Algorithms to Detect Multiprotein Modularity Conserved During Evolution

Luqman Hodgkinson and Richard M. Karp

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

Abstract. Detecting essential multiprotein modules that change infrequently during evolution is a challenging algorithmic task that is important for understanding the structure, function, and evolution of the biological cell. In this paper, we present a linear-time algorithm, Produles, that improves on the running time of previous algorithms. We present a biologically motivated graph theoretic set of algorithm goals complementary to previous evaluation measures, demonstrate that Produles attains these goals more comprehensively than previous algorithms, and exhibit certain recurrent anomalies in the performance of previous algorithms that are not detected by previous measures.

Keywords: modularity, interactomes, evolution, algorithms.

1   Introduction

Interactions between proteins in many organisms have been mapped, yielding large protein interaction networks, or interactomes [1]. The present paper continues a stream of scientific investigation focusing on conservation of modular structure of the cell, such as protein signaling pathways and multiprotein complexes, across organisms during evolution, with the premise that such structure can be described in terms of graph theoretic properties in the interactomes [2–6]. This stream of investigation has led to many successes, discovering conserved modularity across a wide range of evolutionary distances. However, there remain many challenges, such as running time, false positive predictions, coherence of predicted modules, and absence of a comprehensive collection of evaluation measures.

Evidence of conservation in the interaction data across organisms is essential for modules claimed by an algorithm to be conserved over a given evolutionary distance [7]. Due to the additivity of the scoring function for some previous algorithms in the interaction densities across organisms, a very dense network in one organism can be aligned with homologous proteins in another organism that have zero or few interactions among them. In this case, the interaction data does not support the claim of conservation across the given organisms.
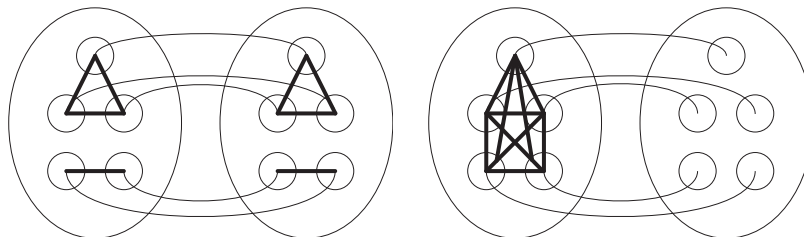
Fig. 1. Cartoons describing difficulties with additivity across data types and organisms. Organisms are represented by ovals. Proteins are represented by circles. Protein interactions are represented by thick lines. Proteins with high sequence similarity are connected with thin lines. Algorithms that are additive across the interaction and sequence data may predict the module on the left to be conserved due to high sequence similarity. In this case, the module boundaries are not well-defined, most likely containing portions of multiple modules that may have no relation with each other. Algorithms that are additive in the interactions across organisms may predict the module on the right to be conserved though there is no evidence for module conservation across the organisms in the protein interaction data.

Good boundaries are important for the modules that are returned by an algorithm. Some previous algorithms, such as NetworkBlast [3] and Graemlin [5], use a scoring function that is a sum of multiple scores: one score based on protein sequence similarity, and one score from each organism based on the density of interactions among the module proteins in the interactome for that organism. These algorithms then use a greedy search on this scoring function to find conserved modules. Due to the additivity, module pairs similar to the cartoons in Fig. 1 may receive high scores and be reported as conserved.

Produles is an important step to address these issues. It runs in linear time, scaling better than Match-and-Split [6] and MaWISh [4], and does not exhibit the recurrent anomalies that result from the additivity of the scoring function across organisms and data sources that forms the basis for NetworkBlast and Graemlin. Our objective is to initiate discussion of evaluation measures that are sensitive to these and similar issues by introducing the important algorithm goals described in Section 3.

2   Algorithms

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, $h : V_1 \times V_2 \to \mathbb{R}^+$, defined only for the most sequence similar pairs of proteins appearing in the interactomes. In this study, h is derived

from BLAST [8] E-values. 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 easily applied to any pair of interactomes, including those for newly studied organisms.

Modularity, Conductance, and Degree Bounds

A modular system consists of parts organized in such a way that strong interactions occur within each group or module, but parts belonging to different modules interact only weakly [9]. Following this, a natural definition of multiprotein 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 multiprotein module is a set of proteins $M \subset V$ such that $|M| \ll |V|$ and M has a large value of

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

where $E(M)$ is the set of interactions with both interactants in M, and $\mathtt{cut}(M, V \setminus M)$ is the set of interactions spanning M and $V \setminus M$. Of the interactions involving proteins in M, the fraction contained entirely within M is given by $\mu(M)$. This is similar to the earlier definition of λ-module [10].

