Protein function prediction via analysis of interactomes

Elena Nabieva

Mona Singh

Department of Computer Science & Lewis-Sigler Institute for Integrative Genomics

January 22, 2008

1 Introduction

Genome sequencing efforts have resulted in an explosion of organisms whose entire protein complements have been determined. Nevertheless, for many proteins, little is known beyond their sequences, and for the typical proteome, between one-third and one-half of its proteins remain uncharacterized. As a result, a major challenge in modern biology is to develop methods for determining protein function at the genomic scale.

Computational methods to assign protein function have traditionally relied on identifying sequence similarity to proteins of known function. In recent years, however, other computational methods for predicting protein function have been developed (review, [33]). Many of these non-homology based methods still utilize sequence information, but can predict that two proteins share a function even when they have no sequence similarity. For example, in gene fusion methods [29,51], two proteins are believed to be related functionally if they appear as parts of a single protein in some other organism. Phylogenetic profiles [32,55] predict proteins to be functionally related if they have similar patterns of occurrences across multiple genomes. Genomic context methods [22, 54] predict functional coupling between proteins if they tend to be contiguous in several genomes.

Increasingly, computational techniques for predicting protein function have analyzed data resulting from new high-throughput technologies. While there is a fascinating array of new functional genomics technologies that have enabled prediction of protein function, in this chapter we examine a family of methods that are based on analyzing large-scale protein-protein interaction data. Currently, several types of protein interactions have been determined via high-throughput experimental technologies. These include interactions between proteins that interact physically, that participate

in a synthetic lethal or epistatic relationship, that are coexpressed, or where one phosphorylates or regulates another (review, [77]). Together, these interactions comprise the interactome and can be represented as networks or graphs, where interactions are undirected in the case of symmetric interactions, and directed otherwise.

Here, we focus primarily on predicting protein function via analysis of networks comprised of physical interactions. Most of these methods are based on the principle of *guilt-by-association*, where proteins are annotated by transferring the functions of the proteins with which they interact. The methods differ in whether they use local or global properties of the interactome in annotating proteins, in which particular topological features of the interactome they utilize, in whether they rely on first identifying tight clusters of proteins within the interactome before transferring annotations, and in whether they use guilt-by-association explicitly or employ some other similarity measure. While the focus of this chapter is on protein-protein physical interaction networks, it is often straightforward to apply the same methods to other types of networks. However, as the underlying topological features of these networks may differ, the methods may perform quite differently on them. We refer the reader to other reviews [2, 64] for alternative viewpoints that additionally consider function prediction methods that integrate physical interaction networks with network data derived from other experimental sources.

2 Further background

Physical interaction networks Large-scale physical interaction networks for several organisms have been obtained via two-hybrid experiments, where an interaction between a pair of proteins is determined via transcriptional activation in yeast [30]. An alternative high-throughput technology determines interactions of proteins via affinity purification of the target protein followed by mass spectrometry identification of the associated proteins (review, [9]). These two types of experiments are the most commonly used approaches for large-scale determination of physical interactions and have uncovered tens of thousands of interactions. However, they do impose certain features on the data that may be less than ideal. The yeast two-hybrid method may discover interactions that do not take place under physiological conditions and may miss interactions that do. The pull-down methods do not specify if the interactions inferred for a target protein are direct or are instead mediated through other associated proteins. Moreover, as with all experiments, especially high-throughput ones, a certain amount of noise is present in the results; this amount may differ between

different experiments and between subsets of interactions found by the same experiment. To some extent, this noise can be handled computationally by incorporating an assessment of interaction reliability into the computational approach (see Section 3). It is worth noting as well that the interactomes determined to date are incomplete, and that comparisons between existing data sets for the same organism reveal only partial overlap; the latter is due both to noise in the data as well as different sets of proteins under consideration.

Protein function Protein function is a broad concept that has different meanings depending on context. In computational settings, protein function is typically described via terms from one of several controlled vocabularies. Because of the differing degrees of specificity with which protein function can be described, these controlled vocabularies are usually arranged as hierarchies or directed acyclic graphs that relate the different terms to each other. The Gene Ontology (GO) [5] is the most prevalent of such controlled vocabulary systems; it classifies protein function into three separate categories, each of which consists of a set of terms that may be related to each other via is-a or part-of relations; these relations can be represented as a directed acyclic graph. Protein function in the usual sense is described by two of the categories, molecular function and biological process. The molecular function of a protein describes its biochemical activity, whereas its biological process specifies the role it plays in the cell, or the pathway in which it participates. Additionally, GO has a cellular component category which describes the places where the protein is found. These views of protein function are largely orthogonal: for example, proteins with the same molecular function can play a role in different pathways, and a pathway is built of proteins of various molecular functions. This distinction affects which methods are the most applicable for computational prediction of protein function of each type. Because molecular function corresponds to the intrinsic features of the protein (e.g., its catalytic activity), it is often predicted based on sequence or structural similarity to proteins of known function. Biological processes, on the other hand, are fundamentally collaborative; therefore, it is natural to predict them based on a protein's interaction partners. In this chapter, when we refer to a protein's function, we will typically mean its biological process, though network analysis of interactomes can also be useful for predicting a protein's cellular component; for example, several of the clustering methods reviewed here focus as much on predicting membership within protein complexes (which are described by cellular component annotations) as on predicting biological processes.

Mathematical formulation It is natural to represent the collection of protein physical interactions discovered for an organism as an undirected graph or network, where the vertices represent proteins and the edges connect vertices whose corresponding proteins interact. Each vertex is then labeled with zero or more controlled vocabulary terms corresponding to the protein's function(s). The terms used as labels may furthermore participate in a relation described by a system like the Gene Ontology. The function prediction problem then becomes the task of assigning labels to all vertices in a network. This labeled graph representation makes the function prediction problem amenable to the wealth of techniques developed in the graph theory and network analysis communities. For example, the idea of guilt-by-association, which is used by most approaches, turns the problem of function prediction into the problem of identifying (possibly overlapping) regions in the network that participate in the same biological process (i.e., should be assigned the same vertex label). Broadly speaking, most of the methods used for the network-based functional annotation utilize and extend well-understood concepts from graph theory, graphical models and/or clustering.

Notation More formally, a protein-protein interaction network is represented as a graph $G = (V, E)$, where there is a vertex $v \in V$ for each protein, and an edge $(u, v) \in E$ between two vertices u and v if the corresponding proteins interact. Since we are considering physical interactions between proteins, these edges are undirected. Throughout the chapter, we ignore self-interactions. Let N denote the number of proteins in the network. The network can also be represented by its $N \times N$ adjacency matrix A, where $A_{uv} = 1$ if $(u, v) \in E$ and 0 otherwise. Let $\mathcal{F}$ be the set of possible protein functional annotations. Each protein may be annotated with one or more annotations from $\mathcal{F}$. That is, each vertex $v \in V$ may have a set of labels associated with it. The edges in the network may be weighted; typically the weight $w_{u,v}$ on the edge between u and v reflects how confident we are of the interaction between u and v. If each interaction given in the network is considered equally trustworthy, the network may be considered unweighted or with unit-weighted edges.

Many approaches discussed below utilize the "neighborhood" of a protein. Let $\mathcal{N}_r(u)$ denote the neighborhood of protein u within radius r; that is, $\mathcal{N}_r(u)$ is the set of proteins where each protein has some path in the network to u that is made up of at most r edges. Then $\mathcal{N}_0(u)$ consists of protein u, $\mathcal{N}_1(u)$ consists of protein u and all proteins that interact with u, $\mathcal{N}_2(u)$ consists of the proteins in $\mathcal{N}_1(u)$ along with all proteins that interact with any of the proteins in $\mathcal{N}_1(u)$, and so on. Note that the number of interactions of a protein u is given by $|\mathcal{N}_1(u)| - 1$, since self-interactions are not considered.

3 Incorporating interaction reliability

All methods for predicting protein function based on interaction networks face the issue of data quality, as it is well known that high-throughput physical interaction data are noisy, and that different experimental data sets have varying reliability, even if they are based on the same underlying technology (e.g., see [24,66,74]). A common practice to address the issue of noise is to include edge weights that are chosen to reflect the reliability of interactions. Here, we review a simple scheme for assessing physical interaction reliability [53], that is essentially the same as the ones used in several approaches for the more general problem of data integration [41,73].

