23. Structural Pattern Discovery in Protein–Protein Interaction Networks

Tamás Nepusz, Alberto Paccanaro

Most proteins in a cell do not act in isolation, but carry out their function through interactions with other proteins. Elucidating these interactions is therefore central for our understanding of cellular function and organization. Recently, experimental techniques have been developed, which have allowed us to measure protein interactions on a genomic scale for several model organisms. These datasets have a natural representation as weighted graphs, also known as protein–protein interaction (PPI) networks. This chapter will present some recent advances in computational methods for the analysis of these networks, which are aimed at revealing their structural patterns. In particular, we shall focus on methods for uncovering modules that correspond to protein complexes, and on random graph models, which can be used to denoise large scale PPI networks. In Sect. 23.1, the state-of-the-art techniques and algorithms are described followed by the definition of measures to assess the quality of the predicted complexes and the presentation of a benchmark of the detection algorithms on four PPI networks. Section 23.2 moves beyond protein complexes and explores other structural patterns of protein–protein interaction networks using random graph models.

Biological processes in our cells are carried out by groups of biochemical compounds interacting with each other in elaborate ways. Proteins are probably the most important class of these compounds, consisting of chains of amino acids that are folded into a wide variety of complex fibrous or spherical shapes. Interactions between proteins are facilitated by specific domains on the amino acid chains that allow them to bind to each other. Some of these interactions are short-lived (i. e., a protein may be carrying another protein from one part of the cell to another), while others are stable and facilitate the formation of long-lived structures called protein complexes. The accurate knowledge of interactions and complexes is crucial to deepen our understanding of biological processes and to assign a cellular function to proteins in the cell. It is thus no surprise that the web of interactions between proteins in living organisms has been the focus of scientific research for years in experimental [23.1–8] and computational (in silico) studies alike [23.9–15].

A natural representation of the interactions between proteins is an undirected graph, where vertices of the graph correspond to individual proteins and the edges between vertices denote pairwise interactions. Such networks are usually inferred via high-throughput experimental techniques such as yeast-2-hybrid (Y2H)

experiments or co-affinity purification followed by mass spectrometry (AP-MS). Most frequently, the edges of this network are also labeled with confidence scores that represent how likely the existence of a given interaction is; these confidence scores are usually derived from the same experimental procedure that was used to produce the list of interacting protein pairs. Partial protein–protein interaction networks are now available for all major model organisms, ranging from the yeast *Saccharomyces cerevisiae* [23.1, 2, 7, 8, 12] to *Homo sapiens* [23.5, 6], opening up possibilities to analyze them using machine-learning algorithms in order to uncover their interesting structural patterns.

This chapter will first focus on the most intensively studied structural pattern, namely the modular architecture of PPI networks. We will see that PPI networks contain subsets of proteins that are connected to each other more densely than to the rest of the network and that these subsets show a good correspondence with protein complexes. In Sect. 23.1, we describe the state-of-the-art techniques and algorithms for detecting protein complexes in PPI networks, followed by the definition of measures to assess the quality of the predicted complexes. To conclude the section, we present a benchmark of the described protein complex detection algorithms on four PPI networks of the yeast *Saccharomyces cerevisiae*, a well-known model organism in systems biology, genetics, and cell biology.

In Sect. 23.2, we move beyond protein complexes and explore other structural patterns of PPI networks using random graph models. First, we introduce the most promising random graph models for modeling PPI networks and describe how they can be fitted to a specific observed interaction dataset. After that, we will evaluate the predictive power of these graph models using standard machine-learning approaches (ROC and precision-recall curves) and present a case study that applies the degree-corrected stochastic blockmodel [23.16] to predict putative novel interactions between the proteins of *Homo sapiens*. Finally, we validate some of these predictions with the help of hand-curated and reviewed biological databases, such as the gene ontology (GO) [23.17] and the Kyoto Encyclopedia for Genes and Genomes (KEGG) [23.18].

23.1 Detecting Protein Complexes from Protein–Protein Interaction Networks

Protein complexes are groups of two or more associated polypeptide chains that bind together in a living organism and form a stable molecular structure that carries out a given biological function. They are the key components of many biological processes, and scientists generally think about a cell as a collection of modular protein complexes, each of which is responsible for an independent biological process. For instance, ribosome, one of the most well-known protein complexes, creates other proteins from amino acids and an RNA sequence that encodes the structure of the protein being built. This is achieved by a complex interplay of more than 70 proteins in eukaryotes [23.19]. The accurate description of protein complexes is thus one of the first steps towards understanding how different biological functions are carried out by the cells of a living organism.

Proteins in a complex are linked together by a dense network of PPI, and although there are quite a few interactions between proteins of different complexes, it is generally the case that the interaction density between members of the same complex is much higher than the interaction density between members of different complexes.

23.1.1 Complex Detection by Traditional Graph Clustering

The earliest algorithms for detecting protein complexes from pairwise interactions were mostly based on the application of a traditional graph clustering method to the graph representation of the set of interactions. The classical problem of graph clustering aims to decompose the vertex set V of a graph $G(V, E)$ into disjoint subsets $V_1, V_2, \ldots, V_k$ such that the intracluster edge densities (i. e., the number of edges within $V_1, V_2, \ldots, V_k$ divided by the number of theoretically possible edges within these sets) are maximal and the number of edges between clusters is minimal. However, since these methods were not designed specifically with PPI data in mind, they may fail to capture some peculiar properties of PPI networks. One such property is the fact that a protein may participate in more than one complex; in other words, protein complexes are not disjoint like clusters in the result of a graph clustering algorithm. Another property is that not all proteins participate in protein complexes, while graph clustering methods tend to classify each vertex into one of the clusters even when there is not enough evidence to support the membership

of a vertex in *any* of the clusters. Despite this disadvantage, traditional graph clustering techniques are still being used to identify protein complexes, and no review would be complete without them. In this section, we will describe two of the most successful ones: restricted neighborhood search clustering (RNSC) [23.20] and Markov clustering (MCL) [23.21, 22].

Restricted Neighborhood Search Clustering

RNSC [23.20, 23] is a cost-based local search algorithm that starts from an initial random clustering and iteratively moves a randomly selected node from one cluster to another in order to improve the result. The quality of a clustering $\mathcal{C}$ is measured by two functions, the *naive* and the *scaled* cost function. Earlier stages of the algorithm use the naive cost function as it is faster to calculate, while later stages switch to the scaled cost function. The naive cost function for a graph $G(V, E)$ and its clustering $\mathcal{C}$ is defined as

$$C_{\mathrm{n}}(G, \mathcal{C}) = \frac{1}{2} \sum_{v \in V} \left[\#^1_{\mathrm{out}}(G, \mathcal{C}, v) + \#^0_{\mathrm{in}}(G, \mathcal{C}, v) \right], \tag{23.1}$$

where $\#^1_{\mathrm{out}}(G, \mathcal{C}, v)$ is the total number of edges incident on v that lead out of v's cluster in the clustering $\mathcal{C}$, and $\#^0_{\mathrm{in}}(G, \mathcal{C}, v)$ is the number of vertices in the cluster of v that are *not* adjacent to v. Intuitively, a good clustering should contain only a small number of inter-cluster edges (which decreases the first term in the sum), and a large number of intra-cluster edges (which decreases the second term), therefore this cost function seems plausible. However, note that the cost function is biased towards small clusters. *King* [23.23] noted that for graphs with density less than 0.5, a single giant cluster including all the points will always have a higher cost than a clustering that puts every vertex into its own cluster. The scaled cost function resolves this problem by weighing the contribution of each cluster [23.23],

$$C_{\mathrm{s}}(G, \mathcal{C}) = \frac{n-1}{3} \sum_{v \in V} \frac{1}{N(v) \cup C_v} \cdot \left[\#^1_{\mathrm{out}}(G, \mathcal{C}, v) + \#^0_{\mathrm{in}}(G, \mathcal{C}, v) \right], \tag{23.2}$$

where $N(v)$ is the neighborhood of v (i. e., the set of nodes adjacent to v) and C_v is the cluster in which v resides.

Local search algorithms typically suffer from the problem of getting stuck in suboptimal local minima. RNSC resolves this problem by occasionally inserting diversification moves into the optimization process. A diversification move either places a subset of vertices into randomly selected clusters or destroys a cluster entirely and distributes its vertices among the remaining clusters. Both moves usually increase the cost of the clustering temporarily, but it may help the algorithm to climb out from local minima and allow it to decrease the cost by further optimization steps in later stages.

The RNSC algorithm also employs a tabu search heuristic [23.24] to prevent cycling around a local minimum. This is implemented by a fixed capacity first-in, first-out queue of vertices, called the *tabu queue*. When a vertex is moved, it enters the queue at one end and pushes all the vertices already in the queue towards the other end. Vertices within the tabu queue are not allowed to be moved between clusters until they leave the queue at the other end – which they will eventually do as the size of the tabu queue is fixed. The algorithm terminates after a fixed number of moves and it is restarted a fixed number of times from different random configurations. The final result will then be the clustering with the smallest scaled cost found among all the iterations in all trials.

The algorithm so far is a generic graph clustering algorithm. To incorporate domain-specific knowledge, *King* et al. [23.20] extended the algorithm by a post-processing stage that excludes clusters that are smaller than a given size or sparser than a given density threshold; their recommendations keep predicted protein complexes containing at least four proteins and having a density larger than 0.7. In the final stage, the proteins in each cluster are evaluated for functional homogeneity using their annotations from the GO [23.17], and only those clusters are kept that contain at least one overrepresented gene ontology annotation. These steps allow some of the proteins to be left out entirely from protein complexes.

Although RNSC was shown to perform favorably on protein interaction data for the yeast *Saccharomyces cerevisiae* [23.20], it should be noted that the algorithm has a large number of parameters (including the search length, the number of repetitions, the frequency and length of diversification steps, the maximum number of clusters to consider, and the size of the tabu queue), all of which have to be tuned to the specific PPI network at hand. Even with careful tuning, the algorithm often converges to wildly different clusterings in different runs with the same parameter settings, which may require the evaluation of many different sets of predicted complexes to find the ones worthy of further small-scale follow-up experiments.

Markov Clustering

MCL (MCL) [23.22] is based on the idea that a random walk on the vertices of the graph that is started from an arbitrary node is more likely to spend more time walking within a densely connected region (i. e., a cluster) of the graph than walking between different regions, assuming that the moves are only allowed along the edges and the random walker cannot jump arbitrarily between vertices. MCL deterministically approximates transition probabilities of such random walks by iteratively using two operators on row-stochastic matrices (also called Markov matrices, hence the name of the algorithm) until a well-defined set of clusters emerge.

A formal, step-by-step description of the algorithm on undirected, unweighted graphs without isolated vertices is as follows:

1. Let $\boldsymbol{M} = (m_{ij})$ be the matrix of random walks on the graph $G(V, E)$; in other words, let m_{ij} be the probability that a random walker at node i moves to node j in one step. Since the random walker chooses each incident edge of node i with equal probability, $m_{ij} = A_{ij}/d_i$, where A_{ij} is 1 if vertices i and j are connected and zero otherwise, and d_i is the number of edges incident on vertex i. Note that each row of $\boldsymbol{M}$ sums up to 1, so $\boldsymbol{M}$ is row-stochastic.
2. Calculate the two-step random walk matrix by multiplying $\boldsymbol{M}$ with itself. This step is called the *expansion* step.
3. Inflate the probabilities of high-probability paths and deflate the probabilities of low-probability paths in $\boldsymbol{M}^2$. This is achieved by raising each element of $\boldsymbol{M}^2$ to a power $\gamma > 1$ and then re-normalizing the row sums of the result to 1 again to make the matrix row-stochastic. The re-normalized matrix is then stored in $\boldsymbol{M}'$. This step is called the *inflation* step.
4. If the difference between $\boldsymbol{M}'$ and $\boldsymbol{M}$ is less than a given threshold, the algorithm has converged. Otherwise we assign $\boldsymbol{M}'$ to $\boldsymbol{M}$ and continue from step 2 with another expansion-inflation cycle.

The result of the above process is a doubly idempotent matrix $\boldsymbol{M}$ that remains the same after an expansion-inflation iteration. It can be proven that the above procedure converges quadratically around the equilibrium state and it is conjectured that the process always converges if the input graph is symmetric [23.22]. In practical applications, the convergence is noticeable after 310 iterations. It can also be shown that the graph constructed from the equilibrium state matrix $\boldsymbol{M}$ consists of different connected directed components, each component having a star-like structure. The components are then interpreted as the clusters of the original graph.