The conductance of a set of vertices in a graph is defined as

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

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

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

Thus, when searching for relatively small modules in a large interactome, minimizing conductance is equivalent to maximizing modularity. This relationship allows us to modify powerful algorithms from theoretical computer science designed for minimizing conductance [11]. It has been shown that conductance in protein interaction networks is negatively correlated with functional coherence, validating both this definition of modularity and the notion that biological systems consist of functional modules [12].

Assuming we are searching for modules of size at most b with modularity at least d, the vertices in any such module have bounded degree. Let $\delta(u)$ be the degree of u in G.

Theorem 1. *If $d > 0$, the objective function in the optimization problem*

$$\max_{G,M,u} \quad \delta(u)$$

$$s.t. \quad u \in M$$
$$\quad |M| = b$$
$$\quad \mu(M) \geq d$$
$$\quad \mu(M) > \mu(M\backslash\{u\})$$

satisfies the bound $\delta(u) < (b-1)(1+d)/d$.

Proof. Let $M' \triangleq M\backslash\{u\}$. Let $y \triangleq |E(M')|$. Let $x \triangleq |\texttt{cut}(M',\{u\})|$.

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

so

$$|\texttt{cut}(M',V\backslash M')| > \frac{y(1-\mu(M))}{\mu(M)}$$

Thus,

$$\mu(M) = \frac{x+y}{[\delta(u)-x] + [|\texttt{cut}(M',V\backslash M')| - x] + [x+y]} < \frac{x+y}{\delta(u)-x+y+\frac{y(1-\mu(M))}{\mu(M)}}$$

which implies

$$\mu(M) < \frac{x}{\delta(u)-x}$$

As $\mu(M) \geq d$,

$$\delta(u) < \frac{x(1+d)}{d} \leq \frac{(b-1)(1+d)}{d}$$

$\square$

The motivation for the restriction $\mu(M) > \mu(M\backslash\{u\})$ is that when searching for modules with high modularity, there may be proteins with such high degrees that it always improves the modularity to remove them from the module. It can be shown that this bound is tight and that neither requiring connectivity of M in the underlying graph nor requiring connectivity of $M\backslash\{u\}$ in the underlying graph can allow the bound to be further tightened.

Modularity Maximization Algorithm

PageRank-Nibble [11] is an algorithm for finding a module with low conductance in a graph $G = (V, E)$. Let A be the adjacency matrix for G. Let D be a diagonal matrix with diagonal entries $D_{ii} = \delta(i)$ where $\delta(i)$ 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 that with probability $1/2$ remains at the current vertex and

with probability 1/2 randomly walks to an adjacent vertex. 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 and s is a row vector distribution on the vertices of the graph called the preference vector. Define the distribution that places all mass at vertex v

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

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 [11].

Let p be a distribution on the vertices of G. Let the vertices be sorted in descending order by $p(\cdot)/\delta(\cdot)$ where ties are broken arbitrarily. Let $S_j(p)$ be the set of the first j vertices in this sorted list. For $j \in \{1, ..., |V|\}$, the set $S_j(p)$ is called a sweep set [11].

The PageRank-Nibble algorithm consists of computing an approximate Page-Rank vector with $s = \chi_v$, defined as $\mathrm{apr}(\alpha, s, r) = \mathrm{pr}(\alpha, s) - \mathrm{pr}(\alpha, r)$, where r is called a residual vector, and then returning the sweep set $S_j(\mathrm{apr}(\alpha, \chi_v, r))$ with minimum conductance [11].

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, $0 = \mathrm{apr}(\alpha, \chi_v, \chi_v)$. After initializing $p_1 = 0, r_1 = \chi_v$, the solution is improved iteratively. Each iteration, called a push operation, chooses an arbitrary vertex u such that $r_i(u)/\delta(u) \geq \epsilon$. 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 $(u, v) \in E$, $r_{i+1}(v) = r_i(v) + (1 - \alpha)r_i(u)/(2\delta(u))$

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 [11].

