels at most $n_G + n_H$. A leaf is any algorithm call that terminates recursion (including any that outputs a maximal solution), an internal node is any call that invokes other calls (its children) recursively, and a level is a set of calls at the same recursion depth from the initial call on the input graphs (see Fig. 1 of the Supplemental text for some illustrations).

First, we prove the quadratic bound on the number of leaves by arguing any node pair in $G, H$ is present in at most one leaf of the call tree (and hence in at most one output solution too). As the $G_i$'s and the $H_j$'s are node-disjoint, a call on $G, H$ *partitions* the node pairs in $G, H$ among its children calls on $G_i, H_j$ for all $i = 0$ to $c$, $j = 0$ to $d$ (where we add dummy leaf children involving $G_0 \doteq G - G'$ or $H_0 \doteq H - H'$

for analysis, when these graphs are non-empty). If a node pair is in two leaves, then the least common ancestor of the leaves should have sent this same node pair along two of its children, a contradiction! Next, observe that the number of internal nodes is at most the number of leaves as each internal node has at least two children (including dummy leaves). Finally, the number of levels is at most $n_G + n_H$ as each children on $G_i, H_j$ of any internal node on $G, H$ satisfies $n_{G_i} + n_{H_j} \leq n_G + n_H - 1$. If it were not true, then $n_{G_i} + n_{H_j} = n_G + n_H$, thereby making $G, H$ a maximal solution and this internal node on $G, H$ a leaf instead!

Consider now the class of functions $f(.,.)$ that satisfy $\sum_{i=1}^{c} \sum_{j=1}^{d} (f(G_i, H_j) - f_0) \leq (f(G, H) - f_0)$ for any $G, H$. To complete the proof of the running time for such $f(.,.)$, we bound the process time (of the match and split steps, excluding the constant $f_0$ terms) at each level of the recursion call tree. The condition above simply says that the process time at *all* children of an internal node in the call tree is at most the process time at the internal node itself. We partition the calls (nodes) in a level $l$ into sibling groups, and bound the time of each sibling group by that of their single parent in level $l - 1$. So the time at all nodes in a level is at most that in the preceding level. Cascading these relations, the process time at each level is at most that at the root level call on $G, H$, which is $(f(G, H) - f_0)$.

*2.2.4. Generality of the algorithm.* Our algorithm framework admits quite general schemes to search for similar subgraph pairs and score them, and is hence attractive in a biological setting. The searching scheme is flexible as our problem variant and algorithm works for different connectivity and local matching criteria. The scoring scheme is flexible as it is decoupled from the searching scheme. Besides, the number of maximal solutions is only quadratic, so it is not expensive to compute a sophisticated, biologically inspired score for every solution.

We discuss the flexibility of the local matching and connectivity criteria in more detail. We already saw different monotone local matching criteria. We could also combine such monotone criteria to get a new monotone criterion. For example, declaring two nodes as locally matching if they are so with respect to similar length-$p$ paths *or* $s$-similar neighborhoods gives a less stringent criterion. Many connectivity criteria are possible too. We could replace connectedness with biconnectedness (Baase, 1991) for instance, by simply changing the algorithm's split step and still obtain a provably efficient algorithm. The number of output solutions is then at most $m_G m_H$ and the running time $O(m_G m_H f(G, H))$, the proofs of which (omitted) are a simple extension of the connectedness proofs and use the property that the biconnected components of a graph are edge-disjoint.

## 2.3. Overall method

Our method of detecting conserved modules between two protein networks involves a searching scheme to find similar subgraph pairs (*candidate conserved modules* or *candidates*) using our graph-matching algorithm above, and a scoring scheme to rank these candidates using statistics of similar paths between a subgraph pair. We now describe these schemes and their place in the overall Match-and-Split method.

*2.3.1. Searching scheme (via graph-matching).* To adapt the generic graph-matching algorithm Match-and-Split to the specific task of comparing protein networks, we make default choices for certain graph-matching parameters and incorporate a clustering heuristic to handle solutions that are large. In this section, our algorithm's maximal solutions are referred to simply as solutions.

**Choosing parameters.** Our default parameter choices result in a lenient graph-matching criterion, for we would like to detect as many functional modules as possible from noisy protein networks of divergent organisms (e.g., yeast and human). The default choices we made on exploring a limited parameter space follow.

Connectedness defines our connectivity criterion. We choose it over biconnectedness as some functional modules (e.g., linear signaling pathways) are not biconnected, and even those over highly interacting proteins (e.g., protein complexes) may not appear biconnected due to incomplete interaction data.

Similar length-$p$ paths (for $p = 1, 2$) defines our local matching criterion. We choose it over $s$-similar neighborhoods (for $s = 1, 2$) again on the basis of sensitivity. For $p, s = 1$, both options are equivalent as they yield the same $lm(S, T)$ node-set (see Section 2.2.2). However for $p, s = 2$, this node-set from similar paths is a superset of the one from similar neighborhoods, so similar paths is a more lenient criterion.

