Algorithms to detect multi-protein modularity conserved during evolution

Luqman Hodgkinson
Richard M. Karp

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2011-7
http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-7.html

January 26, 2011

Algorithms to detect multi-protein modularity conserved during evolution

Luqman Hodgkinson[1], Richard M. Karp[2]

[1] Center for Computational Biology, University of California, Berkeley
luqman@berkeley.edu
[2] Computer Science Division, University of California, Berkeley,
and the International Computer Science Institute
karp@icsi.berkeley.edu

Abstract. A multi-protein module is a collection of proteins exhibiting modularity in their interactions. Multi-protein modules may perform essential functions and be conserved by purifying selection. Detecting essential multi-protein modules that change infrequently during evolution is a challenging algorithmic task. A new linear-time algorithm named Produles offers significant algorithmic advantages over previous approaches. An algorithmic framework for evaluation is presented that facilitates evaluation of algorithms for detecting conserved modularity with respect to their algorithmic goals.

1 Introduction

The biological cell is a fascinating complex system with much to be understood. Proteins are molecular machines of the cell. Each protein functions in a neighborhood, interacting with other proteins and molecules to perform its tasks. There are approximately 22,000 genes in the human genome [1] and still more proteins due to alternative splicing [2]. As it is difficult to determine exactly how most proteins interact, preliminary work has focused on determining whether two given proteins can bind to each other [3]. Binding relationships for the proteins of an organism yield a graph, the interactome, with vertices representing proteins and edges representing potential interactions [3]. The interactome is only an approximation to cell organization as two proteins that could bind to each other may be carefully separated into separate locations by cell machinery and may never interact in vivo [4]. Moreover, in multi-cellular organisms, there are many cell types, and only a subset of genes are expressed in each cell type [4]. There are also temporal separations in which various genes are expressed at different times in the cell cycle [4]. Yet, the interactome is a reasonable beginning for understanding organization of protein interactions in the cell.

There is primary evidence of multi-protein modularity conserved for hundreds of millions of years including fundamental molecular mechanisms underlying cell-to-cell communication such as the Notch pathway [5]. How widespread are these instances of conserved multi-protein modularity? How can we effectively use computational techniques to identify conserved modularity in interactomes with thousands of proteins? Detecting modularity in large biological data sets can be challenging, both because modules have imprecise boundaries in biological systems [6] and because data sets are incomplete and imprecise [7].

Early ground-breaking studies searched for conserved pathways in interactomes for H. pylori and S. cerevisiae [8], and for conserved complexes in interactomes for C. elegans, D. melanogaster, and S. cerevisiae [9]. Additional attempts to identify conserved modularity in various interactomes have been subsequently published [10–15].

Following this introduction, Section 2 describes the input data for this study. Section 3 introduces Produles, a new linear-time algorithm to detect conserved multi-protein modules under the premise that a multi-protein module is a set of proteins with low conductance in an interactome. Section 4 describes software tools that facilitate comparison of algorithms on current interactomics data. Section 5 describes a collection of quality measures for evaluating algorithms. Section 6 discusses previously-published algorithms for detecting conserved modularity. Section 7 evaluates the performance of Produles with respect to algorithmic goals and compares with other algorithms. Section 8 discusses findings of the study. Section 9 concludes the paper.

2 Input data

An interactome is an undirected graph $G = (V, E)$, where V is a set of proteins and $(v_1, v_2) \in E$ iff protein v_1 interacts with protein v_2. In this study the input is restricted to a pair of interactomes, $G_i = (V_i, E_i)$ for $i \in \{1, 2\}$ and protein sequence similarity values, $h : V_1 \times V_2 \rightarrow \mathbb{R}^+$, based on BLAST [16], defined only for the most sequence similar pairs of proteins appearing in the interactomes. As BLAST E-values change when the order of the interactomes is reversed, h is defined with the rule

$$h(v_1, v_2) = h(v_2, v_1) = \frac{E(v_1, v_2) + E(v_2, v_1)}{2}$$

where $E(v_1, v_2)$ is the minimum BLAST E-value for $v_1 \in V_1, v_2 \in V_2$ when v_1 is tested for homology against the database formed by V_2. Any algorithm using only this data is a general tool as it can be applied easily to any pair of interactomes, including those for newly studied species.

3 Produles

One definition of a modular system is a system with parts organized in such a way that strong interactions occur within each group or module, but parts belonging to different modules interact only weakly [17,18]. Following this, a natural definition of multi-protein modularity is that proteins within a module are more likely to interact with each other than to interact with proteins outside of the module. Let $G = (V, E)$ be an interactome. A multi-protein module is a set of proteins $M \subset V$ such that $|M| \ll |V|$ and M has a large value of

$$\mu(M) = \frac{|E(M)|}{|\texttt{cut}(M, V \backslash M)| + |E(M)|}$$

where $E(M)$ is the set of interactions with both interactants in M, and $\texttt{cut}(M, V\backslash M)$ is the set of interactions spanning M and $V\backslash M$. Of the interactions involving proteins in M, the fraction contained entirely within M is given by $\mu(M)$.

The conductance of a set of vertices in a graph is a well-studied quantity in graph theory [19] defined as

$$\Phi(M) = \frac{|\texttt{cut}(M, V\backslash M)|}{|\texttt{cut}(M, V\backslash M)| + 2\min(|E(M)|, |E(V\backslash M)|)}.$$

When $|E(M)| \leq |E(V\backslash M)|$, as for all applications in this study,

$$\Phi(M) = \frac{|\texttt{cut}(M, V\backslash M)|}{|\texttt{cut}(M, V\backslash M)| + 2|E(M)|} = \frac{1 - \mu(M)}{1 + \mu(M)}.$$

Thus, when searching for relatively small modules in a large interactome, minimizing conductance is equivalent to maximizing modularity.

The Produles algorithm begins by finding a small module,

$$M \subset V_1$$

with high modularity in G_1 using a local module-finding algorithm as described in Appendix E. Let

$$\mathcal{H}_T(M) = \{v \mid \exists\ u \in M \text{ such that } h(u, v) \leq T\}$$

Modules corresponding to the connected components of the subgraph of G_2 induced by $\mathcal{H}_T(M)$ are candidates for conservation with M. Let these modules be $N_1, N_2, ..., N_k$. For $i = 1, ..., k$, let

$$\mathcal{R}_T(M, N_i) = \{u \in M \mid \exists\ v \in N_i \text{ such that } h(u,v) \leq T\}$$

If the following are true:

$$\begin{aligned} a &\leq |\mathcal{R}_T(M,N_i)| \leq b \\ a &\leq |N_i| \leq b \\ \frac{1}{c}|N_i| &\leq |\mathcal{R}_T(M,N_i)| \leq c|N_i| \\ \mu(\mathcal{R}_T(M,N_i)) &\geq d \\ \mu(N_i) &\geq d \end{aligned}$$

where a is a lower bound on size, b is an upper bound on size, c is a balance parameter, and d is a lower bound on desired modularity, and if $\mathcal{R}_T(M, N_i)$ yields a connected induced subgraph of G_1, then Produles returns the pair $(\mathcal{R}_T(M, N_i), N_i)$ as a conserved multi-protein module. Each protein is used exactly once as a starting vertex for the local module-finding algorithm. When all proteins in G_1 have been used as starting vertices, the roles of G_1 and G_2 are reversed, and the entire process is repeated.

With careful implementation and refinement of the basic idea sketched above, Produles requires time linear in the size of the input. Appendix B contains algorithm details and proof of linear running time.

4 Software tools

In order to compare various algorithms on current data, certain software development was necessary.

4.1 EasyProt

EasyProt [20] is a parallel software architecture for acquiring and processing proteomics and interactomics data, and for analyzing results of algorithms that detect conserved multi-protein modules. EasyProt builds on the dataflow concepts from SEDA [21]. In EasyProt, each task, called an element, executes in parallel and passes messages along a DAG in which the elements are vertices. A type system has been developed suitable for proteomics and interactomics data that is used for the messages passed along the DAG. Each message stores a description that identifies the elements through which it and its predecessors have passed, along with any parameters that were used. The user specifies the elements, sets their parameters, and specifies the DAG edges, using a simple graph language that is compiled into Java using ANTLR [22]. Several message-passing protocols are supported, including broadcast and round-robin.

Elements have been developed for several classes of tasks: obtaining interactomics and proteomics data, managing a cache for intermediate storage, running protein homology detection algorithms, running various algorithms for detecting conserved multi-protein modules and converting to standardized formats, analyzing sets of conserved multi-protein modules, and generating VieprotML for visualization in VieProt as described in Subsection 4.2.

All steps from data acquisition to final analysis are entirely within the EasyProt framework. This ensures that all algorithms are treated fairly, running on the same data sets and receiving the same analysis. The EasyProt framework allows previously published proof-of-concept implementations for detecting conserved multi-protein modules to be used robustly as practical tools. Currently Produles, NetworkBlast-M [23], MaWISh [10], and Match-and-Split [13] are fully supported in EasyProt and can be run on any data set with clear algorithmic evaluation of results.

4.2 VieProt

VieProt [24] is a tool for visualizing conserved multi-protein modules with a dynamic force-directed layout, that accepts data in a custom XML format, VieprotML, generated by EasyProt. VieProt is an enormous improvement on the most current alternate visualization tools such as VANLO [25]. With VieProt, it is easy to evaluate visually new and old algorithmic ideas for detecting conserved multi-protein modularity. Proteins, interactions and interaction sources, protein sequence similarities, GO annotations [26], and algorithmic measures of quality are displayed in a visually-appealing format. An image from VieProt is included as Appendix H.

5 Algorithm quality measures

A collection of quality measures for individual conserved multi-protein module pairs is defined. Algorithms are evaluated by taking expectations and standard deviations over the sets of module pairs that they return.

Let

$$\mathcal{M} = \{(M_1^i, M_2^i) \mid i \in \{1, ..., k\}\}$$

be k pairs of conserved modules returned by an algorithm. Let

$$(M_1, M_2) \in \mathcal{M}$$

Let

$$M \in \{M_1, M_2\}$$

Let

$$G_{int}(M) = (M, E(M))$$

be the interaction graph corresponding to M where $E(M)$ is the set of interactions with both interactants in M. Let

$$G_{hom}(M) = (M, H(M))$$

be the homology graph corresponding to M where, for $p_1, p_2 \in M$, $(p_1, p_2) \in H(M)$ iff $h(p_1, p_2)$ is defined. Let

$$S(M) = |M|$$

Let

$$\Delta(M) = |E(M)| / \binom{|M|}{2}$$

be a measure of module interaction density: the actual number of interactions divided by the maximum possible number of interactions. Let $C(M)$ be the number of connected components in $G_{int}(M)$. Let

$$\begin{aligned} f_a(M_1, M_2) &= (f(M_1) + f(M_2))/2 \\ f_d(M_1, M_2) &= |f(M_1) - f(M_2)| \end{aligned}$$

where $f \in \{\mu, S, \Delta, C\}$. The subscript a denotes average over the two modules in the pair and the subscript d denotes difference between the two modules in the pair.

