Survey: Enhancing Protein Complex Prediction in PPI Networks with GO Similarity Weighting

True Price, Francisco I Peña III, Young-Rae Cho
Department of Computer Science
Baylor University
Waco, TX, USA
{*true_price, frank_pena, young-rae_cho*}*@baylor.edu*

***Abstract*—Predicting protein complexes from protein-protein interaction (PPI) networks has been the focus of many computational approaches over the last decade. These methods tend to vary in performance based on the structure of the network and the parameters provided to the algorithm. Here, we evaluate the merits of enhancing PPI networks with semantic similarity edge weights using Gene Ontology (GO) and its annotation data. We compare the cluster features and predictive efficacy of six well-known unweighted protein complex detection methods (Clique Percolation, MCODE, DPClus, IPCA, Graph Entropy, and CoAch) against updated weighted implementations. We conclude that incorporating semantic similarity edge weighting in PPI network analysis unequivocally increases the performance of these methods.**

***Keywords*-protein-protein interactions, PPI, PPI networks, protein interaction networks, semantic similarity, protein complexes, weighted networks**

I. Introduction

The study of protein-protein interactions (PPIs) determined by high-throughput experimental techniques has created the need for methods allowing us to discover new information about biological function. These interactions can be thought of as a large-scale network, with nodes representing proteins and edges signifying an interaction between two proteins. Often, PPI networks can be on the scale of an entire genome, called the interactome [1], [2]. With the rapid pace of model organism interactome generation, the development of computational approaches to analyze these networks is increasingly in demand. In a PPI network, we can potentially find protein complexes or functional modules as densely connected subgraphs. A protein complex is a group of proteins that interact with each other at the same time and place creating a quaternary structure. Functional modules are composed of proteins that bind each other at different times and places and are involved in the same cellular process [3]. Various graph clustering algorithms [4] have been applied to PPI networks to detect protein complexes or functional modules, including several designed specifically for PPI network analysis.

The quality of PPI network analysis, however, is strongly tied to the performance of the graph clustering method being used. A balance must be struck between eliminating extraneous proteins – leading to smaller clusters with a low coverage rate – and selecting large modules as predicted complexes – possibly generating a large number of high-coverage clusters poorly matching actual complexes. These are issues of modularity and noise: Does a highly-dense region of the PPI network contain one large complex, or several smaller complexes? Similarly, how are we able to distinguish between noise and actual complexes in low-connectivity regions of the network? To address these problems, current research is largely oriented around improving or creating new, PPI-tailored methods.

The process of analyzing a "raw" PPI network is inherently limited by proteins that interact with each other but are not involved in the same protein complex. To this end, the effectiveness of protein complex prediction methods requires enhancing the information inherently stored in the PPI network. In this paper, we analyze the experimental value of using semantic similarity scores to weight the PPIs of *S. cerevisiae* in the BioGRID database [5]. We use Gene Ontology (GO) data [6] to calculate two different metrics of semantic similarity scoring previously presented by Cho *et al.* [7] and use these measures as edge weights for the PPI network. To assess the benefits of semantic similarity weighting, we have converted several unweighted PPI network analysis methods to a weighted schema: Clique Percolation [8], MCODE [9], DPClus [10], IPCA [11], Graph Entropy [12], and CoAch [13]. We compare the results of these modified weighted methods to their original unweighted implementations and analyze the predictive efficacy of weighted PPI analysis using the set of known protein complexes of *S. cerevisiae* [14].

II. Survey of Methods

A. Protein Complex Detection

We analyze six protein complex prediction methods in three different categories: one clique-merging approach (Clique Percolation), four seed-growth approaches (MCODE, DPClus, IPCA, Graph Entropy), and one combined approach (CoAch). The methods chosen have performed well in previous surveys of PPI analysis methods [4], and, importantly, are all centered around detecting highly dense subgraphs in the network. Additionally, while methods for weighted network analysis have been defined for some of the six methods, to our knowledge, none have published software for accomplishing this task, with the exception of the CFinder software for clique percolation.

Seed-growth approaches require that a seed (either a single vertex or a small-sized dense subgraph) be chosen at the beginning and grown based on a density function. These seed-growth style algorithms have the advantage of efficiently detecting highly modular clusters in large PPI networks, but their performance is often hugely dependent on user-supplied thresholds for seed selection and cluster growth.

1) Network Terminology: Consider a PPI network $G = (V, E)$, where V is a set of vertices, representing proteins, and E is a set of edges connecting pairs of these vertices, representing protein interactions. The *degree* of a node $v \in V$ is the number of edges connected to V; it is also the size of v's set of *neighbors*:

$$\deg(v) = |\{(u, v) \mid u \in V and (u, v) \in E\}|$$

A *subgraph* $G' = (V', E')$ of a graph G is a subset of vertices of G plus the edges by which they are connected. We can define, as a general term, the *density* of a graph or subgraph as the number of edges in the graph divided by the number of edges possible in a graph of that size:

$$d(G) = \frac{2 \times |E|}{|V| \times (|V| - 1)}$$

We describe specific modifications for converting the methods listed above into weighted network implementations. The weighted degree of a vertex v in a graph $G = (V, E)$ is given by [4]:

$$\deg_w(v) = \Sigma_{e=(u,v)\in E} w(e),$$

where $w(e)$ is the weight of edge e. In a similar fashion, the weighted density of a graph is defined as:

$$d_w(G) = \frac{2 \times \Sigma_{e\in E} w(e)}{|V| \times (|V| - 1)}$$

In methods such as DPClus that calculate vertex and edge weights from graph structure, we refer to these values as *global* weights and the method-specific values as *local* weights.

2) Clique Percolation: The method of clique percolation was first introduced by Derényi *et al.* [15] for random graphs and was formally defined by Palla *et al.* [8] (in supplementary material). Because it is designed to discover highly dense communities of nodes, clique percolation has become a popular method for protein-complex prediction in PPI networks. Several clique percolation algorithms have been proposed, notably CFinder [16], sequential clique percolation (SCP) [17], and most recently, a very efficient implementation using a binary tree hierarchy and Bloom filters [18].

A *clique* is a completely connected subset of vertices in a graph, where an edge exists between every pair of vertices in the subset. The goal of clique percolation is, given a network and some k, to find all cliques of size k in the network and build percolation communities from them. Two k-cliques are said to percolate into each other if they have $k - 1$ vertices in common. Such k-cliques can be said to be *adjacent* to one another, and a k-clique is *reachable* from another k-clique if there exists a sequence of adjacent k-cliques by which the first clique can be transformed into the second. Percolation communities are formed from the union of maximally reachable k-cliques, where every k-clique is reachable from every other k-clique in the set.

To find percolation communities, the CFinder and Bloom filter methods start by discovering maximal cliques (cliques which are not contained in any larger clique within the given network) and applying adjacency filtering for a given k. CFinder uses a clique-clique overlap matrix to find percolation communities, while the Bloom filter method eliminates non-adjacent k-cliques in a binary tree hierarchy. In a different approach, SCP finds all k-cliques in the network and percolates based on common size-($k - 1$) sub-cliques, using theoretical bipartite graph methods to increase efficiency. The Bloom filter method empirically outperforms the other methods for all values of k, and SCP has been found to be most efficient for low k.

Farkas *et al.* [19] discuss procedures for applying clique intensity in order to adapt the clique percolation method for weighted network analysis. The intensity I of a subgraph $G = (V, E)$ is defined as the geometric mean of its weights [20]:

$$I(G) = \left(\prod_{e\in E} w_e \right)^{-|E|}.$$

Given a k, we can order the k-cliques of a graph by intensity, $w_1 \geq w_2 \geq \ldots \geq w_n$. Farkas *et al.* suggest a method, which we adopt here, of thresholding the cliques used in the clique percolation based on intensity, that is, excluding cliques whose intensity is below a certain value. If this threshold is above w_1, no percolation clusters are found because no clique meets the intensity theshold. Conversely, if the threshold is below w_n, the method is equivalent to unweighted clique percolation, since no cliques are filtered out. To balance these two extremes – no clusters yielded versus potentially giant clusters yielded – we iteratively decrease this threshold, starting at w_1 and working down, until the ratio of the size of the two largest clusters reaches two. While this method can fluctuate for small-sized networks, it is well-suited for PPI analysis.