Sequence similarity defines our node similarity function as in previous network comparison methods. We in fact choose the same criteria used in two previous methods for fair evaluation. The two criteria are: ($A$) $sim(u, v)$ is true whenever the BLAST E-value of proteins $u, v$ is at most $10^{-7}$ and each protein is among the 10 best BLAST matches of the other (Sharan et al., 2005), and ($B$) $sim(u, v)$ is true whenever the BLAST E-value of $u, v$ is less than that of 60% of ortholog pairs in some ortholog database (as reviewed in Koyuturk et al., 2006).

**Incorporating clustering heuristic.** Increased sensitivity from the lenient graph-matching criterion above comes at a cost. Sometimes, a solution is over a large number of proteins and hence less specific. For instance, a solution from a preliminary comparison of yeast and human networks covers more than 500 yeast and human proteins. To split such large solutions, we incorporate a betweenness clustering heuristic in our Match-and-Split algorithm. This clustering splits a graph into highly-connected, smaller clusters based on iterative computations of an edge betweenness centrality measure (Girvan and Newman, 2002) (see Supplemental text for details). One could also cluster a graph using other methods such as the popular spectral clustering methods (Weiss, 1999).

We incorporate the clustering by replacing our algorithm's "[base-case]" statement with the code block below. As before $n_G$ refers to the number of nodes in $G$, and we may assume $n_G \geq n_H$ without loss of generality. The parameter $n_{\max}$ (say 25) indicates when a solution is large.

[base case] code block:
**if** ($n_G \leq n_{\max}$ and $n_H \leq n_{\max}$)
  Output the maximal solution $G, H$.
**else** [large solution]
  Split $G$ into clusters $G_1, \ldots, G_e$ using betweenness clustering.
  **for** $i = 1$ to $e$
    *Match-and-Split*($G_i, H$).

A betweenness clustering of $G$ takes $O(n_G m_G^2)$ running time (Girvan and Newman, 2002). In practice, incorporating the clustering heuristic is not expensive as our experiments result in very few large solutions (mostly one), each covering just a few hundred nodes.

*2.3.2. Scoring scheme (via similar paths).* We score a candidate conserved module based on the number of similar length-$p$ paths, and express the statistical significance of the score as a $P$-value. We use the $P$-values both to rank the candidates from the searching scheme and to retain only those with $P$-values at 10% significance level after multiple testing. Our scoring scheme is flexible, as seen in Section 2.2.4, in permitting complicated scoring measures including measures in previous network comparison methods. Still we use a simple scoring measure and the promising results we obtain shows the strength of our searching scheme based on the graph-matching algorithm.

The score of a candidate conserved module $S \subseteq G, T \subseteq H$, where $G, H$ are the input protein networks, is simply the number of pairs of similar length-$p$ paths between them (see Section 2.2.1). We evaluate the $P$-value of this score using a null model that randomizes the edges and node similarity function of $G, H$ to exclude the mechanism of interest viz., conservation of protein modules. To provide a stringent control, the randomization loosely preserves the degree sequence and node similarity distribution as in previous methods. The simplicity of our scoring measure and null model allows us to develop an analytical bound on the $P$-value (see Supplemental text). This bound can also incorporate reliabilities of noisy protein interactions.

*2.3.3. Implementation pipeline and dataset.* The overall Match-and-Split method proceeds in a pipeline to detect candidate conserved modules between two protein networks. First our searching scheme uses the novel graph-matching algorithm to produce candidates, then a size filter retains only medium-sized candidates, and finally our scoring scheme ranks the candidates by their $P$-values and retains those at 10% significance level (after multiple testing). A fast implementation of the method is publicly available (see Supplemental text for website reference), and it takes only a minute or few (at most 4) on a 3.4 GHz Pentium Linux machine to compare two studied networks.

The size filter, similar to one in a previous method (Sharan et al., 2005), retains any candidate subgraph pair $S, T$ whose number of nodes $n_S, n_T$ satisfy $n_{\min} \leq n_S, n_T \leq n_{\max}$ ($n_{\min} = 3, n_{\max} = 25$ in our

experiments, and $n_{\max}$ is same as in the searching scheme's clustering heuristic). We focus on such medium-sized candidates for the following reasons. A large module as a whole is likely to correspond to a less specific function and worse causes artifactual increase in sensitivity in our evaluation studies. A small module, over say two proteins, is likely to result from a spurious match occurring simply by chance.