An appropriate measure of overlap between pairs of conserved modules considers that a module may have undergone duplication in a lineage, leading to the same module being conserved in two different locations of the interactome for the lineage that underwent module duplication. Thus, for module pair (M_1^i, M_2^i), we define its overlap by other module pairs in the set to be

$$\mathcal{O}^i = \max_{j \neq i} \min\{|M_1^j \cap M_1^i| / |M_1^i|, |M_2^j \cap M_2^i| / |M_2^i|\}$$

A value of $\mathcal{O}^i = x$ implies that no module pair $j \neq i$ exists that covers more than fraction x of each module in module pair i.

Let (M_1, M_2) be a conserved module pair. Consider a pair of sets (P_1, P_2) where

$$P_1 \subseteq M_1, P_2 \subseteq M_2$$

and all proteins in $P_1 \cup P_2$ form a single connected component in G_{hom}. It is reasonable to assume that all proteins in $P_1 \cup P_2$ evolved either from a single ancestral protein or from a small number of ancestral proteins whose domains have become commingled in the descendants. By a slight abuse of notation, we call $p = (P_1, P_2)$ an ancestral protein. P_i is called the projection of p on M_i for $i \in \{1, 2\}$.

Let M_a be the ancestral module defined as the set of ancestral proteins. Note that M_a is a function of (M_1, M_2). Let $p, q \in M_a$ where $p = (P_1, P_2), q = (Q_1, Q_2)$. For $i, j \in \{1, 2\}$, if there is an interaction in G_i between some $p' \in P_i$ and some $q' \in Q_i$ but no interaction in G_j between any $p'' \in P_j$ with any $q'' \in Q_j$, then there is said to be relationship disagreement concerning the ancestral interaction between p and q. The maximum possible number of relationship disagreements is

$$\mathcal{R}_{max}(M_1, M_2) = \binom{|M_a|}{2}$$

If (M_1, M_2) has $\mathcal{R}(M_1, M_2)$ relationship disagreements, then the modules in the pair disagree on fraction

$$E_r(M_1, M_2) = \mathcal{R}(M_1, M_2)/\mathcal{R}_{max}(M_1, M_2)$$

of ancestral relationships.

The ancestral module is a hypothesis for the number of protein losses and protein duplications in the lineages leading to the present-day modules. For $\ell \in \{1, 2\}$, let

$$\pi_\ell(M_a) = \{p \in M_a \mid P_\ell \neq \emptyset\}$$

be the set of ancestral proteins that have a nonempty projection on module M_ℓ in interactome ℓ. If the projection of an ancestral protein on M_ℓ contains k distinct proteins, then we say that $k - 1$ duplications of the ancestral protein occurred. Thus, the number of protein duplications in M_ℓ is $|M_\ell| - |\pi_\ell(M_a)|$. The total number of protein duplications in the module pair is

$$\mathcal{D}(M_1, M_2) = |M_1| - |\pi_1(M_a)| + |M_2| - |\pi_2(M_a)|$$

The maximum possible number of protein duplications occurs when $|M_a| = 1$,

$$\mathcal{D}_{max}(M_1, M_2) = |M_1| + |M_2| - 2$$

The fraction of possible protein duplications exhibited by M_1 and M_2 is

$$E_d(M_1, M_2) = \mathcal{D}(M_1, M_2)/\mathcal{D}_{max}(M_1, M_2)$$

The number of protein losses is

$$\begin{aligned} \mathcal{L}(M_1, M_2) &= (|M_a| - |\pi_1(M_a)|) + (|M_a| - |\pi_2(M_a)|) \\ &= 2|M_a| - |\pi_1(M_a)| - |\pi_2(M_a)| \end{aligned}$$

which is maximized over M_a when each of the two differences in parentheses is maximized. The maximum possible number of protein losses occurs when $|M_a| = |M_1| + |M_2|$,

$$\mathcal{L}_{max}(M_1, M_2) = |M_2| + |M_1|$$

The fraction of possible protein losses exhibited by M_1 and M_2 is

$$E_\ell(M_1, M_2) = \mathcal{L}(M_1, M_2)/\mathcal{L}_{max}(M_1, M_2)$$

(E_r, E_d, E_ℓ) is a measure of purifying selection. Under the assumption that (M_1, M_2) is a conserved module pair, when E_r, E_d, and E_ℓ are small, few changes took place during evolution, evidence of strong purifying selection. $||(E_r, E_d, E_\ell)||_2$ is a summary measure of the strength of purifying selection.

$|M_a|$ is a measure of significance of a conserved module pair. It describes how many ancestral proteins participated in the ancestral module.

Let $C(M_a)$ be the number of connected components in an interaction graph with vertex set M_a, where an interaction is defined between two ancestral proteins $p, q \in M_a$ if any protein in the projection of p on M_i interacts with any protein in the projection of q on M_i, for some $i \in \{1, 2\}$. Any value of $C(M_a) > 1$ implies that the module pair is not well-defined as there is no evidence that the various connected components belong in the same ancestral module.

Let (M_1, M_2) be a random variable distributed according to the empirical distribution function on the module pairs:

$$\mathbf{prob}[(M_1, M_2) = (M_1^i, M_2^i)] = \frac{1}{k}$$

for $i \in \{1, ..., k\}$. Let $\mathcal{O}$, $|M_a|$, $C(M_a)$, E_r, E_d, E_ℓ, and $||(E_r, E_d, E_\ell)||$ be random variables distributed according to the empirical distribution functions on $\mathcal{O}^i$, $|M_a^i|$, $C(M_a^i)$, E_r^i, E_d^i, E_ℓ^i, and $||(E_r^i, E_d^i, E_\ell^i)||_2$, respectively, where $M_a^i = M_a$ evaluated on the module pair (M_1^i, M_2^i), and $E_x^i = E_x(M_1^i, M_2^i)$ for $x \in \{r, d, \ell\}$. For each random variable, a summary quality measure is the pair $(\mathcal{E}(\cdot), \sqrt{\mathbf{var}(\cdot)})$, the expected value and standard deviation, evaluated on the random variable.

Two non-statistical measures of quality are the number of conserved modules k and the fraction of proteins in the interactomes that participate in conserved multi-protein modules. Let $\mathcal{U}_i$ be the set of unique proteins that are part of conserved modules in interactome i. Let $\mathcal{C}_i = |\mathcal{U}_i|/|V_i|$. Then $\mathcal{C} = (\mathcal{C}_1 + \mathcal{C}_2)/2$ is a measure of proteome coverage.

6 Previous algorithms

6.1 NetworkBlast

NetworkBlast [9] is based on a maximum-likelihood scoring model that gives high scores to module pairs inducing dense subgraphs with high sequence-similarity between proteins in the module pair. The model is additive in the densities of the two modules in the pair so the difference in the densities of the modules in a reported conserved pair is often large. A significant fraction of the reported conserved module pairs from NetworkBlast-M [23], including some of its highest scoring results, consist of module pairs such that one module induces a dense subgraph and the other module induces a subgraph with zero or a small number of interactions. This leads to large values of Δ_d, C_a, C_d, and E_r. Moreover, it is often the case that $C(M_a) > 1$. The search algorithm starts with high-scoring module pairs consisting of one or a few proteins in each interactome and grows them with a greedy algorithm based on the scoring function [9, 23].

6.2 MaWISh

MaWISh [10] is based on an evolutionary duplication-divergence scoring model that penalizes protein interaction divergence and infers protein duplications directly from sequence similarity, rewarding recent protein duplications and penalizing ancient protein duplications. The search algorithm is similar to that for NetworkBlast, starting with high-scoring seeds and growing them with a greedy algorithm based on the scoring function. Appendix G describes the scoring model in detail, explaining why MaWISh returns some module pairs with large E_d value and with $C(M_a) > 1$.

6.3 Match-and-Split

Match-and-Split [13] attempts a symmetric split of both interactomes, recursively matching all pairs of inter-interactome subnetworks that result from the split, where the two subnetworks in each recursive call are induced subgraphs of the two interactomes. Match-and-Split splits the networks by removing proteins in a network that do not have matching proteins in the other network, and then defining the subnetworks as the resulting connected components. Two proteins are considered matching if they are sequence similar and if both have neighboring proteins in the interactomes that are sequence similar to each other. Unfortunately, this rarely leads to a symmetric split. In most cases each network splits into a large component and many tiny components. The only meaningful recursive comparisons in this case are comparisons involving a large component. The large number of recursive calls in Match-and-Split affects the running time adversely. Several nice features include guarantees that $\mathcal{O} = 0$, $C_a = 0$, and $C(M_a) = 0$.

7 Results

7.1 Evaluation of Produles

All variants are applied to current iRefIndex [27] interactomes for H. sapiens and D. melanogaster, Release 6.0, filtered to retain binary interactions with UniProtKB [28] identifiers. The resulting networks consist of 74,554 interactions on 13,065 proteins for H. sapiens and 40,004 interactions on 10,050 proteins for D. melanogaster. Protein amino acid sequences were obtained from UniProtKB. The blastp program from stand-alone NCBI BLAST [29] was applied with threshold 10^{-9} on E-values yielding 138,824 pairs of homologous proteins. Using this threshold none of the data is expected to be spurious as the total number of comparisons is $2 * (13,065 * 10,050) < 4 * 10^8$. Conserved modules are taken as induced subgraphs. The subroutines based on Nibble and PageRank-Nibble, as described in Appendix E, are modified to consider only connected sweep sets with at most 20 proteins. All experiments were conducted using a MacBook Pro with 2.53 GHz Intel Core i5 processor and 4GB 1067 MHz DDR3 memory running Mac OS X Version 10.6.4. Detailed results tables are listed in Appendix A and discussed in Section 8.

7.2 Comparison of Algorithms

Unless otherwise indicated, data sets, programming environment, and parameters are identical to those described in Subsection 7.1. For Produles-P, $d = 0.05$.

Algorithms are compared after using a filter to remove very large, very small, and unbalanced module pairs. Only pairs of conserved modules with 5-20 proteins per species and with each module having no more than 1.5 times as many proteins as its conserved partner are considered. For each algorithm, the number of module pairs failing the filter and the reasons for the failure are listed in Appendix F.

For MaWISh, intra-species BLAST E-values were computed with threshold 10^{-9}, yielding 150,326 pairs of homologous proteins in H. sapiens and 51,956 pairs of homologous proteins in D. melanogaster. For NetworkBlast-M, all interactions are given equal confidence of 1.0. For MaWISh, stringent sequence similarity thresholds are used in place of COGs: inter-species pairs are considered orthologous if they have sequence similarity h values at most T_1 and intra-species pairs are considered in-paralogous if they have sequence similarity h values at most T_2 where $T_2 < T_1$. Homology values are converted to MaWISh similiarity scores using the algorithm in their paper [10]. All algorithms are tested with varying sequence similarity threshold T where for MaWISh, $T_1 = T$ and $T_2 = T * 10^{-20}$. The full data set is used consistently for evolutionary model evaluation.