One difficulty to this approach is that in determining clique intensities, every k-clique in the network must be generated. The CFinder software suite has an implementation for this weighted graph approach, but we found it intractable when given our PPI network. Moreover, since the Bloom filter method is intended for use with maximal cliques, it is not suited for the generation of all k-cliques. Therefore, we chose to implement a weighted version of the SCP algorithm, which naturally generates all cliques of size k and is therefore convenient for such adaption.

3) MCODE: Introduced by Bader and Hogue in 2003 [9], MCODE is the earliest seed-growth method for predicting protein complexes from PPI networks. MCODE works in two steps: 1) vertex weighting, and 2) molecular complex prediction. In the vertex weighting step, the weight of a vertex v in the PPI network is calculated from the highest k-core of v's neighborhood, including v. The k-core of a graph is a subgraph where every node is of degree k or greater; the highest k-core is simply the k-core with the highest value of k. The weight of v is defined as this maximum k times the density of the corresponding k-core. In the current implementation of MCODE published online (http://baderlab.org/Software/MCODE), the density of a k-core $C_k = (V_k, E_k)$ is defined as

$$d(k) = \frac{|E_k|}{|V_k|^2}.$$

To incorporate edge weights, the equation becomes:

$$d_w(k) = \frac{\Sigma_{e \in E} w_e}{|V_k|^2}.$$

After all the vertices in the PPI network have been weighted, MCODE applies a seed-growth procedure to build predicted protein complexes. A seed is selected as the highest weighted vertex, and the cluster is recursively grown to include all neighboring vertices within a user-defined vertex weight percentage (VWP) of the original seed weight. Once the cluster is formed, its vertices are marked as visited, making them ineligible for inclusion in future clusters. This process is repeated until all vertices have been marked as visited.

MCODE offers two optional post-processing techniques. The first, "haircut," removes vertices that share only one edge in the predicted complex. The second, "fluff," allows for overlapping of predicted complexes by adding all neighbors of the cluster whose highest k-core density is greater than a given fluff parameter. Vertices added in the fluff step are not marked as visited.

4) DPClus: DPClus, defined by Amin *et. al* in [10], projects weights onto an unweighted graph using a common neighbors approach. In DPClus, the weight of an edge $(u, v) \in E$ is defined as the number of common neighbors between u and v. Similarly, the weight of a vertex is its weighted degree – the sum of all edges connected to the vertex. This algorithm works much like MCODE, except the stopping condition for cluster formation is based on cluster density, rather than vertex weight.

DPClus relies on two stopping thresholds for cluster formation: cluster density (d_{in}) and cluster property (cp_{in}). The density d_k of a cluster $C_k = (V_k, E_k)$ is defined as the ratio of the number of edges in the cluster to the number of possible edges in the cluster, or

$$d_k = \frac{E_k}{|V_k| \times (|V_k| - 1)}.$$

The cluster property cp_{nk} between a node n and a cluster C_k is defined as the inverse density of C_k (without n) weighted by the proportion of cluster nodes which are adjacent to n:

$$cp_{nk} = \frac{|N_n \cap V_k|}{d_k \times |V_k|},$$

where N_n is the set of vertices adjacent to n.

For seed node selection, the seed is chosen as the vertex with the highest weight, unless the highest weight is zero (meaning that no two nodes in the graph share any common neighbors), in which case the vertex with the highest degree is chosen as the seed. The cluster is grown by iteratively selecting the highest priority vertex from its set of neighbor vertices. The priority of a neighbor vertex is determined by 1) the number of cluster nodes adjacent to the neighbor and 2) the sum of the edge weights between the neighbor and its adjacent cluster nodes. Cluster growth stops when no neighbor vertex can be added without either the cluster density (including the neighbor) or the cluster property (between the neighbor and the original cluster) dropping below its respective threshold. At this point, all nodes in the cluster are removed from the network, and the edge and vertex weights of the resulting graph are recalculated. The algorithm terminates when no edges in the graph remain.

Additionally, the authors of DPClus define a 'fine-tuning' procedure in the special case that all neighbors of a cluster are connected to the cluster by a single edge. If this occurs, neighbor vertices are prioritized by the number of other neighbor vertices to which they are adjacent, minus the number of neighbor vertices to which their connected cluster node is adjacent. In our experiments, this special case was never observed.

Like MCODE, DPClus does not natively generate overlapping clusters but does allow for overlapping cluster nodes can be added in a post-processing step. To do this, DPClus considers all neighbors of the original cluster, both visited and unvisited. Using this neighbor set, it applies the same iterative prioritize-add-reweight procedure as before, but does not expand the neighborhood frontier when a neighbor is added to the cluster. That is, only neighbors of the original cluster are considered as candidates for expansion.

The method for incorporating edge weights to DPClus affects much of the algorithm. The seed node for a new cluster is simply chosen as the highest global weighted-degree vertex. This weighted degree is computed from the remaining graph, just as in the unweighted version.

When prioritizing neighbor vertices, only the first criterion is altered, ordering by: 1) the sum of the global edge weights between the vertex and the cluster node, and 2) the sum of the local edge weights between the vertex and the cluster node. Finally, when determining cluster density with a candidate neighbor vertex included, the general formula for weighted subgraph density is employed using the global edge weights. Local edge and vertex weights of the remaining graph are recalculated every iteration in the exact same manner as before.

5) IPCA: IPCA was introduced by Li *et. al* [11] as a modified version of DPClus. In contrast to DPClus, this method focuses on the maintaining the *diameter* of a cluster, defined as the maximum shortest distance between all pairs of vertices, rather than its density. In doing so, the seed growth aspect of IPCA emphasizes structural *closeness* of a predicted protein complex, as well as structural connectivity.

Like DPClus, IPCA computes local vertex and edge weights by counting the number of common neighbors shared between two vertices. However, IPCA calculates these values only once at the beginning of the algorithm, rather than updating them every time a discovered cluster is removed from the graph. This allows overlap to occur naturally between clusters, as cluster nodes are not permanently removed from the graph; however, it can also lead to a large amount of cluster overlap.

IPCA can incorporate two different neighbor-ordering thresholds: $(SP \leq d)$ and $(ASP \leq d)$. $(SP \leq d)$ requires that the diameter of a cluster never exceed d, while $(ASP \leq d)$ considers the average length of the shortest paths between all pairs vertices in a cluster. Because the currently published implementation of IPCA only implements $(SP \leq d)$, we focus on this threshold alone.

Seed node selection in IPCA works the same as in DPClus, where the unvisited vertex with the highest weight is chosen, with ties going to the vertex with the highest degree. A cluster K is extended by ordering its neighbors by the value IN_{vK}, defined as

$$IN_{vK} = \frac{m_{vK}}{|K|},$$

where m_{vK} is the number of edges between a neighbor vertex v and the cluster. The highest-valued neighbor v is added to the cluster if it meets two conditions: 1) $IN_{vK} \geq T_{in}$, and 2) $SP(K+v) \leq d$. Here, T_{in} is a user-specified threshold dictating the minimum percentage of adjacent cluster nodes that v must share in order to be added to the cluster. The second condition requires that the maximum shortest path length between v and every member of K be no greater than d. Typically, d is set to two.

To integrate edge weights to IPCA, the value IN_{vK} subsequently becomes:

$$IN_{vK} = \frac{\Sigma_{u \in K} w(u,v)}{|K|}$$

Importantly, the requirement of IPCA that $IN_{vK} \geq T_{in}$ becomes much more stringent in a weighted network context. If a vertex is chosen as a seed node but has no associated global edge weights greater than T_{in}, no neighbors will be added to the cluster. In this event, the seed node is still marked as visited, but the size-1 cluster is not reported as a predicted protein complex. This greatly reduces the number of clusters reported by the IPCA method.

If a neighboring vertex meets both of the above criteria, it is added to the cluster, and the priorities of the neighbors of this new cluster are recalculated. If no such neighbor is found, cluster growth stops, and the cluster is output as a predicted protein complex. The vertices in a generated cluster are marked as visited, making them ineligible as seed nodes but still able to be included in future clusters.