The protein networks for model organisms *Saccharomyces cerevisiae*, *Drosophila melanogaster*, and *Caenorhabditis elegans* (referred to hereafter as yeast, fly, and worm, respectively) are experimentally-derived (e.g., two-hybrid, immunoprecipitation) interactions collected in the DIP database (Salwinski et al., 2004). The version of the networks used and the interaction reliabilities based on a logistic regression model are taken from a previous study (Sharan et al., 2005). Our human protein network is from the HPRD database (Peri et al., 2003) (binary or direct interactions; September 2005 version), and we assume unit reliability for these interactions as they are literature-based.

### 2.4. Evaluation measures

We describe some measures that we would need in the Results section to evaluate and interpret the conserved modules from pairwise network comparisons. The notation yeast-human comparison denotes the comparison between yeast and human protein networks, and yeast-human modules the resulting conserved modules. Similar notations apply for other two-species comparisons too.

*2.4.1. Performance (sensitivity and specificity).* The performance of a pairwise network comparison method depends on the quality of candidate conserved modules it outputs, which we could measure by how well the candidate set aligns with a reference set of conserved modules. But such reference sets, even if available, are not comprehensive or applicable for our purpose (e.g., STKE (*http://stke.sciencemag.org/cm/*, June 2005) contains only a couple of fly-worm conserved signaling pathways and no protein complexes; Biocarta (*www.biocarta.com/genes/allPathways.asp*, January 2006) has many human-mouse conserved pathways but available mouse interaction data is sparse).

A viable alternative is to use known functional modules in a *single species* as reference to evaluate part of a candidate set. Our reference modules are the literature-based yeast protein complexes present in MIPS (Mewes et al., 2004) (May 2006 version), at level at most 3 in its complex category hierarchy. We consider only complexes with 3 to 25 proteins due to our study's focus on medium-sized modules (for reasons, see Section 2.3.3). To test a method, we compare the yeast network against the human, fly or worm network (for dataset details, see Section 2.3.3), collect the yeast subgraphs $S$ from each output candidate $S, T$, and measure the overlap of these single-species candidate modules with the reference modules.

Our main measures are module-level sensitivity and specificity, which are the fraction of reference modules covered by some candidate module and that of candidate modules covered by some reference module respectively. A module $S$ of proteins from a single species (yeast) is covered by another module $S'$ if $|S \cap S'|/|S| \geq 50\%$, a stringent criterion given the noisy interaction data.

We also present measures at the interaction and protein levels to assess the quality of different components of candidate modules—similar to measures at the gene, exon, and nucleotide levels in gene structure prediction methods (Burset and Guigo, 1996). Define the protein interactions *spanned* by a set of modules as the union of interactions present in each of these modules. If set $A$ denotes the interactions spanned by all candidate modules and $B$ that by all reference modules, then interaction-level sensitivity is $|A \cap B|/|B|$ and specificity is $|A \cap B|/|A|$. Protein-level measures are defined similarly.

*2.4.2. GO analysis measures.* We use the GO resource (The Gene Ontology Consortium, 2000) for functional annotations of genes. Specifically, all our functional descriptions are based on known GO Biological Process annotations (August 2006 version) of proteins to terms at level at least 4 in the GO hierarchy. Given these annotations, we next define many GO-related concepts that we would need later.

The *best GO term* of a protein module is the term the module's proteins are enriched for with the least hypergeometric $P$-value (computed by GO::TermFinder [Boyle et al., 2004] at 10% significance level with Bonferroni correction). Given two GO terms, the *match* between them is the overlap $\min(|A \cap B|/|A|, |A \cap B|/|B|)$ between the set $A, B$ of terms they imply, where a GO term implies itself and its ancestors in the GO ontology. This match is based on a well-justified measure (Kiritchenko et al., 2005).

We call a protein module *functionally homogeneous* if at least 50% of its proteins are annotated with the best GO term the whole module is enriched for. We call a candidate conserved module $S, T$ (e.g., yeast-

human candidate) as *functionally similar* with respect to GO if $S$ and $T$ are each functionally homogeneous and the match between their best GO terms is at least 75%.

When validating a predicted GO term of a protein, we use a well-justified criterion as for match between terms. This criterion requires the respective sets $A, B$ of terms implied by the predicted term and some known term annotated to the protein to have a high overlap $|A \cap B|/|A| \geq 75\%$.

## 3. RESULTS

### 3.1. Performance against previous methods

We test our Match-and-Split method against two previous methods, NetworkBLAST and MaWISh. We try two Match-and-Split versions, $p = 1$ and $p = 2$, which specify the local matching criterion of similar length-$p$ paths. We use the default versions of NetworkBLAST and MaWISh that detect conserved protein complexes or subnetworks (see Supplemental text for software references). We don't test the Bayesian alignment (Berg and Lassig, 2006) and Græmlin (Flannick et al., 2006) methods as they are from very recent studies with results focusing on coexpression or prokaryotic networks, whereas the current study's focus is eukaryotic protein networks. Fundamentally though, all these methods work for the networks of any species.