For each experimental source i (e.g., each high-throughput experiment may be considered one source, and the collection of all small-scale experiments may be considered as a single different source), let r_i denote the probability that an interaction observed in this experiment is a true physical interaction. Assuming that the observations and sources of error are independent for each experimental source, one can estimate the probability of a physical interaction between proteins u and v as:

$$1 - \Pi_i(1 - r_i),$$

where the product is taken over all experiments i which observe an interaction between u and v. This estimate can then be used as the weight $w_{u,v}$ of the edge between u and v. If r_i is chosen to be identical for all experimental sources, this approach simply gives higher reliability to physical interactions that have been observed multiple times. A more meaningful approach is to estimate r_i for each experimental source i by, for example, computing the fraction of interactions coming from that source that connect proteins with a known shared function. It has been shown that a wide range of network analysis algorithms perform better in predicting protein function when utilizing this scheme for assessing interaction reliability than when considering all interactions as equally likely [18, 53]. There are other alternatives for estimating data set reliability. For example, it is common for high-throughput experimental publications to report, along with data, some measure of reliability for each reported interaction; this measure may be as simple as the number of times an interaction has been observed or may be based on more sophisticated schemes (e.g., [34]). Regardless of the specific method used to assess the reliability of an interaction, the importance of treating different data sources separately has been demonstrated [67].

For well-studied organisms, the reliability of a physical interaction may also be estimated utilizing data integration schemes that attempt to combine many different types of data (e.g., ex-

pression, localization and physical and genetic interaction) in order to functionally link proteins (e.g., [40, 48, 68, 73]). Each link is associated with a weight that represents the probability, or some other confidence measure, that the two corresponding proteins are functionally related. Physical interaction reliabilities may be justifiably estimated using functional linkage scores since a higher functional similarity between two proteins suggests that the observed interaction is more likely to be true. More generally, weighted networks derived via data integration techniques can themselves be used for protein function prediction. Note however that though the problems are closely related, predicting functional linkages is not the same as predicting the function of a protein, as a protein can be linked with varying levels of confidence to several proteins with multiple biological process annotations; some method or rule, such as one of those reviewed here, is still necessary to decide which annotations are transferred.

4 Algorithms

A wide range of methods have been developed for analysing protein-protein interaction networks in order to predict protein function. In the discussion below, we review some of these and categorize them based upon their underlying algorithmic ideas as well as upon the extent to which they utilize network structure. The approaches are also briefly outlined in Table 1.

4.1 Neighborhood approaches

The assumption of guilt-by-association naturally gives rise to a prediction method based on majority vote that assigns to each protein the biological process that is most frequent among its direct interactions [63]. In this case, the score for assigning to a protein u a particular annotation a could be the number of proteins that u interacts with that are annotated with a; alternatively, the score may be computed as the fraction of u's interactions that have annotation a. In the case of weighted interaction networks, a weighted sum can be used instead. This majority or neighborhood-counting method is limited in that it only uses local neighborhood information and takes no advantage of more global properties of the network; it also has limited efficiency for poorly annotated proteomes. Subsequent graph-theoretic approaches have attempted to generalize this principle to consider linkages beyond the immediate neighbors in the interaction graph, both to provide a systematic framework for analyzing the entire interactome as well as to make predictions for proteins with no annotated interaction partners.

A simple way to extend the majority approach is to look at all proteins within a neighborhood of specified radius and use the most over-represented functional annotation [38] as the prediction for the protein of interest. That is, for each protein u and a fixed radius r, this neighborhood approach considers all proteins in $\mathcal{N}_r(u)$ and then for each function, computes a score based on the χ^2 test. In particular, the score is computed as $\frac{(f-e)^2}{e}$, where f is the number of proteins within the neighborhood having the function under consideration and e is the number of proteins expected to have that function within the neighborhood, given the frequency of the function in the entire network. The function with the highest χ^2 score is assigned to the protein. With radius one, this approach is similar to the simpler majority approach; note, however, that if two functions annotate the same number of a protein's direct neighbors, the neighborhood approach favors the one that annotates fewer proteins in the entire interactome. While this approach moves beyond direct neighbors, it does not consider the network topology within the local neighborhood. For example, Figure 1 shows an interaction network where proteins u and v have the same count for each annotation within radius two; thus the neighborhood approach treats these proteins equivalently when considering a radius of two, despite the fact the evidence for protein u having the annotation depicted by the color black is much stronger than it is for protein v. Perhaps because the method completely ignores network topology within neighborhoods, its biological process predictions are best when considering neighborhoods of radius one [38]. Moreover, even the radius-one predictions perform worse than majority vote [53], suggesting that the decision to penalize more frequent candidate functions may not be optimal; in fact, some of the methods we consider later in the chapter, such as those based on Markov network techniques, use a function's *a priori* frequency in the opposite way. A recent extension of the neighborhood approach attempts to include proteins at radius two while additionally utilizing some information about network topology by assigning weights to each protein in the neighborhood by favoring the number of shared interactors it has with the protein being annotated, and then scoring each function based on its weighted frequency in the neighborhood [18].

4.2 Graph cuts

One systematic approach to consider the entire network and its annotations in a way that uses information about network connectivity is to utilize the concept of graph cuts. A k-cut is defined as a partition of the vertices of a graph into k sets, and the cost of the cut is sum of the weights of the edges between vertices in different sets. This framework provides a natural appli-

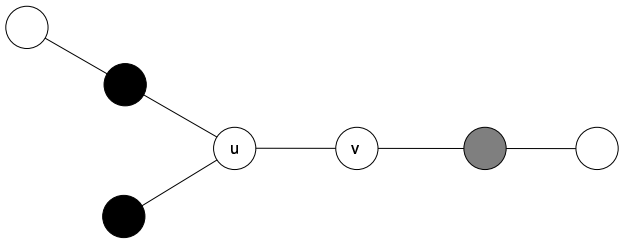

Figure 1: A protein interaction graph annotated with two functions, depicted using black and grey. White nodes correspond to proteins that do not have biological process annotations. When annotating proteins u and v, a neighborhood approach [38] with radius two would make the same prediction, even though the evidence in favor of predicting the function depicted by black is much stronger for protein u than for protein v, and vice versa for the function depicted by grey.

cation of the guilt-by-association assumption at the full-network scale, as the cut problem can be formulated so as to annotate proteins in a way that minimizes the weighted number of the edges that violate this assumption (i.e., connect proteins having different function). Several cut-based methods for function prediction have been developed [42, 53, 72]; they can either consider functions simultaneously [53, 72], or just one at a time [42].

If all functions are considered at the same time, the function prediction problem is a generalization of the computationally difficult minimum multiway k-cut problem [21], where the goal is to partition a graph in such a way that each of the k terminal nodes belongs to a different subset of the partition and such that the weighted number of edges that are "cut" in the process is minimized. In the more general version of the multiway-cut problem relevant to the protein functional annotation problem, the goal is to assign a function to all unannotated nodes so as to minimize the sum of the weights of the edges joining nodes that have no (assigned or previously known) function in common (i.e., these edges define the cuts). Formally, the problem in the case of function prediction can be stated as minimizing

$$-\sum_{u,v} J_{uv}\delta(\sigma_u, \sigma_v) - \sum_{u} h_u(\sigma_u),$$

where σ_u is functional assignment to node u, $\delta(x, y) = 1$ if $x = y$ and 0 otherwise, J_{uv} is the adjacency matrix for unlabeled vertices, and $h_u(\sigma_u)$ is the number of classified neighbors of vertex u labeled with σ_u [72]. For the weighted version, J_{uv} and h_u can be easily modified to reflect edge

weights.

In the case where one function is considered at a time, each protein that is known to have that function is labeled as a "positive" and each protein that is known to have some function but not the one being considered is labeled as a "negative." The optimization problem in that case can be stated as minimizing

$$-\sum_{u}\sum_{v\neq u} w_{u,v}s_u s_v,$$

where $w_{u,v}$ is the weight of edge (u, v), and s_u is 1 if the vertex is labeled with the function being evaluated and -1 otherwise [42]. If the graph is unweighted, $w_{u,v}$ can be set uniformly to 1. It is straightforward to see that this is a basic minimum cut/maximum flow problem, and thus exact solutions are obtainable in polynomial time (e.g., see [19]).