6) Graph Entropy Method: Kenley and Cho [12] recently introduced a new seed-growth style approach using an information-theoretic model to detect clusters of high modularity in a complex PPI network. This method takes advantage of the use of entropy with regard to information theory. Entropy is a measure of uncertainty involved in a random variable. This approach uses a new definition, *Graph Entropy*, as a measure of structural complexity in a graph. This algorithm incorporates a seed-growth technique. Unlike the other seed-growth style methods, however, the graph entropy approach does not require any predetermined threshold because it searches for an optimal solution by minimizing graph entropy.

This method finds locally optimal clusters with minimal graph entropy. A seed vertex is selected at random from a candidate set of seed vertices. Then, an initial cluster which is composed of the seed vertex and its immediate neighbors is created. Next, the neighbors are iteratively evaluated for removal to minimize the initial entropy of the graph outer boundary vertices are then added recursively if their addition causes the entropy of the graph to decrease.

The entropy of a vertex v is defined as

$$e(v) = -p_{\mathrm{i}}(v)\log_2 p_{\mathrm{i}}(v) - p_{\mathrm{o}}(v)\log_2 p_{\mathrm{o}}(v),$$

where $p_{\mathrm{i}}(v)$ is the ratio of the number of neighbors of v in the cluster to the total number of neighbors of v, and $p_{\mathrm{o}}(v) = 1 - p_{\mathrm{i}}(v)$.

The entropy of the graph is defined as the sum of the entropies for all vertices in the graph

$$e(G) = \sum_{v \in V} e(v).$$

For weighted graph entropy, the value $p_{\mathrm{i}}(v)$ of a vertex v becomes the weighted ratio of the sum of inner links (edges connecting the vertex to the current cluster V') to the weighted degree of v:

$$p_{\mathrm{i}}(v) = \frac{\sum_{v_i \in V'} w(v, v_i)}{\sum v_k \in V w(v, v_k)}.$$

The formulas for vertex entropy and total graph entropy remain unchanged for the weighted graph entropy method.

7) CoAch: The motivation behind the core-attachment (CoAch) algorithm [13] comes from the observation that protein complexes often have a dense core of highly interactive proteins. CoAch works in two steps, first discovering highly connected regions ("preliminary cores") of a network and then expanding these regions by adding strongly associated neighbors. The algorithm operates with three user-specified parameters: minimum core density d (for preliminary cores), maximum core affinity t (similarity threshold for distinct preliminary cores), and minimum neighbor closeness (for attaching non-core neighbors to preliminary cores). Minimum neighbor closeness is typically given as one-half.

The first step in CoAch is to discover protein-complex cores. To this end, a set of candidate cores is generated for every vertex in the graph, with similar cores consolidated to be maximally dense. The process for generating candidate cores from a vertex v is as follows: First, the algorithm determines G_v, the 2-core graph of v's neighborhood, including v. This step removes nodes that have no connections in v's neighborhood graph apart from sharing an edge with v. Next, the core vertices of G_v are selected. This is the set of vertices in G_v whose degree is greater than or equal to the average degree of G_v. After that, CoAch applies a recursive core-removal algorithm to G_v. This algorithm works by dividing G_v into non-core clusters of connected nodes if the density of G_v is less than a user-specified density threshold. These clusters are further divided into non-core components until the density threshold is met. Core nodes are recursively added back to the cluster once a sufficiently dense subgraph is found. Once the core-removal algorithm is complete, additional post-processing of the discovered cores is applied to make them maximally dense. This gives us a set of candidate cores generated from v.

The candidate cores of a single vertex are consolidated into a global set of preliminary cores. To apply this "redundancy filtering" step, every candidate core is scored against every preliminary core using the neighborhood affinity measure:

$$NA(A, B) = \frac{|V_A| \cap |V_B|}{|V_A| \times |V_B|}$$

for some candidate core A and some preliminary core B. If the NA score between the candidate core and its most similar preliminary core B_{max} is less than the user-defined affinity threshold, the candidate core is added to the set of preliminary cores. Otherwise, if the density-weighted size of the candidate core ($d_A \times |V_A|$) is greater than that of B_max, then the candidate core replaces B_max in the set of preliminary cores.

After preliminary core discovery has been completed on every vertex, the preliminary cores are expanded into predicted protein complexes. For every preliminary core $C = (V_C, E_C)$, CoAch takes the set of directly adjacent neighbors, N_C. Then, the algorithm adds all $v \in N_C$ adjacent to more than half of the vertices in C. This measure is also known as the *closeness* of v to C:

$$closeness(v, C) = \frac{|N_v| \cap |V_C|}{|V_C|},$$

where N_v is the neighbor set of v.

The weighted version of CoAch simply uses weighted degree and density functions.

B. Semantic Similarity Methods

An ontology is a formal way of representing knowledge which is described by concepts and their relationships [21]. As a collaborative effort to specify biological ontologies, GO addresses the need for consistent descriptions of genes and gene products across species [22]. It provides a collection of well-defined biological concepts, called GO terms, spanning three domains: biological processes, molecular functions and cellular components. GO is structured as a Directed Acyclic Graph (DAG) by specifying general-to-specific relationships between parent and child terms.

As another intriguing feature, GO maintains annotations for genes and gene products to their most specific GO terms, called direct annotations. Because of the general-to-specific relationships in the ontology, a gene that is annotated to a specific term is also annotated to all its parent terms on the paths towards the root. These are called inferred annotations. Considering both direct and inferred annotations, we can quantify the specificity of a GO term by the proportion of the annotating genes on the term to the total genes in the ontology.

Semantic similarity is a function computing closeness in meaning between terms within an ontology. A variety of semantic similarity measures have been proposed previously [23], [24], [25], [26], aiming at quantifying functional similarity between genes. They have utilized the GO structure and annotations. The existing semantic similarity measures can be grouped into four broad categories: *path-length-based methods*, *information-content-based methods*, *common-term-based methods*, and *hybrid methods*.

1) Path-length-based methods: Path-length-based methods compute path length between GO terms in the ontology structure. For instance, semantic similarity of two GO terms C_1 and C_2 can be considered as the inverse of the shortest path length between them. Semantic similarity is also measured by the depth to the most specific common ancestor term (SCA) C_0 that subsumes two GO terms C_1 and C_2, i.e., the shortest path length from the root to SCA [27]. The greater the depth to SCA of two terms is, the more similar they are. Considering not only commonality but also a difference between C_1 and C_2, we can normalize this method by the average depth of the terms [28].

$$sim_{ND}(C_1, C_2) = \frac{2 \cdot len(C_r, C_0)}{len(C_0, C_1) + len(C_0, C_2) + 2 \cdot len(C_r, C_0)},$$

where $len(C_i, C_j)$ is the shortest path length between C_i and C_j in GO. C_r denotes the root term.

2) Information-content-based methods: The methods in this category use the notion of information content of GO terms. The information content of a term C is defined as the negative log likelihood of C, $-\log P(C)$. In the application to GO, the likelihood $P(C)$ is calculated by the proportion of the annotating proteins on C to the entire proteins annotating in the ontology. This information theoretic measure, information content, assigns higher values to the GO terms that have higher specificity.

For semantic similarity between C_1 and C_2, Resnik [29] used the information content of the most specific common ancestor term (SCA) C_0 of C_1 and C_2. The more specific information the two terms share, the more similar they are.

$$sim_{R}(C_1, C_2) = -\log P(C_0).$$

Lin [30] took into consideration the ratio of the information content of C_0 to the average information content of C_1 and C_2.

$$sim_{L}(C_1, C_2) = \frac{2 \cdot \log P(C_0)}{\log P(C_1) + \log P(C_2)}.$$

Jiang and Conrath [31] proposed measuring the sum of differences of information contents between C_0 and two terms C_1 and C_2 and taking its inverse as the semantic similarity between C_1 and C_2.

$$sim_{JC}(C_1, C_2) = \frac{1}{1 - \log P(C_1) - \log P(C_2) + 2 \cdot \log P(C_0)}.$$

Cho *et al.* [7] recently proposed a combined method, called simICND, of normalizing Resnik's method by Jiang's method.

$$sim_{ICND}(C_1, C_2) = \frac{-\log P(C_0)}{1 - \log P(C_1) - \log P(C_2) + 2 \cdot \log P(C_0)}.$$

This method gives a penalty to Resnik's semantic similarity if the information contents of C_1 and C_2 are greater than the information content of C_0, i.e. their SCA.