Let $\mathcal{U}_i$ be the set of unique proteins in interactome i that an algorithm reports to be part of conserved modules. Let $\mathcal{Y}_i$ be the analogous set reported by another algorithm. Let $\ddot{\mathcal{C}}_i = |\mathcal{U}_i \cap \mathcal{Y}_i| / |\mathcal{Y}_i|$.

Let $\ddot{\mathcal{C}} = (\ddot{\mathcal{C}}_1 + \ddot{\mathcal{C}}_2)/2$. The tables in Appendix A report $\ddot{\mathcal{C}}$ where the rows correspond to the algorithms used as $\mathcal{U}$ and the columns correspond to the algorithms used as $\mathcal{Y}$. Let $\{(M_1^j, M_2^j) \mid j \in \{1, ..., k_1\}\}$ and $\{(N_1^i, N_2^i) \mid i \in \{1, ..., k_2\}\}$ be the output of two algorithms. Let

$$\ddot{\mathcal{O}}^i = \max_{1 \leq j \leq k_1} \min\{|M_1^j \cap N_1^i|/|N_1^i|, |M_2^j \cap N_2^i|/|N_2^i|\}$$

Let $\ddot{\mathcal{O}}$ be distributed according to the empirical distribution on $\ddot{\mathcal{O}}^i$. The tables in Appendix A report the pair $(\mathcal{E}(\ddot{\mathcal{O}}), \sqrt{\mathbf{var}(\ddot{\mathcal{O}})})$ where the rows correspond to the algorithms used as M and the columns correspond to the algorithms used as N.

Detailed results tables are listed in Appendix A and discussed in Section 8.

8 Discussion

8.1 Algorithmic Goals

Clear algorithmic goals form an interface between biological problems and algorithm design, allowing the following two questions to be examined separately: 1) if the algorithmic goals were perfectly attained, would the biological problem be solved? and 2) how can algorithms be best designed to attain the algorithmic goals? This work proposes algorithmic goals that may be useful for evaluating algorithms to detect conserved modularity across interactomes. An algorithm, Produles, has been designed that attains these algorithmic goals more closely than previous algorithms.

In the paper for NetworkBlast [9], from which NetworkBlast-M derives its scoring model, the algorithmic goal is well-specified: "The goal is to find two sets of proteins, one in yeast and one in bacteria, such that many of the proteins in each set have orthologous counterparts in the other, there is a high level of interaction among the proteins in each set, and the patterns of interaction in the two sets are similar [9]." The algorithm often does not achieve this algorithmic goal due to density imbalance among the modules reported as conserved.

The MaWISh paper [10] recognizes the importance of modularity, stating, "functional modules are likely to be densely connected while being separable from other modules [10]." Despite consequences of the MaWISh scoring model described in Appendix G that produce a few module pairs with $C(M_a) > 1$, MaWISh does well with respect to the algorithmic goals presented in this work.

In the paper for Match-and-Split [13], the algorithmic goal is well-specified: "A locally matching subset pair comprises protein pairs that have similar sequences and similar neighborhood or context in the networks [13]." The algorithm seems to achieve this algorithmic goal, though inefficiently.

8.2 Produles

It is remarkable that the E_r values for Produles are so low and that the discovered module pairs have similar topologies given that there is no attempt by Produles to match topologies. On arbitrary graphs, Produles would match modules of differing densities and topologies. Produles overwhelmingly finds module pairs with similar topologies in the interactomes seemingly due to conservation of these multi-protein modules during evolution.

Over 10% of proteins in the interactomes for human and fly are found by Produles-P with $d = 0.05$ to be part of conserved modules, a remarkable result, as the ancestors of chordates and arthropods diverged more than 500 million years ago [30], and as the interactome data remains incomplete [7].

Given incomplete and sometimes unreliable interactomics data [7], Produles attempts to find regions of the interactomes that are reliable by ignoring those regions that do not exhibit modularity in both interactomes. Less than 15% of the interactomes are found to be part of conserved multi-protein modules even with the most lenient parameter settings. While this may reflect the actual extent of conservation, it is possible that as interactomics data becomes more complete, a larger fraction of the interactomes will be found to be part of conserved multi-protein modules.

When an algorithm detects conserved multi-protein modules that share proteins, these modules can be combined into a larger composite module. Whenever this union takes place, Appendix D shows that the modularity of the composite module is at least as large as the minimum modularity of the modules being combined. This allows inspection of a hierarchy of modularity with larger conserved modules consisting of smaller conserved modules. To investigate these composite modules, all module pairs from Produles-P with $d = 0.05$ were combined into composite modules. When two module pairs had overlapping proteins in both interactomes, they were combined. A summary of the results is listed in Appendix A. The modularity for these composite modules is similar to that of the original modules. The size increased significantly. Using VieProt for visual inspection, many large reasonable pairs of composite modules are readily seen.

8.3 Comparison of Algorithms

Only Produles and Match-and-Split have $C(M_a) = 1$ in all cases.

Because of the large number of recursive calls, Match-and-Split cannot run on large data sets. Match-and-Split is able to run on the smaller data set with $T = 10^{-100}$ when there are 5,675 homologous pairs and less than 50% of the proteins can possibly be considered. MaWISh can process somewhat larger networks but also encounters difficulties when many homologous proteins are considered. MaWISh creates a graph with a vertex for each pair of homologous proteins and then creates edges among these vertices as described in Appendix G.

In MaWISh, duplications are penalized equally for each duplicate pair so the penalty grows quadratically with the number of duplicates, which, while desirable computationally, is not a reasonable model of evolution. The duplication model in this study has similarities to the linear duplication model described in the MaWISh paper [10]. The interaction model of MaWISh penalizes each duplicate equally for interaction loss or gain. During evolution, if a duplicate loses an interaction and then duplicates again, the subsequent duplicate never participated in the interaction and should not be penalized for interaction loss. Even when a copy of the interaction is truly lost, this is not uncommon as duplication weakens purifying selection on redundant copies [31]. Development of a truly new interaction, complete loss of an interaction, complete loss of a protein, and, to a lesser extent, duplication of a protein, are primary rare events when multi-protein modularity is conserved by purifying selection [31].

Produles uses the evolutionary model defined in this paper only for evaluation. The evolutionary model could be used directly to improve the evolutionary model score. If $||(E_r, E_d, E_\ell)||_2 > E^*$, for some threshold E^*, this variant of Produles would neither report the modules as conserved nor remove their proteins, other than the starting vertex, from future consideration, as done with modularity. This may not, however, be desirable as natural modules would be carved to remove those regions with lower similarity. As MaWISh explicitly attempts to optimize an evolutionary model score, it may carve regions with better scores from natural modules. With MaWISh and this variant of Produles, the reported conserved modules may be the most-highly conserved subsets of actual conserved modules.

In the applications of MaWISh in their paper [10], T_1 is set to a BLAST E-value smaller than 60% of the BLAST E-values between orthologous proteins in COG [32]. For fixed thresholds, using a database of orthologous proteins as a reference leads to the same number of nonzero $S(\cdot)$ values as our applications, so the running time and the sizes of data sets MaWISh can process remain similar. To verify that the omission of a database of orthologous proteins as a reference does not affect MaWISh results significantly, we used the set of all PHOG-T(D) orthologous proteins from PhyloFacts 3.0 [33] between the H. sapiens and D. melanogaster proteins as a reference. Any PHOG with more than 30 proteins was removed from consideration. MaWISh was run with the resulting set of PHOGs as a reference and $T_1 = 3 * 10^{-41}$, as 78% of h values between orthologous proteins in this set of PHOGs were above $3 * 10^{-41}$. Detailed results are listed in Appendices A and F.

Having discovered an algorithmic flaw in the NetworkBlast model, that density balance is not considered, the module pairs that minimize this flaw, those with lowest Δ_d value, were evaluated. Of all module pairs returned by NetworkBlast-M on the full data set, a summary of the 245 module pairs with lowest Δ_d value is listed in Appendix A. The evolutionary model results show improvement over the 245 module pairs with highest NetworkBlast-M score.

To further evaluate the various algorithms, the interactomes were randomized while keeping the protein sequence similarities fixed. The randomization, which consisted of a sequence of edge endpoint swaps, maintained the degree distribution. More precisely, if $(u_1, u_2), (u_3, u_4)$ are two randomly chosen edges in an interactome, these edges are replaced by the edges $(u_1, u_4), (u_3, u_2)$ unless the four endpoints are not distinct, in which case they are left alone. After all edges in the interactomes were randomized using this algorithm, the various algorithms for conserved module detection were applied to the randomized interactomes. Produles-P with values of d at least 0.05 did not report any conserved module pairs in the random graphs. All other algorithms reported conserved module pairs in the random graphs. MaWISh with $T_1 = 10^{-100}, T_2 = 10^{-120}$ reported 60 conserved module pairs, but did not report any module pair with more than four proteins in either module. Match-and-Split with $T = 10^{-100}$ reported four conserved module pairs, but did not report any module pair with more than five proteins in either module. NetworkBlast-M with $T = 10^{-9}$ reported 4,281 conserved module pairs in the same size range as it reported on the real interactomes. A table with quality measures for the conserved module pairs reported by NetworkBlast-M with $T = 10^{-9}$ on the real interactomes and the random graphs is listed in Appendix A. For NetworkBlast-M, the modularity and density are smaller on the random graphs. The evolutionary model scores on the real interactomes and the random graphs are similar. The density shows improved balance on the random graphs.

The overlap among the module pairs from the various algorithms tends to be significant as shown by tables in Appendix A, indicating that all algorithms are primarily searching for conserved modules among the same proteins. The difference seems to be largely in how well the boundaries of the conserved modules are discovered.

Though none of the algorithms in this study allow protein losses, and thus $E_\ell = 0$ in all cases, allowing protein losses may improve results, possibly by lowering C_a or increasing μ_a. Algorithms that allow indirect interactions, such as PathBlast [8] and the variant of MaWISh with $\overline{\Delta} > 1$ [10], do allow protein losses.

Only Produles and NetworkBlast-M apply to larger networks that include a large number of homologous protein pairs. The module pairs discovered by Produles at higher values of T are of similar algorithmic quality to those discovered at lower values of T. Some of these module pairs contain short homologous proteins with sequence similarity h values above 10^{-20}. These module pairs may be missed by algorithms that cannot process networks with higher values of T.

Graemlin [11] either returned no module pairs or ran for hours or days until being terminated with parameter settings suggested by the authors, so it was not included in the comparisons. In the future we plan to study additional algorithms such as PathBlast [8], Hirsh-Sharan [12], CAPPI [14], and DOMAIN [15].

9 Conclusion

Several algorithms that detect multi-protein modularity conserved during evolution have been examined. A new algorithm named Produles has been designed that does well on clear algorithmic goals. Promising future directions are discussed in Appendix I.