Several techniques have been applied to solve these cut problems for interactomes. In the case where one function at a time is considered, a deterministic approximation algorithm has been applied to obtain a single solution per function [42]. In this application, a version is also considered where edges are assigned (positive) weights based on the correlation of the corresponding proteins' expression profiles. In subsequent work, this formulation has been solved exactly using a minimum cut algorithm [52]. In the case where multiple functions are considered at once, simulated annealing has been applied and solutions from several runs have been aggregated [72]. That is, the score of a function for a particular protein is given by the number of runs in which the simulated annealing solution annotates the protein with the function. The simulated annealing approach is a heuristic and thus does not guarantee an optimal solution to the underlying optimization problem. However, an integer linear programming (ILP) formulation for the generalized multiway-cut problem has also been proposed [53]. While ILP is computationally difficult from a theoretical point of view, in practice optimal solutions to this ILP, and thus the original optimization problem, can be readily obtained for existing physical interactomes using AMPL [31] and the commercial solver CPLEX [39].

An important shortcoming of the basic cut formulation is that it ignores distance in the network. For example, the network in Figure 2 has four minimum cuts of value one, and the cut criterion does not favor any one cut over the other. However, we expect proteins that are closer together in the network to have more similar biological process annotations than those that are further apart. Thus, in the network in Figure 2, we would want the proteins closer to the black node to be annotated with its function, and the proteins closer to the grey node to be annotated with its function. As suggested by [53], one may begin to address this problem in the cut-based framework

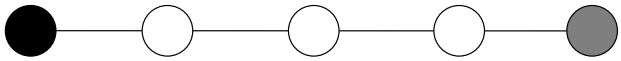

Figure 2: A protein interaction graph with two annotated functions, represented as black and grey nodes. White nodes do not have biological process annotations. There are four ways to annotate proteins so that only one edge is "cut". However, the second protein from the left is more likely to have the function depicted by the colored black than the second protein from the right. A single cut of the graph does not take into account such distance effects.

by considering the multiplicity of optimal solutions. If we find all optimal cuts for the graph in Figure 2, we observe that proteins closer to the black node are found more frequently in the same cut as the black node than in the same cut as the grey node. Thus, the set of all optimal solutions contains a sense of distance to annotated nodes. In the earlier simulated annealing approach proposed for this problem, information from multiple solutions is utilized [72]. If each run does indeed converge to an optimal solution, considering multiple runs amounts to sampling from the space of optimal solutions. The ILP can also be modified to find multiple solutions [53]. The score for a function for a protein is then the number of obtained solutions in which this function is assigned to the protein.

4.3 Flow-based methods

One attempt to overcome the cut-based methods' ignorance of distances in the network has been proposed based on another concept from computer science, namely, network flow [53]. Intuitively, network flow problems treat the graph as a collection of pipes having limited capacity (represented as weights), and pose the question of the maximum amount of liquid that can be sent from a specified *source* node to a specified *sink* node using those pipes. The network flow problem is dual to the notion of graph cut (e.g., see [19]), as the size of the minimum cut between the source and the sink turns out to be the limiting factor to maximum flow, and vice versa.

Network flow has been used as the inspiration for a simulation method for function prediction [53]. Informally, each protein of known functional annotation is an infinite "source" of "functional flow" that can be propagated to unannotated nodes, using the edges in the interaction graph as a conduit. Each protein has a "reservoir" which represents the amount of flow that the node can pass on to its neighbors at the next iteration, and each edge has a capacity (its weight) limiting the amount of flow that can pass through the edge in one iteration. Each iteration of the algorithm

updates the reservoirs using simple local rules, whereby flow only spreads from proteins with more filled reservoirs to those with less filled reservoirs, and a node pushes its flow to its neighbors proportionally to the capacities of the respective edges. The simulation is run for a fixed number of steps, and a functional score for each protein is obtained by summing the total amount of flow for that function that the protein has received over the course of the simulation. This method exploits network connectivity as multiple disjoint paths between functional sources and a protein results in more flow to the protein. It also incorporates a notion of distance in the network as the effect of each annotated protein on any other protein decreases with increasing distance between them: if the algorithm is run for d iterations, then a source's immediate neighbor in the graph receives d iterations worth of flow from the source, while a node that is two links away from the source receives $d-1$ iterations worth of flow, and so on. Similarly, the number of iterations for which the algorithm is run determines the maximum number of interactions that can separate a recipient node from a source in order for the flow to propagate from the source to the recipient. For the protein interaction context, a relatively small number of iterations has worked well in practice (e.g., less than half the diameter of the network). The reader is referred to [53] for the exact formulation of the functional flow algorithm.

In subsequent work, a similar deterministic flow-based simulation approach has also been applied for finding clusters in protein interaction networks [17].

4.4 Markov network-based methods

Cut-based methods for functional annotation have a more general probabilistic counterpart in methods based on Markov networks [23,26,49], and these formulations can more fully address some of the weaknesses of the cut-based methods. A Markov network, also known as a Markov random field, is an undirected graphical model that represents the joint probability distribution of a set of random variables. It is specified by an undirected graph where each vertex represents a random variable and each edge represents a dependency between two random variables, such that the state of any random variable is independent of all others given the states of its neighbors. The joint distribution represented by a Markov random field is computed by considering a potential function over each of its cliques. For N random variables X_i, the probability of an assignment of the states is given by:

$$\Pr(X_1 = x_1, \ldots, X_N = x_N) = \frac{1}{Z} e^{-\sum_k \Phi_k(X_{\{k\}})},$$

where k enumerates all cliques, Φ_k is the potential function associated with the k-th clique, $X_{\{k\}}$ gives the states of the k-th clique's random variables, and Z is a normalizing constant.

In applications to network-based function annotation, one function has been considered at a time [23, 26]. Each protein has a random variable associated with it, and its state corresponds to whether the function under consideration is assigned to the protein or not. It is assumed that the joint distribution can be expressed in terms only of cliques of size at most two (i.e., edges). This means that the potential function evaluating the network is a linear expression composed of terms over the vertices and edges. So,

$$\Pr(X_1 = x_1, \ldots, X_N = x_N) = \frac{1}{Z} e^{-\left(\sum_{u \in V} \phi_1(X_{\{u\}}) + \sum_{(u,v) \in E} \phi_2(X_{\{u,v\}})\right)},$$

where ϕ_1 computes the vertex "self-term" and the ϕ_2 computes the pairwise edge term. The self-term potential is chosen to correspond to the prior probability for annotating a protein with a particular function; it takes into account the frequency of the function in the network. Note that this is the opposite of what is done by the neighborhood method [38], which prefers less frequent terms to those that are more frequent. The pairwise edge potential is chosen to have different values corresponding to the three cases where either the interacting proteins both have the function under consideration, or they both do not have that function, or one has that function and the other does not; these values are determined using a quasi-likelihood method. Note that these values are not necessarily the same for each function. As noted earlier [25], this model is a generalization of the per-function cut-based method [42], and is similar to that of the multiple function cut formulation [72]. In particular, the cut-based models assume the same fixed value for interactions between proteins of the same function (or for interactions between a protein of one function and any other), regardless of function; this may not be the best assumption, as the guilt-by-association assumption may be true to different degrees for different functions. To make a functional prediction for a protein, the posterior probability that a protein has the function of interest is computed using Gibbs sampling, and then if this value is above a chosen threshold, the function is predicted. Importantly, an exact computation of the posterior probability considers the probability of all assignments of the random variables, and thus implicitly incorporates a distance effect, where the impact of a protein's function on unannotated proteins decreases with distance.

An alternate Markov network approach for protein function annotation [49] assumes that the number of neighbors of a protein that have a particular functional annotation is binomially distributed according to a parameter that differs depending on whether the protein has that function

or not. The posterior probabilities for each protein are computed via a heuristic modification of belief propagation (review, [76]).

4.5 Clustering

Another broad family of methods begins by first identifying components in the interaction network that are likely to correspond to functional units, and then assigning functions to proteins based on their membership in the functional unit. The underlying philosophy for most of these methods is that cellular networks are organized in a modular fashion [37], and that these modules correspond to sets of proteins that take part in the same cellular process or together comprise a protein complex. Identification of functional modules is thus a somewhat stronger goal than simple functional assignment. Most of the methods for identifying modules operate on the underlying assumption that proteins within modules are more tightly connected than proteins in different modules; one may think of this as the module-discovery problem's analog of the guilt-by-association assumption.