3) Common-term-based methods: Common-term-based methods measure the overlap of two ancestor term sets of C_1 and C_2. The more ancestor GO terms C_1 and C_2 share, the higher similarity they have. The intersection of two ancestor term sets can be normalized by the union of them, called simUI [27], or by the smaller of them, called NTO [32].

$$sim_{UI}(C_1, C_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|},$$

and

$$sim_{NTO}(C_1, C_2) = \frac{|S_1 \cap S_2|}{\min(|S_1|, |S_2|)},$$

where S_1 and S_2 are the sets of all ancestor GO terms of C_1 and C_2, respectively.

4) Hybrid methods: Hybrid methods incorporate the aspects of two different categories. For example, Wang *et al.* [33] proposed the semantic similarity that integrates a common-term-based method with a path-length-based method. Their method, called G-SESAME, computes the ratio of common ancestor GO terms, but gives different weights to the the ancestor terms according their depth.

SimGIC [34] integrates a common-term-based method with an information-content-based method. In Formula II-B3, instead of counting the terms in S_1 and S_2, it calculates the sum of information contents of the terms.

$$sim_{GIC}(C_1, C_2) = \frac{\sum_{C_i \in S_1 \cap S_2} \log P(C_i)}{\sum_{C_j \in S_1 \cup S_2} \log P(C_j)},$$

where S_1 and S_2 are the sets of all ancestor GO terms of C_1 and C_2, respectively.

IntelliGO [35] is a vector representation model that combines the normalized depth with information contents as weights. Jain and Bader [36] introduced a novel approach, called TCSS, which applies clustering of GO terms to find sub-ontologies and measures semantic similarity by Resnik's method considering whether two terms are included in the same sub-ontology. This method attempts to solve the problem of unequal depth in different branches of GO.

Cho *et al.* [7] recently proposed a combined method, called simICNP, of normalizing Resnik's method as an information-content-based approach by a path-length-based method.

$$sim_{ICNP}(C_1, C_2) = \frac{-\log P(C_0)}{1 + len(C_1, C_2)}.$$

This method gives a penalty to Resnik's semantic similarity if C_1 and C_2 are located farther from their SCA in GO.

III. Experimental Results and Discussion

A. Experimental Design

To evaluate the effectiveness of weighted protein complex prediction methods, we used the protein-protein interaction data set of *S. cerevisiae* available from BioGRID [5]. This data set has 4416 proteins and 83151 interactions. To incorporate edge weights to the PPI network, we applied the simICND and simICNP semantic similarity scores, which have been shown to perform the best in [7]. For semantic similarity labeling, we use the molecular function (MF) and biological processes (BP) terms published by the GO Consortium [6]. Once the simICND and simICNP edge weights were calculated for the BioGRID data set, we removed zero-weight edges – that is, edges showing interactions between protein pairs with no semantic similarity – resulting in 4318 proteins and 81123 interactions.

For unweighted clique percolation, we use the Bloom filter implementation provided by Reid *et. al* (available online at http://sites.google.com/site/CliquePercComp). All other

Table I
METHOD PARAMETERS USED FOR PROTEIN COMPLEX PREDICTION

Clique Percolation (CPM)	Run with $k = 3$ and $k = 4$
MCODE	VWP= 0.2, fluff disabled
DPClus	$d_{\text{in}} = 0.9$, $cp_{\text{in}} = 0.5$
IPCA	$d = 2$, $T_{in} = 0.5$
Graph Entropy	N/A
CoAch	$d = 0.7$, $t = 0.225$

algorithms, weighted and unweighted, were implemented in the Python programming language.

For fairness in comparison, we have evaluated each method using the default parameters given in the software originally published by the method's authors. A list of these parameters is given in Table I. The clique percolation method (referred to as CPM in figures and tables) only uses a parameter of clique size, k, which we evaluated at both $k = 3$ and $k = 4$. The graph entropy method has no parameters. The same parameters were used for both unweighted and weighted tests, as we have no *a priori* knowledge of the effect of parameter values for the weighted implementation.

The ground-truth protein complexes of *S. cerevisiae* were obtained by combining two data sets determined by small-scale and high-throughput experiments, respectively, in a previous study [14]. All proteins in this ground-truth data set are present in the BioGRID PPI network after removal of zero-weighted edges.

B. Comparison of Cluster Features

In this section, we consider the features of predicted complexes – namely, modularity and density – generated by each of the six methods. We assess the improvement of weighted methods over their original unweighted implementations based on these features. Mainly, we seek to answer how well, at face value, the predicted complexes match our expectations about the protein-protein interactions observed in actual complexes. In general, we observe that enhancing methods with weighted network analysis appears to produce predicted complexes more similar to expectations, with the exception of MCODE.

Tables II and III give statistics about the clusters returned by each method, weighted and unweighted. For each prediction method, we give the number of predicted complexes, the average size and standard deviation in size of these complexes, the average density, and the number of proteins covered by all predicted complexes. Density is reported as unweighted, drawing from the original network with zero-weighted edges removed, and weighted, drawn from the collective edge weights of a predicted complex generated using either simICND or simICNP. The protein coverage for a method is the number of proteins in the PPI network that are involved in any predicted protein complex. Note that, aside from weighted density, the results for the unweighted prediction methods are identical for both simICND and simICNP. This is because the zero-weighted edges removed from the PPI network are conserved between the two semantic similarity methods.

We present ground-truth cluster features as a simple benchmark for expected prediction outcomes. In ground-truth data, there exist several hundred complexes covering roughly half (2360 out of 4318) of the proteins in our PPI network. On average, these complexes consist of five proteins, and they tend to have moderate density (slightly more than 0.7 unweighted and slightly more than 0.5 for weighted).

Regarding the modularity of complexes generated by each method, we note that the number of complexes predicted by unweighted methods varies quite considerably (see Figure 1). Unweighted IPCA, for one, predicts 2711 different complexes and covers every protein. On the other hand, some methods, such as clique percolation, predicted a very small amount of clusters but covered more than a third of all proteins. Weighted implementations tended toward cluster sizes closer to those found in the ground-truth data set. Additionally, the protein coverage seems to balance away from the extremes when weighted analysis is applied. On the low end, graph entropy coverage in simICND increases from 678 proteins to 1791; DPClus decreases from 3183 to 2893. Similar trends can be seen in simICNP data. This seems to imply that semantic similarity weighting is effective in augmenting the output of the methods analyzed. That is, it is able to attenuate methods that tend to over- or underestimate protein involvement, and it is able to increase complex modularity in terms of number and size.

Cluster density between methods is shown for unweighted density in Figure 2, and for weighted density in Figure 3. For clique percolation and graph entropy, we see small decreases in unweighted graph density, although the values are still quite high. IPCA and CoAch show increases in unweighted density, and CoAch's predicted complexes average almost exactly the same as the ground-truth data. Interestingly, DPClus shows a sharp drop in unweighted density for simICND, but has an average of almost complete connectivity (density of 1.0) for simICNP. MCODE shows very poor unweighted density.

All methods except for MCODE showed increase in average weighted density. In the unweighted methods, semantic density was fairly consistent, between 0.2 and 0.4. For the weighted methods, DPClus again showed a substantial increase for simICNP. From the results, we see clique percolation and IPCA tend toward high-weighted-density complexes when a weighted network is applied. CoAch keeps a low level of semantic coherence among its predicted complexes. The weighted version of graph entropy and DPClus, on the other hand, has an average weighted density very close to that of the ground-truth data set, indicating that the method tends to keep semantic groupings at a rate

Table II
COMPARISON OF CLUSTER FEATURES FOR PREDICTED PROTEIN COMPLEXES BETWEEN UNWEIGHTED AND WEIGHTED METHODS (SIMICND)