The only parameter of the algorithm in the above description is the inflation exponent γ, whose values are typically between 1.2 and 5. The exponent controls the granularity of the obtained clustering; values close to 1 yield a few large clusters, while higher values generate a larger number of dense clusters. In practice, the algorithm also adds a pruning step after every expansion-inflation iteration, which removes entries close to zero from $\boldsymbol{M}$; this ensures that the matrix remains sparse and matrix multiplications can be performed efficiently, enabling MCL to scale up to graphs containing millions of vertices and edges. The algorithm can also be extended naturally to directed and weighted graphs. In directed graphs, we consider only the outgoing edges when building the initial random walk matrix, and add loop edges to each vertex to ensure the convergence of the algorithm. Weights can be incorporated by making the random walk biased towards edges with larger weights; the probability of moving from vertex i to vertex j in a single step then becomes proportional to w_{ij} instead of A_{ij}, while keeping $\boldsymbol{M}$ row-stochastic. A post-processing step is commonly used to discard clusters with only one or two members or a density that is smaller than the expected density of real protein complexes.

MCL was first applied to identify sets of evolutionally related proteins [23.21] and it quickly became very popular in the bioinformatics community. It was subsequently applied for the identification of yeast protein complexes from high-throughput data by *Krogan* et al. [23.7]. However, note that MCL still does not solve the problem of overlapping protein complexes as it puts every protein into one and only one of the clusters.

23.1.2 Overlapping Clustering Methods for PPI Networks

Besides traditional graph clustering techniques, researchers have also proposed alternative algorithms that have been tailored to PPI networks. A common property of these methods is that they are able to put one protein into more than one predicted complex, or to leave out a protein entirely from all the predicted complexes if there is not enough evidence to support the assignment of that protein into any of the complexes. In this section, we will describe three such algorithms: molecular complex detection

(MCODE) [23.25], the k-clique percolation method (also called CFinder [23.26]), and clustering by overlapping neighborhood expansion (ClusterONE [23.27]).

Molecular Complex Detection

The MCODE [23.25] algorithm takes a different approach to classical graph clustering algorithms: instead of partitioning the entire vertex set of the graph, it tries to identify locally dense regions and disregards those parts of the graph that are not likely to belong to any of the clusters. The algorithm consists of three steps: vertex weighting, complex prediction, and post-processing. The vertex weighting step assigns a weight to all the vertices that quantifies how likely a given vertex is to be a core of a putative protein complex. The complex prediction stage selects highly weighted vertices from the previous step and grows protein complexes in its neighborhood. The post-processing step expands the protein complexes found in the previous step to allow overlaps between them.

The keystone of the algorithm is the vertex weighting scheme, which is based on two components: 1) the so-called core-clustering coefficient of the vertices and 2) the highest k-core of the immediate neighborhood of the vertices. To understand these measures, we need a few definitions first.

Definition 23.1 Clustering Coefficient

The clustering coefficient C_i of a vertex i is defined as the number of edges between neighbors of a vertex, divided by the number of theoretically possible edges between neighbors. Formally, given a vertex i with k_i neighbors and m edges between those neighbors, the clustering coefficient C_i is given by

$$C_i = \frac{2m}{k_i(k_i-1)} \,. \tag{23.3}$$

One can think about the clustering coefficient as a local density measure; in fact, it is exactly the density of the subgraph spanned by the neighbors of a given vertex. Intuitively, vertices with high clustering coefficients are good candidates for proteins to be included in protein complexes as they have a densely connected neighborhood. However, many PPI networks contain hub proteins that interact with a diverse set of other proteins, potentially also including some that have only a single connection (the one towards the hub protein). These dangling links may rapidly decrease the clustering coefficient of a hub protein that could nevertheless have connections towards another, densely connected region as well. *Bader* and *Hogue* [23.25] resolved this problem by introducing the *k-core indices* and the *core-clustering coefficient*.

Definition 23.2 k-Core

The k-core of a graph $G(V; E)$ is a subset of vertices $V_0 \subseteq V$ such that the degrees of the subgraph spanned by V_0 are all larger than or equal to k.

Definition 23.3 k-Core Index

The k-core index of a vertex v in a graph G is the largest k such that v is the member of the k-core of G but not the $(k+1)$-core of G. The k-core index of a graph G is the largest k such that the k-core of G is not empty.

It is easy to see that the k-core index of vertices with d links is at most d, and the k-core of G is smaller than or equal to the maximal vertex degree in the graph.

Definition 23.4 Core-Clustering Coefficient

The core-clustering coefficient of a vertex v in a graph G is the density of the highest k-core of the subgraph spanned by v and its neighbors.

Note that the core-clustering coefficient is similar to the clustering coefficient measure in the sense that it is also a local density measure, but it does not suffer from the negative effect of dangling links. Given a highly connected hub protein v, the highest k-core of its neighborhood will not include the dangling links, and the density of this vertex set will not be affected negatively. The MCODE algorithm uses the product of the core-clustering coefficient of v and the highest k-core index in the immediate neighborhood of v as the weight of vertex v.

Once the weights of the vertices have been determined, MCODE starts to grow protein complexes from highly weighted vertices as listed:

1. Select the vertex with the largest weight in the graph that has not yet been included in any of the predicted complexes. This vertex is called the *seed vertex* and it is the only initial member of the complex being grown. We denote the seed vertex with v and its weight with w_v.
2. Add those vertices in the neighborhood of v to the complex being grown that are not added to any other complex yet and have a weight larger than $(1-\alpha)w_v$, where α is called the *vertex weight percentage*. Typical values of α are between 0 and 0.2.

3. Recursively keep on adding vertices adjacent to the complex being grown and not being in any other complex yet if their weight is larger than $(1-\alpha)w_v$.
4. If there are no more vertices that have not been considered yet and have not been added to any of the predicted complexes, go back to step 1. Otherwise, start the post-processing stage.

The complex prediction stage gives us a list of nonoverlapping complex candidates. Every protein in the network is assigned to *at most* one of the complex candidates, but it may also have happened that a protein was not assigned to any of the complexes. The final post-processing stage first discards predicted complexes without a 2-core (i. e., a subgraph where all the degrees are larger than or equal to 2), then tries to introduce overlaps between the complexes by an operation called *fluffing* and trims unnecessary proteins from complexes by another step called *haircut*. Both the fluffing and the haircut steps are optional, but the former is required if we wish to see overlaps between predicted complexes.

The fluffing step considers every vertex in every predicted complex one by one. For each vertex v, the process adds the neighbors of v to the complex containing v if the density of the subgraph spanned by v and its neighbors is above the value of the fluff parameter of the algorithm. At this stage, it is possible that a vertex u becomes part of two or more complexes if it is part of two or more dense neighborhoods. The haircut step removes vertices from complexes that are connected to the rest of the complex with only a single edge, effectively reducing each complex to its 2-core.

One of the advantages of the MCODE algorithm is that it can be run in an exploratory analysis mode where the dense neighborhood of a selected seed vertex is extracted from the network. This is achieved by growing a single cluster from the seed vertex and performing the additional fluffing or haircut operations on it. This mode is very useful for researchers who are interested in the role of a particular protein within the cell and its interactions with other proteins. On the other hand, MCODE tends to be very strict in the sense that proteins are usually very tightly connected in MCODE clusters and it usually misses smaller molecular complexes, especially if the experimental data is noisy and there are a lot of missing interactions in the input data. It is also not able to take edge weights into account, therefore PPI networks with associated confidence information for each interaction have to be binarized and low-confidence edges have to be discarded before an MCODE analysis, adding the optimal confidence threshold to the set of MCODE parameters to be tuned.

Complexes from k-Cliques: the CFinder Method

CFinder [23.26] was one of the first general-purpose overlapping clustering algorithms, which was later also applied successfully to biological networks [23.28]. The cluster definition of the algorithm is based on k-cliques, i. e., sets of vertices of size k where any pair of vertices is connected directly by an edge. The algorithm first constructs the k-clique reachability graph of the input graph G. The vertices of the k-clique reachability graph represent the k-cliques of the original graph, and two vertices in the reachability graph are connected by an edge if the corresponding k-cliques share at least $k-1$ vertices. The clusters of the original graph are then derived from the connected components of the k-clique reachability graph such that each cluster becomes the union of k-cliques in one connected component of the reachability graph. This approach naturally includes overlaps between vertices of the original graph as a vertex may be a member of multiple k-cliques, and these k-cliques can end up in different connected components of the reachability graph. The only parameter of the algorithm is the value of k, which controls the granularity of the clustering; larger values of k yield densely connected clusters. $k=2$ is equivalent to finding the connected components of the input graph, therefore k is always larger than 2 in practical applications of the algorithm.

One can easily recognize that it is enough to enumerate the maximal cliques of the original network that have at least k vertices instead of finding all k-cliques, as every subset of a maximal clique is also a clique, therefore a maximal clique of size n will be mapped to a connected subgraph consisting of $\binom{n}{k}$ vertices in the k-clique accessibility graph. Such subgraphs can be shrunk into a single vertex that will represent the whole maximal clique without affecting the connectivity properties of the k-clique accessibility graph.

The original CFinder publication [23.26] was not concerned with edge weights. Later on, a weighted extension of CFinder was proposed [23.29], which introduces a second parameter I, acting as an intensity threshold for the detected cliques. In the weighted CFinder algorithm, the product of the edge weights in a clique must exceed I in order to include that clique in the accessibility graph. This is computationally more prohibitive than the unweighted variant as we have to enumerate all k-subcliques of maximal cliques and check their intensities explicitly. In fact, the reference

implementation of CFinder did not provide results of a PPI network. Nevertheless, the CFinder algorithm is still a very promising approach to detecting complexes from unweighted PPI networks and it is also applicable to weighted PPI data after the selection of an appropriate weight threshold.

Clustering by Overlapping Neighborhood Expansion

Clustering by overlapping neighborhood expansion (or ClusterONE in short) [23.27] is a recent overlapping clustering algorithm that grows clusters from seed proteins, similarly to the MCODE algorithm [23.25], and also allows researchers to explore the densely connected region around a given seed protein. The key difference is that ClusterONE explicitly strives to make use of the confidence scores (weights) that are usually associated to interactions in PPI datasets instead of discarding them after a thresholding step. The core of the algorithm is a quality measure called *cohesiveness*, which quantifies how likely it is for a set of proteins to be a good complex candidate based on structural considerations. The definition of cohesiveness for a set of vertices $V_0 \subseteq V$ in a graph $G(V, E)$ is

$$f(V_0) = \frac{w^{\text{in}}(V_0)}{w^{\text{in}}(V_0) + w^{\text{bound}}(V_0) + p|V_0|} , \qquad (23.4)$$

where $w^{\text{in}}(V_0)$ is the total weight of internal edges in group V_0, $w^{\text{bound}}(V_0)$ is the total weight of edges that connect V_0 to the rest of the network, and $p|V_0|$ is a penalty term. The exact form of the cohesiveness function is based on the following considerations.

1. A good protein complex candidate should have many edges with large weights within the complex, hence $w^{\text{in}}(V_0)$ should be as large as possible.
2. Similarly, a good candidate should have only a few edges that connect the complex to the rest of the network, therefore $w^{\text{bound}}(V_0)$ should be small.
3. In PPI networks derived from experimental procedures, it is usually the case that not all the interactions are included in the network; some connections may be missing due to the limitations of the experiment. In the extreme case, it may happen that protein A in the PPI network is connected only to a putative complex with a single interaction, but in reality, protein A also interacts with other proteins not in the complex. Adding protein A to the complex is plausible if we know for sure that A interacts with a member of the complex and *with no other proteins*, but it is not justified if we know in advance that many connections of A are likely to be uncharted yet. The penalty term $p|V_0|$ accounts for this by assuming that every protein has p missing connections in addition to the ones in the PPI network. We note that this definition could easily be extended to employ different p values for different proteins based on biological assumptions, thus a well-studied protein may have a lower p value since it is less likely to possess undiscovered interactions.