Once functional modules, or clusters, are identified, they can be used for annotating uncharacterized proteins, as the most common functional annotation within a cluster can be transferred to its uncharacterized proteins. Alternatively, one can look at overrepresentation instead of frequency and transfer the functions that are enriched in a cluster according to the hypergeometric distribution. Such an approach computes a p-value for a particular function in a cluster as:

$$p = 1 - \sum_{i=0}^{i=f-1} \frac{\binom{F}{i}\binom{N-F}{n-i}}{\binom{N}{n}},$$

where N is the number of proteins in the network, F is the number of proteins in the network annotated with the function under consideration, n is the size of the cluster, and f is the number of proteins within the cluster annotated with that function. Like the neighborhood overrepresentation method of [38], if two functions annotate the same number of proteins within a cluster, this method favors the function that annotates fewer proteins in the interactome. We also note that one feature of cluster-based function prediction methods is that it is possible and indeed not uncommon for certain modules not to contain any annotated proteins, in which case functional assignment to such a cluster cannot be made in a straightforward fashion.

Cluster analysis is a rich area with applications in many diverse fields. A large number of clustering methods have been developed, both for the more familiar problem of clustering general data that comes with some natural measure of similarity, and, to a lesser extent, for the more specific problem of graph clustering. Many of these methods have been applied to interactome

data. Broadly speaking, the clustering methods we consider are either specific to the network domain, or are based on standard distance- or similarity-based clustering techniques; in the latter case, the key issue is typically in deciding on a suitable measure of distance or similarity between two proteins in an interaction network. Additionally, the methods differ in the extent to which the network features they exploit are local. In this regard, we note that some methods use only local neighborhood information when clustering whereas others use more global features of the network; nevertheless, even when using local features to cluster proteins, clustering can be performed on the entire interactome, and thus in some sense, such clustering approaches incorporate the global organization of the interactome as well.

4.5.1 Network-based clustering

Of the clustering approaches, those based on network clustering are perhaps the closest in spirit to the cut- or flow-based annotation schemes: they explicitly attempt to partition the network into contiguous components in such a way that there are more connections between proteins within a component than between proteins belonging to different components. However, unlike the former group of methods, cluster-based approaches typically do not begin with the prior information about the partial assignment of function to neighbors; moreover, several graph clustering-based methods focus on the more specific problem of identifying protein complexes.

Local clustering A number of local clustering approaches attempt to isolate highly connected or dense components within the larger protein interaction network. The density of a set of vertices may be defined in many ways. The density of a set of vertices V' is sometimes computed as the total number of edges among the vertices in V' divided by the total number of possible edges within V' (i.e., $\binom{|V'|}{2}$). Finding the densest subgraph of a particular size is a computationally hard problem, and thus a number of heuristic approaches have been developed. In one approach, a Monte Carlo procedure is developed that attempts to find a set of k nodes with maximum density [65]. A special case of the density measure that has also been exploited to uncover dense components is the clustering coefficient. It is computed for a vertex v as the density of the neighbors of v (i.e., $\mathcal{N}_1(v)$ with v excluded). In [7], each vertex is weighted using a measure similar to its clustering coefficient, but that instead tries to exclude the effects of low-degree vertices. Low degree vertices are frequent in protein interaction networks, and may artificially lower the clustering coefficients of highly connected vertices in dense regions of the network that are also connected to several vertices

of low degree. The clustering coefficient is thus computed instead over a k-core of the neighbors of each vertex, where k-cores are maximal subgraphs of degree $\geq k$. The vertex with the highest weight seeds the search process, and clusters are greedily expanded from it, with vertices being included in the cluster if their weights are above a given threshold. Once no more vertices can be added, this process is repeated for the next highest weighted unseen vertex in the network.

A greedy graph clustering approach is also taken by [3]. Here, a cluster is grown so as to maintain the density of the cluster above a particular threshold, and to ensure that each vertex that is added to the cluster is connected to a large enough number of vertices already in the cluster. The process is initialized by finding the vertex that takes part in the largest number of triangles (i.e., has the largest number of common neighbors with its neighbors).

Dense substructures within protein networks have also been uncovered via spectral analysis [13]. Here, eigenvalues and eigenvectors of the adjacency matrix of the network are computed. For each positive eigenvalue, its corresponding eigenvector is used to group together proteins. In particular, the proteins corresponding to the larger components of the eigenvector tend to form dense subgraphs. Groupings are further filtered to be of sufficient size and to have large enough interconnectivity.

Seeded module discovery Rather than finding clusters in protein-protein physical interaction networks without any functional annotations, a few approaches start with a set of proteins in the interaction network and attempt to identify modules around these "seed" proteins [6, 8]. In the context of protein function prediction, the seeds are proteins that are known to share some biological process or take part in the same complex. In [8], each interaction is labeled with confidence or reliability value in the range of 0 and 1, and a protein is added to the cluster if there exists a path from any seed protein to it such that the product of the reliabilities of the edges in the path is greater than a preselected threshold; for each protein, this corresponds to computing its shortest path to any seed protein when mapping each edge reliability to its negative logarithm. This approach thus scores the membership of a protein to the initial seed set using the probability of its connection via the single-most probable path. In [6], random networks are used to compute the probability that protein u is a member of the same group as the seed set of proteins. This probability is estimated as the fraction of random networks in which a path exists from u to any protein in the seed set. Each random network is generated by taking every edge in the original network, and adding it into the network with probability proportional to its reliability in the original network. This approach

thus attempts to compute the probability of a connection to the initial seed set using any path in the network.

Divisive hierarchical network clustering Girvan and Newman have proposed a divisive hierarchical clustering procedure that is based on edge betweenness [35]. For any edge, its betweenness is defined as the number of shortest paths between all pairs of vertices that run through that edge. This technique, thus, uses global information about the protein network. Edges between modules are expected to have more shortest paths through them than those within modules, and therefore should have higher betweenness values. The overall hierarchical procedure partitions the network by successively deleting edges with highest betweenness values. It has been applied to yeast and human interaction data [27]. The Girvan-Newman algorithm has also been modified so that shortest paths are computed on weighted networks. In one approach, instead of counting the total number of shortest paths through an edge, the total number of "non-redundant" shortest paths through an edge are counted by considering paths that do not share an endpoint [16]. Edge weights are also considered by this method; in this case, weights correspond to dissimilarities between endpoints, rather than similarities or edge reliabilities.

The Girvan-Newman algorithm has also been modified so that the edge with lowest edge clustering coefficient is iteratively deleted [58]. The edge clustering coefficient is a generalization of the usual clustering coefficient, and measures the number of triangles to which a given edge belongs, normalized by the number of triangles that might potentially include it. To deal with the special case where the edge is found in no triangles, the edge clustering coefficient for edge (u, v) is actually defined as:

$$ECC(u, v) = \frac{z_{u,v} + 1}{\min\{|\mathcal{N}_1(u)| - 2, |\mathcal{N}_1(v)| - 2\}},$$

where $z_{u,v}$ gives the number of triangles that edge (u, v) participates in. Unlike the edge betweenness measure, the edge clustering coefficient is a local measure; however, in principle, this definition can be extended to handle higher-order cycles as well. The edge clustering coefficient has been used to uncover modules in yeast [75]. A related algorithm that combines both the global edge betweenness measure with a local measure similar to the edge clustering coefficient has also been proposed [75]. This algorithm computes a local measure called the commonality index for each edge as

$$C(u, v) = \frac{z_{u,v} + 1}{\sqrt{|\mathcal{N}_1(u) - 1| \cdot |\mathcal{N}_1(v) - 1|}}.$$

The edge evaluation measure is then based on the observation that an edge connecting different

modules should have a low commonality and high edge betweenness. Therefore, the algorithm removes edges (u, v) in the order of decreasing $B(u, v)/C(u, v)$ ratio, where $B(u, v)$ is the Girvan-Newman betweenness, and $C(u, v)$ is the commonality index.