Method	Number of Complexes	Avg. Size (St. Dev.)	Max Size	Avg. Unweighted Density	Avg. Weighted Density	Proteins Covered
CPM(k=3)	34	115.05 (641.81)	3802	0.924	0.386	3839
CPM(k=3) Weighted	240	6.93 (13.30)	173	0.890	0.870	1595
CPM(k=4)	130	27.76 (263.19)	3017	0.953	0.316	3138
CPM(k=4) Weighted	122	10.16 (17.48)	169	0.900	0.861	1169
MCODE	59	27.66 (43.45)	180	0.604	0.408	1632
MCODE Weighted	67	23.83 (43.42)	343	0.463	0.282	1597
DPClus	887	4.23 (4.07)	56	0.983	0.392	3183
DPClus Weighted	556	6.03 (6.93)	78	0.745	0.551	2893
IPCA	2711	39.29 (32.63)	104	0.644	0.214	4318
IPCA Weighted	1419	9.16 (12.17)	66	0.852	0.726	3340
Graph Entropy	302	3.32 (4.07)	33	0.941	0.463	678
Graph Entropy Weighted	737	4.32 (8.20)	111	0.885	0.562	1791
CoAch	1818	32.04 (29.07)	207	0.604	0.204	3542
CoAch Weighted	1740	13.91 (17.06)	126	0.731	0.249	3215
Ground Truth	734	5.08 (9.39)	176	0.734	0.577	2360

Table III
COMPARISON OF CLUSTER FEATURES FOR PREDICTED PROTEIN COMPLEXES BETWEEN UNWEIGHTED AND WEIGHTED METHODS (SIMICND)

Method	Number of Complexes	Avg. Size (St. Dev.)	Max Size	Avg. Unweighted Density	Avg. Weighted Density	Proteins Covered
CPM(k=3)	34	115.05 (641.81)	3802	0.924	0.371	3839
CPM(k=3) Weighted	211	7.80 (15.30)	180	0.875	0.826	1563
CPM(k=4)	130	27.76 (263.19)	3017	0.953	0.311	3138
CPM(k=4) Weighted	119	10.73 (17.88)	170	0.882	0.805	1169
MCODE	59	27.66 (43.45)	180	0.604	0.394	1632
MCODE Weighted	69	21.05 (26.62)	124	0.516	0.300	1453
DPClus	887	4.23 (4.07)	56	0.983	0.383	3183
DPClus Weighted	648	3.20 (2.54)	24	0.998	0.976	1967
IPCA	2711	39.29 (32.63)	104	0.644	0.209	4318
IPCA Weighted	1440	8.60 (11.51)	61	0.874	0.712	3306
Graph Entropy	302	3.32 (4.07)	33	0.941	0.442	678
Graph Entropy Weighted	671	4.31 (7.48)	86	0.897	0.579	1595
CoAch	1818	32.04 (29.07)	207	0.604	0.199	3542
CoAch Weighted	1717	13.64 (16.82)	133	0.732	0.245	3178
Ground Truth	734	5.08 (9.39)	176	0.734	0.553	2360

similar to known complexes.

Overall, it appears that weighted network analysis tends to reduce unweighted complex density toward that of the ground-truth data set, with the exception of DPClus for simICNP, and MCODE. Considering weighted density, the methods tend toward dense semantic clusterings, above that found in ground-truth complexes. This may be the result of the methods' original unweighted design, which was mean to account for the noisiness of PPI data by discovering as dense of regions in the PPI network as possible. Further research into weighted protein complex prediction might benefit in expanding methods that better match the semantic densities of known complexes, such as the method of graph entropy.

C. Predictive Efficacy

To evaluate the quality of predicted protein complexes made by the methods, we use the f-score statistic, defined as the the harmonic mean of both precision and recall:

$$f = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}.$$

Recall and precision are defined as:

$$\text{Recall} = \frac{|X \cap P|}{|P|}$$

$$\text{Precision} = \frac{|X \cap P|}{|X|}.$$

where X is a predicted protein complex and P is a ground-truth protein complex. We match each predicted protein complex to the corresponding ground-truth complex with the highest f-score.

Figure 4 gives a comparison of the average f-score for all complexes generated by a particular method. In every case, except for MCODE, we observed an increase in this value from an unweighted method to its weighted counterpart.

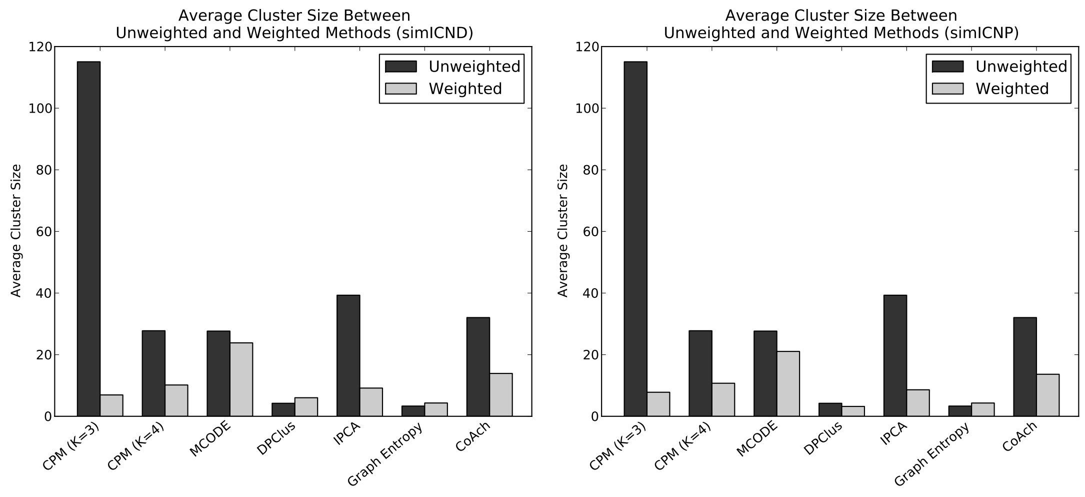


Figure 1. Average size of predicted protein complexes, per method.

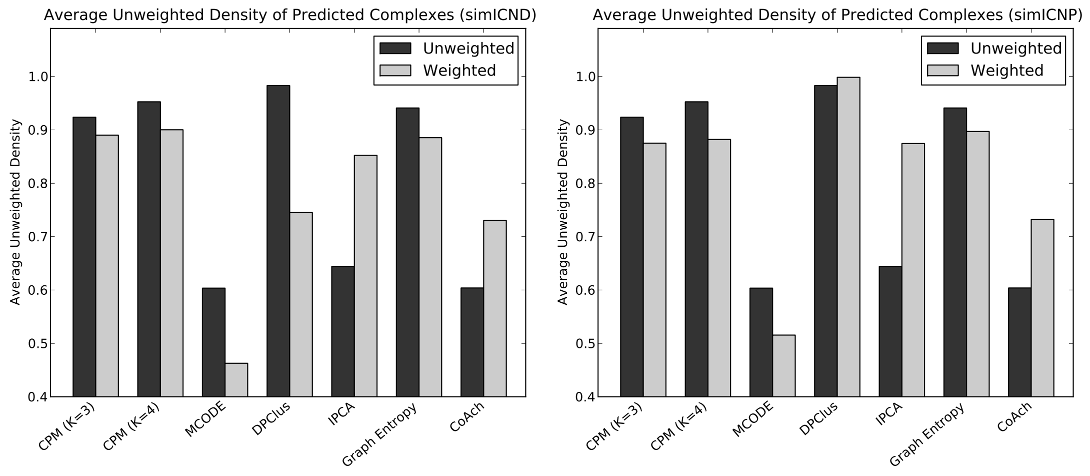


Figure 2. Average unweighted density of predicted protein complexes, per method.

We use the Mann-Whitney test to evaluate the difference in f-score distributions between the unweighted and weighted protein complex detection methods. The Mann-Whitney test is non-parametric test for comparing the distributions of two different groups [37]. It calculates a statistic, U, as the number of all possible pairs from each of two samples (say, X and Y) where $x_i < y_j$. U is formally given as

$$U = n_1 n_2 + \frac{1}{2} n_1 (n_1 + 1) - T,$$

where n_1 and n_2 are the sizes of the two samples, and T is the sum of the ranks in the smaller group when the values from both groups are collectively ranked from least to greatest. The ratio $\frac{U}{n_1 n_2}$ is the estimated probability that an observation from the first group will be less than an observation from the second.

When comparing the distribution of f-scores for unweighted and weighted methods, we apply the hypothesis that the distribution of values in the weighted version will be greater than the values of the unweighted version. As seen in tables IV and V, the null hypothesis is rejected for all methods with the exception of MCODE. Full charts of f-score distributions for every method can be found in Figure 6.