References

1. International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431:931–945, 2004.

2. Timothy W. Nilsen et al. Expansion of the eukaryotic proteome by alternative splicing. *Nature*, 463:457–463, 2010.

3. Marc Vidal. Interactome modeling. *FEBS Letters*, 579:1834–1838, 2005.

4. Bruce Alberts et al. *Molecular Biology of the Cell*. Garland Science, Fifth Reference edition, 2008.

5. José F. De Celis. The Notch signaling module. In *Modularity in Development and Evolution*. University of Chicago Press, 2004.

6. Gerhard Schlosser et al. Recognition and modeling of modules. In *Modularity in Development and Evolution*. University of Chicago Press, 2004.

7. Luke Hakes et al. Protein-protein interaction networks and biology–what's the connection? *Nature Biotechnology*, 26(1):69–72, 2008.

8. Brian P. Kelley et al. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences*, 100(20):11394–11399, 2003.

9. Roded Sharan et al. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *Journal of Computational Biology*, 12(6):835–846, 2005.

10. Mehmet Koyutürk et al. Pairwise alignment of protein interaction networks. *Journal of Computational Biology*, 13(2):182–199, 2006.

11. Jason Flannick et al. Graemlin: general and robust alignment of multiple large interaction networks. *Genome Research*, 16:1169–1181, 2006.

12. Eitan Hirsh et al. Identification of conserved protein complexes based on a model of protein network evolution. *Bioinformatics*, 23:e170–e176, 2006.

13. Manikandan Narayanan et al. Comparing protein interaction networks via a graph match-and-split algorithm. *Journal of Computational Biology*, 14(7):892–907, 2007.

14. Janusz Dutkowski et al. Identification of functional modules from conserved ancestral protein-protein interactions. *Bioinformatics*, 23:i149–i158, 2007.

15. Xin Guo et al. Domain-oriented edge-based alignment of protein interaction networks. *Bioinformatics*, 25:i240–i246, 2009.

16. Stephen F. Altschul et al. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

17. Ulrich Krohs. The cost of modularity. In *Functions in Biological and Artificial Worlds: Comparative Philosophical Perspectives*. MIT Press: Vienna Series in Theoretical Biology, 2009.

18. Herbert A. Simon. The structure of complexity in an evolving world: the role of near decomposability. In *Modularity: Understanding the Development and Evolution of Natural Complex Systems.* MIT Press: Vienna Series in Theoretical Biology, 2005.

19. Béla Bollobás. *Modern Graph Theory.* Springer: Graduate Texts in Mathematics, 1998.

20. Luqman Hodgkinson et al. EasyProt: parallel message-passing architecture for utilizing proteomics and interactomics data. *To be submitted to BMC Bioinformatics.*

21. Matt Welsh et al. SEDA: an architecture for well-conditioned, scalable internet services. *SOSP 2001*, pages 230–243, 2001.

22. Terence J. Parr et al. ANTLR: a predicated-LL (k) parser generator. *Software-Practice and Experience*, 25(7):789–810, 1995.

23. Maxim Kalaev et al. Fast and accurate alignment of multiple protein networks. *RECOMB 2008*, pages 246–256, 2008.

24. Luqman Hodgkinson et al. VieProt: visualizing conserved multi-protein modularity with a dynamic force-directed layout. *To be submitted to BMC Bioinformatics.*

25. Steffen Brasch et al. VANLO - interactive visual exploration of aligned biological networks. *BMC Bioinformatics*, 10(327), 2009.

26. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

27. Sabry Razick et al. iRefIndex: a consolidated protein interaction database with provenance. *BMC Bioinformatics*, 9(405), 2008.

28. The UniProt Consortium. The universal protein resource (UniProt) in 2010. *Nucleic Acids Research*, 38:D142–D148, 2010.

29. Eric W. Sayers et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acid Research*, 37:D5–D15, 2009.

30. Douglas J. Futuyma. *Evolution.* Sinauer, Second edition, 2009.

31. Thomas E. Creighton. *Proteins: Structures and Molecular Properties.* Freeman, Second edition, 1993.

32. Roman L. Tatusov et al. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research*, 28(1):33–36, 2000.

33. Ruchira Datta et al. Berkeley PHOG: PhyloFacts orthology group prediction web server. *Nucleic Acid Research*, 37:D5–D15, 2009.

34. Daniel A. Spielman et al. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR abs/0809.3232*, 2008.

35. Reid Andersen et al. Local graph partitioning using PageRank vectors. *FOCS 2006*, pages 475–486, 2006.

36. Konstantin Voevodski et al. Finding local communities in protein networks. *BMC Bioinformatics*, 10(297), 2009.

37. Lawrence Page et al. The PageRank citation ranking: bringing order to the Web. *Stanford Digital Library Technologies Project*, Technical Report, 1999.

38. Reid Andersen et al. Local graph partitioning using PageRank vectors (full version). Available from homepage of Kevin Lang. Accessed August 2010.

39. Marc Vidal. A unifying view of 21st century systems biology. *FEBS Letters*, 583(24):3891–3894, 2009.

40. Stanley Fields. Interactive learning: lessons from two hybrids over two decades. *Proteomics*, 9:5209–5213, 2009.

41. Sylvie Lalonde et al. Molecular and cellular approaches for the detection of protein-protein interactions: latest techniques and current limitations. *The Plant Journal*, 53(4):610–635, 2008.

42. Alexandra Madeira et al. Coupling surface plasmon resonance to mass spectrometry to discover novel protein-protein interactions. *Nature Protocols*, 4(7):1023–1037, 2009.

43. Alessio Masi et al. Optical methods in the study of protein-protein interactions. In *Integrins and Ion Channels: Molecular Complexes and Signaling.* Springer Science: Advances in Experimental Medicine and Biology, 2010.

44. Reid Andersen et al. Finding sparse cuts locally using evolving sets. *STOC 2009*, pages 235–244, 2009.

Appendix A: Results tables

Best entries are in bold.

Evaluation of Produles

Parameters are

- All variants: $a = 5, \quad b = 20, \quad c = 1.5, \quad e = 50$
- P variants: $\alpha = 10^{-3}, \quad \epsilon = 10^{-5}$
- N variants: $t_{last} = 10^3, \quad \epsilon = 10^{-5}$

Produles-N	$d = 0.05$	$d = 0.06$	$d = 0.07$	$d = 0.08$	$d = 0.09$	$d = 0.10$
Running time	4m	4m	4m	4m	4m	4m
k	169	151	138	118	101	89
μ_a	(0.13,0.04)	(0.14,0.04)	(0.14,0.04)	(0.15,0.03)	**(0.16,0.03)**	**(0.16,0.03)**
μ_d	(0.08,0.06)	(0.08,0.06)	(0.08,0.06)	(0.08,0.05)	**(0.08,0.05)**	**(0.07,0.05)**
S_a	(7.93,2.86)	(7.93,2.74)	(7.91,2.74)	(7.92,2.90)	(7.90,2.87)	(7.84,2.83)
S_d	(1.56,1.34)	(1.52,1.33)	(1.51,1.28)	(1.58,1.31)	(1.60,1.30)	(1.60,1.23)
Δ_a	(0.32,0.10)	(0.32,0.10)	(0.33,0.10)	(0.33,0.11)	(0.32,0.09)	(0.32,0.09)
Δ_d	(0.07,0.09)	(0.07,0.09)	(0.07,0.09)	(0.08,0.10)	(0.07,0.07)	(0.07,0.07)
C_a	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	0.10	0.09	0.08	0.07	0.06	0.05
$\mathcal{O}$	**(0.28,0.33)**	**(0.28,0.34)**	(0.31,0.35)	(0.29,0.36)	(0.29,0.37)	(0.32,0.37)
$\|M_a\|$	(5.75,2.43)	(5.87,2.50)	(5.82,2.46)	(5.80,2.61)	**(5.91,2.44)**	(5.81,2.35)
$C(M_a)$	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
E_r	(0.05,0.13)	(0.05,0.12)	(0.05,0.12)	**(0.04,0.12)**	**(0.03,0.10)**	**(0.04,0.11)**
E_d	(0.30,0.26)	(0.29,0.26)	(0.29,0.26)	(0.30,0.26)	**(0.28,0.24)**	**(0.28,0.25)**
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\|\|(E_r, E_d, E_\ell)\|\|$	(0.32,0.28)	(0.31,0.27)	(0.31,0.27)	(0.32,0.27)	**(0.29,0.25)**	**(0.29,0.26)**

Produles-P	$d = 0.05$	$d = 0.06$	$d = 0.07$	$d = 0.08$	$d = 0.09$	$d = 0.10$
Running time	**3m**	**2m**	**2m**	**2m**	**3m**	**2m**
k	**248**	**217**	179	160	129	107
μ_a	(0.13,0.04)	(0.13,0.04)	(0.14,0.04)	(0.14,0.04)	(0.15,0.04)	**(0.16,0.04)**
μ_d	(0.08,0.06)	(0.07,0.06)	(0.07,0.06)	(0.07,0.05)	(0.07,0.06)	**(0.07,0.06)**
S_a	(7.61,2.62)	(7.61,2.68)	(7.70,2.73)	(7.65,2.81)	(7.75,2.87)	(7.88,3.06)
S_d	(1.39,1.27)	(1.40,1.28)	(1.42,1.25)	(1.38,1.19)	(1.49,1.25)	(1.54,1.33)
Δ_a	(0.34,0.10)	(0.34,0.10)	(0.34,0.10)	(0.34,0.10)	(0.33,0.09)	(0.33,0.09)
Δ_d	(0.08,0.10)	(0.08,0.11)	(0.08,0.11)	(0.08,0.11)	(0.08,0.10)	(0.08,0.10)
C_a	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	**0.13**	0.11	0.10	0.09	0.07	0.06
$\mathcal{O}$	(0.34,0.32)	(0.34,0.32)	(0.32,0.33)	(0.33,0.35)	(0.32,0.36)	(0.33,0.37)
$\|M_a\|$	(5.43,2.19)	(5.48,2.21)	(5.49,2.23)	(5.48,2.34)	(5.67,2.24)	(5.72,2.28)
$C(M_a)$	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
E_r	(0.06,0.14)	(0.05,0.14)	(0.05,0.14)	(0.05,0.14)	**(0.04,0.12)**	(0.05,0.13)
E_d	(0.31,0.27)	(0.31,0.27)	(0.31,0.27)	(0.31,0.27)	(0.29,0.25)	**(0.28,0.25)**
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\|\|(E_r, E_d, E_\ell)\|\|$	(0.34,0.28)	(0.32,0.28)	(0.33,0.28)	(0.33,0.29)	**(0.30,0.26)**	**(0.30,0.27)**

Comparison of Algorithms

$\mathbf{T = 10^{-100}}$ (5,675 homologous protein pairs)