Divisive methods do not necessarily specify how to get modules or clusters from the hierarchical grouping process. One proposed approach is to consider a set of vertices $V' \subset V$ as a module if, for each of its vertices, the number of interactions it has within V' (its indegree) is greater than the number of interactions it has with vertices in $V - V'$ (its outdegree) [58]. This condition can be weakened so that a module only requires that the sum of the indegrees for the all vertices in the module be greater than the sum of their outdegrees. The partitioning of the network can now be performed so that an edge with highest edge betweenness or lowest edge clustering coefficient is only removed if it results in two modules [58]. A modified definition considers a set V' a module if the ratio of the number of edges within V' to the number of edges from vertices in V' to vertices outside of this set is greater than one [50]; this is almost the same criterion as that for a weak module [58], except that edges within V' are not counted twice. This definition has been used to uncover modules in an agglomerative procedure, where singleton vertices are considered initially and edges are added back in, using the reverse Girvan-Newman ordering, only if the edge is not between two modules. Modules have also been defined in terms of the structure of the hierarchical cluster subtrees [75]. Here, a module consists of the nodes of a maximal subtree where all non-leaf nodes have at least one child being a leaf, and two modules that have the same parent are merged when the maximal commonality of edges between them is larger than a pre-defined cutoff.

Other network clustering approaches In [43], an initial random partitioning of the network is modified by iteratively moving one protein from one cluster to another in order to improve the clustering's cost. For each protein, the cost measure considers the number of proteins within its assigned cluster with which it does not interact, as well as the number of interactions from it to proteins not assigned to its cluster; both should be small in ideal clusterings. In order to avoid local minima, the local search is modified so as to occasionally disperse the contents of a cluster at random. Additionally, a list of forbidden moves is kept to prevent cycling back to a previous partition. Resulting clusters are then filtered for size, density, and functional homogeneity.

Another approach for clustering is based on uncovering so-called k-clique percolation clusters [1]. A k-clique is a complete subgraph over k nodes, and two k-cliques are considered adjacent if they share exactly $k - 1$ nodes. A k-clique percolation cluster consists of nodes that can be reached via

chains of adjacent k-cliques from each other. An advantage of such an approach is that each protein can belong to several clusters. Since a protein can have different roles in the cell, membership in several clusters is biologically meaningful, and it may be useful to identify a strategy that can recover multiple functions.

A clustering approach based on (modified) random walks within a network has also been developed [28, 70]. The interaction network is transformed into a Markov process, where transition probabilities from u to v and v to u are associated with each edge (u, v); that is, the adjacency matrix is converted to a stochastic matrix. The stochastic-flow algorithm alternates between an expansion step, which causes flow to dissipate within clusters, and an inflation step, which eliminates flow between different clusters. In the expansion step, the probability transition matrix is squared; this corresponds to taking one more step in a random walk. In the inflation step, each entry in the stochastic matrix is raised to the r-th power and then normalized to ensure that the resulting matrix is stochastic again; for $r \geq 1$, the inflation step tends to favor higher probability transitions, and thus tends to boost the probabilities of intra-cluster walks and demote those of inter-cluster walks. This process is repeated until convergence, at which point the connected directed components are evident. Note that in this algorithm, the inflation step distinguishes it from simply taking (traditional) random walks on a graph. This stochastic flow-based clustering procedure has been applied to a protein interaction network that has been transformed into a line graph [56]. Here, each vertex in the new graph represents an interaction in the original network, and any two vertices are adjacent if the corresponding interactions in original network involve a common protein. Note that the line graph formulation allows the stochastic flow-based clustering to place each protein into several clusters.

4.5.2 General distance-based clustering

Rather than use the guilt-by-association assumption directly and explicitly attempt to keep connected nodes in the same cluster, many approaches to clustering interactomes rely instead on assumptions about the similarity of cluster co-members' patterns of connections to *other vertices* in the graph. This makes it possible to use standard distance-based clustering techniques, such as hierarchical clustering, on the resulting similarity or distance matrix. Various similarity measures have been proposed for clustering interaction networks. In one approach [62], the similarity between two proteins is determined by considering each protein's interactions, and computing the significance of their number of shared interactions via the hypergeometric distribution. An alternate

approach that also measures the overlap between the sets of interactions for each pair of proteins uses the Czekanowski-Dice distance [12]. For proteins u and v, this is given by:

$$CD(u,v) = \frac{|\mathcal{N}_1(u)\Delta\mathcal{N}_1(v)|}{|\mathcal{N}_1(u) \cup \mathcal{N}_1(v)| + |\mathcal{N}_1(u) \cap \mathcal{N}_1(v)|},$$

where Δ computes the symmetric difference between two sets. In addition to these two measures [12, 62], there are a number of other ways of computing the similarity or distance between two proteins by considering only the overlap among their direct interactions [36, 46]. In contrast to these purely local measures, a more global measure can be used where the distance between two proteins is calculated as the shortest path distance between them in the network [4]. In a related earlier approach [59], each protein is associated with a vector that contains its shortest path distance to all other proteins in the network. A similarity between two proteins is computed as the correlation coefficient between their corresponding shortest-path vectors. Since global and local similarity measures may be quite different, this global shortest-path based similarity measure has also been used in conjunction with a local connectivity coefficient based on the common interactors of two proteins [57].

For any of these measures, agglomerative hierarchical clustering is then performed by progressively merging groups of proteins that are closest or most similar to each other. Neighbor-joining [61] has also been used in the context of clustering interactomes [12]; it favors merging items that are close to each other but also considers distances from the remaining items. As discussed earlier, hierarchical clustering methods do not automatically give the final partitioning of the network. In [12], the separation into clusters is performed using existing biological process annotations, whereby each cluster must have at least half of its proteins annotated by the same term. This function is then transferred to the other proteins in the cluster.

In some applications of distance-based hierarchical clustering, there can be a problem where distances among several items are identical. This is the case, for example, when setting the distance between two proteins as their shortest path distance in the network. One possible solution to this problem is a two phase approach [4]. In the first phase, hierarchical clustering is performed multiple times, and each time there is a "tie in proximity," a random pair is chosen for merging. Each clustering run is stopped according to a threshold that considers the distances between all proteins in a cluster. In the second phase, the fraction of solutions in which each protein pair is clustered together is then used as a similarity measure for a final round of clustering.

4.6 Kernel-based learning methods

Discriminative learning methods are another broad area in computer science that has been applied to the problem of predicting protein function using interaction networks. The methods discussed here use support vector machines (SVMs), machine learning methods which embed positive and negative examples in a feature space and then find a maximal separating hyperplane in this space between the positive and negative examples [14, 71]. In the context of function prediction via network analysis, SVMs have been applied by considering each function in turn and labeling each protein as either positive or negative based upon whether it is annotated with the function of interest [47, 69]. The key technical difficulty is how each protein u in the network is mapped to a point x_u in the feature space. If proteins are "close" in the network, then they should also be close in the feature space. The mapping to the feature space can be given implicitly via a positive definite kernel matrix K specifying the inner product (i.e., $K_{uv} = x_u^T x_v$); since the discriminant function for SVMs is specified via inner products, explicit representations of the points are not necessary.

In [47], two kernels are considered. First, a linear kernel is created where each entry K_{uv} is the dot product of the N-dimensional vectors representing the interactions of proteins u and v. The more similar the interaction patterns for the proteins, the larger this value is in the kernel matrix; this kernel is similar in spirit to local clustering methods based on the similarity of immediate interactors. It does not capture more global properties of the network. Second, a diffusion kernel [45] is created where the kernel value K_{uv} can be interpreted as the probability that a random walk starting from u will be at v after infinite time steps; the transition probabilities between nodes are dependent on a parameter specifying the rate of diffusion. The diffusion kernel accounts for all possible paths connecting two proteins, and nodes that are connected with shorter paths or by several paths are considered more similar. Thus this kernel utilizes some of the same network features as the flow-based function prediction method and the stochastic-flow clustering approach. It has been shown that the diffusion kernel captures the global constraint that the sum of the Euclidean distances between connected samples is bounded, but that this can lead to large variances in the pairwise distances [69]. This observation has led to the development of a locally constrained diffusion kernel, which captures additional local constraints requiring that the Euclidean distance between connected samples be more tightly bound. SVMs using the locally constrained diffusion kernel are found to better predict protein function than those using the original diffusion kernel.