Note that $f(V_0)$ has a minimum value of zero (when there are no internal edges) and a maximum value of 1 (when there are no external edges, there is at least a single internal edge and $p = 0$), which leads to an intuitive probabilistic interpretation. Suppose we are selecting a random edge from the graph such that selection probabilities are proportional to edge weights. In this case, $f(V_0)$ is the conditional probability of selecting an edge with *two* internal endpoints, given that at least *one* endpoint is internal. An additional pleasant property of $f(V_0)$ is that when $f(V_0) > 1/3$, vertices of the subgraph have more internal weight than external weight on average, satisfying the conditions of being a community in the weak sense [23.30].

ClusterONE builds on the concept of cohesiveness and consists of two major steps: 1) finding a set of groups with high cohesiveness and 2) merging redundant cohesive groups and post-processing the results. However, we also note that the algorithm is not tied to the specific form of the cohesiveness function we introduced above: the same framework may be used in conjunction with any alternative local quality measure defined for sets of vertices in a graph [23.31–33].

A straightforward way to find a subgraph with a high cohesiveness is to adopt a greedy strategy: starting from a single seed vertex, one can extend the group one vertex at a time so that the newly added vertex always increases the cohesiveness of a group as much as possible. Removals are also allowed if removing a vertex from the group increases its cohesiveness. This approach yields locally optimal cohesive groups in the sense that the cohesiveness of the group cannot be improved any more by adding or removing a single vertex. The procedure to grow a locally optimal cohesive group from a seed vertex is as listed:

1. Select a seed vertex v_0 and let $V_0 = \{v_0\}$. Set the step number $t = 0$.
2. Calculate the cohesiveness of V_t and let $V_{t+1} = V_t$.

3. For every external vertex v incident on at least one boundary edge, calculate the cohesiveness of $V' = V_t \cup \{v\}$. If $f(V') > f(V_{t+1})$, let $V_{t+1} = V'$.
4. For every internal vertex v incident on at least one boundary edge, calculate the cohesiveness of $V'' = V_{t+1} \setminus \{v\}$. If $f(V'') > f(V_{t+1})$, let $V_{t+1} = V''$.
5. If $V_t \neq V_{t+1}$, increase t and return to step 2. Otherwise, declare V_t a locally optimal cohesive group.

This process is repeated from different seeds in order to form multiple, possibly overlapping groups as follows. We first select the vertex of G with the highest degree as the first seed and grow a locally optimal cohesive group around it. After a locally optimal cohesive group is found, the next seed vertex will always be the one that has the highest degree among those that have not yet been included in any other cohesive group. It may happen that the original seed vertex is not included in the detected cohesive group (because it was removed earlier in step 4); in this case, the vertex is declared an outlier. The process continues until all vertices are either included in at least one cohesive group or declared as outliers.

As an illustration of the above steps, let us return to the example graph in Fig. 23.1. Assuming $p = 0$ (i. e., no presumed undiscovered connections), the cohesiveness of the marked set is 10/15. In steps 3 and 4, the algorithm can either extend the current set by adding C, F, or G, or contract the set by removing A, B, D, or E. The best move for the algorithm is to add C to the set, since it converts three boundary edges to internal ones (in other words, it decreases w^{bound} by 3 and increases w^{in} by 3) and does not bring in any new boundary edges. After adding C, the cohesiveness of the group increases to 13/15 and the group becomes locally optimal, as adding F would result in a cohesiveness of 14/17 and adding G would yield 14/18.

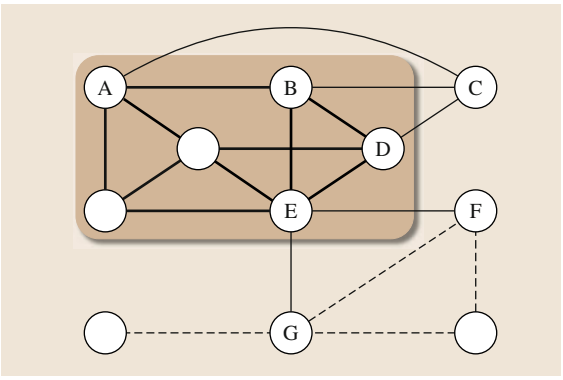

Fig. 23.1 Illustration of the greedy cohesive group detection process. The group itself is denoted by a *shaded background*. *Thick black edges* are internal, *thin black edges* are boundary edges, while *thin gray dashed edges* are completely external. Vertices *marked by a letter* are incident on at least one boundary edge, therefore only these vertices will be considered for addition or removal by the algorithm. The best choice is to extend the group by vertex C as it would convert three boundary edges to internal ones and would not add any additional boundary edges

The procedure outlined above will generate locally optimal cohesive groups that may overlap with each other to some extent. While the primary purpose of this procedure was, indeed, to generate overlapping clusters, some of the generated clusters will differ only in one or two proteins, which further complicates experimental validation of the results. It is, therefore, desirable to merge pairs of highly overlapping cohesive groups before declaring the clusters final. The algorithm uses the overlap score ω proposed in [23.25] to quantify the overlap between two protein sets A and B,

$$\omega(A, B) = \frac{|A \cap B|^2}{|A||B|} \,. \tag{23.5}$$

Pairs of groups having ω above a specified threshold are then merged in the predicted set of complexes. Finally, clusters smaller than a predefined size threshold or having a density less than a predefined density threshold are discarded.

23.1.3 Assessing the Quality of Complex Predictions

In this section, we compare the different techniques outlined in the previous section by benchmarking them on four PPI datasets for yeast. We first describe the datasets we are about to use in our tests and the set of reference complexes, then define some quality measures that will be employed to quantify how well a predicted set of complexes represents the reference complexes. Finally, we will test each of the algorithms and compare the quality scores obtained. We used the original implementations released by the authors of these algorithms and we followed the published protocols closely. Predicted complexes containing less than three proteins were discarded from the results of each algorithm unless the originally published protocol suggested otherwise: in the case of the RNSC algorithm, we therefore used a size threshold of 4 instead of 3.

Datasets and Reference Complexes

In our experiments, we have used the PPI networks for the yeast *Saccharomyces cerevisiae*, as published by *Krogan* et al. [23.7] and *Gavin* et al. [23.8], as well as the dataset of *Collins* et al. [23.12], which merges the Krogan and Gavin datasets using computational methods. The Krogan dataset was published in two variants in the original paper: the core dataset contained only highly reliable interactions (all having a confidence score larger than 0.273), while the extended dataset contained more interactions with less overall reliability (the smallest confidence score was 0.101). Both variants of the Krogan dataset were tested separately. Self-interactions and isolated proteins were discarded.

All the datasets that we used contained confidence values associated to each PPI. For the Gavin dataset, we re-scaled the original socio-affinity scores to the range [0, 1] for fairness to those algorithms that are not prepared for edge weights outside the range [0, 1]. Finally, in order to apply those algorithms that do not support edge weights (namely MCODE, RNSC, and CFinder), we first had to binarize the networks. This was done using the threshold values for the weights that were originally suggested by the authors of the datasets.

The set of reference complexes were extracted from the most recent version of the MIPS catalogue of protein complexes [23.34] (dated 18 May 2006). The MIPS dataset (Munich Information Center for Protein Sequences) is organized hierarchically: complexes in MIPS may consist of subcomplexes extending to at most five hierarchy levels deep. We considered MIPS categories containing at least 3 and at most 100 proteins as protein complexes, since these are plausible lower and upper bounds for protein complexes that are expected to appear in experimental PPI datasets. We also excluded MIPS category 550 and all its descendants, as these categories correspond to protein complexes that were predicted by computational methods and have not been confirmed experimentally.

In some cases, not all of the proteins in a given MIPS complex were found in a given benchmark dataset, and we cannot expect a protein complex detection algorithm to predict a complex that is only partially represented in the input data. To this end, MIPS complexes where less than half the proteins were represented in a given dataset were removed from the reference set for that dataset.

Clustering-Wise Sensitivity, Positive Predictive Value and Accuracy

In order to assess the performance of overlapping and nonoverlapping clustering algorithms within the same benchmark, we need to be able compare an arbitrary set of predicted complexes with a predefined gold standard complex set without relying on the assumption that a protein may belong to only one predicted and only one reference complex. The comparison is made difficult by the fact that often a match between a predicted complex and a gold standard one is only partial. Moreover, a gold standard complex can have a (partial) match with more than one predicted complex, and vice versa.

One of the first attempts to compare nonoverlapping and overlapping protein complex detection methods using a general quality score was published by *Brohée* and *van Helden* [23.35]. They introduced several measures, three of which will be used in our comparisons as well. These scores are the *clustering-wise sensitivity* (CWS), the *clustering-wise positive predictive value* (PPV) and the *geometric accuracy* (Acc).

CWS and PPV are based on the confusion matrix $\boldsymbol{T}$ between a predicted complex set and the set of reference complexes. Given n reference and m predicted complexes, let t_{ij} denote the number of proteins that are found both in reference complex i and predicted complex j, and let n_i denote the number of proteins in reference complex i. The CWS and PPV measures are then defined as

$$\mathrm{CWS} = \frac{\sum_{i=1}^{n} \max_{j=1}^{m} t_{ij}}{\sum_{i=1}^{n} n_i} , \tag{23.6}$$

$$\mathrm{PPV} = \frac{\sum_{j=1}^{m} \max_{i=1}^{n} t_{ij}}{\sum_{j=1}^{m} \sum_{i=1}^{n} t_{ij}} . \tag{23.7}$$

The Acc is the simply the geometric mean of CWS and PPV

$$\mathrm{Acc} = \sqrt{\mathrm{CWS} \times \mathrm{PPV}} . \tag{23.8}$$

Note that the CWS and PPV measures are very similar but not entirely symmetric; the denominator of CWS includes the total size of reference complexes, while the denominator of PPV is simply the sum of elements in the confusion matrix $\boldsymbol{T}$. When overlaps between complexes are allowed, n_i may be larger than, smaller than, or equal to the sum of row i in the confusion matrix (which we will denote with t_{i*} from now on), thus the denominator of CWS may also be larger than, smaller than, or equal to the denominator of PPV. This leads to a somewhat counter-intuitive behavior for these mea-

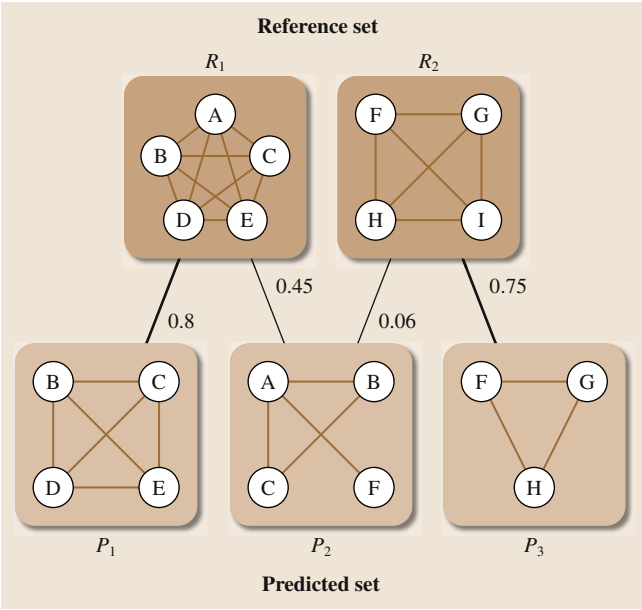

Fig. 23.2 Illustration of the maximum matching ratio between a reference and a predicted complex set. R_1 and R_2 are members of the reference set, while P_1, P_2 and P_3 are three predicted complexes. An edge connects a reference complex and a predicted complex if their overlap score is larger than zero. The maximum matching is shown by the *thick edges*. Note that P_2 was not matched to R_1 since P_1 provides a better match with R_1. The maximum matching ratio in this example is $(0.8+0.75)/2 = 0.775$

sures: in the case of a perfect agreement between the reference complexes and the set of predicted complexes, the CWS is equal to 1, but the positive predictive value may be lower when overlaps are present between the reference complexes. Since PPV is a component of the Acc score, this has a negative impact on the perceived performance of overlapping clustering algorithms when they are measured by the Acc score alone. It is even possible to construct a set of reference complexes such that a perfect clustering algorithm receives a lower score than a dummy algorithm that places every protein in a separate cluster; an example of this is given in the supplementary materials of [23.27].