Each push operation maintains the invariant [11]

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

When no additional pushes can be performed, the final residual vector r satisfies

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

The running time for computing $\mathrm{apr}(\alpha, \chi_v, r)$ is $\mathcal{O}(1/(\epsilon\alpha))$ [11]. If we set ϵ and α to constants, which is reasonable given their meanings, and if we consider only the first b sweep sets, the algorithm runs in constant time. As we desire the degrees of the vertices in the final set to be bounded, we do not consider any sweep sets that contain vertices with degree $(b - 1)(1 + d)/d$ or greater, and we also require connectivity in the underlying graph.

Algorithm to Detect Conservation

The algorithm begins by finding a multiprotein module,

$$M \subset V_1$$

with high modularity in G_1 using the algorithm described previously. 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:

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

where a is a lower bound on size, b is an upper bound on size, c is a size 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 we report the pair $(\mathcal{R}_T(M, N_i), N_i)$ as a conserved multiprotein module.

Each protein is used exactly once as a starting vertex for the modularity maximization algorithm. A counter is maintained for each protein in G_1. When a protein is placed in a module by the modularity maximization algorithm, the counter for the protein is incremented. Each counter has maximum value e for some constant e. If the modularity maximization 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. 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.

Proof of Linear Running Time

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 a union of vertex lists and does not require additional considerations 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

$$\sum_{M} \sum_{v \in M} |h(v, \cdot)| = \sum_{v} \sum_{M : v \in M} |h(v, \cdot)|$$
$$\leq e \sum_{v} |h(v, \cdot)|$$
$$= e|h(\cdot, \cdot)|$$

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 desire N_i such that $\mu(N_i) \geq d$ and $|N_i| \leq b$, which ideally do not contain any vertex u such that $\mu(N_i \backslash \{u\}) > \mu(N_i)$, we can discard, by Theorem 1, any vertex $v \in \mathcal{H}_T(M)$ with degree in G_2 of $(b-1)(1+d)/d$ or greater. A modified depth-first search that transitions only among vertices in $\mathcal{H}_T(M)$ is then used to compute $N_1, N_2, ..., N_k$. This requires time

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

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 a union of lists stored at the vertices in the N_i requires time $\mathcal{O}(\sum_i |N_i| b \log b) = \mathcal{O}(|\mathcal{H}_T(M)|)$. Testing for connectivity of a single $\mathcal{R}_T(M, N_i)$ with a modified depth-first search that transitions only 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 degree bounded by $(b-1)(1+d)/d$. 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 computing the sum of degrees of the vertices in U and the number of edges with both endpoints in U. These can be computed in constant time when $|U| \leq b$ as each vertex in U has degree bounded by $(b-1)(1+d)/d$.

3   Biologically Motivated Algorithm Goals

These goals address the challenges described in the introduction. Goal 1 is a measure of how many sequence dissimilar proteins participate in the module. Goal 2 is a measure of quality of module boundaries. Goal 3 is a measure of evidence for the claim of conservation in the interaction data. Goal 4 measures fit to an evolutionary model that includes interaction formation and divergence, protein duplication and divergence, and protein loss. Goal 5 measures proteome coverage and module overlap. We now quantify these goals mathematically.

Definition 1. *(Algorithm output)* Let k pairs of conserved modules returned by an algorithm be $\mathcal{M} = \{(M_1^i, M_2^i) \mid i \in \{1, ..., k\}\}$. Let $(M_1, M_2) \in \mathcal{M}$. Let $M \in \{M_1, M_2\}$.

Definition 2. *(Filled module)* Let $G_{int}(M) = (M, E(M))$.

Definition 3. *(Module homology graph)* Let $G_{hom}(M_1, M_2) = (M_1 \cup M_2, H(M))$, where, for $p_1 \in M_1$, $p_2 \in M_2$, $(p_1, p_2) \in H(M)$ iff $h(p_1, p_2)$ is defined.

Definition 4. *(Module size)* Let $S(M) = |M|$.

Definition 5. *(Module density)* Let $\Delta(M) = |E(M)|/\binom{|M|}{2}$.

Definition 6. *(Interaction components)* Let $C(M)$ be the number of connected components in $G_{int}(M)$.

Definition 7. *(Module average)* Let $f_a(M_1, M_2) = (f(M_1) + f(M_2))/2$, where $f \in \{\mu, S, \Delta, C\}$.

Definition 8. *(Module difference)* Let $f_d(M_1, M_2) = |f(M_1) - f(M_2)|$, where $f \in \{\mu, S, \Delta, C\}$.

Definition 9. *(Module overlap)* Let $\mathcal{O}^i(M_1^i, M_2^i) = \max_{j \neq i} \min\{|M_1^j \cap M_1^i|/|M_1^i|,$ $|M_2^j \cap M_2^i|/|M_2^i|\}$. 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.