5 Assessment of prediction quality

It is natural to ask how different network-based methods for the function prediction problem perform in comparison to each other. Unfortunately, a comprehensive comparative evaluation of these methods has not been done. Therefore, we briefly outline a couple of evaluation frameworks that have been proposed and showcase the performance of some of the reviewed methods in these frameworks. Overall, it is difficult to judge the comparative performance of different methods by surveying the literature. This is due in part to differences in the evaluation frameworks; such differences include the measures used to assess performance quality, the treatment of multiple annotations and predictions, the selection of a gold standard for functional annotation, the treatment of the functional hierarchy, and the precise (and always changing) interaction networks under consideration.

Some common features of evaluation frameworks are that most of the existing testing has been performed in the baker's yeast *Saccharomyces cerevisiae*, because of the quality and quantity of data available for that organism, and that all frameworks use cross-validation testing. In this type of testing, the annotations of one (or more) protein are considered as unknown, and the annotations of the remaining proteins, along with the network, are used to predict its annotations.

Evaluation frameworks One way to treat the issue of multiple predictions and multiple annotations is by using each *prediction* in the calculation of performance measurements. This is the approach taken by [26], using annotations from YPD functional categories [20] and considering all predictions with score above a cutoff. In this work, for each annotated protein u with at least one annotated interaction partner, it is assumed to be unannotated and its function is predicted. Then, performance measurements are computed in terms of: k_u, the number of known functions for protein u; p_u, the number of predicted functions for protein u; and o_u, the amount of overlap between the set of known and predicted functions. The precision (or positive predictive value) is defined as:

$$\text{Precision} = \frac{\sum_u o_u}{\sum_u p_u}.$$

The recall (or sensitivity) is defined as:

$$\text{Recall} = \frac{\sum_u o_u}{\sum_u k_u}.$$

In follow up work [25], 134 GO biological process terms are chosen for consideration if they annotate more than 50 proteins and if none of their child biological process terms annotate the same set of

proteins. Since GO is a directed acyclic graph and functional terms can be related to each other via is-a or part-of relationships, the authors suggest modifications to this basic scheme to accommodate this hierarchical structure. A possible weakness in this per-prediction framework is that proteins that have more annotations will have a larger effect on performance measurements.

An alternative approach [53] is to treat each *protein* as a data point when measuring performance. In particular, for each protein, if the top scoring functional annotation is above some threshold, it is the prediction for the protein. If a prediction is a known functional annotation, it is considered a true positive, and otherwise, it is a false positive. Measuring performance per-protein avoids the problem of proteins with many multiple annotations or predictions from dominating the results, and makes the performance measures easily interpretable in terms of the number of proteins that can be annotated at a certain confidence. This criterion still permits ties between top-scoring predictions; in this case, a protein's predicted annotation is counted as a true positive if more than half of its top-scoring predictions are correct, and a false positive otherwise. This approach is taken as a compromise between two extreme cases. In the first case, a prediction for a protein can be counted as a true positive if at least one of the predictions made for it is correct; however, in this case, a method that predicts every protein to participate in every function would only have true positives in this framework. At the other extreme, a protein can be counted as a true positive if every prediction made for it is correct. This, however, would count as false positives those proteins that get many correct predictions and only one incorrect one. An alternative and perhaps better approach would be to compute the precision and recall per-protein, and then average the results over proteins. Here, a flat set of functional terms coming from the MIPS [60] functional hierarchy was used for evaluation, with 72 biological process terms chosen from the second level of hierarchy.

A number of clustering approaches are evaluated in [11], based on how well they recapitulate known yeast protein complexes. While this is not the same as assessing the performance of function prediction, there is likely to be a relationship between the two; moreover, this study is likely to be useful in designing a similar evaluation of clustering approaches in the context of function prediction. The clustering algorithms are run both on simulated networks where complexes are embedded into the graph, and edges are added and removed at various proportions, as well as on data sets obtained in high-throughput experiments. Performance is measured by computing recall values (i.e., for each complex, find the cluster which has the highest fraction of its proteins) and precision values (i.e., for each cluster, find the maximal fraction of its proteins found in the same annotated complex). In theory at least, it is also possible to use either of the above approaches [26, 53] to evaluate

how well the enriched biological processes in each cluster predict protein function. We expect the evaluation of clustering for prediction of complexes to give different results than clustering for function prediction, as, on the one hand, complex prediction may be a more specific problem than function prediction, but, on the other hand, the dense network components that are readily identified by clustering methods may be "easy cases" for function prediction, while more ambiguous proteins in sparse regions may be left out of the clusters identified by some of the methods.

Comparative performance Nabieva *et al.* [53] test the majority, neighborhood, multiway-cut and flow formulations in two-fold cross-validation on the yeast proteome using Receiver Operating Characteristic (ROC) analysis. They find that the flow-based method generally outperforms other methods. They also find, perhaps surprisingly, that the next best method is majority, which outperforms neighborhood and multiway-cut formulations and performs as well as flow-based method for proteins with at least three neighbors annotated with the same function.

The multiway-cut formulation was previously found to outperform the majority method [72]. However, the measure of success used to judge performance there was the fraction of times the top prediction for each protein is correct, and the score of the top prediction was not considered. ROC analysis, as in [53], with a varying threshold gives a more complete picture of performance, particularly with respect to high-confidence predictions, and shows that majority outperforms the cut-based method over a large false positive range, but the cut method is able to make predictions when majority cannot. A subsequent paper [52] also finds that a cut-based approach does not outperform a strictly local approach which predicts function based on the fraction (instead of number) of neighbors with a particular function. In their case, the cut-based approach considered is the pairwise min-cut problem of [42].

In [26], the authors find in leave-one-out testing that the Markov network approach [26] outperforms the majority [63] and neighborhood approaches [38] on the yeast interactome. The significant added generality of the Markov network approach over the cut-based approach and its implicit use of distance in the network may potentially explain why it performs better than majority whereas the related cut-based methods do not; however, a weakness with the testing as performed in [26] is that the a functional prediction for a protein is scored according to its rank when using the majority and neighborhood methods. This means that the strength of the evidence for a functional prediction from the protein's neighborhood is not considered; for example, for the majority method, it does not matter in this testing framework if the top-scoring function for a protein appears nine

times or one time among its direct interactions—both are treated equivalently. It remains to be seen whether the Markov random field approach will outperform the local method when scores—not ranks—are considered.

Other findings revealed in cross-validation testing include the necessity of multiple solutions for the cut-based method in order to get higher confidence predictions, and a deteriorating performance of the neighborhood method with increasing radius, reflecting the peril of using more distant nodes without considering their distance or connectivity to the target node [53]. It is also observed that all methods, including majority, multiway-cut, and functional flow improve when incorporating interaction reliability [18, 53].

These evaluations show that the strength of the functional signal from the local neighborhood is the best indicator of whether or not a high-confidence prediction can be made: if a protein is interacting with many proteins with known annotation, a majority scheme performs well, as do other methods. Also, the results suggest that the information from immediate neighbors can be used directly, and statistical information, such as that used in the χ^2 criterion, is not necessarily helpful. On the other hand, when a protein is known to interact with only unannotated proteins, local approaches such as majority cannot make any predictions, whereas the cut, flow, Markov network, and clustering methods can. More broadly, for proteins with few interactions or few interactions with annotated proteins, which is likely to be the case for more recently characterized proteomes, more global methods are necessary for functional predictions. Thus, global methods are likely to be an important tool in characterizing proteins in unusual or less-studied proteomes.

As mentioned earlier, clustering methods have largely not been evaluated with respect to function prediction. However, the study of [11] finds that the stochastic flow-based clustering procedure [70] is robust to alterations in the simulated data and clearly outperforms the other methods tested [7, 10, 43] in extracting complexes from high-throughput physical interaction datasets.

6 Conclusions

The emergence of high-throughput techniques for determining protein physical interactions at the genomic scale has provided large amounts of data that can be used for answering the challenge of predicting protein function. Here, we have reviewed a number of methods that have been developed for this problem. There are several promising directions for further research in this area.