We evaluate a method by measuring how well the yeast modules it outputs (on pairwise protein network comparisons of yeast and other species) align with a reference set of yeast protein complexes in MIPS (see Section 2.4.1). For fair evaluation, we attempt to use common input, default parameter values, and common output processing for all methods. For instance, the input networks *and* node similarity function $sim(.,.)$ are the same across methods. The $3, 25$ size filter thresholds are same too, so we evaluate all methods only on the *medium-sized candidates* they output (see Section 2.3.3).

Different $sim(.,.)$ criteria could yield quite varying results, and we try two criteria used in previous methods (see Section 2.3.1). The main text presents criterion $A$ results, and the Supplemental text presents some criterion $B$ results to show few changes in the relative performance of the methods between the two criteria (mainly in yeast-fly comparison where module level results of NetworkBLAST are better than other methods with criterion $B$ and not $A$).

First, we discuss module-level results of Match-and-Split ($p = 1$) relative to the other two methods. In yeast-human comparison (Table 1), Match-and-Split and MaWISh have comparable performance with Match-and-Split being somewhat better, and NetworkBLAST has better sensitivity than other methods at the cost of very low specificity and an output of too many candidates. A similar though less pronounced trend appears in the yeast-fly case (Table 2). In the yeast-worm case (Table 1 of the Supplemental text), NetworkBLAST has much better specificity than other methods at comparable sensitivity.

TABLE 1. EVALUATION OF OUTPUT CANDIDATES FROM YEAST-HUMAN NETWORK COMPARISON USING SENSITIVITY AND SPECIFICITY MEASURES (EXPRESSED AS ROUNDED PERCENTAGES) AT THE MODULE, INTERACTION, AND PROTEIN LEVELS

| *Method* | *No. of output modules (interactions, proteins)* | *Module* | | *Interaction* | | *Protein* | |
|---|---|---|---|---|---|---|---|
| | | *Sens.* | *Spec.* | *Sens.* | *Spec.* | *Sens.* | *Spec.* |
| Match-and-Split | | | | | | | |
| $p = 1$ | 80 (667, 421) | 25.0 | 47.5 | 20 | 35 | 25 | 48 |
| $p = 2$ | 72 (664, 411) | 28.0 | 41.7 | 20 | 34 | 25 | 47 |
| NetworkBLAST | 421 (1311, 606) | 40.9 | 18.5 | 34 | 30 | 37 | 48 |
| MaWISh | 151 (508, 389) | 20.5 | 43.7 | 15 | 33 | 23 | 46 |

The second column shows the number of yeast modules (candidates) output, and the number of interactions and proteins spanned by these yeast modules. The reference set comprises 132 medium-sized (size 3–25) yeast complexes in MIPS that span 1144 interactions and 791 proteins. For relevant definitions, see Section 2.4.1.

Sens., sensitivity; spec., specificity.

TABLE 2. EVALUATION OF OUTPUT CANDIDATES FROM YEAST-FLY NETWORK COMPARISON, USING THE SAME FORMAT AS TABLE 1

| *Method* | *No. of output modules (interactions, proteins)* | *Module* | | *Interaction* | | *Protein* | |
|---|---|---|---|---|---|---|---|
| | | *Sens.* | *Spec.* | *Sens.* | *Spec.* | *Sens.* | *Spec.* |
| Match-and-Split | | | | | | | |
| $p = 1$ | 27 (155, 123) | 6.8 | 51.9 | 5 | 35 | 8 | 49 |
| $p = 2$ | 25 (131, 110) | 7.6 | 48.0 | 4 | 37 | 7 | 53 |
| NetworkBLAST | 77 (354, 206) | 8.3 | 39.0 | 9 | 28 | 12 | 46 |
| MaWISh | 26 (81, 87) | 5.3 | 50.0 | 3 | 40 | 6 | 52 |

Sens., sensitivity; spec., specificity.

Moving from relative to absolute performance, the values of module-level sensitivity and specificity are low (e.g., a mere 25.0% and 47.5%, respectively, by Match-and-Split ($p = 1$), a competitive method in yeast-human case). Low sensitivity could be due to factors like noisy interaction data, poor conservation of complexes across compared organisms, or differing computational and biological definitions of a functional module or complex. Finding which of these factors is key is a subject of future work. Low specificity is probably due to incompleteness of the reference set, and it does not indicate the output candidates are spurious (a claim supported by further analysis in Section 3.4).

### *3.2. Single-species versus pairwise network analysis*

Pairwise network comparison methods use cross-species conservation in an attempt to improve over single-species methods in detection of functional modules (see Section 1). But previous network comparison studies have not evaluated their methods against single-species ones. Here we undertake this evaluation by testing a popular single-species method MCODE (Bader and Hogue, 2003) on the yeast network under the same measures and size range (3 to 25 proteins) as before. It is beyond the scope of this study to test all single-species methods.