	Produles-P	NetworkBlast-M	MaWISh	Match-and-Split
Running time	2m	1m	**0m**	74m
k	16	134	22	13
μ_a	**(0.11,0.04)**	(0.05,0.02)	(0.05,0.03)	(0.07,0.04)
μ_d	(0.07,0.07)	(0.03,0.03)	(0.04,0.06)	(0.05,0.06)
S_a	(6.03,0.94)	(13.84,1.00)	(7.41,2.13)	(6.96,1.81)
S_d	(0.94,0.83)	(1.67,1.46)	(1.00,0.95)	(1.62,1.27)
Δ_a	(0.44,0.13)	(0.30,0.09)	(0.33,0.10)	(0.52,0.10)
Δ_d	(0.16,0.19)	(0.31,0.30)	(0.04,0.04)	(0.25,0.17)
C_a	(1.00,0.00)	(3.47,2.26)	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(4.50,4.57)	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	0.01	0.06	0.01	0.01
$\mathcal{O}$	(0.16,0.27)	(0.54,0.22)	(0.06,0.10)	**(0.00,0.00)**
$\lvert M_a \rvert$	(3.88,1.36)	**(9.96,2.94)**	(5.41,2.19)	(3.54,1.39)
$C(M_a)$	**(1.00,0.00)**	(1.05,0.22)	**(1.00,0.00)**	**(1.00,0.00)**
E_r	(0.03,0.09)	(0.38,0.29)	**(0.00,0.01)**	(0.06,0.10)
E_d	(0.44,0.23)	**(0.31,0.20)**	(0.33,0.25)	(0.56,0.26)
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\lvert\lvert(E_r, E_d, E_\ell)\rvert\rvert$	(0.45,0.24)	(0.55,0.25)	**(0.33,0.25)**	(0.57,0.26)

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M	MaWISh	Match-and-Split
Produles-P		0.09, (0.05,0.11)	0.11, (0.09,0.17)	0.20, (0.18,0.33)
NetworkBlast-M	0.77, (0.67,0.27)		0.70, (0.47,0.22)	0.89, (0.73,0.23)
MaWISh	0.20, (0.22,0.29)	0.15, (0.11,0.12)		0.28, (0.21,0.22)
Match-and-Split	0.21, (0.24,0.41)	0.11, (0.16,0.18)	0.16, (0.10,0.12)	

$T = 10^{-40}$ (25,346 homologous protein pairs)

	Produles-P	NetworkBlast-M	MaWISh	Match-and-Split
Running time	**2m**	**2m**	5m	NA
k	128	385	100	NA
μ_a	**(0.10,0.03)**	(0.07,0.03)	(0.05,0.03)	NA
μ_d	(0.06,0.05)	(0.05,0.04)	(0.04,0.05)	NA
S_a	(7.14,2.12)	(14.07,0.84)	(7.33,2.58)	NA
S_d	(1.34,1.15)	(1.48,1.42)	(1.14,1.07)	NA
Δ_a	(0.35,0.09)	(0.31,0.10)	(0.39,0.13)	NA
Δ_d	(0.07,0.10)	(0.26,0.27)	(0.08,0.09)	NA
C_a	(1.00,0.00)	(2.74,1.93)	(1.00,0.00)	NA
C_d	(0.00,0.00)	(2.97,3.86)	(0.00,0.00)	NA
$\mathcal{C}$	0.07	0.17	0.04	NA
$\mathcal{O}$	(0.30,0.33)	(0.55,0.23)	**(0.22,0.18)**	NA
$\|M_a\|$	(5.31,2.01)	**(10.10,3.02)**	(4.43,2.26)	NA
$C(M_a)$	**(1.00,0.00)**	(1.09,0.29)	**(1.00,0.00)**	NA
E_r	**(0.03,0.11)**	(0.32,0.27)	**(0.03,0.09)**	NA
E_d	**(0.29,0.23)**	(0.31,0.22)	(0.45,0.30)	NA
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	NA
$\|\|(E_r, E_d, E_\ell)\|\|$	**(0.30,0.24)**	(0.50,0.26)	(0.47,0.30)	NA

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M	MaWISh
Produles-P		0.27, (0.17,0.16)	0.29, (0.15,0.22)
NetworkBlast-M	0.72, (0.57,0.28)		0.77, (0.45,0.22)
MaWISh	0.20, (0.11,0.19)	0.20, (0.15,0.12)	

$\mathbf{T = 10^{-25}}$ (50,831 homologous protein pairs)

	Produles-P	NetworkBlast-M	MaWISh	Match-and-Split
Running time	**2m**	4m	>120m	NA
k	179	589	NA	NA
μ_a	**(0.12,0.04)**	(0.07,0.03)	NA	NA
μ_d	(0.06,0.06)	(0.05,0.04)	NA	NA
S_a	(7.39,2.51)	(14.11,0.79)	NA	NA
S_d	(1.34,1.26)	(1.43,1.41)	NA	NA
Δ_a	(0.34,0.10)	(0.31,0.10)	NA	NA
Δ_d	(0.07,0.09)	(0.23,0.25)	NA	NA
C_a	(1.00,0.00)	(2.52,1.74)	NA	NA
C_d	(0.00,0.00)	(2.63,3.38)	NA	NA
$\mathcal{C}$	0.10	0.23	NA	NA
$\mathcal{O}$	**(0.30,0.31)**	(0.55,0.22)	NA	NA
$\|M_a\|$	(5.62,2.01)	**(9.73,3.12)**	NA	NA
$C(M_a)$	**(1.00,0.00)**	(1.06,0.24)	NA	NA
E_r	**(0.04,0.11)**	(0.32,0.26)	NA	NA
E_d	**(0.26,0.22)**	(0.33,0.23)	NA	NA
E_ℓ	(0.00,0.00)	(0.00,0.00)	NA	NA
$\|\|(E_r, E_d, E_\ell)\|\|$	**(0.27,0.23)**	(0.52,0.25)	NA	NA

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M
Produles-P		0.28, (0.14,0.15)
NetworkBlast-M	0.68, (0.51,0.27)	

$\mathbf{T = 10^{-9}}$ (138,824 homologous protein pairs)

	Produles-P	NetworkBlast-M	MaWISh	Match-and-Split
Running time	**3m**	14m	NA	NA
k	248	974	NA	NA
μ_a	**(0.13,0.04)**	(0.07,0.03)	NA	NA
μ_d	(0.08,0.06)	(0.05,0.04)	NA	NA
S_a	(7.61,2.62)	(14.25,0.76)	NA	NA
S_d	(1.39,1.27)	(1.30,1.39)	NA	NA
Δ_a	(0.34,0.10)	(0.32,0.10)	NA	NA
Δ_d	(0.08,0.10)	(0.20,0.21)	NA	NA
C_a	(1.00,0.00)	(2.28,1.41)	NA	NA
C_d	(0.00,0.00)	(2.01,2.74)	NA	NA
$\mathcal{C}$	0.13	0.30	NA	NA
$\mathcal{O}$	**(0.34,0.32)**	(0.53,0.21)	NA	NA
$\|M_a\|$	(5.43,2.19)	**(9.10,3.13)**	NA	NA
$C(M_a)$	**(1.00,0.00)**	(1.07,0.26)	NA	NA
E_r	**(0.06,0.14)**	(0.33,0.23)	NA	NA
E_d	**(0.31,0.27)**	(0.39,0.23)	NA	NA
E_ℓ	(0.00,0.00)	(0.00,0.00)	NA	NA
$\|\|(E_r, E_d, E_\ell)\|\|$	**(0.34,0.28)**	(0.55,0.24)	NA	NA

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M
Produles-P		0.28, (0.12,0.14)
NetworkBlast-M	0.68, (0.45,0.26)	

Additional Comparisons

Produles	Produles-P	composite Produles
μ_a	(0.13,0.04)	(0.13,0.04)
μ_d	(0.08,0.06)	(0.07,0.06)
S_a	(7.61,2.62)	(11.17,7.37)
S_d	(1.39,1.27)	(1.76,1.60)
Δ_a	(0.34,0.10)	(0.29,0.14)
Δ_d	(0.08,0.10)	(0.08,0.11)
C_a	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	0.13	0.13
$\mathcal{O}$	(0.34,0.32)	**(0.00,0.00)**
$\lvert M_a \rvert$	(5.43,2.19)	**(7.65,6.28)**
$C(M_a)$	(1.00,0.00)	(1.00,0.00)
E_r	**(0.06,0.14)**	(0.07,0.16)
E_d	**(0.31,0.27)**	(0.39,0.29)
E_ℓ	(0.00,0.00)	(0.00,0.00)
$\lvert\lvert(E_r, E_d, E_\ell)\rvert\rvert$	**(0.34,0.28)**	(0.42,0.31)

MaWISh	$T_1 = 10^{-40}, T_2 = 10^{-60}$	PHOG-T(D), $T_1 = 3 * 10^{-41}$
Running time	**5m**	7m
k	100	93
μ_a	(0.05,0.03)	(0.05,0.02)
μ_d	(0.04,0.05)	(0.04,0.05)
S_a	(7.33,2.58)	(7.75,2.83)
S_d	(1.14,1.07)	(1.03,1.03)
Δ_a	**(0.39,0.13)**	(0.35,0.11)
Δ_d	(0.08,0.09)	(0.07,0.07)
C_a	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	0.04	**0.05**
$\mathcal{O}$	(0.22,0.18)	**(0.20,0.20)**
$\lvert M_a \rvert$	(4.43,2.26)	**(5.08,2.81)**
$C(M_a)$	(1.00,0.00)	(1.00,0.00)
E_r	**(0.03,0.09)**	(0.04,0.09)
E_d	(0.45,0.30)	**(0.41,0.27)**
E_ℓ	(0.00,0.00)	(0.00,0.00)
$\lvert\lvert(E_r, E_d, E_\ell)\rvert\rvert$	(0.47,0.30)	**(0.42,0.27)**

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	MaWISh	MaWISh-PHOG-T(D)
MaWISh		0.69, (0.70,0.37)
MaWISh-PHOG-T(D)	0.75, (0.64,0.37)	

NetworkBlast-M	full set	highest score	lowest Δ_d
k	1021	248	248
μ_a	(0.07,0.03)	(0.09,0.03)	(0.07,0.03)
μ_d	(0.05,0.04)	(0.06,0.04)	(0.03,0.03)
S_a	(14.12,0.98)	(14.65,0.67)	(14.24,0.87)
S_d	(1.56,1.82)	(0.70,1.34)	(1.16,1.45)
Δ_a	(0.32,0.10)	**(0.45,0.06)**	(0.30,0.11)
Δ_d	(0.21,0.21)	(0.28,0.29)	**(0.02,0.01)**
C_a	(2.28,1.39)	(2.36,1.61)	(1.52,0.62)
C_d	(2.03,2.70)	(2.27,3.21)	(0.53,0.67)
$\mathcal{C}$	0.32	0.09	0.14
$\mathcal{O}$	(0.52,0.21)	(0.65,0.20)	**(0.44,0.27)**
$\|M_a\|$	(8.95,3.15)	(8.80,3.35)	**(9.88,2.94)**
$C(M_a)$	**(1.07,0.25)**	(1.09,0.28)	(1.10,0.31)
E_r	(0.33,0.23)	(0.36,0.29)	**(0.15,0.16)**
E_d	(0.39,0.23)	(0.43,0.25)	**(0.33,0.22)**
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\|\|(E_r, E_d, E_\ell)\|\|$	(0.56,0.24)	(0.64,0.23)	**(0.38,0.25)**