First of all, it is clear that the area of function prediction via network analysis is in need of

a comprehensive and systematic evaluation framework. Such an evaluation will ideally attempt to answer not only which methods perform better but also why. We expect different methods to perform well in different circumstances, and ideally an evaluation would bring to light which method should be used in which situation. In particular, it should be possible to relate topological features and annotation density of the network to performance. For example, local methods may be expected to perform well on dense or well-annotated networks. Since the experimentally determined interactomes of various organisms in their present state differ with respect to their coverage, network density, and known annotations, such an evaluation will be vital for guiding researchers towards an appropriate prediction method for their particular needs.

In terms of methodological directions, a potentially fruitful area that is in need of principled exploration is a closer study of protein annotations, and in particular, of the relationships between functions. One promising line of research involves developing techniques for exploiting the hierarchical nature of protein function classification. Currently, many methods address the issue only at the evaluation step, and often use a flat set of terms which are then treated as unrelated labels; the flat set may include the leaf terms of the functional hierarchy (i.e., the most specific descriptors) or a hand-picked set of terms. In the latter case, more specific functional annotations may be "upcast" to their ancestor term(s) in the flat set, and less specific annotations are ignored. The development of methods that more directly exploit the functional hierarchy as part of the prediction algorithm is likely to be fruitful. A related research direction is to accommodate functional relationships between interacting proteins that go beyond guilt-by-association, which forms the basis of most methods currently used for network-based function prediction. Simply stated, if guilt-by-association were completely true, all proteins in an organism would be engaged in the same non-trivial biological process. This is clearly not the case; moreover, biological "cross-talk" is evident in interactomes [63], as there are many pairs of different biological processes recurring as annotations for interacting proteins. Understanding and leveraging the interplay between biological processes should benefit future methods for predicting protein function. Promising research along both of these lines has been initiated [15, 44]. Lastly, an intriguing possibility is to relate modularity to the distance in the network along which functional connections hold. The assumption of modularity suggests that guilt-by-association may hold on mezoscale, along the size of a functional module, but at larger network distances understanding of the cross-talk between processes may become more relevant for function prediction.

The area of function prediction via network analysis is based on recently available data and is

thus relatively new, yet its graph-theoretic formulation enables it to tap into decades of algorithmic and methodological advances in computer science and applied mathematics. In the coming years, we expect to see further methodological developments in this area, as well as the establishment of more uniform testing frameworks. Together with the growth of interaction data and the improvement of the accuracy of experimental techniques for interaction determination, these developments promise to give network analysis methods a position of increasing prominence in computational function prediction.

7 Acknowledgments

This work has been supported by NSF CCF-0542187, NSF IIS-0612231, NIH GM076275, and the NIH Center of Excellence grant P50 GM071508.

Table 1: Summary of methods for predicting protein function via network analysis

Neighborhood approaches	Majority: consider how often a function is seen as annotation of a proteins' immediate interactors [63].
	Neighborhood: consider neighborhood of radius 1, 2, or 3 and compute over-representation of a function in that neighborhood, as judged by the χ^2 test [38].
	Weighted neighborhood: consider neighborhood of radius 2, and assign function based on weighted paths from the target protein to neighborhood proteins [18].
Cut-based	Multiway cut: consider all functions simultaneously [72] (NP-hard). Solve approximately via Monte Carlo approach [72] or exactly via ILP [53].
	Mincut: consider one function at a time [42]. Solve approximately via heuristic [42] or exactly via flow [52].
Flow-based	Assign functions via simulation of "functional flow" from annotated nodes [53].
Markov network	Use pairwise potential over interacting proteins [26] or assume that the number of neighbors of a protein that have a particular function is binomially distributed according to whether the protein has the function in question or not [49]. One function is modeled at a time.
Local graph clustering	Find high-density subgraphs of specified size via Monte Carlo methods [65].
	Starting from a locally dense node as seed, greedily add vertices according to their local neighborhood density (k-core clustering coefficient) [7], or according to their connectedness to the cluster while maintaining cluster density and vertex cluster property above a cutoff [3].
	Spectral analysis: build clusters consisting of nodes corresponding to the larger components of eigenvectors for positive eigenvalues of adjacency matrix [13].
Seeded module discovery	Add proteins to cluster that have sufficiently reliable paths to any seed protein [8].
	Add proteins to cluster that are grouped together with the seed proteins in sufficiently many random networks [6].
Network-based hierarchical clustering	Apply Girvan-Newman (GN) algorithm, building a hierarchical clustering by removing edges with highest edge-betweenness [27]; extend the GN algorithm to weighted graphs and modify to consider non-redundant paths [16]; extend the GN algorithm to additionally consider local measure (edge commonality) [75]; perform agglomerative clustering in the reverse order of the GN edge removal [50].
Distance-based hierarchical clustering	Cluster proteins according to the overlap between their common interactors using hypergeometric distribution [62]; or Czekanowski-Dice distance [12].
	Cluster proteins according to a distance based on their shortest path length and using randomization to break ties [4].
	Cluster proteins according to the similarity of their all-pairs shortest-path profiles [59]; combine this global measure with local measure based on direct interactors [57].
Other graph clustering	Starting with a random initial clustering, apply moves to improve the clustering cost, which favors few missing edges within clusters and few present edges between clusters [43].
	Cluster proteins that belong to a path of adjacent k-cliques [1].
	Stochastic-flow clustering: alternate random-walk steps with steps that amplify the inter-cluster distance [70]. Has been applied to line graph transformation of network [56].
Supervised learning	Train SVM utilizing an appropriate kernel that captures the distance between two proteins in the network. Linear, diffusion, and locally-constrained diffusion kernels have been applied [47, 69].

References

[1] B. Adamcsek, G. Palla, I. Farkas, I. Derenyi, and T. Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22:1021–1023, 2006.

[2] T. Aittokallio and B. Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, 7:243–255, 2006.

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

[4] V. Arnau, S. Mars, and I. Marin. Iterative cluster analysis of protein interaction data. *Bioinformatics*, 21:364–378, 2005.

[5] M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, et al. Gene ontology: tool for the unification of biology. The gene ontology consortium. *Nat. Genet.*, 25(1):25–29, 2000.

[6] S. Asthana, O. King, F. Gibbons, and F. Roth. Predicting protein complex membership using probabilistic network reliability. *Genome Res.*, 14:1170–1175, 2004.

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

[8] J. Bader. Greedily building protein networks with confidence. *Bioinformatics*, 19:1869–1874, 2003.

[9] A. Bauer and B. Kuster. Affinity purification-mass spectrometry. *Eur. J. Biochem.*, 270:570–578, 2003.

[10] M. Blatt, S. Wiseman, and E. Domany. Superparamagnetic clustering of data. *Phys. Rev. Lett.*, 76:3251–3254, 1996.

[11] S. Brohee and J. van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7:488, 2006.

[12] C. Brun, F. Chevenet, D. Martin, J. Wojcik, A. Guenoche, and B. Jacq. Functional classification of proteins for the prediction of cellular function from a protein-protein interaction network. *Genome Biol.*, 5:R6, 2003.

[13] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, et al. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucl. Acids. Res.*, 31:2443–2450, 2003.

[14] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[15] S. Carroll and V. Pavlovic. Protein classification using probabilistic chain graphs and the Gene Ontology structure. *Bioinformatics*, 22:1871–1878, 2006.

[16] J. Chen and B. Yuan. Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, 22:2283–2290, 2006.

[17] Y.-R. Cho, W. Hwang, M. Ramanathan, and Aidong Zhang. Semantic integration to identify overlapping functional modules in protein interaction networks. *BMC Bioinformatics*, 8:265, 2007.

[18] H. Chua, W.-K. Sung, and L. Wong. Exploiting indirect neighbors and topological weight to predict protein function from protein-protein interactions. *Bioinformatics*, 22:1623–1630, 2006.

[19] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[20] M.C. Costanzo, M. E. Crawford, J. E. Hirschman, J. E. Kranz, P. Olsen, L. S. Robertson, M. S. Skrzypek, B. R. Braun, K. L. Hopkins, P. Kondu, C. Lengieza, J. E. Lew-Smith, M. Tillberg, and J. I. Garrels. YPD^{TM}, PombePD^{TM}, and WormPD^{TM}: Model organism volumes of the BioKnowledge library, an integrated resource for protein information. *Nucl. Acids Res.*, 29:75–79, 2001.