The performance of the pairwise methods relative to the single-species MCODE is varied. Under proper homolog and species selection, Match-and-Split performs better than MCODE in detecting reference yeast complexes. Comparing Table 1 on yeast-human comparison with Table 3, Match-and-Split ($p = 1$) is much more sensitive than MCODE at the same specificity, and MaWISh performs similar to MCODE. The choice of homologs is important as changing the $sim(.,.)$ criterion from $A$ here to $B$ in Table 2 of the Supplemental text reduces the benefit of pairwise methods over MCODE. Species selection is also crucial as the pairwise methods have worse sensitivity than MCODE in the yeast-fly (Table 2) and yeast-worm (see Table 1 of the Supplemental text) cases. Low sensitivity in the yeast-worm case could be due to the sparse worm interaction data, but the reason in the yeast-fly case is unclear as the fly network is interaction-rich.

Our Match-and-Split searching scheme includes a betweenness clustering heuristic. The heuristic is by itself a single-species method (denoted Split-only) for it can cluster the full yeast network into highly connected modules. Table 4 compares Match-and-Split and Split-only to show the boost in specificity from

TABLE 3. EVALUATION OF OUTPUT CLUSTERS FROM YEAST NETWORK ANALYSIS BY MCODE, A SINGLE-SPECIES CLUSTERING METHOD, USING THE SAME FORMAT AS TABLE 1

| *Method* | *No. of output modules (interactions, proteins)* | *Module* | | *Interaction* | | *Protein* | |
|---|---|---|---|---|---|---|---|
| | | *Sens.* | *Spec.* | *Sens.* | *Spec.* | *Sens.* | *Spec.* |
| MCODE | 53 (614, 323) | 16.7 | 47.2 | 19 | 36 | 20 | 48 |

We focus on medium-sized (size 3–25) clusters as in other evaluations.
Sens., sensitivity; spec., specificity.

TABLE 4. EVALUATION OF MATCH-AND-SPLIT ($p = 1$) ON PAIRWISE YEAST-HUMAN NETWORK COMPARISON AGAINST SPLIT-ONLY ON SINGLE-SPECIES YEAST NETWORK CLUSTERING, AGAIN USING THE SAME FORMAT AS TABLE 1

| *Method* | *No. of output modules (interactions, proteins)* | *Module* | | *Interaction* | | *Protein* | |
|---|---|---|---|---|---|---|---|
| | | *Sens.* | *Spec.* | *Sens.* | *Spec.* | *Sens.* | *Spec.* |
| Match-and-Split | 81 (634, 410) | 24.2 | 48.2 | 20 | 35 | 25 | 49 |
| Split-only | 569 (3597, 2776) | 50.0 | 12.3 | 58 | 18 | 76 | 22 |

Similar results from Match-and-Split ($p = 2$) is omitted. As betweenness clustering of large graphs is compute-intensive, Split-only uses a quicker version of it (see Supplemental text; Split-only still requires orders of magnitude more time than Match-and-Split). For fairness, the Match-and-Split version here uses the quicker clustering inside its searching scheme.

Sens., sensitivity; spec., specificity.

adding pairwise match criterion to plain clustering. Split-only detects more reference yeast complexes but at the cost of outputting too many candidate modules at very low specificity. Adding pairwise match criterion also drastically reduces running time by restricting the size of graphs that need to be betweenness clustered.

### 3.3. Select conserved modules

The results above evaluate the candidates from our method on a global scale. Here we discuss the biology of a select few candidates. We start with a flavour of some top-ranked candidates from the Match-and-Split ($p = 1$) yeast-human comparison in Tables 5 and 6. These tables contain functional descriptions based on some known GO Biological Process annotations (see Section 2.4.2). The format of these tables is inspired from a previous study (Koyuturk et al., 2005).

Consider the candidate ranked 2 in Table 5 and shown in Figure 4. From literature-based descriptions in SGD (Hong et al., 2006) and UniProt (Apweiler et al., 2004), the yeast and human proteins of this candidate are each components of the TIM23 complex, a mitochondrial inner membrane translocase. This complex mediates translocation of preproteins across the mitochondrial inner membrane. Typical preproteins are nuclear-encoded, synthesized in the cytosol and contain a targeting sequence (presequence or transit peptide) to direct transport.