The Maximum Matching Ratio

Recognizing the shortcomings of the Acc measure, *Nepusz* et al. introduced an alternative quality score called the *maximum matching ratio* (MMR) that is designed specifically for the comparison of overlapping clusterings [23.27]. The measure is based on the overlap score (23.5) and a maximum matching in a bipartite graph. To calculate the MMR, we begin by building a bipartite graph where the two sets of nodes represent the reference and predicted complexes, respectively, and an edge connecting a reference complex with a predicted one is weighted by the overlap score between the two (Fig. 23.2). We then select the maximum weighted bipartite matching on this graph; that is, we choose a subset of edges such that each predicted and reference complex is incident on at most one edge in the chosen set and the sum of the weights of such edges is maximal. In this way, the chosen edges represent an optimal assignment between reference and predicted complexes in a way that no reference complex is assigned to more than one predicted complex and vice versa. The MMR between the reference and the predicted complex set is then given by the total weight of the selected edges divided by the number of reference complexes. This ratio measures how accurately the predicted complexes represent the reference complexes.

Note that the maximum matching ratio divides the weight of the maximum matching by the number of reference complexes instead of the number of predicted complexes. This is motivated by the fact that the gold standard sets are very often incomplete [23.36], therefore a predicted complex that does not match any of the reference complexes may belong to a valid but previously uncharacterized complex.

Figure 23.2 shows an example to illustrate this concept. The maximum matching ratio offers a natural, intuitive way to compare predicted complexes with a gold standard, and it explicitly penalizes cases when a reference complex is split into two or more parts in the predicted set, as only one of its parts is allowed to match the original reference complex.

Benchmarks

The standard procedure for evaluating the performance of a machine-learning algorithm on some dataset starts by dividing the dataset into a training and a testing set. The parameters of the algorithm are then tuned on the training set, and the optimal parameters are used to calculate the final performance score of the algorithm on the testing set. However, this procedure assumes that the input dataset can be naturally decomposed into problem *instances* such that 1) each instance is a complete input of the machine-learning algorithm on its own and 2) each instance is independent of others. Unfortunately, neither of these assumptions hold for graph clustering algorithms in biological contexts, where we usually have a single network to work with. The network itself is indivisible, since removing a predefined fraction of

edges from a network changes its structural properties in a way that affects the outcome of a clustering algorithm significantly; in other words, removing some edges from a network is similar to adding noise to a feature vector in a standard machine-learning algorithm rather than to putting a set of problem instances aside in a testing set. Therefore, we cannot simply divide the input data into training and testing sets, and we cannot turn to the well-established methodology of k-fold cross-validation to evaluate the performance of a clustering method and to avoid the over-optimization of algorithm parameters to a given dataset or a given quality score [23.37]. To this end, we adopted the following counter-measures against overfitting in our benchmarks:

1. We tested each of the algorithms on four different datasets: three high-throughput experimental datasets [23.7, 8] and a computationally derived network that integrates the results of these studies [23.12].
2. We used more than one quality score to assess the performance of each algorithm: the fraction of matched complexes with a given overlap score threshold ($\omega \geq 0.25$), the Acc [23.35], and the maximum matching ratio that we proposed. These scores were combined into a composite quality score by taking the sum of the three individual scores.
3. For each algorithm *except* ClusterONE, the final results were obtained after having optimized the algorithm parameters to yield the best possible results as measured by the maximum matching ratio on the MIPS gold standard complex set [23.34], while the results for ClusterONE were obtained with one common parameter setting for all the datasets. Therefore, the scores of ClusterONE represent its performance when the method is adapted to a wider problem domain (i. e., detecting overlapping protein complexes from high-throughput experimental PPI networks in general), while the scores of other algorithms measure their performance when they are optimized to a *specific* dataset. It is thus rightly expected that these latter scores are optimistic estimates.

Figure 23.3 shows the results of our benchmarks, indicating that ClusterONE outperforms alternative approaches on the tested datasets, matching more complexes with a higher accuracy and providing a better one-to-one mapping with reference complexes in almost all the datasets. Only for the Gavin dataset did MCL achieve a better accuracy but a lower MMR, which can be explained by the bias of the accuracy score towards nonoverlapping clusterings, as discussed above. Also note that ClusterONE is more tolerant to the presence of low-confidence edges than other methods, as its maximum matching ratio on the Krogan extended dataset (which contains a high number of low-confidence interactions) is better than the maximum matching ratio of the runner-up MCL on the Krogan core dataset, which contains highly confident interactions only.

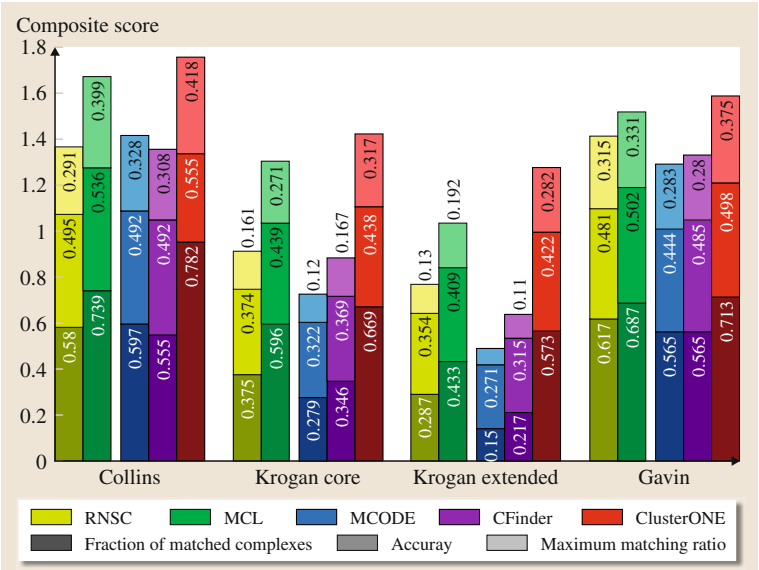

Fig. 23.3 Benchmark results of the tested algorithms on the MIPS gold standard. *Colors* correspond to the various algorithms, *shades of the same color* denote the individual components of the composite score of the algorithm (*dark* = fraction of matched complexes, *medium* = geometric accuracy, *light* = maximum matching ratio). The *numbers on the bars* show the exact scores for each component of the composite score. The total height of each column is the value of the composite score for a given algorithm on a given dataset. Larger scores are better

So far in this chapter we have focused our analysis on the structure of PPI networks to the discovery of those densely connected submodules that correspond to protein complexes. These dense subgraphs are one of the peculiar structural properties of PPI networks; however, they are not the only ones. In Sect. 23.2 we will try to capture the broader picture and provide models for the underlying organizational principles of PPI networks using random graphs. Moreover, we shall demonstrate how these models can be used for predicting putative novel interactions in PPI datasets.

23.2 Random Graph Models for Protein–Protein Interaction Networks

A random graph model is an algorithm that tells us how can one generate specific types of graphs using random numbers and a few predefined parameters. For instance, a simple random graph model could be specified as outlined here.

> *Take n vertices. Consider each pair of vertices once and for each such pair, add an edge between them with probability p.*

The motivation behind the usage of random graphs for modeling PPI networks is as follows. Assume that we are able to find a well-fitting random graph model for PPI networks. This has two immediate advantages. First, we can use the generating mechanism of the random graph models to *reason* about how real PPI networks have evolved and what the general organizational principles behind them are, and second, we can use the random graph model to *make predictions* about the existence of yet uncharted connections between proteins as we can simply *ask* the model about the probability of the existence of every edge in the network. For instance, if it turns out that the above-mentioned simple model (which is one of the variants of the Erdős–Rényi random graph model [23.38]) is the best fitting model for PPI networks, we could simply infer that the connections between proteins are essentially distributed at random, and the probability of finding a new interaction between two proteins that are not known to interact yet is equal to p. Such a discovery (were it to be taken seriously) would probably put an end to all further research into the structure of protein interaction networks, but as we will see later, this is luckily not the case and there are still plenty of unanswered questions about how PPI networks evolved and why they look the way they do.

In the next subsections, we will introduce several random graph models that are thought to be good models for the structure of PPI networks. A comparative analysis will then follow; the main purpose of this analysis will be to assess the predictive power of these models with the underlying assumption that a well-fitting random graph model should have the greatest predictive power among all of them. In particular, we will test the models on the PPI network of the yeast *Saccharomyces cerevisiae* and examine how accurately they are able to predict new connections in this network. Finally, we will present a case study where we apply the best model to identify putative connections in the interactome of *Homo sapiens*.

23.2.1 The Configuration Model

The configuration model was first proposed by *Molloy* and *Reed* [23.39], and was later applied to PPI networks by *Préulj* et al. [23.40] under the name *stickiness index based model*. The model assumes that the connections between proteins depend on the abundance of binding domains on the proteins themselves: two proteins are more likely to interact with each other if both of them have many and/or easily accessible binding domains. Formally, each protein has a stickiness index θ_i that determines the affinity of the protein to interact with others; the probability of an interaction between proteins i and j is then given by $\theta_i\theta_j$. Given a graph $G(V, E)$, one can easily calculate the likelihood of a given parameterization $\boldsymbol{\theta}$, i. e., the probability that the model generates G using these parameters

$$\mathcal{L}[\boldsymbol{\theta}|G(V,E)] = \prod_{(i,j)\in V\times V} \left[\theta_i\theta_j A_{ij} + (1-\theta_i\theta_j)(1-A_{ij})\right] , \tag{23.9}$$

where A_{ij} is 1 if and only if vertices i and j are connected, zero otherwise, and the product iterates over *unordered* pairs of vertices.

When fitting the model to an observed network with n proteins, θ_i is usually set to $k_i / \sum_{j=1}^{n} k_j$, where k_i is the number of observed interactions that involve protein i. This ensures that the expected degree of each protein in the networks generated by the configuration

model is equal to the observed degree of the protein in the original PPI network. Note that the model does not preserve the degree distribution on the level of individual networks; a protein with k_i observed interactions may have more or less interactions in any single generated random network, but it will have k_i interactions on average across the entire ensemble of random networks if we generate enough random instances. Also note that loop edges have a nonzero probability in this model, and explicitly disallowing loop edges would skew the expected degrees downwards, but the probability of loop edges is usually negligible for sparse networks.

23.2.2 Geometric Random Graphs

The configuration model was shown to be a good fit to PPI networks [23.40]. However, it has been noted in the literature that PPI networks may have a bias against connections between hub proteins (i. e., proteins with high degree) [23.41], and therefore the basic assumption of the configuration model may not be appropriate to describe connections between hubs. To this end, geometric random graphs [23.42] were suggested later as an alternative model for PPI networks [23.13].

The generation process of geometric random graphs first embeds the n vertices of the graph in an d-dimensional unit hypercube and connects vertex pairs with probabilities depending on the distances between them; more precisely, the probability of an edge between proteins i and j is equal to $f(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$, where $\boldsymbol{x}_i$ is the position of the point corresponding to protein i and $\|\dots\|^2$ denotes the Euclidean norm. The positions of the vertices and the function f can be considered as parameters of the model. In the simplest case, $f(x)$ is 1 if x is less than a predefined distance threshold δ and 0 otherwise – this function would connect all pairs of proteins that are closer to each other than δ. From a biological point of view, the model assumes that the proteins can be placed in an abstract d-dimensional *chemical space*, and proteins with similar chemical properties have a higher chance of interacting with each other. The likelihood of a given parameterization on a given graph $G(V, E)$ is given as

$$\mathcal{L}(\boldsymbol{X}, f \mid G) = \prod_{(i,j)\in V\times V} \Big\{ f(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) A_{ij} + [1 - f(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)](1 - A_{ij}) \Big\} \,. \tag{23.10}$$

The geometric random graph model is very flexible and it is naturally able to generate graphs with a clustered structure. This property is very useful to us as we have already seen that PPI networks tend to contain tightly connected modules of proteins. However, placing the vertices in a d-dimensional space and choosing the function f in a way that produces a given PPI network with a high probability is a challenging task. The algorithm proposed in [23.13] and [23.15] consists of two major steps:

1. Estimate the position matrix $\boldsymbol{X}$ by applying multidimensional scaling in d dimensions to a matrix $\boldsymbol{M} = (m_{ij})$ where m_{ij} is equal to the square root of the length of the shortest path from protein i to protein j in the network. The resulting configuration of the vertex positions minimizes the total squared elementwise difference between $\boldsymbol{M}$ and the distance matrix of the placed vertices in the d-dimensional space.
2. Assume that f is the mixture of three Gaussian curves,

$$f(x) = \sum_{i=1}^{3} m_i \left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \mathrm{e}^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \right) ,$$

where μ_i and σ_i are the means and standard deviations of the corresponding Gaussian distributions and the m_i values are the mixing parameters. The optimal values of m_i, μ_i and σ_i for $i = 1, 2, 3$ are then found by expectation-maximization.