NetworkBlast-M	interactomes	random graphs
k	1021	4281
μ_a	**(0.07,0.03)**	(0.04,0.01)
μ_d	(0.05,0.04)	**(0.02,0.01)**
S_a	(14.12,0.98)	(13.95,0.76)
S_d	(1.56,1.82)	(1.27,0.99)
Δ_a	**(0.32,0.10)**	(0.19,0.02)
Δ_d	(0.21,0.21)	**(0.06,0.05)**
C_a	(2.28,1.39)	(1.50,0.50)
C_d	(2.03,2.70)	(0.84,0.90)
$\mathcal{C}$	0.32	0.44
$\mathcal{O}$	(0.52,0.21)	**(0.31,0.10)**
$\|M_a\|$	(8.95,3.15)	**(10.99,1.68)**
$C(M_a)$	(1.07,0.25)	**(1.00,0.02)**
E_r	**(0.33,0.23)**	(0.37,0.07)
E_d	(0.39,0.23)	**(0.23,0.11)**
E_ℓ	(0.00,0.00)	(0.00,0.00)
$\|\|(E_r, E_d, E_\ell)\|\|$	(0.56,0.24)	**(0.44,0.09)**

Appendix B: Produles details and proof of running time

As described in Appendix E, local algorithms that run in constant time not depending on the size of the input networks can be used to find modules with high modularity, with size at most b, and with the degree in the interactome of each module vertex bounded above by $b^2(1+d)/d$ for constants b, d as defined in Section 3. A counter is maintained for each protein in G_1. When a protein is placed in a module by the local module-finding algorithm, the counter for the protein is incremented. Each counter has maximum value e for some constant e. If the local module-finding algorithm returns a module containing any protein with counter value e, the entire module is ignored. If a protein in G_1 is reported to be in a conserved module, the counter for the protein is set to e in order to reduce module overlap. Each value of $h(v,\cdot)$ for $v \in V$ is considered only when constructing $\mathcal{H}_T(M)$ for $\{M : v \in M\}$, so each value of $h(v,\cdot)$ is considered at most e times. If v is stored at each vertex in $\mathcal{H}_T(M)$ when constructing $\mathcal{H}_T(M)$, then constructing $\mathcal{R}_T(M,N_i)$ is simply a union of vertex lists and does not require additional consideration of $h(v,\cdot)$ values. As for all $v \in V_1$,

$$|\{M : v \in M\}| \leq e$$

the number of consideration of h values is

$$\begin{aligned}\sum_M \sum_{v\in M} |h(v,\cdot)| &= \sum_v \sum_{M:v\in M} |h(v,\cdot)| \\ &\leq e\sum_v |h(v,\cdot)| \\ &= e|h(\cdot,\cdot)|\end{aligned}$$

After finding $\mathcal{H}_T(M)$, it is necessary to compute $N_1, N_2, ..., N_k$. This can be problematic if any of the vertices in $\mathcal{H}_T(M)$ have large degree, which could conceivably be as large as $|V_2| - 1$. However, as we are only interested in N_i such that $\mu(N_i) \geq d$ and $|N_i| \leq b$, we can discard any vertex $v \in \mathcal{H}_T(M)$ with degree greater than $b^2(1+d)/d$ in G_2 as shown in Appendix C. We can then run a modified depth-first search that only transitions among vertices in $\mathcal{H}_T(M)$ to compute $N_1, N_2, ..., N_k$. This requires time

$$\mathcal{O}((\frac{b^2(1+d)}{d})|\mathcal{H}_T(M)|) = \mathcal{O}(|\mathcal{H}_T(M)|)$$

As

$$|\mathcal{H}_T(M)| \leq \sum_{v\in M} |h(v,\cdot)|$$

all of these depth-first searches over the full run of the algorithm require time

$$\mathcal{O}(\sum_M |\mathcal{H}_T(M)|) = \mathcal{O}(\sum_M \sum_{v\in M} |h(v,\cdot)|) = \mathcal{O}(|h(\cdot,\cdot)|)$$

For a given M, constructing all $\mathcal{R}_T(M,N_i)$ by concatenating lists of vertices stored at the vertices in the N_i requires time $\mathcal{O}(|\mathcal{H}_T(M)|)$. Testing for connectivity of a single $\mathcal{R}_T(M,N_i)$ with a modified depth-first search that only transitions among vertices in $\mathcal{R}_T(M,N_i)$ requires constant time as $|\mathcal{R}_T(M,N_i)| \leq b$ and as each vertex in M has bounded degree. At most $|\mathcal{H}_T(M)|/a$ of the sets $\mathcal{R}_T(M,N_i)$ are constructed when processing M. Thus, all of these constructions and depth-first searches over the full run of the algorithm can be completed in time

$$\mathcal{O}(\sum_M |\mathcal{H}_T(M)|) = \mathcal{O}(|h(\cdot,\cdot)|)$$

Computing the modularity of module $U \in \{N_i, \mathcal{R}_T(M,N_i)\}$ requires simply computing the sum of degrees of the vertices in U and the number of edges with both endpoints in U. The sum of degrees of the vertices in U can be computed in constant time as $|U| \leq b$. The number of edges with both endpoints in U can also be computed in constant time as each vertex in U has degree bounded by a constant, so the endpoints of all edges incident on each vertex can be tested for set inclusion in U in constant time.

Appendix C: Proof of module degree bound

Theorem: Any module M in a graph $G = (V, E)$, such that $|M| \leq b$ and $\mu(M) \geq d$ consists of vertices with degree in G no greater than $b^2(1+d)/d$.

Proof: By observing that

$$\Phi(M) = \frac{\sum_{v \in M} d(v) - 2|E(M)|}{\sum_{v \in M} d(v)}$$

where $d(v)$ is the degree of v in G, we see that if any $v \in M$ has $d(v) > b^2(1+d)/d$ where $|M| \leq b$, it would be the case that

$$\begin{aligned}
\Phi(M) &= \frac{\sum_{v \in M} d(v) - 2|E(M)|}{\sum_{v \in M} d(v)} \\
&> \frac{b^2(1+d)/d - 2|E(M)|}{b^2(1+d)/d} \\
&\geq \frac{b^2(1+d)/d - 2b^2}{b^2(1+d)/d} \\
&= \frac{1-d}{1+d}
\end{aligned}$$

which would imply

$$\mu(M) < d$$

Appendix D: Proof of modularity for composite modules

Theorem: For modules M_1, M_2, it is true that $\mu(M_1 \cup M_2) \geq \min\{\mu(M_1), \mu(M_2)\}$.

Proof:

$$\mu(M_1 \cup M_2) = \frac{|E(M_1 \cup M_2)|}{|\texttt{cut}(M_1 \cup M_2, V \backslash (M_1 \cup M_2))| + |E(M_1 \cup M_2)|}$$

$$= \frac{|E(M_1)| + |E(M_2)| + |\texttt{cut}(M_1, M_2)|}{|\texttt{cut}(M_1, V \backslash M_1)| + |\texttt{cut}(M_2, V \backslash M_2)| - 2|\texttt{cut}(M_1, M_2)| + |E(M_1)| + |E(M_2)| + |\texttt{cut}(M_1, M_2)|}$$

$$= \frac{|E(M_1)| + |E(M_2)| + |\texttt{cut}(M_1, M_2)|}{|\texttt{cut}(M_1, V \backslash M_1)| + |\texttt{cut}(M_2, V \backslash M_2)| + |E(M_1)| + |E(M_2)| - |\texttt{cut}(M_1, M_2)|}$$

$$\geq \frac{|E(M_1)| + |E(M_2)|}{|\texttt{cut}(M_1, V \backslash M_1)| + |E(M_1)| + |\texttt{cut}(M_2, V \backslash M_2)| + |E(M_2)|}$$

$$\geq \min\{\frac{|E(M_1)|}{|\texttt{cut}(M_1, V \backslash M_1)| + |E(M_1)|}, \frac{|E(M_2)|}{|\texttt{cut}(M_2, V \backslash M_2)| + |E(M_2)|}\}$$

$$= \min\{\mu(M_1), \mu(M_2)\}$$

The final inequality follows from the lemma

$$\frac{a+b}{c+d} \geq \min\{\frac{a}{c}, \frac{b}{d}\} \text{ for } a, b \geq 0 \text{ and } c, d > 0$$

which can be proved by observing that

$$\frac{a+b}{c+d} < \frac{a}{c} \Rightarrow c(a+b) < a(c+d) \Rightarrow bc < ad$$

whereas

$$\frac{a+b}{c+d} < \frac{b}{d} \Rightarrow d(a+b) < b(c+d) \Rightarrow ad < bc$$

which cannot both be true.

Appendix E: Conductance minimization algorithms

Nibble [34] and PageRank-Nibble [35] are two algorithms for finding sets of vertices with low conductance in a graph. Reasonable adaptations of the algorithms allow them to run in constant time and to search only for small modules. The project of adapting Nibble and PageRank-Nibble to search only for small modules was initiated in [36]. The adaptations described in [36] do not guarantee constant running time for Nibble.

Nibble

Nibble [34] is an algorithm for finding a set of vertices with low conductance in a graph G with n vertices. Let A be the adjacency matrix for G. Let D be a diagonal matrix with diagonal entries $D_{ii} = d(i)$ where $d(i)$ is the degree of vertex i in G. Let $W = (AD^{-1} + I)/2$ where I is the identity matrix. W is a lazy random walk transition matrix for G that with probability $1/2$ remains at the current vertex and with probability $1/2$ randomly walks to an adjacent vertex. Let q, r be vectors representing distributions on the vertices of G, not necessarily normalized. Define the truncation operator

$$[q]_\epsilon(u) = \begin{cases} q(u) \text{ if } \ q(u) \geq d(u)\epsilon \\ 0 \text{ otherwise} \end{cases}$$

Define the distribution that places all mass at vertex v

$$\chi_v(u) = \begin{cases} 1 \text{ if } \ u = v \\ 0 \text{ otherwise} \end{cases}$$

Each iteration of Nibble at time step t generates the vectors

$$q_t = \begin{cases} \chi_v \text{ if } \ t = 0 \\ Wr_{t-1} \text{ otherwise} \end{cases}$$

$$r_t = [q_t]_\epsilon$$

Nibble is run for t_{last} iterations. After each iteration, the vertices are sorted by $q_t(\cdot)/d(\cdot)$. Let $S_j(q_t)$ be a set of j vertices with highest values of $q_t(\cdot)/d(\cdot)$ where ties are broken arbitrarily while maintaining $S_j(q_t) \subset S_{j+1}(q_t)$. These sets $S_j(q_t)$ are called sweep sets and there are always n of them. After each iteration the conductance is computed for at most the first b sweep sets for some constant b, never including any vertex v with $q_t(v) = 0$. No further sweep sets are considered if a vertex with degree greater than $b^2(d+1)/d$ is reached, where d is a constant parameter. This ensures that all vertices in modules returned by the algorithm have degrees bounded by $b^2(d+1)/d$, which is useful for reasons described in Appendix B. The sweep set with minimum conductance over all iterations so far is stored. In original Nibble, b is not a constant but rather a function of the sum of degrees of vertices in the sweep sets, and there is no guarantee that vertices in returned modules have bounded degree.