We now elaborate on the candidate ranked 72 in Table 6 and shown in Figure 4. The candidate may seem too heterogeneous to be conserved but it actually contains many homologous complexes as inferred from literature-based comments at SGD and UniProt. This example illustrates how a lenient matching

TABLE 5. FIVE TOP-RANKED CANDIDATES FROM MATCH-AND-SPLIT ($p = 1$) YEAST-HUMAN COMPARISON

| *Rank* | *P-value (score)* | *Size* | *Best GO term of module (% annotated proteins)* | | *Terms' match* |
|---|---|---|---|---|---|
| | | | *Yeast* | *Human* | |
| 1 | 3.33e-13 (8) | 5, 3 | Purine ribonucleoside salvage (100%) | Nucleoside metabolism (100%) | 53% |
| 2 | 1.05e-12 (3) | 3, 4 | Protein import into mitochondrial matrix (100%) | Protein targeting to mitochondrion (75%) | 89% |
| 3 | 1.38e-12 (4) | 3, 3 | Postreplication repair (100%) | DNA repair (100%) | 94% |
| 4 | 1.40e-12 (6) | 4, 3 | ER to Golgi vesicle-mediated transport (100%) | ER to Golgi vesicle-mediated transport (100%) | 100% |
| 5 | 1.89e-12 (2) | 3, 3 | Processing of 20S pre-rRNA (100%) | rRNA processing (100%) | 95% |

The size of a candidate (third column) is the number of its yeast, human proteins. The "% annotated proteins" is the fraction of proteins in a module annotated with the module's best GO term, and the match shown is between the best GO terms of yeast module $S$ and human module $T$ in a candidate $S, T$ (for definitions, see Section 2.4.2).

TABLE 6. FIVE TOP-RANKED CANDIDATES WITH AT LEAST 10 YEAST AND 10 HUMAN PROTEINS FROM MATCH-AND-SPLIT ($p = 1$) YEAST-HUMAN COMPARISON, PRESENTED IN THE SAME FORMAT AS TABLE 5

| | | | *Best GO term of module (% annotated proteins)* | | |
|---|---|---|---|---|---|
| *Rank* | *P-value (score)* | *Size* | *Yeast* | *Human* | *Terms' match* |
| 50 | 7.76e-10 (40) | 12, 10 | Ubiquitin-dependent protein catabolism (100%) | Ubiquitin-dependent protein catabolism (100%) | 100% |
| 72 | 1.02e-08 (16) | 12, 12 | DNA-dependent DNA replication (75%) | DNA metabolism (91.7%) | 83% |
| 74 | 1.48e-08 (95) | 16, 17 | Protein amino acid phosphorylation (81.2%) | Phosphorus metabolism (88.2%) | 35% |
| 77 | 6.40e-08 (18) | 15, 13 | Transcription initiation (86.7%) | Transcription initiation (69.2%) | 100% |
| 78 | 1.28e-07 (79) | 20, 23 | Actin filament organization (65%) | Rho protein signal transduction (43.5%) | 11% |

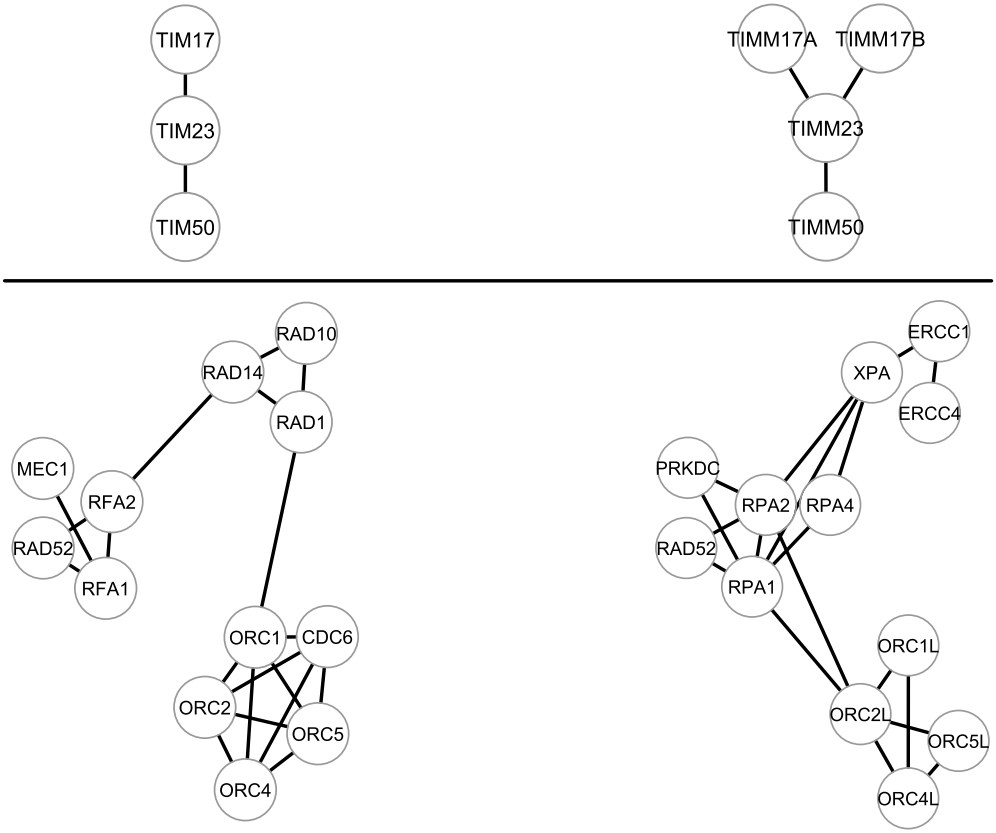


**FIG. 4.** Select candidates from Match-and-Split ($p = 1$) yeast-human comparison. Each candidate is a conserved module of yeast **(left)** and human **(right)** proteins. Two proteins similar by the *sim*(.,.) function are roughly aligned horizontally.