The application of the geometric random graph model to de-noise PPI datasets (i. e., to predict new protein interactions and to point out false positive interactions in experimental data) shows that the model has good predictive power for PPI prediction without using any additional data (e.g., protein sequences or the 3-D structure of the proteins) [23.15].

23.2.3 Stochastic Blockmodels

Stochastic blockmodels originate from the field of sociology [23.43,44] where they have been applied to model social structures and to gain insights into the structure of observed relationships between individuals. However, the model has many applications outside the field of sociology; for instance, stochastic blockmodels have been used to predict yet uncharted connections in a network model of the visuotactile cortex of the macaque monkey [23.45,46].

Stochastic blockmodels assume that each vertex of the graph corresponds to exactly one of d discrete types, and the probability of the existence of an edge depends

on the types of the two endpoints of the edge. The two principal parameters of the model are a type vector $\boldsymbol{t} = [t_1, t_2, \ldots, t_n]$ and a symmetric probability matrix $\boldsymbol{P} = [p_{ij}]$ of size $d \times d$; in this setup, an edge is generated between vertices i and j with probability p_{t_i,t_j}. The likelihood of a given parameterization of a stochastic blockmodel for a graph $G(V, E)$ is given by

$$\mathcal{L}(d, \boldsymbol{t}, \boldsymbol{P} \mid G) = \prod_{(i,j)\in V\times V} [p_{t_i,t_j} A_{ij} + (1 - p_{t_i,t_j})(1 - A_{ij})] \,. \tag{23.11}$$

Stochastic blockmodels can also exhibit a wide variety of structural patterns; for instance, high p_{ii} values generate clusters comprising of vertices of type i, while high p_{ij} $(i \neq j)$ values generate nearly complete bipartite subgraphs between vertices of type i and type j. These patterns are particularly important for protein interaction networks, as clusters may correspond to well-known protein complexes (to which we have dedicated the entire first half of this chapter), and nearly bipartite subgraphs may easily be generated by the lock-and-key mechanism of protein binding [23.47], where it is assumed that specific pairs of binding sites act like locks and keys, and a protein possessing a "key-type" binding site is, therefore, able to interact with almost any other protein that possesses the corresponding "lock-type" binding site. This is naturally represented in stochastic blockmodels via the type vector and the preference matrix – a high p_{ij} $(i \neq j)$ value in the preference matrix then identifies a putative lock–key pairing between proteins of type i and type j.

In order to fit a stochastic blockmodel to an observed network G, one has to find the optimal values for the number of types d, the type of each vertex (i. e., the vector $\boldsymbol{t}$) and the connection probabilities between types (the matrix $\boldsymbol{P}$). A possible way to do this is via maximum likelihood (ML) or maximum a posteriori (MAP) estimates. Assuming that d and the type vector $\boldsymbol{t}$ are known, one can estimate the probability matrix $\boldsymbol{P}$ by maximizing the likelihood $\mathcal{L}(\boldsymbol{P} \mid G, k, \boldsymbol{t})$, and it is easy to show that the maximum likelihood estimate for $\boldsymbol{P}$ is

$$p_{rs} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}\delta_{t_i r}\delta_{t_j s}}{\sum_{i=1}^{n}\delta_{t_i r}\sum_{j=1}^{n}\delta_{t_j s}} \,, \tag{23.12}$$

where A_{ij} is 1 if and only if vertices i and j are connected (zero otherwise), and δ_{xy} is 1 if and only if $x = y$ (zero otherwise). Consequently, $\boldsymbol{P}$ can be treated as a quantity that depends on the type vector $\boldsymbol{t}$ and the graph G instead of a parameter on its own. The problem is then simplified to finding the optimal type vector $\boldsymbol{t}$ and the number of required types, which is typically achieved using Markov chain Monte Carlo methods. The fitting algorithm that we propose to use will be described later.

One disadvantage of the stochastic blockmodel is the expected degree distribution of the network it generates; it can be shown that the degrees of the vertices are drawn from a mixture of Poisson distributions, and the mixing coefficients are determined by the relative frequencies of vertex types in the type vector $\boldsymbol{t}$ [23.48]. This distribution is able to mimic many other degree distributions from uniform to scale-free if the number of types is large enough, but increasing the number of types arbitrarily has the risk of overfitting the model to the observed network. In the most extreme case, the number of vertex types is equal to the number of vertices in the network (therefore each vertex has its own type), and the preference matrix is equal to the adjacency matrix of the original graph. Needless to say, such a configuration reproduces the original network perfectly, but does not tell us anything about the hidden structure of the network as it essentially treats each vertex as a unique entity. A possible solution to this problem is to control the number of vertex types using one of the well-known techniques for model selection (e.g., minimizing *Akaike*'s information criterion [23.49] or the Bayesian information criterion [23.50] instead of the log-likelihood), but it leaves us with a degree distribution that is inconsistent with the original network. To this end, the degree-corrected variant of stochastic blockmodels was introduced recently [23.16], which allows the generated networks to reproduce the degree distribution of the original network even with a small number of types, at the expense of some extra parameters.

The degree-corrected stochastic blockmodel combines the idea of the original blockmodel with the configuration model that we already described in Sect. 23.2.1. In this model, each vertex i has a type index t_i and an *activity* parameter θ_i, which is similar to the *stickiness* index in the configuration model. The probability matrix $\boldsymbol{P}$ is replaced by a rate matrix $\boldsymbol{\Omega}$. The *number* of edges between vertices i and j is then drawn from a Poisson distribution with rate parameter $\theta_i\theta_j\omega_{t_i,t_j}$; in other words, the number of edges between two vertices depends both on their individual activity levels and their affinity towards each other. Note that this model may generate multiple edges between two vertices, which lack a suitable interpretation in case of PPI networks, but one can simply collapse multiple edges into a single one in a post-processing step.

The likelihood function for degree-corrected stochastic blockmodels is [23.16]

$$\mathcal{L}(d, \boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\Omega}) = \prod_{(i,j)\in V\times V} \frac{(\theta_i\theta_j\omega_{t_i,t_j})^{A_{ij}}}{A_{ij}!} \exp(-\theta_i\theta_j\omega_{t_i,t_j}) \times \prod_{i\in V} \frac{(\frac{1}{2}\theta_i^2\omega_{t_i,t_i})^{A_{ii}/2}}{(A_{ii}/2)!} \exp\left(-\frac{1}{2}\theta_i\theta_i\omega_{t_i,t_i}\right) .$$

Note that parts of the vertex activity vector $\boldsymbol{\theta}$ that belong to vertices within the same group are arbitrary to within a multiplicative constant, since multiplying all θ_i where i is in a given group g by a constant c can be counterbalanced by dividing $\omega_{t_i,j}$ and ω_{j,t_i} by c (where $j \neq t_i$) and ω_{t_i,t_i} by c^2. Without loss of generality, we can thus normalize θ_i such that the sum of θ_i values for all i within the same group is equal to 1. This allows one to re-write the likelihood function as

$$\mathcal{L}(d, \boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\Omega}) = C \times \prod_{i\in V} \theta_i^{k_i} \times \prod_{r=1}^{d}\prod_{s=r}^{d} \omega_{rs}^{m_{rs}/2} \cdot \exp\left(-\frac{1}{2}\omega_{rs}\right) , \qquad (23.13)$$

where k_i is the degree of vertex i, C is a constant that depends solely on the adjacency matrix of the graph and not on the model parameters, and m_{rs} is the number of edge vertices of type r and s if $r \neq s$ and *twice* the number of edges between vertices of type r if $r = s$.

Similarly to the uncorrected stochastic blockmodel described above, one can factor out the rate matrix $\boldsymbol{\Omega}$ and the vertex activity vector $\boldsymbol{\theta}$ and simply treat them as dependent variables on $\boldsymbol{t}$ and the input graph G. The maximum likelihood estimates for $\boldsymbol{\Omega}$ and $\boldsymbol{\theta}$ are as [23.16]

$$\hat{\theta}_i = \frac{k_i}{\sum_{j=1}^{d} m_{t_i j}}, \quad \hat{\omega}_{ij} = m_{ij} , \qquad (23.14)$$

where m_{ij} is the number of edges between vertices of type i and j if $i \neq j$ and *twice* the number of edges between vertices of type i if $i = j$. The above choice yields an expected degree k_i for vertex i, thus the expected degree sequence of the generated graphs will match the degree sequence of the graph the model is being fitted to (neglecting loop and multiple edges).

Fitting the uncorrected and the degree-corrected stochastic blockmodels to a given network with a given type count d can then be achieved using Markov chain Monte Carlo methods. Note that in both cases, the only parameter we really have to optimize is the type vector $\boldsymbol{t}$; once it is known, all the remaining parameters ($\boldsymbol{P}$ in the uncorrected case or $\boldsymbol{\Omega}$ and $\boldsymbol{\theta}$ in the degree-corrected case) can be calculated quickly. To optimize $\boldsymbol{t}$, we consider a Markov chain whose states consist of different type configurations, and propose a simple strategy to update the state of the Markov chain based on the Metropolis–Hastings algorithm as listed:

1. We choose a single vertex and select a new candidate type for this vertex.
2. We calculate the difference in the log-likelihood between the old and the new configuration. Since the changes introduced in the log-likelihood function by a single point mutation are local, not all the terms in the log-likelihood function are affected, and the difference can be calculated easily without re-evaluating the entire log-likelihood function.
3. When $\log \mathcal{L}_{\text{new}} > \log \mathcal{L}_{\text{old}}$, we accept the change unconditionally. Otherwise, we accept the change with probability $\exp(\log \mathcal{L}_{\text{new}} - \log \mathcal{L}_{\text{old}})$. If the change was rejected, the Markov chain stays in the same type configuration as in the previous step.
4. If the Markov chain did not converge to a stationary distribution yet, return to step 1.

Two points have to be clarified in the algorithm in order to make it fully specified. The first point is the problem of initialization, i. e., the starting configuration should we use for $\boldsymbol{t}$ to obtain a result with a high log-likelihood quickly. The second point is the problem of termination, i. e., the conditions we use to determine whether the Markov chain consisting of the type vectors in consecutive steps has converged or not. In the simplest case, the chain can simply be initialized by a random type vector. However, our experiments with the uncorrected stochastic blockmodel [23.46, 48] showed that the following, greedy initialization heuristic yields results in similar quality in a considerably shorter amount of time, as it brings the initial state of the Markov chain closer to the mode of the distribution:

1. First, we select a random type vector $\boldsymbol{t}$.
2. Each vertex is then given the opportunity to change its type if it is able to increase its own contribution to the log-likelihood by doing so. Type changes for all the vertices are done in a synchronous manner; in other words, each vertex assumes that none of the other vertices change their types when they decide on their own new types they wish to belong to.
3. When the new configuration is equivalent to the old one, we consider the initialization to have converged

to a steady state and we start the Markov chain from the new configuration. Otherwise, the initialization algorithm returns to step 2.

The problem of termination can be solved by calculating the average log-likelihoods of two consecutive blocks of a large number of samples and stop the Markov chain when the difference between the two averages becomes smaller than a predefined threshold. This is motivated by the fact that the average log-likelihood of a block of samples is an estimator of the entropy of the distribution of likelihood values, and the entropy should be constant in a Markov chain that has converged to its stationary distribution. In our experiments, we use a block size of $2^{13} = 8192$ samples and an entropy difference threshold of 1.

Note that the fitting procedure described above can be used only if d (the number of types) is specified in advance. In most practical model fitting problems, d is unknown and must be chosen automatically. In our experiments, we have employed the following strategy to select the optimal d value:

1. Start from $d = 1$.
2. Run the optimization process outlined above with the given d until the Markov chain converges.
3. Let $\log \mathcal{L}_d$ denote the best log-likelihood encountered in step 2.
4. If $d < \sqrt{n}$, the square root of the number of proteins, increase d by one and go back to step 2.
5. For each d, calculate *Akaike*'s information criterion [23.49] for the model with the best log-likelihood as

$$\mathrm{AIC}_d = 2r - 2\log \mathcal{L}_d \,, \tag{23.15}$$

where r is the number of parameters in the model. For the original blockmodel, $r = d(d+1)/2 + n + 1$, while for the degree-corrected blockmodel, $r = d(d+1)/2 + 2n + 1$.
6. Let $d = \mathrm{argmin}_i \, \mathrm{AIC}_i$.