Table IV
Comparison of Weighted and Unweighted F-measure Distributions for Predicted Protein Complexes using the Mann-Whitney Test (simICND)

Method	Unweighted Mean	Weighted Mean	p-Value
CPM(k=3)	0.275	0.563	< 0.001
CPM(k=4)	0.352	0.669	< 0.001
MCODE	0.373	0.278	0.014
DPClus	0.342	0.392	< 0.001
IPCA	0.277	0.448	< 0.001
Graph Entropy	0.235	0.312	< 0.001
CoAch	0.280	0.328	< 0.001

In Figure 5, we show for each method the percentage of

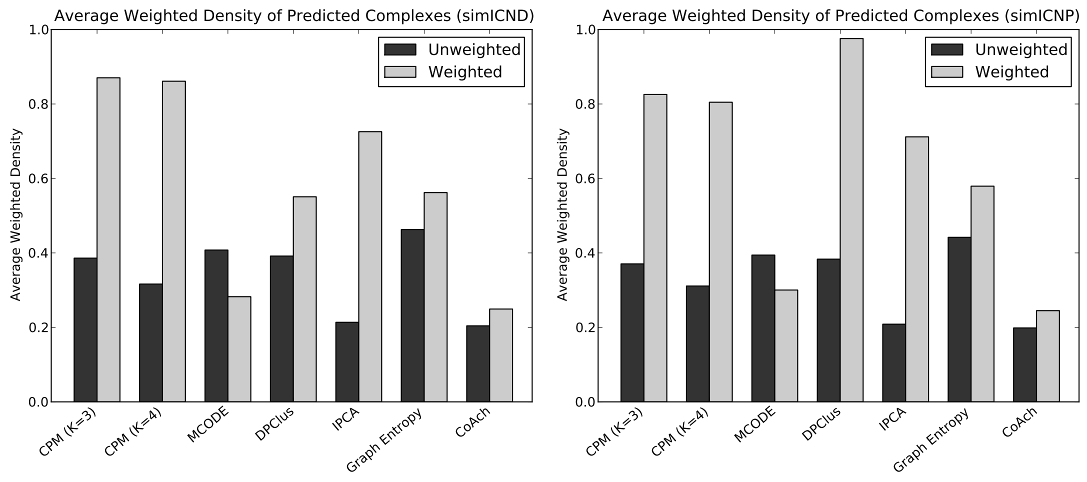


Figure 3. Average weighted density of predicted protein complexes, per method.

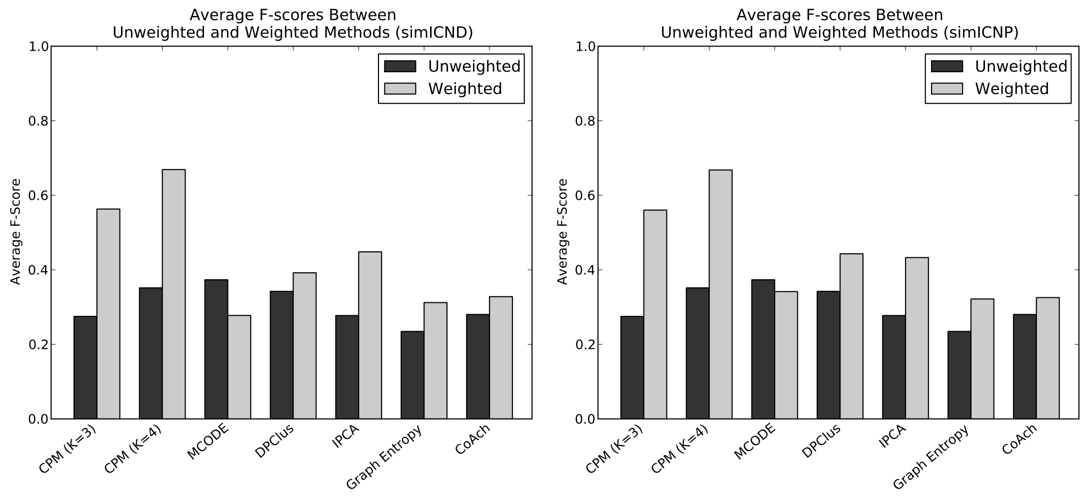


Figure 4. Average maximum f-score values for predicted protein complexes compared to ground-truth complexes.

Table V
COMPARISON OF WEIGHTED AND UNWEIGHTED F-MEASURE DISTRIBUTIONS FOR PREDICTED PROTEIN COMPLEXES USING THE MANN-WHITNEY TEST (SIMICNP)

Method	Unweighted Mean	Weighted Mean	p-Value
CPM(k=3)	0.275	0.560	< 0.001
CPM(k=4)	0.352	0.668	< 0.001
MCODE	0.373	0.341	0.325
DPClus	0.342	0.443	< 0.001
IPCA	0.277	0.433	< 0.001
Graph Entropy	0.235	0.322	< 0.001
CoAch	0.280	0.326	< 0.001

clusters having an f-score of one. These are predicted complexes that match a known ground-truth complex exactly. We see substantial increases for the clique percolation method and IPCA, and interestingly, DPClus shows a substantial increase, but only for the simICNP graph. These results indicate that weighted clique percolation, while returning relatively few clusters compared to other methods

We see from the distribution of f-scores for each method that weighted network analysis substantially enhances the predictive efficacy of protein complex detection algorithms, with the exception of MCODE. The low performance of MCODE may be linked to the low number of predicted complexes the method tends to output; its weighted output has by far the lowest number of complexes of any method. Interestingly, MCODE had the highest average f-score of any unweighted method, but again, this is most likely tied to the small output of the algorithm. These findings agree with the results of Li *et al.* [4] in their survey of protein complex prediction methods.

Regarding the remaining algorithms, clique percolation showed substantial improvements in cluster size and f-score alike. While the method returned a relatively small amount of complexes, the complexes it did find scored very well in both precision and recall when compared to known

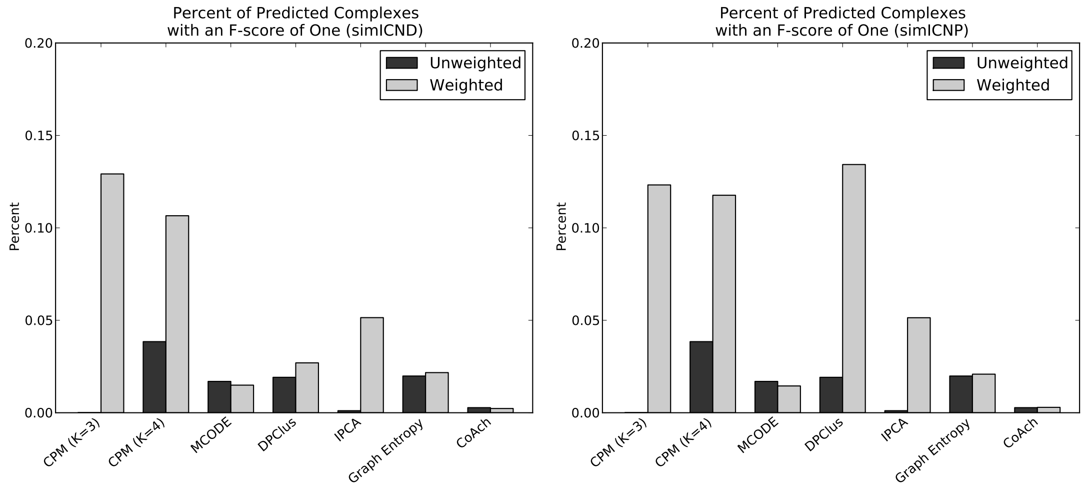


Figure 5. Percent of predicted protein complexes exactly matching a known ground-truth complex, per method.

complexes. The DPClus and graph entropy methods both showed an increase in f-score values while adopting cluster size and protein coverage scores closer to those found in the ground-truth data set. IPCA and CoAch became more selective in the clusters and proteins chosen, resulting in significantly higher f-scores, as well.

IV. Conclusion

We reviewed six computational methods that were designed to predict protein complexes from PPI networks. We modified these approaches to incorporate weighted edges, based on two different semantic similarity scores using GO data. These approaches and their modifications were compared for predictive efficacy and cluster features.