It remains to show that the algorithm can be implemented to run in constant time for constant t_{last}, ϵ. Since t_{last} is constant, it suffices to show that a single iteration requires constant time. Let $\sigma(\cdot)$ be the support function that returns the set of vertices with positive values in its distribution argument. Define the volume of a set of vertices as the sum of degrees:

$$\text{vol}(S) = \sum_{v \in S} d(v)$$

Rather than computing $q_t = Wr_{t-1}$ using matrix multiplication, q_t can be computed by explicitly passing messages to neighbors in the graph. Each vertex $v \in \sigma(r_{t-1})$ keeps half of $r_{t-1}(v)$ and partitions half of $r_{t-1}(v)$ equally among its neighbors. By keeping a linked list of references to vertices with nonzero distribution values, this requires $\mathrm{vol}(\sigma(r_{t-1}))$ messages, leading to $|\sigma(q_t)| \leq \mathrm{vol}(\sigma(r_{t-1}))$. The truncated distribution $[q_t]_\epsilon$ can be computed simply by removing references from the linked list for any vertex v such that $q_t(v) < d(v)\epsilon$. Only vertices with nonzero values of $q_t(\cdot)/d(\cdot)$ need be sorted. If the degree of each vertex is stored at the vertex, making degree lookup a constant-time operation, these vertices can be sorted in $\mathcal{O}(\mathrm{vol}(\sigma(r_{t-1})) \log \mathrm{vol}(\sigma(r_{t-1})))$ time. Conductances for the first b sweep sets can be computed in $\mathcal{O}(b^5(d+1)\log b/d)$ time by observing that the conductance of $S_j(q_t)$ can be computed by knowing the sum of degrees of vertices in $S_j(q_t)$ and the number of edges with both endpoints in $S_j(q_t)$. The former can be computed in $\mathcal{O}(|S_j(q_t)|) = \mathcal{O}(b)$ time, and the latter can be computed with at most $b * b^2(d+1)/d$ set inclusion tests in a balanced binary search tree of size at most b, which follows from the bound on the degree of each vertex in $S_j(q_t)$.

It remains to show that $\mathrm{vol}(\sigma(r_{t-1}))$ never exceeds a constant value. For any $v \in \sigma(r_{t-1})$, by the truncation operation, $r_{t-1}(v) \geq d(v)\epsilon$. Because the distribution starts with $r_0 = [\chi_v]_\epsilon$ that has total value at most 1 and never increases in total value,

$$1 \geq \sum_{v \in \sigma(r_{t-1})} r_{t-1}(v) \geq \epsilon \sum_{v \in \sigma(r_{t-1})} d(v)$$

which implies

$$\mathrm{vol}(\sigma(r_{t-1})) \leq \frac{1}{\epsilon}$$

PageRank-Nibble

PageRank-Nibble [35] is an algorithm based on PageRank [37] and Nibble [34] for finding a module with low conductance in a graph $G = (V, E)$. A PageRank vector is a row vector solution $\mathrm{pr}(\alpha, s)$ to the equation

$$\mathrm{pr}(\alpha, s) = \alpha s + (1-\alpha)\mathrm{pr}(\alpha, s)W^T$$

where $\alpha \in (0, 1]$ is a teleportation constant, s is a row vector distribution on the vertices of the graph called the preference vector, and $W^T = (D^{-1}A + I)/2$ is a lazy random walk transition matrix in a form suitable for row vector distributions. Intuitively, when $s = \chi_v$, a PageRank vector can be viewed as a weighted sum of the probability distributions obtained by taking a sequence of lazy random walk steps starting from v, where the weight placed on the distribution obtained after t walk steps decreases exponentially in t [35].

There is a unique PageRank vector since

$$\begin{aligned} p &= \alpha s + (1-\alpha)pW^T \\ p[I - (1-\alpha)W^T] &= \alpha s \\ p &= \alpha s[I - (1-\alpha)W^T]^{-1} \end{aligned}$$

which follows as the matrix in brackets is strictly diagonally dominant and, thus, nonsingular.

The PageRank-Nibble algorithm consists of computing an approximate PageRank vector with $s = \chi_v$, defined as $\mathrm{apr}(\alpha, \chi_v, r) = \mathrm{pr}(\alpha, \chi_v) - \mathrm{pr}(\alpha, r)$, where r is called a residual vector, and returning the sweep set $S_j(\mathrm{apr}(\alpha, \chi_v, r))$ with minimum conductance among the first b sweep sets.

From the definition, if p is a vector that satisfies $p + \mathrm{pr}(\alpha, r) = \mathrm{pr}(\alpha, \chi_v)$, then $p = \mathrm{apr}(\alpha, \chi_v, r)$. Thus, $0 = \mathrm{apr}(\alpha, \chi_v, \chi_v)$. We can initialize $p_1 = 0, r_1 = \chi_v$ and improve the solution iteratively. Each iteration, called a push operation, chooses an arbitrary vertex u such that $r_i(u)/d(u) \geq \epsilon$. Then $p_{i+1} = p_i$ and $r_{i+1} = r_i$ except for the following changes:

1. $p_{i+1}(u) = p_i(u) + \alpha r_i(u)$
2. $r_{i+1}(u) = (1-\alpha) r_i(u)/2$
3. For each v such that $(u,v) \in E$, $r_{i+1}(v) = r_i(v) + (1-\alpha) r_i(u)/(2d(u))$

Intuitively, $\alpha r_i(u)$ probability is sent to $p_{i+1}(u)$, and the remaining $(1-\alpha)r_i(u)$ probability is redistributed in r_{i+1} using a single lazy random walk step.

It is shown that each push operation maintains the invariant

$$p_i + \mathrm{pr}(\alpha, r_i) = \mathrm{pr}(\alpha, \chi_v)$$

The proof can be found in the text and appendix of [38]. When no additional pushes can be performed, the final residual vector r satisfies

$$\max_{u \in V} \frac{r(u)}{d(u)} < \epsilon$$

The running time for computing $\mathrm{apr}(\alpha, \chi_v, r)$ is $\mathcal{O}(1/(\epsilon\alpha))$. This follows directly from the claim that if T is the total number of push operations and d_i is the degree of the vertex pushed at the ith iteration, then

$$\sum_{i=1}^{T} d_i \leq \frac{1}{\epsilon\alpha}$$

To prove this claim, observe that the vertex u pushed at iteration i satisfies

$$r_i(u) \geq \epsilon d_i$$

As $\alpha r_i(u)$ probability is sent to $p_{i+1}(u)$,

$$\begin{aligned} ||r_{i+1}||_1 &= ||r_i||_1 - \alpha r_i(u) \\ &\leq ||r_i||_1 - \alpha\epsilon d_i \end{aligned}$$

Because the initial residual vector is $r_1 = \chi_v$ with $||\chi_v||_1 = 1$, the ℓ_1 norm of the residual vector cannot decrease by more than 1 over all iterations, so

$$\alpha\epsilon \sum_{i=1}^{T} d_i \leq 1$$

from which the claim follows.

For PageRank-Nibble, it is not necessary to bound the degree explicitly for each vertex in the considered sweep sets in order to guarantee that these degrees are bounded above by a constant. PageRank-Nibble guarantees that

$$\mathrm{vol}(\sigma(\mathrm{apr}(\alpha, \chi_v, r))) \leq \frac{2}{(1-\alpha)\epsilon}$$

This follows from the observation that the final push on a vertex $v \in \sigma(\mathrm{apr}(\alpha, \chi_v, r))$ occurred at an iteration i when $r_i(v) \geq \epsilon d(v)$ and a fraction $(1-\alpha)/2$ of that probability remained at $r_{i+1}(v)$. Thus, for each $v \in \sigma(\mathrm{apr}(\alpha, \chi_v, r))$,

$$r(v) \geq \frac{1-\alpha}{2} \cdot \epsilon d(v)$$

Thus

$$1 \geq \sum_{v \in \sigma(\mathrm{apr}(\alpha, \chi_v, r))} r(v) \geq \frac{(1-\alpha)\epsilon}{2} \cdot \mathrm{vol}(\sigma(\mathrm{apr}(\alpha, \chi_v, r)))$$

from which the claim follows.

Appendix F: Unfiltered algorithm results

Let P be the set of module pairs returned by an algorithm. Let $P_f \subseteq P$ be the set of module pairs that fail the filter. Let $P_k \subseteq P_f \backslash (P_1 \cup \cdots \cup P_{k-1})$ be the set of module pairs with at least one module consisting of only k proteins, for $k \in \{1,2,3,4\}$. Let $P_{21+} \subseteq P_f \backslash (P_1 \cup P_2 \cup P_3 \cup P_4)$ be the set of module pairs with at least one module consisting of more than 20 proteins. Let $P_b = P_f \backslash (P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_{21+})$ be the set of module pairs that fail the balance requirement. Let $k = |P|$. Let $k_s = |P_s|$ where s is any subscript.

NetworkBlast-M / T	10^{-100}	10^{-40}	10^{-25}	10^{-9}
k	149	400	614	1021
k_f	15	15	25	47
k_1	0	0	0	0
k_2	1	0	0	0
k_3	0	0	0	0
k_4	0	1	0	1
k_{21+}	0	0	0	0
k_b	14	14	25	46

Filtering has little effect on the results from NetworkBlast-M.

MaWISh / T	10^{-100}	10^{-40}	PHOG-T(D)	10^{-25}	10^{-9}
k	395	982	976	NA	NA
k_f	373	882	883	NA	NA
k_1	0	0	0	NA	NA
k_2	263	437	476	NA	NA
k_3	72	286	256	NA	NA
k_4	30	104	102	NA	NA
k_{21+}	5	24	20	NA	NA
k_b	3	31	29	NA	NA

Most MaWISh results consist of modules that are conserved single interactions on two proteins which have little significance, and conserved modules on three proteins. Some modules on four proteins may be meaningful and could have been allowed to pass the filter, but they are less significant than larger conserved modules and do not affect the conserved modules found in the range of 5-20 proteins which are the focus of this study. MaWISh returns some modules containing hundreds of proteins with more than 5% of all proteins each, which have $C(M_a) > 1$ and seem to have little significance. Most modules that did not pass the filter were removed by the size filter and relatively few by the balance filter.

Match-and-Split / T	10^{-100}	10^{-40}	10^{-25}	10^{-9}
k	63	NA	NA	NA
k_f	50	NA	NA	NA
k_1	0	NA	NA	NA
k_2	0	NA	NA	NA
k_3	28	NA	NA	NA
k_4	21	NA	NA	NA
k_{21+}	0	NA	NA	NA
k_b	1	NA	NA	NA

Most of the conserved module pairs reported by Match-and-Split involve modules with three or four proteins.