Finally, note that using $d = 1$ (i. e., one vertex type only) for the degree-corrected stochastic blockmodel reproduces the configuration model as $\hat{\theta}_i = k_i/2m$ and $\hat{\omega}_{11} = 2m$ (where m is the number of edges), producing $p_{ij} = k_i k_j/2m$.

23.2.4 Hierarchical Random Graphs

The last random graph model that we are going to investigate in this chapter is the hierarchical random graph model [23.51]. This model has already been used to describe the structure of several real-world networks ranging from food webs to terrorist contact networks. One of the advantages of this model is that it avoids the problem of group count selection (unlike stochastic blockmodels). Instead of groups, the model uses a binary dendrogram, where the leaves of the dendrogram correspond to the vertices of the graph being generated. Since each intermediate node of the dendrogram joins two leaves or other intermediate nodes, it is easy to see that the dendrogram has n leaves and $n-1$ intermediate nodes. These intermediate nodes have associated probability values (denoted by a vector p_k, where k is an index in $[1; n-1]$ that identifies an intermediate node), and the probability of an edge between vertices i and j is equal to $p_{\mathrm{LCA}(\mathcal{D},i,j)}$, where $\mathrm{LCA}(\mathcal{D}, i, j)$ is the lowest common ancestor node of leaves i and j in the dendrogram $\mathcal{D}$. The likelihood function of the model is then

$$\begin{aligned}\mathcal{L}[\mathcal{D}, \boldsymbol{p} \mid G(V, E)] &= \prod_{(i,j)\in V\times V} [p_{\mathrm{LCA}(\mathcal{D},i,j)} A_{ij} \\ &\quad + (1 - p_{\mathrm{LCA}(\mathcal{D},i,j)})(1 - A_{ij})] \,. \end{aligned} \tag{23.16}$$

From a biological point of view, hierarchical random graph models assume that the proteins in a PPI network evolved from a remote common ancestor, and the affinity of two proteins towards each other depends on their ancestry; in particular, the node in the dendrogram where the two proteins diverged from each other. However, the model does not require that closely related proteins have a larger connection probability than remotely related proteins. In general, the idea of protein ancestry is well-known in biology, and several projects are dedicated to grouping proteins based on evolutionary relatedness. For instance, *protein families* in the SCOP (structural classification of proteins) database [23.52] *are clearly evolutionarily related*, and *superfamilies* contain *p roteins that have low sequence identities, but whose structural and functional features suggest that a common evolutionary origin is probable* [23.53]. The basic assumption of hierarchical random graph models is, therefore, entirely plausible in the context of PPI networks.

The fitting procedure of hierarchical random graphs is very similar to the technique that we employed for stochastic blockmodels. It is easy to recognize that the probability vector $\boldsymbol{p} = (p_i)$ can be considered a dependent variable on the structure of the dendrogram as follows. Let L_i be the set of leaf nodes in the left subtree of intermediate node i, R_i be the set of leaf nodes

in the right subtree and m_i is the number of edges going between L_i and R_i. The maximum likelihood estimate of p_i conditioned on the dendrogram is then given as $m_i/(|L_i||R_i|)$. Therefore, if we know the dendrogram, we can easily calculate the corresponding probabilities that maximize the log-likelihood. This makes it possible to employ Markov chain Monte Carlo optimization strategies again to find the optimal dendrogram structure and probability vector for a given network. The details of the process are given in [23.51].

23.2.5 Evaluating the Predictive Power of Random Graph Models

As mentioned earlier in Sect. 23.2, our primary motivation for using random graph models to analyze PPI networks is to facilitate the prediction of novel, yet uncharted interactions between proteins by fitting a model to the known part of a PPI network and then *asking* the fitted model for the probability of putative connections. In this section, we will, therefore, evaluate the predictive power of the random graph models presented in earlier sections.

The input data of our benchmarks will be the protein interaction dataset of *Collins* et al. [23.12], which combines two earlier experimental PPI datasets [23.7,8] into a self-consistent view of the interactome of the yeast *Saccharomyces cerevisiae*. Since it is based on two independent experimental datasets, the data of Collins et al. is thought to be one of the most accurate PPI networks for yeast. The dataset includes 5437 proteins and assigns a confidence score for each protein pair. Collins et al. also proposed a confidence score threshold that yields 9074 high-confidence interactions between 1622 proteins. These interactions can be turned into a graph containing 1622 vertices and 9074 edges. Since the geometric random graph model cannot handle disconnected networks, we extracted the largest connected component of this graph, leading to a network of 1004 proteins and 8323 interactions.

Since the fitting algorithms of random graph models require the whole network as input data, the standard machine learning approach (which divides the input data into training and testing sets) cannot be applied here either; for instance, it is not possible to calculate the multidimensional scaling required for fitting a geometric random graph model with only a fragment of the entire adjacency matrix. However, training and testing on the same dataset is likely to result in overfitting and, therefore, should be avoided. *Kuchaiev* et al. [23.15] circumvented the problem by training the random graph models on the entire high confidence network and testing the model predictions on the set of mid and low-confidence edges, where they considered the mid-confidence edges as further positive examples. During the training process, mid-confidence edges are considered nonexistent, therefore a method that is prone to overfitting will learn the mid-confidence edges as negative examples and will have a lower performance in the test phase, where the mid-confidence edges are assumed to exist. The threshold for the confidence score of mid-confidence edges was set to 0.2, yielding 5352 mid-confidence edges and 489 831 negative testing examples.

Predictions for the geometric random graph model were obtained by calculating the probability of the existence of all test edges according to the models. For the stochastic blockmodels and the hierarchical random graph model, 10 000 samples were taken from the Markov chains after convergence, with sampling probability 0.1, and the final predicted probability for an edge were calculated by averaging the predicted probabilities for this edge across all the 10 000 samples. For instance, if an edge was predicted to exist with probability 0.7 by half of the samples and with probability 0.4 by the other half of the samples, the final predicted probability for that edge became 0.55. The geometric random graph model used five dimensions according to the recommendations of *Kuchaiev* et al. [23.15]. To account for the imbalance between the number of positive and negative examples in the test set, the results were evaluated both by ROC curves (Fig. 23.4) and by precision-recall curves (Fig. 23.5) [23.54].

Before drawing any conclusions on the difference between the AUC (area under the curve) values, we performed paired permutation tests on the curves [23.55] to assess the significance of these differences. All the differences between ROC curves were deemed significant (largest p-value $< 10^{-5}$), which allows us to conclude that the degree-corrected blockmodel shows superior performance to other alternative models in terms of predictive power, as measured by both the ROC and the precision-recall curves. Similarly, the configuration model showed the worst performance. The remaining three models were ranked differently by the two curves. The area under the ROC curve was larger for the geometric random graphs, while the precision-recall curve ranked the hierarchical random graph and the uncorrected stochastic blockmodel before geometric random graphs. The difference can be explained by looking at the precision-recall curves: the geometric random graph shows a consistently low

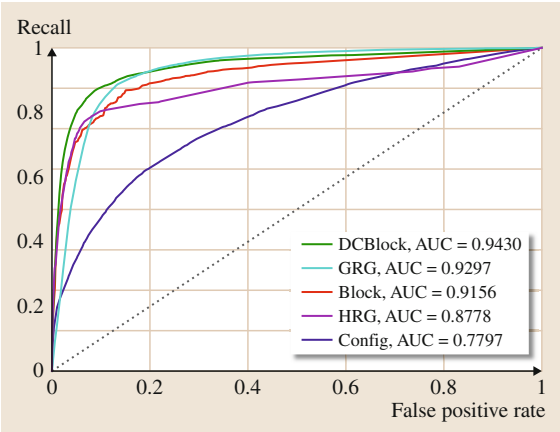

Fig. 23.4 ROC curves for different random graph models on the test set derived from the PPI network of *Collins* et al. [23.12]. Block = stochastic blockmodel, DCBlock = degree-corrected stochastic blockmodel, GRG = geometric random graph, HRG = hierarchical random graph, Config = configuration model

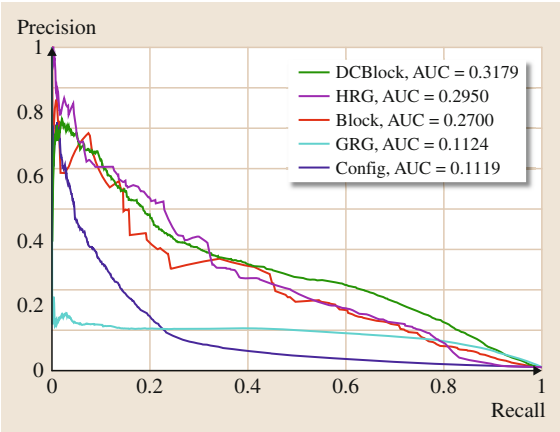

Fig. 23.5 Precision-recall curves for different random graph models on the test set derived from the PPI network of *Collins* et al. [23.12]. AUC = area under the curve, larger is better. Block = stochastic blockmodel, DCBlock = degree-corrected stochastic blockmodel, GRG = geometric random graph, HRG = hierarchical random graph, Config = configuration model

precision for almost any level of recall, while the performance of the other models generally improves as the recall rate is lowered. At a reasonable recall of 35%, the precision achieved by both stochastic blockmodels and the hierarchical random graph model is about 33%, which is more than twice as high as the precision reported for geometric random graphs by *Kuchaiev* et al. [23.15] (precision = 15%, confirmed by our replicate).

These numbers also allow us to estimate the performance of these random graph models on human PPI data. Recent estimates of the size of the human PPI network indicate 154 000–369 000 interactions among 20 000–25 000 proteins [23.56]. At a recall of 35% and a precision of 33%, we can expect 53 900 novel interactions among only 163 000 predicted ones with stochastic blockmodels or hierarchical random graphs; by comparison, the same number of novel interactions would be expected among 359 000 interactions for the geometric random graph model.

23.2.6 Predicting Novel Interactions in the Human Interactome

The previous section has shown us that degree-corrected stochastic blockmodels outperform alternative random graph models in terms of predictive power. To show the potential of this approach in predicting new PPI, we will apply this graph model to a human protein interaction dataset derived from the BioGRID database [23.57]. The network contained 29 328 physical interactions among 8123 proteins and was constructed as listed:

1. We downloaded version 3.0.66 of the BioGRID database. Note that we intentionally used an older version of BioGRID because this allowed us to confirm some of the predictions by looking them up in a later version of BioGRID.
2. We selected all the physical interactions where both interactors correspond to *Homo sapiens* (species ID: 9606).
3. We kept those interactions annotated by an evidence code denoting a physical interaction between two proteins.
4. We extracted the largest connected component of the resulting network and used this for our experiments.

We fitted a degree-corrected stochastic blockmodel to the above network and used a predicted edge probability threshold of 0.6 to derive a set of predicted interactions, i. e., every connection that was not present in BioGRID but had a predicted probability larger than 0.6 was considered as a putative novel interaction. There were 217 such interactions in the entire network. We then used the GO project [23.17] and KEGG [23.18] to provide biological support for these interactions.

The GO project [23.17] aims to assemble a controlled vocabulary that can be used to annotate biological entities (genes, proteins, etc.) in three different aspects: biological process (BP), cellular component (CC), and molecular function (MF). Terms in the GO (GO) are then connected to each other via directed relations such as *is-a* or *part-of* to form three directed acyclic graphs (DAGs), one for each aspect of the tree. Figure 23.6 shows a subpart of the DAG corresponding to the cellular CC aspect, with all terms that the term *nucleoplasm* is directly or indirectly related to.

Each gene or protein may have multiple annotations in each of the three aspects. Since the GO is hierarchical and each term may have multiple subterms connected by *is-a* relations (e.g., nucleoplasm *is-a* nuclear part), GO annotations may be *direct* (when a GO term is directly assigned to an entity) or indirect (when the term is not assigned directly to an entity but such a relation can be inferred using the inference rules between the GO relations). For instance, the human gene *CDC25B* (and its corresponding protein Cdc25bp) is *directly* annotated by 9 BP, 6 CC and 4 MF terms in the GO at the time of writing, indicating its relatedness to cell division (term ID: GO:0051301 in the BP aspect), protein binding (term ID: GO:0005515 in the MF aspect), and the nucleoplasm (term ID: GO:0005654 in the CC aspect), among others. It is also annotated *indirectly* to superterms of these 19 terms, for instance:

- *CDC25B* is in a nuclear part (GO:0044428), because it is in the nucleoplasm (by direct annotation) and nucleoplasm *is-a* nuclear part.
- *CDC25B* is in an intracellular organelle part (as inferred above) and nuclear part *is-an* intracellular organelle part.
- *CDC25B* is in the nucleus (GO:0005634), because it is in a nuclear part (as inferred above), which is *part-of* the nucleus.