Definition 10. *(Ancestral protein)* Let $p = (P_1, P_2)$, where $P_1 \subseteq M_1, P_2 \subseteq M_2$, and $G_{hom}(P_1, P_2)$ consists of a single connected component.

Definition 11. *(Ancestral protein projection)* For ancestral protein $p = (P_1, P_2)$, P_i is the projection of p on M_i for $i \in \{1, 2\}$.

Definition 12. *(Ancestral module)* Let $M_a(M_1, M_2)$ be the set of ancestral proteins for (M_1, M_2). The arguments, M_1, M_2, may be omitted for brevity when the context is clear.

Definition 13. *(Relationship disagreement)* Let $p, q \in M_a$, where $p = (P_1, P_2)$, $q = (Q_1, Q_2)$. For $i, j \in \{1, 2\}$, relationship disagreement means 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 $q'' \in Q_j$. Let $\mathcal{R}(M_1, M_2)$ be the number of relationship disagreements.

Definition 14. *(Relationship evolution)* Let $E_r(M_1, M_2) = \mathcal{R}(M_1, M_2)/\binom{|M_a|}{2}$, the fraction of possible relationship disagreements.

Definition 15. *(Ancestral module projection)* For $i \in \{1, 2\}$, let $\pi_i(M_a) = \{P_i \mid (P_1, P_2) \in M_a \wedge P_i \neq \emptyset\}$.

Definition 16. *(Number of protein duplications)* Let $\mathcal{D}(M_1, M_2) = |M_1| - |\pi_1(M_a)| + |M_2| - |\pi_2(M_a)|$.

Definition 17. *(Protein duplication evolution)* Let $E_d(M_1, M_2) = \mathcal{D}(M_1, M_2)/(|M_1| + |M_2| - 2)$, the fraction of possible protein duplications.

Definition 18. *(Number of protein losses)* Let $\mathcal{L}(M_1, M_2) = 2|M_a| - |\pi_1(M_a)| - |\pi_2(M_a)|$.

Definition 19. *(Protein loss evolution)* Let $E_\ell(M_1, M_2) = \mathcal{L}(M_1, M_2)/(|M_2| + |M_1|)$, the fraction of possible protein losses.

Definition 20. *(Ancestral components)* Let $C(M_a)$ be the number of connected components in a graph with vertex set M_a, where an edge 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 $i \in \{1, 2\}$.

Definition 21. *(Proteome coverage)* Let $\mathcal{C}_i = |\mathcal{U}_i|/|V_i|$, where $\mathcal{U}_i$ is the set of proteins from V_i that are part of conserved modules. Let $\mathcal{C} = (\mathcal{C}_1 + \mathcal{C}_2)/2$.

Goal 1. $|M_a|$ is the number of ancestral proteins and should be reasonably large for significant multiprotein modules.

Goal 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 module.

Goal 3. Δ_d, C_a, and C_d should be reasonably low to provide evidence for the claim of conservation across organisms. This may be problematic for models that are additive in the interaction densities across organisms.

Goal 4. E_r, E_d, and E_ℓ should be reasonably low for a good fit with evolution.

Goal 5. $\mathcal{C}$ and k should be in reasonable ranges with a low average value of $\mathcal{O}^i$.