criterion over noisy interactions could detect known complexes. The origin recognition complex (of the ORC proteins) with counterparts in yeast and human binds replication origins, and plays a role in DNA replication and transcriptional silencing. The RAD1, RAD10 and RAD14 proteins are subunits of the Nucleotide Excision Repair Factor 1 (NEF1) in yeast and homologous to the ERCC4, ERCC1 and XPA respectively in human. The RFA1, RFA2 in yeast (homologs RPA1, RPA2 in human) are subunits of heterotrimeric Replication Factor A (RF-A), a single-stranded DNA-binding protein involved in DNA replication, repair and recombination.

### *3.4. Annotation transfer from yeast to human*

The candidates output by pairwise network comparison methods, like the ones sampled in last section, enable transfer of functional annotations between organisms. The idea is to annotate a protein module in one organism with a function that a similar module in another organism is known to be enriched for. Our focus here is annotation transfer from yeast to human based on candidate conserved modules (i.e., yeast-human candidates) output by Match-and-Split ($p = 1$), using certain known GO Biological Process annotations described in Section 2.4.2. The Match-and-Split results in this section are competitive with the corresponding results of other tested methods (shown in Tables 3 and 4 of the Supplemental text).

We first present results on functional homogeneity and similarity of yeast-human candidates (as defined with respect to GO in Section 2.4.2). Collect the yeast module $S$ of every yeast-human candidate $S, T$. Then all (100%) of these yeast modules are homogeneous for some function, which could then be transferred. This fraction is 83.8% for the human modules collected similarly. The fraction of yeast-human candidates functionally similar with respect to GO is a reasonable 42.5%, which further supports annotation transfer.

The actual transfer on each yeast-human candidate $S, T$ involves assigning all human proteins in $T$ to the best GO term the yeast module $S$ is enriched for. If this procedure predicts more than one GO term for a human protein, we retain the term with the least hypergeometric $P$-value (see Section 2.4.2). We predict GO Biological Process terms for a total of 462 human proteins (predictions available online; see Supplemental text). This annotation transfer is reliable as a reasonable 295 of these predictions are valid by a stringent, well-justified criterion in Section 2.4.2. This transfer covers only 462 proteins though, a small fraction of the 1882 proteins in the human network sequence-similar to some protein in the yeast network (by the $sim(.,.)$ function).

## 4. DISCUSSION

In the context of comparing two protein networks, this work shows it is possible to design a provably good search algorithm that also translates to promising performance in practice. The algorithmic guarantees of our Match-and-Split method distinguishes it from previous methods based on greedy heuristics. Formal guarantees are important as they lend credibility to the conserved modules found by a bounded-time search procedure.

Our method Match-and-Split performs competitively in tests against two previous pairwise network comparison methods. For instance, Match-and-Split performs comparable to or somewhat better than the MaWISh method. In tests against a single-species method MCODE, Match-and-Split performs better in yeast-human comparison. This single-species test, which was not done in previous pairwise studies, also reveals comparisons (yeast-fly and yeast-worm) where the pairwise methods are poorer than MCODE.

The above evaluations, especially the single-species one, leads to an immediate future question. Are the poor results of pairwise methods in some comparisons mainly due to incomplete interaction data or something intrinsic to the choice of the species pair and homologs between them? The answer could inform the conditions when pairwise methods exploit cross-species conservation to improve single-species detection of functional modules.

The conserved modules our method detects and their functional descriptions would be the findings of most practical interest to biologists. Also of interest would be the reasonably accurate functional predictions resulting from the transfer of GO annotations between conserved modules. We make these findings publicly available (at a website mentioned in the Supplemental text), along with a fast Match-and-Split implementation to facilitate new network comparisons.

Our graph-matching algorithm is flexible in allowing diverse local matching and connectivity criteria. A biologist could for instance design a stringent matching criterion to detect similar instances of a functional module in a single-species network (duplicated modules). We could even compare other types of networks like metabolic networks within the same algorithmic framework. A challenge then is the judicious design of a matching criterion for the biological comparison of interest.