[21] E. Dalhaus, D. S. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis. The complexity of the multiway cuts. In *Proc. 24th Annual STOC*, pages 241–251. ACM, 1992.

[22] T. Dandekar, B. Snel, M. Huynen, and P. Bork. Conservation of gene order: a fingerprint of proteins that physically interact. *Trends Biochem. Sci.*, 23(9):324–328, 1998.

[23] M. Deng, T. Chen, and F. Sun. An integrated probabilistic model for functional prediction of proteins. In *Proc. 7th Annual RECOMB*, pages 95–103. ACM, 2003.

[24] M. Deng, F. Sun, and T. Chen. Assessment of the reliability of protein-protein interactions and protein function prediction. In *Pac. Symp. Biocomput.*, pages 140–151, 2003.

[25] M. Deng, Z. Tu, F. Sun, and T. Chen. Mapping gene ontology to proteins based on protein-protein interaction data. *Bioinformatics*, 20:895–902, 2004.

[26] M. Deng, K. Zhang, S. Mehta, T. Chen, and F. Sun. Prediction of protein function using protein-protein interaction data. *J. Computational Biol.*, 10:947–960, 2003.

[27] R. Dunn, F. Dudbridge, and C. Sanderson. The use of edge-betweenness clustering to investigate biological function in protein interaction networks. *BMC Bioinformatics*, 6:39, 2005.

[28] A. Enright, S. Van Dongen, and C. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res*, 30:1575–1584, 2002.

[29] A.J. Enright, I. Iliopoulos, N. C. Kyrpides, and C. A. Ouzounis. Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402:86–90, 1999.

[30] S. Fields and O.-K. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340:245–246, 1989.

[31] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Brooks/Cole Publishing Company, Pacific Grove, CA, 2002.

[32] T. Gaasterland and M. Ragan. Microbial genescapes: phyletic and functional patterns of ORF distributions among prokaryotes. *Microb. Comp. Genomics*, 3:199–217, 1998.

[33] M. Galperin and E. Koonin. Who's your neighbor? New computational approaches for functional genomics. *Nat. Biotechnol.*, 18:609–613, 2000.

[34] L. Giot, J. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, Y. Li, et al. A protein interaction map of Drosophila melanogaster. *Science*, 302:1727–1736, 2003.

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

[36] D. Goldberg and F. Roth. Assessing experimentally derived interactions in a small world. *Proc. Natl. Acad. Sci. USA*, 100:4372–4376, 2003.

[37] L. Hartwell, J. Hopfield, S. Leibler, and A. Murray. From molecular to modular cell biology. *Nature*, 402:C47–52, 1999.

[38] H. Hishigaki, K. Nakai, T. Ono, A. Tanigami, and T. Takagi. Assessment of prediction accuracy of protein function from protein–protein interaction data. *Yeast*, 18:523–531, 2001.

[39] ILOG CPLEX 7.1, 2000. http://www.ilog.com/products/cplex/.

[40] R. H. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N. Krogan, S. Chung, et al. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302:449–453, 2003.

[41] T. Joshi, Y. Chen, J. Becker, N. Alexandrov, and D. Xu. Genome-scale gene function prediction using multiple sources of high-throughput data in yeast. *OMICS*, 8:322–333, 2004.

[42] U. Karaoz, T. M. Murali, S. Levotsky, Y. Zheng, C. Ding, C. R. Cantor, and S. Kasif. Whole-genome annotation by using evidence integration in functional-linkage networks. *Proc. Natl. Acad. Sci. USA*, 101:2888–2893, 2004.

[43] A. King, N. Przulj, and I. Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20:3013–3020, 2004.

[44] M. Kirac, G. Ozsoyoglu, and J. Yang. Annotating proteins by mining protein interaction networks. *Bioinformatics*, 22:e260–e270, 2006.

[45] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proc. Intl. Conf. on Machine Learning*, pages 315–322, 2002.

[46] R. Krause, C. von Mering, and P. Bork. A comprehensive set of protein complexes in yeast: mining large-scale protein-protein interaction screens. *Bioinformatics*, 19:1901–1908, 2003.

[47] G. Lanckriet, T. Bie, N. Cristianini, M. Jordan, and W. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20:2626–2635, 2004.

[48] I. Lee, S. Date, A. Adai, and E. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306(5701):1555–1558, 2004.

[49] S. Letovsky and S. Kasif. Predicting protein function from protein/protein interaction data: a probabilistic approach. *Bioinformatics*, 19 Suppl 1:i197–i204, 2003.

[50] F. Luo, Y. Yang, C. Chen, R. Chang, J. Zhou, and R. Scheuermann. Modular organization of protein interaction networks. *Bioinformatics*, 23:207–214, 2007.

[51] E. Marcotte, M. Pellegrini, H. Ng, D. Rice, T. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285:751–753, 1999.

[52] T. Murali, C.-J. Wu, and S. Kasif. The art of gene function prediction. *Nat. Biotechnol.*, 24:1474–1475, 2006.

[53] E. Nabieva, K. Jim, A. Agarwal, B. Chazelle, and M. Singh. Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics*, 21 Suppl. 1:i302–i310, 2005.

[54] R. Overbeek, M. Fonstein, M. D'Souza, G. Pusch, and N. Maltsev. The use of gene clusters to infer functional coupling. *Proc. Natl. Acad. Sci. USA*, 96(6):2896–2901, 1999.

[55] M. Pellegrini, E. Marcotte, M. Thompson, D. Eisenberg, and T. Yeates. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proc. Natl. Acad. Sci. USA*, 96(8):4285–4288, 1999.

[56] J. Pereira-Leal, A. Enright, and C. Ouzounis. Detection of functional modules from protein interaction networks. *Proteins*, 54:49–57, 2004.

[57] J. Poyatos and L. Hurst. How biologically relevant are interaction-based modules in protein networks? *Genome Biol.*, 5:R93, 2004.

[58] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA*, 101(9):2658–2663, 2004.

[59] A. Rives and T. Galitski. Modular organization of cellular networks. *Proc. Natl. Acad. Sci. USA*, 100(3):1128–1133, 2003.

[60] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, et al. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Res.*, 32:5539–5545, 2004.

[61] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[62] M. Samanta and S. Liang. Predicting protein functions from redundancies in large-scale protein interaction networks. *Proc. Natl. Acad. Sci. USA.*, 100:12579–12583, 2003.

[63] B. Schwikowski, P. Uetz, and S. Fields. A network of protein-protein interactions in yeast. *Nat. Biotechnol.*, 18:1257–1261, 2000.

[64] R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Molecular Systems Biology*, 3:88, 2007.

[65] V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci. USA.*, 100:12123–12128, 2003.

[66] E. Sprinzak, S. Sattath, and H. Margalit. How reliable are experimental protein-protein interaction data? *J. Mol. Biol.*, 327(5):919–923, 2003.

[67] S. Suthram, T. Shlomi, E. Ruppin, R. Sharan, and T. Ideker. A direct comparison of protein interaction confidence assignment schemes. *BMC Bioinformatics*, 7:360, 2006.

[68] O. Troyanskaya, K. Dolinski, A. Owen, R. Altman, and D. Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in S. cerevisiae). *Proc. Natl. Acad. Sci. USA*, 100:8348–8353, 2003.

[69] K. Tsuda and W. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20 Suppl. 1:i326–i333, 2004.

[70] S. van Dongen. *Graph clustering by flow simulation.* PhD thesis, University of Utrecht, 2000.

[71] V Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[72] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani. Global protein function prediction from protein-protein interaction networks. *Nat Biotechnol.*, 21:697–700, 2003.

[73] C. von Mering, M. Huynen, D. Jaeggi, S. Schmidt, P. Bork, and B. Snel. STRING: a database of predicted functional associations between proteins. *Nucleic Acids Res.*, 31:258–261, 2003.

[74] C. von Mering, R. Krause, B. Snel, M. Cornell, S. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

[75] C. Wang, C. Ding, Q. Yang, and S. R. Holbrook. Consistent dissection of the protein interaction network by combining global and local metrics. *Genome Biol.*, 8:R271, 2007.

[76] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[77] X. Zhu, M. Gerstein, and M. Snyder. Getting connected: analysis and principles of biological networks. *Genes and Development*, 21:1010–1024, 2007.