4   Experiments and Results

Produles, NetworkBlast-M [13], Match-and-Split [6], and MaWISh [4] were applied to iRefIndex [14] binary interactions, Release 6.0, for *Homo sapiens* and *Drosophila melanogaster*, consisting of 74,554 interactions on 13,065 proteins for *H. sapiens* and 40,004 interactions on 10,050 proteins for *D. melanogaster*. The evaluation was performed on the module pairs returned that had 5-20 proteins per organism. This removes a large number of module pairs from Match-and-Split and MaWISh that consist of modules on two or three proteins, single edges or triangles, and the few huge modules from MaWISh with nearly a thousand proteins each and $C(M_a) > 1$, that likely have little information content. This has little effect on NetworkBlast-M for which nearly all of its modules have 13-15 proteins per organism [15]. All programs were run with varying h threshold, corresponding to varying numbers of homologous protein pairs: $h = 10^{-100}$: 5,675 pairs, $h = 10^{-40}$: 25,346 pairs, $h = 10^{-25}$: 50,831 pairs, and
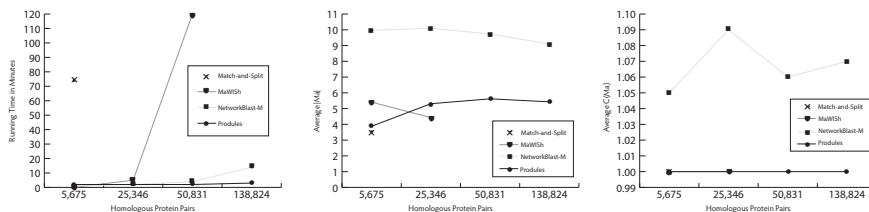
Fig. 2. Comparison of running time and performance on Goal 1 and Goal 2. The x-axis is the number of homologous protein pairs. The y-axis, from left to right, is the running time in minutes, the average $|M_a|$, and the average $C(M_a)$.
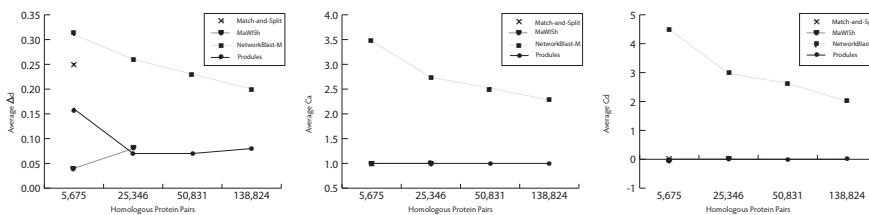

Fig. 3. Comparison of performance on Goal 3. The x-axis is as in Fig. 2. The y-axis, from left to right, is the average Δ_d, the average C_a, and the average C_d.

$h = 10^{-9}$: 138,824 pairs. Considering only the module pairs from NetworkBlast-M with highest NetworkBlast-M score does not significantly change the distributions [15]. Graemlin has nineteen network-specific parameters over a wide range of values. Together with the authors of Graemlin, we were unable to find settings that would yield results for the networks in this study.

As expected, Produles returns multiprotein modules with much higher values of μ than other approaches [15]. What is remarkable is that by focusing only on this measure, other desirable properties are attained. In Fig. 2, the linear running time of Produles is seen to be very desirable. Neither Match-and-Split nor MaWISh could complete on the data set with 50,831 homologous protein pairs. NetworkBlast-M has high average value of $|M_a|$ due mainly to its focus on modules with 13-15 proteins per organism. Both Match-and-Split and Produles guarantee that $C(M_a) = 1$. NetworkBlast-M has a high average value of $C(M_a)$ due to additivity across data types. MaWISh has a few module pairs with $C(M_a) > 1$ in a larger size range. Fig. 3 shows that NetworkBlast-M has difficulty with Goal 3 due to additivity of its scoring model in the interaction densities across organisms. NetworkBlast-M frequently aligns a dense module in one organism with a module that has zero or few interactions in the other organism. For all algorithms, average Δ_a is approximately 0.3 [15]. Fig. 4 shows that Produles performs comparably on the evolutionary model with algorithms that attempt to match topologies. By searching only for modularity, Produles detects conserved multiprotein module pairs in this data set that are consistent with evolution. Fig. 5 shows that NetworkBlast-M produces many overlapping
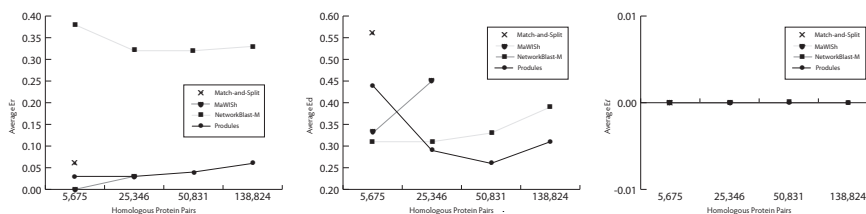
Fig. 4. Comparison of performance on Goal 4. The x-axis is as in Fig. 2. The y-axis, from left to right, is the average E_r, the average E_d, and the average E_ℓ.
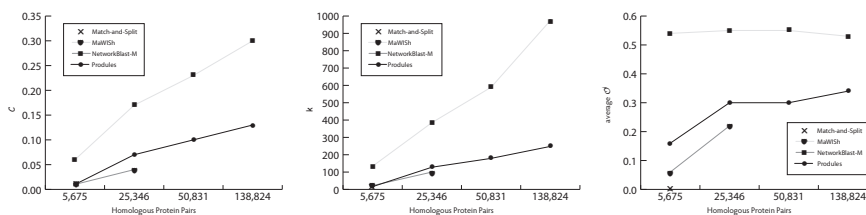

Fig. 5. Comparison of performance on Goal 5. The x-axis is as in Fig. 2. The y-axis, from left to right, is $\mathcal{C}$, k, and the average $\mathcal{O}^i$.

module pairs. As indicated by Figs. 3-4, for many of these, the interaction data does not support the claim of conservation.