Our algorithm guarantees are limited to monotone local matching criteria. Many useful criteria, such as one that declares two nodes locally matching if they are similar and at least half of their neighbors are similar, are non-monotone. A future direction is to explore tractable search formulations for non-monotone criteria. Another limitation with the current study, but not with our framework, is the use of a simple scoring scheme. The simple scores yield reasonably good results; however, network conservation scores that correlate better with the biological significance of a conserved module are a subject of future work.

## ACKNOWLEDGMENTS

We thank Jayanth Kumar Kannan for valuable discussions that led to the algorithm. We thank Sourav Chatterji, Sridhar Rajagopalan, and Ben Reichardt for comments at different stages of the project. We thank Roded Sharan, Silpa Suthram, and Mehmet Koyutürk for help with their previous work. This work was partially supported by NSF (Award 331494).

## REFERENCES

Apweiler, R., Bairoch, A., Wu, C., et al. 2004. UniProt: the Universal Protein Knowledgebase. *Nucleic Acids Res.* 32, D115–D119.

Baase, S. 1991. *Computer Algorithms: Introduction to Design and Analysis*, 2nd ed. Addison-Wesley, San Francisco.

Bader, G., and Hogue, C. 2003. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinform.* 4, 2.

Berg, J., and Lassig, M. 2006. Cross-species analysis of biological networks by Bayesian alignment. *Proc. Natl. Acad. Sci. USA* 103, 10967–10972.

Bork, P., Jensen, L., von Mering, C., et al. 2004. Protein interaction networks from yeast to human. *Curr. Opin. Struct. Biol.* 14, 292–299.

Boyle, E., Weng, S., Gollub, J., et al. 2004. GO::TermFinder—open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes. *Bioinformatics* 20, 3710–3715.

Bunke, H. 1999. Error correcting graph matching: on the influence of the underlying cost function. *IEEE Trans. Pattern Anal. Mach. Intell.* 21, 917–922.

Burset, M., and Guigo, R. 1996. Evaluation of gene structure prediction programs. *Genomics* 34, 353–367.

Flannick, J., Novak, A., Srinivasan, B., et al. 2006. Græmlin: general and robust alignment of multiple large interaction networks. *Genome Res.* 16, 1169–1181.

Garey, M., and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York.

The Gene Ontology Consortium. 2000. Gene Ontology: tool for the unification of biology. *Nat. Genet.* 25, 25–29.

Girvan, M., and Newman, M. 2002. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* 99, 7821–7826.

Hartwell, L., Hopfield, J., Leibler, S., et al. 1999. From molecular to modular cell biology. *Nature* 402, C47–C52.

Hong, E., Balakrishnan, R., Christie, K., et al. 2006. Saccharomyces genome database. Available at: *www.yeastgenome.org/*. Accessed June 15, 2007.

Kiritchenko, S., Matwin, S., and Famili, A. 2005. Functional annotation of genes using hierarchical text categorization. BioLINK SIG: Linking Literature, Information and Knowledge for Biology.

Koyuturk, M., Grama, A., and Szpankowski, W. 2005. Pairwise local alignment of protein interaction networks guided by models of evolution. *Proc. 9th Int. Conf. Comput. Mol. Biol.*, 48–65.

Koyuturk, M., Kim, Y., Topkara, U., et al. 2006. Pairwise alignment of protein interaction networks. *J. Comput. Biol.* 13, 182–199.

Mewes, H., Amid, C., Arnold, R., et al. 2004. MIPS: analysis and annotation of proteins from whole genomes. *Nucleic Acids Res.* 32, D41–D44.

Peri, S., Navarro, J., Amanchy, R., et al. 2003. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Res.* 13, 2363–2371.

Salwinski, L., Miller, C., Smith, A., et al. 2004. The Database of Interacting Proteins. *Nucleic Acids Res.* 32, D449–D551.

Schellewald, C. 2005. *Convex mathematical programs for relational matching of object views* [Ph.D. dissertation]. University of Mannheim, Germany.

Shannon, P., Markiel, A., Ozier, O., et al. 2003. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* 13, 2498–2504.

Sharan, R., and Ideker, T. 2006. Modeling cellular machinery through biological network comparison. *Nat. Biotechnol.* 24, 427–433.

Sharan, R., Suthram, S., Kelley, R., et al. 2005. Conserved patterns of protein interaction in multiple species. *Proc. Natl. Acad. Sci. USA* 102, 1974–1979.

Weiss, Y. 1999. Segmentation using eigenvectors: a unifying view. *Proc. IEEE Int. Conf. Comput. Vision* 975–982.

Address reprint requests to:
*Dr. Manikandan Narayanan*
*Computer Science Division*
*593 Soda Hall*
*University of California, Berkeley*
*Berkeley, CA 94720*

*E-mail:* nmani@cs.berkeley.edu

**This article has been cited by:**

1. W. Ali, C. M. Deane. 2009. Functionally guided alignment of protein interaction networks for module detection. *Bioinformatics* **25**:23, 3166-3173. [CrossRef]