It is reasonable to assume that interacting proteins are situated in the same cellular component and/or participate in the same biological process, and the GO annotations and inference rules give us an opportunity to quantify what fraction of our predicted interacting protein pairs share annotations in the relevant aspects of the GO tree.

Similarly, KEGG [23.18] maintains (among others) an assignment between proteins and the biological pathways they participate in. We can assume that proteins in the same pathway correspond to the same biological process, therefore a shared KEGG pathway annotation for two proteins indicates their relatedness just like a shared GO annotation.

To this end, we calculated the fraction of predicted interaction pairs that 1) share at least one *direct* GO cellular component or biological process annotation or 2) participate in the same KEGG pathway in order to assess the biological plausibility of our predictions. We considered all the *Homo sapiens* pathways from KEGG and all the direct GO annotations ignoring cases when two proteins shared only the root GO terms (GO:0005575: cellular_component or GO:0008150: biological_process). Proteins not corresponding to any of the KEGG pathways were excluded from the KEGG score calculation, and proteins not having any annotations in the GO were excluded from the GO score calculation. In the following paragraphs, we report the fractions as percentages and also calculate the significance score of these fractions by comparing them with the same scores for randomly drawn protein pairs under the assumption of a hypergeometric model.

204 out of our 217 predicted interactions had GO annotations for both of the proteins involved. 171 out of these 204 (83.82%) of these pairs had at least one shared direct GO annotation in either the biological process (BP) or the cellular component (CC) aspect of the tree. The probability of observing such an extreme result by chance is less than 2.7×10^{-38} according to the hypergeometric test we have performed. However, since direct cellular component annotations may sometimes correspond to fairly broad categories (such as *intracellular*) due to lack of information about a given protein, and being in the same cellular component does not necessarily mean that the two proteins interact, we repeated the analysis with the BP aspect of the GO tree only. 187 of our predictions had associated BP annotations for both proteins of the interaction pair, and 82 (43.85%) of them shared at least one BP annotation. The probability of observing such a high number of shared BP annotations when drawing protein pairs at random is less than 4.2×10^{-46}. A similar analysis on the KEGG pathways showed that 111 out of the 217 predicted interactions had KEGG pathway information for both proteins involved, and 63 out of 111 (56.75%, p-value $< 4.6\times10^{-43}$) shared at least one KEGG pathway. These results together confirm the biological significance of our results.

Finally, we compared our predictions with a later version of BioGRID (version 3.1.72) that was published after we have finished our analyses on the BioGRID network and also with the most recent

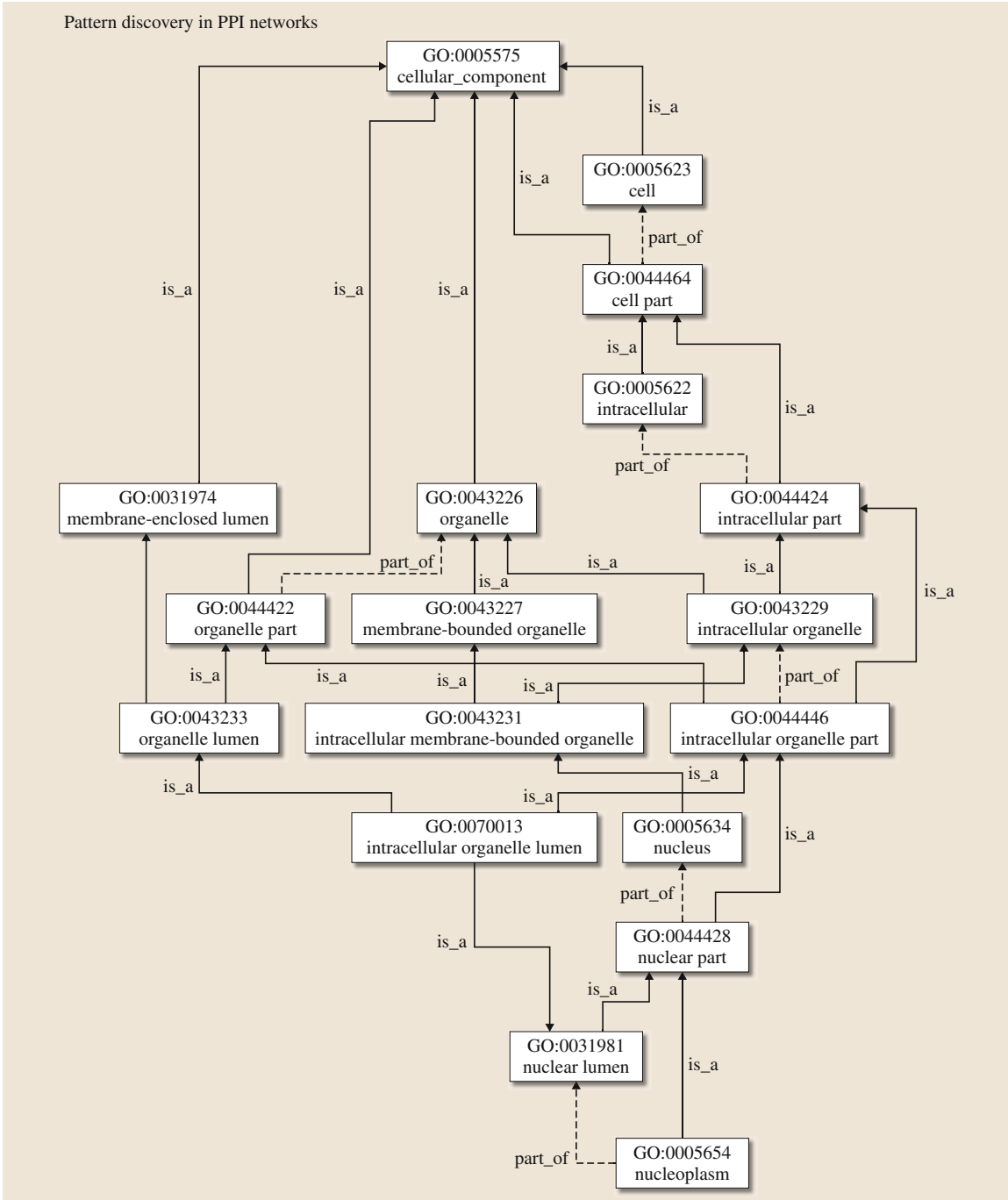

Fig. 23.6 A subpart of the GO, showing terms that the term *nucleoplasm* (at the *bottom* of the figure) is related to. *Solid edges* denote *is_a*, *dashed edges* denote *part_of* relations

version of the Human Protein Reference Database (HPRD [23.58]), an alternative repository of confirmed PPI in humans. Six out of the 217 predicted interactions (PIN1-UBC, ESR1-RELA, MYC-SMARCA2, MYC-SMARCC1, EP300-SMAD3 and SMAD2-UBC) were found in BioGRID 3.1.72 (p-value $< 4.8 \times 10^{-17}$), and the most recent version of HPRD provided confirmation for a further 28 interactions.

23.3 Conclusions

In this chapter, we have presented several computational methods to extract meaningful biological information from PPI networks. The structure we were able to unravel in these networks allowed us to detect the presence of protein complexes (sets of proteins that interact with each other more frequently than with the rest of the network) and even to pinpoint potentially novel interactions in PPI data that were probably overlooked by experimental methods.

We have presented several state-of-the-art techniques to extract protein complexes from PPI networks. Some of these methods (e.g., MCODE [23.25] or RNSC [23.20, 23]) work on the unweighted network only, while others can make use of the confidence scores associated to the interactions (e.g., MCL [23.21, 22] or ClusterONE [23.27]). More recent methods explicitly allow overlaps between the detected complexes [23.26, 27], which is an important feature of complexes found in real PPI networks. We have demonstrated that incorporating weights and the presence of overlaps into the clustering process improves the accuracy of the results of these methods.

In the second part of the chapter, we have given an overview of several random graph models that can be used to model the structure of PPI networks and also to make predictions about false negative or false positive interactions. The latter takes advantage of the fact that after having fitted a random graph model to an experimentally or computationally derived PPI network, we can simply *ask* the random graph model about the probability of the existence of every single connection in the network. Nonexistent connections with a high predicted probability are then candidates for false negative interactions, while existent connections with a very low predicted probability may be treated as false positives. We have shown that stochastic blockmodels [23.16, 43, 44, 46], especially its degree-corrected variant [23.16] have a larger predictive power than several alternative random graph models [23.13, 15, 40, 51] when evaluated on a PPI network derived from two experimental datasets [23.7, 8] and integrated using computational techniques [23.12].

However, despite all the recent advances both in the experimental techniques and the computational methods being used to analyze the produced data, there are still several open questions. For the problem of protein complex detection, it is important to note that some of the detected complexes are always bound to be the byproduct of random noise in the input data, and such noise is mostly unavoidable. Therefore, it would be important to devise computational techniques that are able to estimate the statistical significance of each detected complex in order to help biologists prioritize putative novel complexes for follow-up experiments. For the problem of modeling PPI networks, it is time to go back to the experimental data and see whether we can map the principles of the underlying assumptions of the best random graph models to the underlying biological processes. By being able to do so, we would be able to learn new facts about the biology of protein interactions. At the same time, we should check if all that we know about the biology of the problem is already included in our models and modify them, if necessary, in order to make them more realistic and thus more predictive.

References

23.1 P. Uetz, L. Giot, G. Cagney, T. Mansfield, R. Judson, J. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, J. Rothberg: A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae, Nature **403**(6770), 623–627 (2000)

23.2 T. Ito, K. Tashiro, S. Muta, R. Ozawa, T. Chiba, M. Nishizawa, K. Yamamoto, S. Kuhara, Y. Sakaki: Toward a protein–protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins, Proc. Natl. Acad. Sci. USA **97**(3), 1143–1147 (2000)

23.3 L. Giot, J. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, Y. Li, Y. Hao, C. Ooi, B. Godwin, E. Vitols, G. Vijayadamodar, P. Pochart, H. Machineni, M. Welsh, Y. Kong, B. Zerhusen, R. Malcolm, Z. Varrone, A. Collis, M. Minto, S. Burgess, L. McDaniel, E. Stimpson, F. Spriggs, J. Williams, K. Neurath, N. Ioime, M. Agee, E. Voss, K. Furtak, R. Renzulli, N. Aanensen, S. Carrolla, E. Bickelhaupt, Y. Lazovatsky, A. DaSilva, J. Zhong, C. Stanyon, R. Finley, K. White, M. Braverman, T. Jarvie, S. Gold, M. Leach, J. Knight, R. Shimkets, M. McKenna, J. Chant, J. Rothberg: A protein interaction map of Drosophila melanogaster, Science **302**(5651), 1727–1736 (2003)

23.4 S. Li, C. Armstrong, N. Bertin, H. Ge, S. Milstein, M. Boxem, P. Vidalain, J. Han, A. Chesneau, T. Hao, D. Goldberg, N. Li, M. Martinez, J. Rual, P. Lamesch, L. Xu, M. Tewari, S. Wong, L. Zhang, G. Berriz, L. Jacotot, P. Vaglio, J. Reboul, T. Hirozane-Kishikawa, Q. Li, H. Gabel, A. Elewa, B. Baumgartner, D. Rose, H. Yu, S. Bosak, R. Sequerra, A. Fraser, S. Mango, W. Saxton, S. Strome, S. Van Den Heuvel, F. Piano, J. Vandenhaute, C. Sardet, M. Gerstein, L. Doucette-Stamm, K. Gunsalus, J. Harper, M. Cusick, F. Roth, D. Hill, M. Vidal: A map of the interactome network of the metazoan C. elegans, Science **303**(5657), 540–543 (2004)