As a final test, we computed GO biological process enrichment [16] with Bonferroni correction at 0.05 significance level. All the algorithms in this study performed comparably for modules in each size range. More than 95% of Network-Blast-M modules were in the size range 13-15 proteins for which the percentage of modules enriched were: 100% for Produles, 98% for NetworkBlast-M, 100% for MaWISh, and 100% for Match-and-Split. All remaining NetworkBlast-M modules were in the size range 10-12 proteins for which the percentage of modules enriched were: 79% for Produles, 66% for NetworkBlast-M, 89% for MaWISh, with no modules in this size range for Match-and-Split.

5   Conclusion

We present a linear-time algorithm to detect conserved multiprotein modularity, and a new set of evaluation measures, comparing with leading algorithms and describing reasons for lower performance of earlier approaches. The measures introduced are sensitive to important issues not addressed by previous measures.

References

1. Vidal, M.: Interactome modeling. FEBS Letters 579, 1834–1838 (2005)
2. Kelley, B.P., Sharan, R., Karp, R.M., Sittler, T., Root, D.E., Stockwell, B.R., Ideker, T.: Conserved pathways within bacteria and yeast as revealed by global protein network alignment. Proc. Natl. Acad. Sci. 100(20), 11394–11399 (2003)
3. Sharan, R., Suthram, S., Kelley, R.M., Kuhn, T., McCuine, S., Uetz, P., Sittler, T., Karp, R.M., Ideker, T.: Conserved patterns of protein interaction in multiple species. Proc. Natl. Acad. Sci. 102(6), 1947–1979 (2005)
4. Koyutürk, M., Kim, Y., Topkara, U., Subramaniam, S., Szpankowski, W., Grama, A.: Pairwise alignment of protein interaction networks. Journal of Computational Biology 13(2), 182–199 (2006)
5. Flannick, J., Novak, A., Srinivasan, B.S., McAdams, H.H., Batzoglou, S.: Graemlin: general and robust alignment of multiple large interaction networks. Genome Research 16, 1169–1181 (2006)
6. Narayanan, M., Karp, R.M.: Comparing protein interaction networks via a graph match-and-split algorithm. Journal of Computational Biology 14(7), 892–907 (2007)
7. Beltrao, P., Serrano, L.: Specificity and evolvability in eukaryotic protein interaction networks. PLoS Computational Biology 3(2), e25 (2007)
8. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of Molecular Biology 215(3), 403–410 (1990)
9. Simon, H.A.: The structure of complexity in an evolving world: the role of near decomposability. In: Callebaut, W., Rasskin-Gutman, D. (eds.) Modularity: Understanding the Development and Evolution of Natural Complex Systems. MIT Press: Vienna Series in Theoretical Biology, Cambridge (2005)
10. Li, M., Wang, J., Chen, J., Pan, Y.: Hierarchical organization of functional modules in weighted protein interaction networks using clustering coefficient. In Măndoiu, I., Narasimhan, G., Zhang, Y. (eds.) ISBRA 2009. LNCS (LNBI), vol. 5542, pp. 75-86. Springer-Verlag, Berlin (2009)
11. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using PageRank vectors. In 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pp. 475–486. IEEE Press, New York (2006).
12. Voevodski, K., Teng, S., Xia, Y.: Finding local communities in protein networks. BMC Bioinformatics 10, 297 (2009)
13. Kalaev, M., Bafna, V., Sharan, R.: Fast and accurate alignment of multiple protein networks. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 246-256. Springer, Heidelberg (2008)
14. Razick, S., Magklaras, G., Donaldson, I.M.: iRefIndex: a consolidated protein interaction database with provenance. BMC Bioinformatics 9, 405 (2008)
15. Hodgkinson, L., Karp, R.M.: Algorithms to detect multi-protein modularity conserved during evolution. EECS Department, University of California, Berkeley, Technical Report UCB/EECS-2011-7 (2011)
16. Boyle, E.I., Weng, S., Gollub, J., Jin, H., Botstein, D., Cherry, J.M., Sherlock, G.: 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(18), 3710-3715 (2004)