Appendix G: MaWISh scoring model

The evolutionary model of MaWISh decomposes into an interaction model and a duplication model. There is a score on each pair of edges from the complete bipartite graph on the proteins in the two modules claimed to be conserved. The score of a module pair is the sum of these scores. High scores are considered good. $S : V_1 \times V_2 \rightarrow [0, 1]$ is a monotonically decreasing function of BLAST E-values that gives a positive value to sequence-similar proteins and a value of 0 to sequence-dissimilar proteins. Let (M_1, M_2) be a pair of conserved modules. Let $u_1, u_2 \in M_1$ and $v_1, v_2 \in M_2$. Two scores are associated with this collection of four proteins: $\mathbf{score}((u_1, v_1), (u_2, v_2))$ and $\mathbf{score}((u_1, v_2), (u_2, v_1))$. Each score is defined symmetrically as

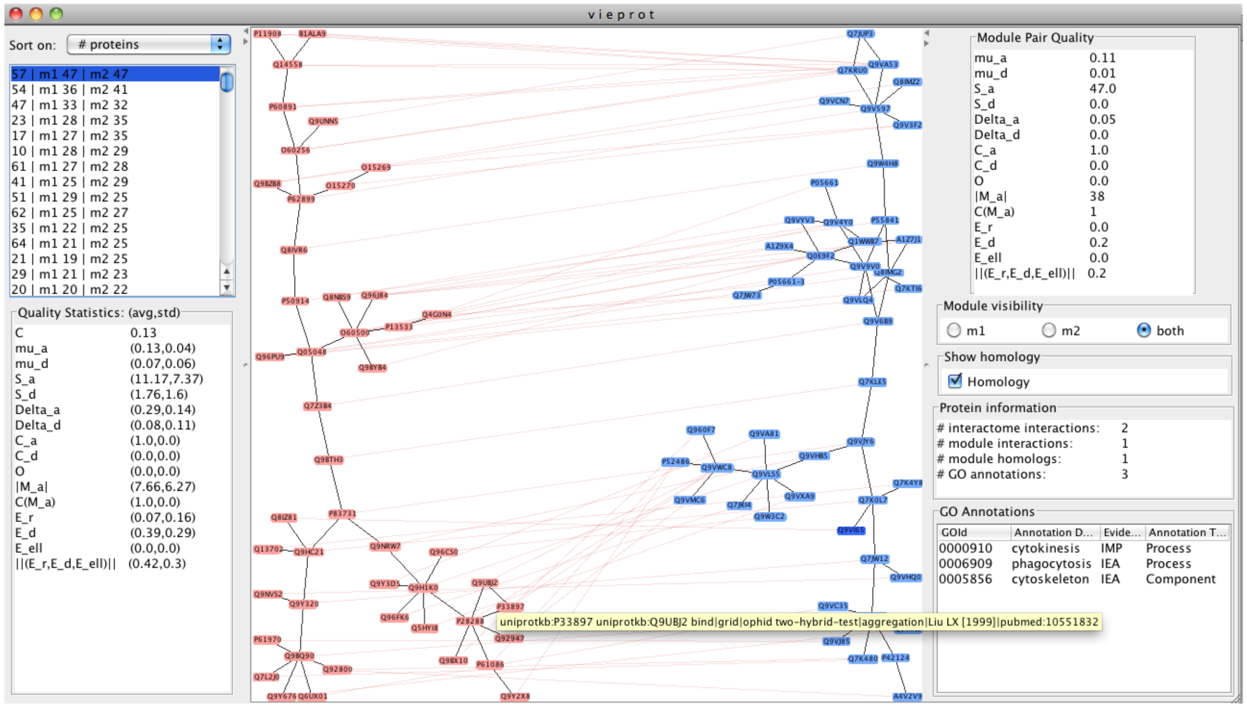
$$\begin{aligned}\mathbf{score}((u_1, v_1), (u_2, v_2)) \quad = \quad & \mathbf{I}(\mathbf{match})S(u_1, v_1)S(u_2, v_2) \\ & -\mathbf{I}(\mathbf{mismatch})S(u_1, v_1)S(u_2, v_2) \\ & +\mathbf{I}(S(u_1, u_2) > 0)(0.1)(S(u_1, u_2) - 0.9) \\ & +\mathbf{I}(S(v_1, v_2) > 0)(0.1)(S(v_1, v_2) - 0.9)\end{aligned}$$

where $\mathbf{I}(\cdot)$ is the indicator function such that $\mathbf{I}(\mathbf{false}) = 0$ and $\mathbf{I}(\mathbf{true}) = 1$. The vast majority of these scores are 0 and are not explicitly represented, as the vast majority of $S(\cdot, \cdot)$ values are 0. The first two terms are the interaction model and the second two terms are the duplication model. **match** and **mismatch** are predicates defined by interactions among the two protein pairs (u_1, u_2) and (v_1, v_2). If both interactions exist, **match** = **true**, **mismatch** = **false** which may lead to a reward. If one interaction exists but not the other, **mismatch** = **true**, **match** = **false** which may lead to a penalty. If neither interaction exists, **match** = **false**, **mismatch** = **false** and the interaction model score is 0.

Considering only $u_1, u_2 \in M_1$, but symmetrically for $v_1, v_2 \in M_2$, the duplication model gives a score of 0 if $S(u_1, u_2) = 0$ or $S(u_1, u_2) = 0.9$. It gives a penalty of $(0.1)(S(u_1, u_2) - 0.9)$ if $0 < S(u_1, u_2) < 0.9$. It gives a reward of $(0.1)(S(u_1, u_2) - 0.9)$ if $S(u_1, u_2) > 0.9$. MaWISh rewards placing highly similar protein pairs in the same module even if they have no interactions between them. As the similarity decreases, the reward gets smaller until it becomes a penalty. As the similarity decreases further, the penalty becomes harsher and harsher until eventually, at a sharp discontinuous cutoff, the penalty vanishes and the duplication model score becomes 0. The reward for placing highly similar proteins in the same module even when they do not involve any interactions leads to a large E_d value.

The MaWISh paper generalizes the scoring model so that the default scoring model given above would be parameterized by $\overline{d} = 0.9, \overline{\delta} = 0.1$. The default value of $\overline{\delta} = 0.1$ places low emphasis on the duplication model relative to the interaction model. However, as MaWISh rewards conserved pairs of proteins with no interactions in either interactome when they are very similar according to $S(\cdot,\cdot)$, even with $\overline{\delta} = 0.1$, a reported conserved module sometimes induces many disconnected subgraphs leading to $C(M_a) > 1$.

Appendix H: VieProt



Sort on: # proteins

57 | m1 47 | m2 47
54 | m1 36 | m2 41
47 | m1 33 | m2 32
23 | m1 28 | m2 35
17 | m1 27 | m2 35
10 | m1 28 | m2 29
61 | m1 27 | m2 28
41 | m1 25 | m2 29
51 | m1 29 | m2 25
62 | m1 25 | m2 27
35 | m1 22 | m2 25
64 | m1 21 | m2 25
21 | m1 19 | m2 25
29 | m1 21 | m2 23
20 | m1 20 | m2 22

Quality Statistics: (avg,std)

C	0.13
mu_a	(0.13,0.04)
mu_d	(0.07,0.06)
S_a	(11.17,7.37)
S_d	(1.76,1.6)
Delta_a	(0.29,0.14)
Delta_d	(0.08,0.11)
C_a	(1.0,0.0)
C_d	(0.0,0.0)
O	(0.0,0.0)
|M_a|	(7.66,6.27)
C(M_a)	(1.0,0.0)
E_r	(0.07,0.16)
E_d	(0.39,0.29)
E_ell	(0.0,0.0)
||(E_r,E_d,E_ell)||	(0.42,0.3)

uniprotkb:P33897 uniprotkb:Q9UBJ2 bind|grid|ophid two-hybrid-test|aggregation|Liu LX [1999]|pubmed:10551832

Module Pair Quality

mu_a	0.11
mu_d	0.01
S_a	47.0
S_d	0.0
Delta_a	0.05
Delta_d	0.0
C_a	1.0
C_d	0.0
O	0.0
|M_a|	38
C(M_a)	1
E_r	0.0
E_d	0.2
E_ell	0.0
||(E_r,E_d,E_ell)||	0.2

Module visibility: m1 m2 [✓ both]

Show homology: [✓ Homology]

Protein information

# interactome interactions:	2
# module interactions:	1
# module homologs:	1
# GO annotations:	3

GO Annotations

GOId	Annotation D...	Evide...	Annotation T...
0000910	cytokinesis	IMP	Process
0006909	phagocytosis	IEA	Process
0005856	cytoskeleton	IEA	Component

Appendix I: Future directions

As cell systems continue to be better understood with increased study of Systems Biology [39], this direction of research has a bright future beyond the current concept of interactome. Interaction assays are being designed to overcome limitations of the common yeast two-hybrid (Y2H) [40] and affinity-purification mass spectrometry (AP-MS) assays [41]; these include surface plasmon resonance (SPR) [41, 42], optical microscopic techniques such as FRET [4, 41, 43], and mating-based split ubiquitin systems (mbSUS) [41]. As techniques are improved, each interactome is likely to be separated into various sub-interactomes distinguished by cell type and interaction type. A natural next step from the work already completed is to study the relationships between algorithms being used to detect conserved modules, experimental assays being used to construct interactomes, and organization of the cell. This should prove to be fertile ground for designing algorithms tailored to specific experimental assays, with the aim to overcome limitations of each assay and to uncover true organization of the cell.

A new algorithm, EvoNibble [44], has been designed for finding sets of vertices with low conductance in a graph, which may possibly be adapted to find small modules in a large interactome. EvoNibble is a randomized algorithm based on a volume-biased evolving set process. We plan to test how EvoNibble compares with PageRank-Nibble and Nibble as a subroutine for Produles. Though a randomized algorithm complicates replication of results, it can be run multiple times to create a high-confidence set that may also include results from other module-detection algorithms.

In subsequent work we plan to use GO annotations [26] to evaluate whether our algorithm, which does well on the algorithmic goals for network alignment, can be used for discovery of new conserved modules. GO annotations are divided into three categories: cellular component, molecular function, and biological process. Cellular component annotations serve as a filter for noisy interaction data: if the proteins are primarily in different components of the cell, they are not likely to interact meaningfully. Enrichment in molecular function annotations indicates that the module contains proteins that work together to perform a particular molecular function such as RNA splicing. Enrichment in biological process annotations indicates that the module contains proteins that may have diverse molecular functions but work together to perform a particular biological process such as signalling in a pathway.

Linear-time algorithms that extend Produles to multiple interactomes are in the design phase. A fast near-progressive approach simply considers proteins homologous to those in the modules already aligned, only slightly refining module boundaries. A more accurate iterative approach applies a module-finding algorithm to the new interactomes, starting at proteins homologous to those in the modules already aligned, more accurately detecting correct module boundaries. These algorithms for detecting multi-protein modularity conserved across multiple interactomes are to be incorporated into a larger functional genomics project that combines both interactomics and conservation data across multiple organisms to support transfer of function annotations.