23.5 U. Stelzl, U. Worm, M. Lalowski, C. Haenig, F. Brembeck, H. Goehler, M. Stroedicke, M. Zenkner, A. Schoenherr, S. Koeppen, J. Timm, S. Mintzlaff, C. Abraham, N. Bock, S. Kietzmann, A. Goedde, E. Toksöz, A. Droege, S. Krobitsch, B. Korn, W. Birchmeier, H. Lehrach, E. Wanker: A human protein–protein interaction network: A resource for annotating the proteome, Cell **122**(6), 957–968 (2005)

23.6 J. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. Berriz, F. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, N. Klitgord, C. Simon, M. Boxem, S. Milstein, J. Rosenberg, D. Goldberg, L. Zhang, S. Wong, G. Franklin, S. Li, J. Albala, J. Lim, C. Fraughton, E. Llamosas, S. Cevik, C. Bex, P. Lamesch, R. Sikorski, J. Vandenhaute, H. Zoghbi, A. Smolyar, S. Bosak, R. Sequerra, L. Doucette-Stamm, M. Cusick, D. Hill, F. Roth, M. Vidal: Towards a proteome-scale map of the human protein–protein interaction network, Nature **437**(7062), 1173–1178 (2005)

23.7 N. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. Tikuisis, T. Punna, J. Peregrin-Alvarez, M. Shales, X. Zhang, M. Davey, M. Robinson, A. Paccanaro, J. Bray, A. Sheung, B. Beattie, D. Richards, V. Canadien, A. Lalev, F. Mena, P. Wong, A. Starostine, M. Canete, J. Vlasblom, S. Wu, C. Orsi, S. Collins, S. Chandran, R. Haw, J. Rilstone, K. Gandi, N. Thompson, G. Musso, P. St. Onge, S. Ghanny, M. Lam, G. Butland, A. Altaf-Ui, S. Kanaya, A. Shilatifard, E. O'Shea, J. Weissman, C. Ingles, T. Hughes, J. Parkinson, M. Gerstein, S. Wodak, A. Emili, J. Greenblatt: Global landscape of protein complexes in the yeast Saccharomyces cerevisiae, Nature **440**(7084), 637–643 (2006)

23.8 A. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. Jensen, S. Bastuck, B. Dumpelfeld, A. Edelmann, M. Heurtier, V. Hoffman, C. Hoefert, K. Klein, M. Hudak, A. Michon, M. Schelder, M. Schirle, M. Remor, T. Rudi, S. Hooper, A. Bauer, T. Bouwmeester, G. Casari, G. Drewes, G. Neubauer, J. Rick, B. Kuster, P. Bork, R. Russell, G. Superti-Furga: Proteome survey reveals modularity of the yeast cell machinery, Nature **440**(7084), 631–636 (2006)

23.9 N. Pržulj, D. Corneil, I. Jurisica: Modeling interactome: Scale-free or geometric?, Bioinformatics **20**(18), 3508–3515 (2004)

23.10 L. Lu, Y. Xia, A. Paccanaro, H. Yu, M. Gerstein: Assessing the limits of genomic data integration for predicting protein networks, Genome Res. **15**(7), 945–953 (2005)

23.11 H. Yu, A. Paccanaro, V. Trifonov, M. Gerstein: Predicting interactions in protein networks by completing defective cliques, Bioinformatics **22**(7), 823–829 (2006)

23.12 S.R. Collins, P. Kemmeren, X.C. Zhao, J.F. Greenblatt, F. Spencer, F.C. Holstege, J.S. Weissman, N.J. Krogan: Toward a comprehensive atlas of the physical interactome of Saccharomyces cerevisiae, Mol. Cell Proteomics **6**, 439–450 (2007)

23.13 D. Higham, M. Rašajski, N. Pržulj: Fitting a geometric graph to a protein–protein interaction network, Bioinformatics **24**(8), 1093–1099 (2008)

23.14 H. Yu, P. Braun, M. Yildirim, I. Lemmens, K. Venkatesan, J. Sahalie, T. Hirozane-Kishikawa, F. Gebreab, N. Li, N. Simonis, T. Hao, J. Rual, A. Dricot, A. Vazquez, R. Murray, C. Simon, L. Tardivo, S. Tam, N. Svrzikapa, C. Fan, A. de Smet, A. Motyl, M. Hudson, J. Park, X. Xin, M. Cusick, T. Moore, C. Boone, M. Snyder, F. Roth, A. Barabási, J. Tavernier, D. Hill, M. Vidal: High-quality binary protein interaction map of the yeast interactome network, Science **322**(5898), 104–110 (2008)

23.15 O. Kuchaiev, M. Rašajski, D. Higham, N. Pržulj: Geometric de-noising of protein–protein interaction networks, PLoS Comp. Biol. **5**(8), e1000454 (2009)

23.16 B. Karrer, M.E.J. Newman: Stochastic blockmodels and community structure in networks, Phys. Rev. E **83**(1 Pt 2), 016107 (2011)

23.17 M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, M. Harris, D. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. Matese, J. Richardson, M. Ringwald, G. Rubin, G. Sherlock: Gene ontology: Tool for the unification of biology, Nat. Genet. **25**(1), 25–29 (2000)

23.18 M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, Y. Yamanishi: KEGG for linking genomes to life and the environment, Nucl. Acids Res. **36**(Database issue), D480–4 (2008)

23.19 B. Alberts, A. Johnson, J. Lewis, M. Raff: *Molecular Biology of the Cell*, 4th edn. (Garland Science, New York 2002), Chap. 6, p. 342

23.20 A. King, N. Pržulj, I. Jurisica: Protein complex prediction via cost-based clustering, Bioinformatics **20**(17), 3013–3020 (2004)

23.21 A.J. Enright, S.V. Dongen, C.A. Ouzounis: An efficient algorithm for large-scale detection of protein families, Nucl. Acids Res. **30**(7), 1575–1584 (2002)

23.22 S. van Dongen: Graph clustering via a discrete uncoupling process, SIAM J. Matrix Anal. Appl. **30**, 121–141 (2008)

23.23 A. King: *Graph Clustering with Restricted Neighborhood Search*, Master's thesis (University of Toronto, Toronto 2004)

23.24 F. Glover, M. Laguna: *Tabu Search* (Kluwer Academic, Dordrecht 1997)

23.25 G.D. Bader, C.W. Hogue: An automated method for finding molecular complexes in large protein interaction networks, BMC Bioinformatics **4**, 2 (2003)

23.26 G. Palla, I. Derényi, I. Farkas, T. Vicsek: Uncovering the overlapping community structure of complex networks in nature and society, Nature **435**(7043), 814–818 (2005)

23.27 T. Nepusz, H. Yu, A. Paccanaro: Detecting overlapping protein complexes from protein–protein interaction networks, Nat. Methods **9**(5), 471–472 (2012)

23.28 B. Adamcsek, G. Palla, I. Farkas, I. Derényi, T. Vicsek: CFinder: Locating cliques and overlapping modules in biological networks, Bioinformatics **22**(8), 1021–1023 (2006)

23.29 I. Farkas, D. Ábel, G. Palla, T. Vicsek: Weighted network modules, New. J. Phys. **9**, 180 (2007)

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

23.31 A. Clauset: Finding local community structure in networks, Phys. Rev. E **72**, 026132 (2005)

23.32 J. Baumes, M. Goldberg, M. Magdon-Ismail: Efficient Identification of Overlapping Communities, LNCS **3495**, 27–36 (2005)

23.33 F. Luo, J.Z. Wang, E. Promislow: Exploring local community structures in large networks, Web Intell. Agent Syst. **6**(4), 387–400 (2008)

23.34 H.W. Mewes, C. Amid, R. Arnold, D. Frishman, U. Güldener, G. Mannhaupt, M. Münsterkötter, P. Pagel, N. Strack, V. Stümpflen, J. Warfsmann, A. Ruepp: MIPS: Analysis and annotation of proteins from whole genomes, Nucl. Acids Res. **32**(Database issue), D41–44 (2004)

23.35 S. Brohée, J. van Helden: Evaluation of clustering algorithms for protein–protein interaction networks, BMC Bioinformatics **7**, 488 (2006)

23.36 R. Jansen, M. Gerstein: Analyzing protein function on a genomic scale: The importance of gold-standard positives and negatives for network prediction, Curr. Opin. Microbiol. **7**(5), 535–545 (2004)

23.37 A.L. Boulesteix: Over-optimism in bioinformatics research, Bioinformatics **26**, 437–439 (2009)

23.38 P. Erdős;, A. Rényi: On random graphs, Publ. Math. **6**, 290–297 (1959)

23.39 M. Molloy, B. Reed: A critical point for random graphs with a given degree sequence, Random Struct. Algorithms **6**, 161–179 (1995)

23.40 N. Pržulj, D. Higham: Modelling protein–protein interaction networks via a stickiness index, J. R. Soc. Interface **3**(10), 711–716 (2006)

23.41 S. Maslov, K. Sneppen: Specificity and stability in topology of protein networks, Science **296**(5569), 910–913 (2002)

23.42 M.D. Penrose: *Random Geometric Graphs*, Oxford Studies in Probability, Vol. 5 (Oxford Univ. Press, Oxford 2003)

23.43 P. Holland, K.B. Laskey, S. Leinhardt: Stochastic blockmodels: Some first steps, Soc. Netw. **5**, 109–137 (1983)

23.44 T.A.B. Snijders, K. Nowicki: Estimation and prediction for stochastic blockmodels for graphs with latent block structure, J. Classif. **14**(1), 75–100 (1997)

23.45 L. Négyessy, T. Nepusz, L. Kocsis, F. Bazsó: Prediction of the main cortical areas and connections involved in the tactile function of the visual cortex by network analysis, Eur. J. Neurosci. **23**(7), 1919–1930 (2006)

23.46 T. Nepusz, L. Négyessy, G. Tusnády, F. Bazsó: Reconstructing cortical networks: Case of directed graphs with high level of reciprocity, Bolyai Soc. Math. Stud. **18**, 325–368 (2008)

23.47 J.L. Morrison, R. Breitling, D.J. Higham, D.R. Gilbert: A lock-and-key model for protein–protein interactions, Bioinformatics **22**(16), 2012–2019 (2006)

23.48 T. Nepusz: Data mining in complex networks: Fuzzy communities and missing link prediction. Ph.D. Thesis (Budapest University of Technology and Economics, Budapest 2008)

23.49 H. Akaike: Likelihood and the Bayes procedure. In: *Bayesian Statistics*, ed. by J.M. Bernardo, M.H. De Groot, D.V. Lindley, A.F.M. Smith (Valencia Univ. Press, Valencia 1980)

23.50 G.E. Schwarz: Estimating the dimension of a model, Ann. Stat. **6**(2), 461–464 (1978)

23.51 A. Clauset, C. Moore, M.E.J. Newman: Hierarchical structure and the prediction of missing links in networks, Nature **453**, 98–101 (2008)

23.52 A. Murzin, S. Brenner, T. Hubbard, C. Chothia: SCOP: A structural classification of proteins database for

the investigation of sequences and structures, J. Mol. Biol. **247**(4), 536–540 (1995)
23.53 http://scop.mrc-lmb.cam.ac.uk/scop/intro.html (last accessed May 16, 2011)
23.54 J. Davis, M. Goadrich: The relationship between precision-recall and ROC curves, ICML '06: Proc. 23rd Int. Conf. Mach. Learn. (ACM, New York 2006) pp. 233–240
23.55 S. Swamidass, C. Azencott, K. Daily, P. Baldi: A CROC stronger than ROC: Measuring, visualizing and optimizing early retrieval, Bioinformatics **26**(10), 1348–1356 (2010)
23.56 G. Hart, A. Ramani, E. Marcotte: How complete are current yeast and human protein-interaction networks?, Genome Biol. **7**(11), 120 (2006)
23.57 C. Stark, B. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, M. Tyers: BioGRID: A general repository for interaction datasets, Nucl. Acids Res. **34**(Database issue), D535–9 (2006)
23.58 T. Keshava Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, A. Venugopal, L. Balakrishnan, A. Marimuthu, S. Banerjee, D. Somanathan, A. Sebastian, S. Rani, S. Ray, C. Harrys Kishore, S. Kanth, M. Ahmed, M. Kashyap, R. Mohmood, Y. Ramachandra, V. Krishna, B. Rahiman, S. Mohan, P. Ranganathan, S. Ramabadran, R. Chaerkady, A. Pandey: Human Protein Reference Database – 2009 update, Nucl. Acids Res. **37**(Database issue), D767–72 (2009)