When semantic similarity weighting is applied to existing methods, the cluster size and unweighted density of predicted protein complexes trends towards those of ground-truth clusters. Average weighted density of predicted complexes tends to favor higher semantic coherence among clustered proteins than seen in known complexes. In ground-truth cluster matching, the f-score of predicted clusters and percent of f-scores equaling one increased for weighted implementations. From our experimental results, we conclude that integrating knowledge of semantic similarity between proteins in PPI networks improves the quality of predicted protein complexes when matched against known protein complexes.

One limitation in analyzing discovered complexes is determining whether a low-scoring complex is a false positive or a novel protein complex. This is evident by the existence of several to many predicted protein complexes with f-scores close to or at zero for every method. This was especially seen in the weighted method implementations. It appears that these methods – particularly with their weighted modifications – have reached a level of feasible predictive accuracy for undiscovered protein complexes. Future research in this field may include lab studies to test the accuracy of significantly low-scoring predicted complexes.

V. Acknowledgements

The authors wish to thank Li and Bader for their correspondence. We would also like to thank the authors of Bloom filter clique perculation, MCODE, and IPCA for making their source available online.

References

[1] Yu, H., et al., "High-quality binary protein interaction map of the yeast interactome network," *Science*, vol. 322, pp. 104–110, 2008.

[2] Venkatesan, K., et al., "An empirical framework for binary interactome mapping," *Nature Method*, vol. 6, no. 1, pp. 83–90, 2009.

[3] Spirin, V. and Mirny, L.A., "Protein complexes and functional modules in molecular networks," *Proc. Natl. Acad. Sci. USA*, vol. 100, no. 21, pp. 12 123–12 128, 2003.

[4] Li, X., Wu, M., Kwoh, C.-K. and Ng, S.-K., "Computational approaches for detecting protein complexes from protein interaction networks: a survey," *BMC Genomics*, vol. 11, no. Suppl 1, p. S3, 2010.

[5] Stark, C., et al., "The BioGRID interaction database: 2011 update," *Nucleic Acids Research*, vol. 39, pp. D698–D704, 2011.

[6] The Gene Ontology Consortium, "The Gene Ontology: enhancements for 2011," *Nucleic Acids Research*, vol. 40, pp. D559–D564, 2012.

[7] Cho, Y.-R., Chiam, T.C. and Lu, Y., "M-Finder: Functional association mining from protein interaction networks weighted by semantic similarity," in *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2012, pp. 95–100.

[8] Palla, G., Derenyi, I., Farkas, I. and Vicsek, T., "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814–818, 2005.

[9] Bader, G.D. and Hogue, C.W., "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 4, p. 2, 2003.

[10] Altaf-Ul-Amin, M., Shinbo, Y., Mihara, K., Kurokawa, K. and Kanaya, S., "Development and implementation of an algorithm for detection of protein complexes in large interaction networks," *BMC Bioinformatics*, vol. 7, p. 207, 2006.

[11] Li, M., Chen, J., Wang, J., Hu, B. and Chen, G., "Modifying the DPClus algorithm for identifying protein complexes based on new topological structures," *BMC Bioinformatics*, vol. 9, p. 398, 2008.

[12] E. C. Kenley and Y.-R. Cho, "Detecting protein complexes and functional modules from protein interaction networks: A graph entropy approach," *PROTEOMICS*, vol. 11, no. 19, pp. 3835–3844, 2011.

[13] Wu, M., Li, X., Kwoh, C.-K. and Ng, S.-K., "A core-attachment based method to detect protein complexes in PPI networks," *BMC Bioinformatics*, vol. 10, p. 169, 2009.

[14] Pu, S., Wong, J., Turner, B., Cho, E. and Wodak, S.J., "Up-to-date catalogues of yeast protein complexes," *Nucleic Acids Research*, vol. 37, no. 3, pp. 825–831, 2009.

[15] Derenyi, I., Palla, G. and Vicsek, T., "Clique percolation in random networks," *Physical Review Letters*, vol. 94, p. 160202, 2005.

[16] Adamcsek, B., Palla, G., Farkas, I.J., Derenyi, I. and Vicsek, T., "CFinder: locating cliques and overlapping modules in biological networks," *Bioinformatics*, vol. 22, no. 8, pp. 1021–1023, 2006.

[17] J. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki, "Sequential algorithm for fast clique percolation," *Physical Review E*, vol. 78, no. 2, p. 026109, 2008.

[18] F. Reid, A. F. McDaid, and N. J. Hurley, "Percolation computation in complex networks," *CoRR*, vol. abs/1205.0038, 2012.

[19] I. Farkas, D. Ábel, G. Palla, and T. Vicsek, "Weighted network modules," *New Journal of Physics*, vol. 9, no. 6, p. 180, 2007.

[20] J. Onnela, J. Saramäki, J. Kertész, and K. Kaski, "Intensity and coherence of motifs in weighted complex networks," *Physical Review E*, vol. 71, no. 6, p. 065103, 2005.

[21] Bard, J.B.L. and Rhee, S.Y., "Ontologies in biology: design, applications and future challenges," *Nature Reviews: Genetics*, vol. 5, pp. 213–222, 2004.

[22] The Gene Ontology Consortium, "The Gene Ontology in 2010: extensions and refinements," *Nucleic Acids Research*, vol. 38, pp. D331–D335, 2010.

[23] Pedersen, T., Pakhomov, S.V.S., Patwardhan, S. and Chute, C.G., "Measures of semantic similarity and relatedness in the biomedical domain," *Journal of Biomedical Informatics*, vol. 40, pp. 288–299, 2007.

[24] Pesquita, C., Faria, D., Falcao, A.O., Lord, P. and Couto, F.M., "Semantic similarity in biomedical ontologies," *PLoS Computational Biology*, vol. 5, no. 7, p. e1000443, 2009.

[25] Wang, J., Zhou, X., Zhu, J., Zhou, C. and Guo, Z., "Revealing and avoiding bias in semantic similarity scores for protein pairs," *BMC Bioinformatics*, vol. 11, p. 290, 2010.

[26] Guzzi, P.H., Mina, M., Guerra, C. and Cannataro, M., "Semantic similarity analysis of protein data: assessment with biological features and issues," *Briefings in Bioinformatics*, vol. 13, no. 5, 2012.

[27] Guo, X., Liu, R., Shriver, C.D., Hu, H. and Liebman, M.N., "Assessing semantic similarity measures for the characterization of human regulatory pathways," *Bioinformatics*, vol. 22, no. 8, pp. 967–973, 2006.

[28] Wu, Z. and Palmer, M., "Verb semantics and lexical selection," in *Proceedings of 32th Annual Meeting of the Association for Computational Linguistics*, 1994, pp. 133–138.

[29] Resnik, P., "Using information content to evaluate semantic similarity in a taxonomy," in *Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1995, pp. 448–453.

[30] Lin, D., "An information-theoretic definition of similarity," in *Proceedings of 15th International Conference on Machine Learning (ICML)*, 1998, pp. 296–304.

[31] Jiang, J.J. and Conrath, D.W., "Semantic similarity based on corpus statistics and lexical taxonomy," in *Proceedings of 10th International Conference on Research in Computational Linguistics*, 1997.

[32] Mistry, M. and Pavlidis, P., "Gene Ontology term overlap as a measure of gene functional similarity," *BMC Bioinformatics*, vol. 9, p. 327, 2008.

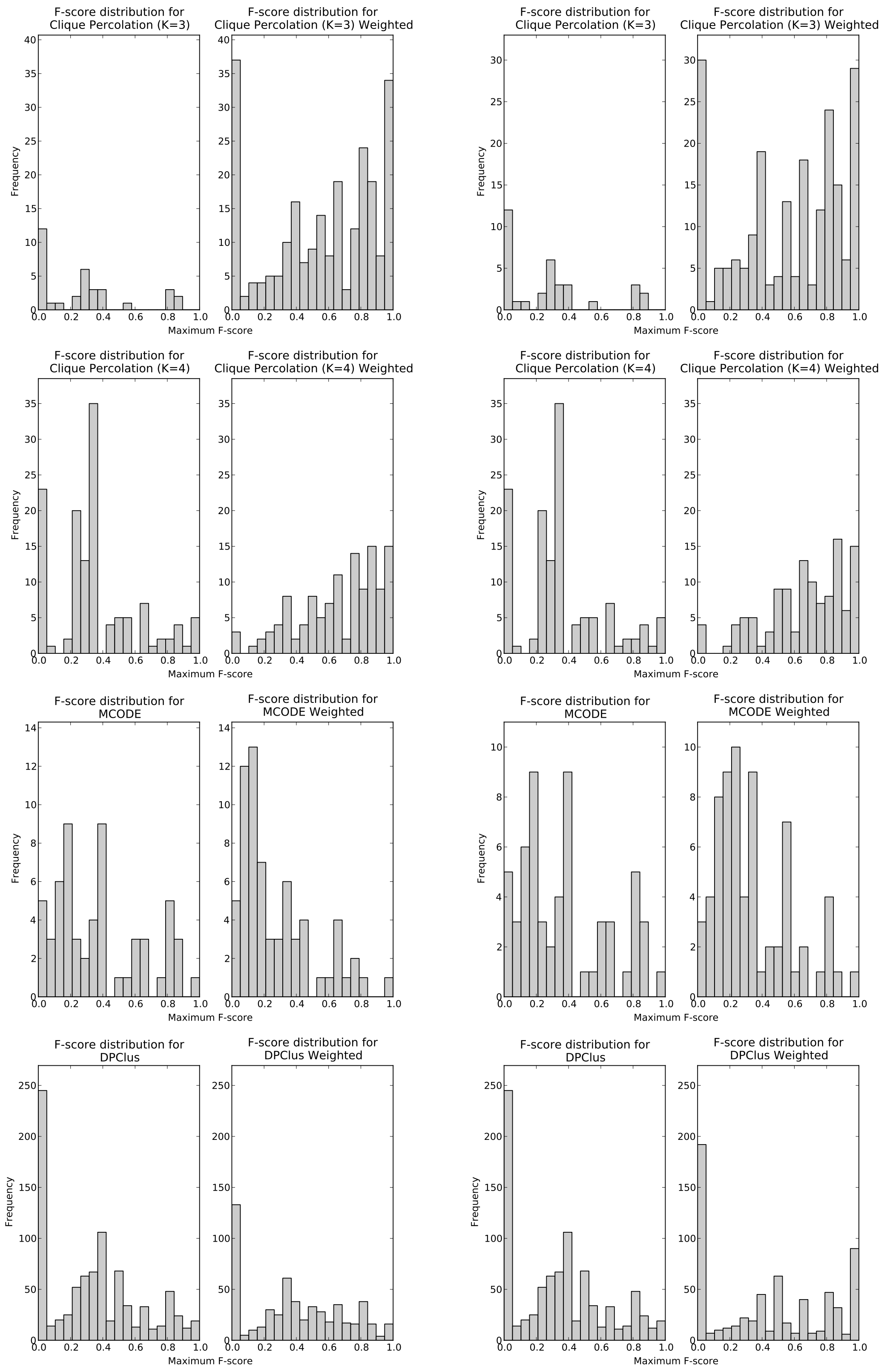
[33] Wang, J.Z., Du, Z., Payattakool, R., Yu, P.S. and Chen, C.-F., "A new method to measure the semantic similarity of GO terms," *Bioinformatics*, vol. 23, no. 10, 2007.

[34] Pesquita, C., Faria, D., Bastos, H., Ferreira, A.E.N., Falcao, A.O. and Couto, F.M., "Metrics for GO based protein semantic similarity: a systematic evaluation," *BMC Bioinformatics*, vol. 9, no. Suppl 5, p. S4, 2008.

[35] Benabderrahmane, S., Smail-Tabbone, M., Poch, O., Napoli, A. and Devignes, M.-D., "IntelliGO: a new vector-based semantic similarity measure including annotation origin," *BMC Bioinformatics*, vol. 11, p. 588, 2010.

[36] Jain, S. and Bader, G.D., "An improved method for scoring protein-protein interactions using semantic similarity within the gene ontology," *BMC Bioinformatics*, vol. 11, p. 562, 2010.

[37] D. Altman, *Practical Statistics For Medical Research*, ser. Texts in Statistical Science Series. Chapman and Hall, 1991.


F-score distribution for Clique Percolation (K=3)
F-score distribution for Clique Percolation (K=3) Weighted
F-score distribution for Clique Percolation (K=4)
F-score distribution for Clique Percolation (K=4) Weighted
F-score distribution for MCODE
F-score distribution for MCODE Weighted
F-score distribution for DPClus
F-score distribution for DPClus Weighted
Frequency
Maximum F-score

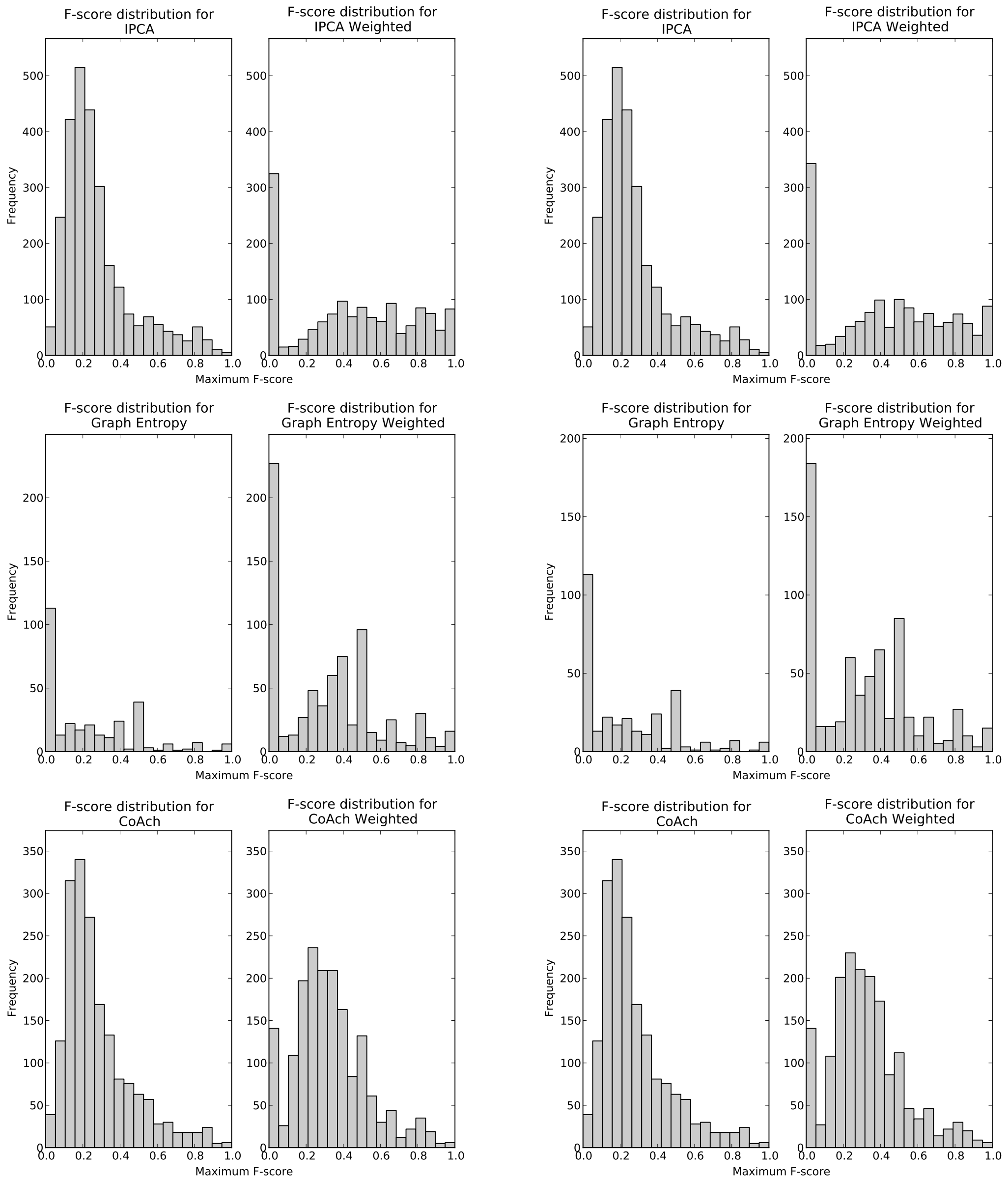


Figure 6. Side-by-side comparisons of maximum f-score distributions for weighted and unweighted methods. The left column contains results based on simICND, and the right column shows results based on simICNP.