Using structure to improve a noisy PPI network

(now semi-coherent!)

Douglas Myers-Turnbull
dmyerstu@ucsd.edu

June 26, 2013

Abstract

Motivation: Protein-protein interaction (PPI) networks are often derived from noisy high-throughput experimental methods such as yeast two-hybrid screening. Consequently, PPI networks are typically both are cumbersome and suffer from low accuracy.

Methods: Here we present a novel method (StructNA) for both reducing the size of PPI networks and improving their accuracy by finding homologs in the network whose interactions are consistent. Homologs are identified in a species-independent way using protein structural information. The method takes as input a single network. It identifies homologs, then merges fully homologous proteins and assigns confidences to interactions that are consistent among homologs. It considers both degree of homology and similarity of network topology for both the merging and scoring.

Results: The method was able to correctly identify conserved interactions both across species and within individual species using data from EBI IntAct compared with benchmark sets derived from Interactome3D. On average, this resulted in a large fraction of interaction confidences in the network being modified, and a reduction of network size between 5% and 10%.

Availability: The algorithm was implemented in Java and is licensed under the Apache License version 2.0. It is freely available for download at `https://github.com/dmyersturnbull/network_merge`.

Introduction

Background

Protein-protein interaction (PPI) networks are often derived from high-throughput experimental assays such as Tandem affinity purification [13], and yeast two-hybrid screening [5]. These techniques are used because they are cost-effective, but they frequently lead to the inclusion of false interactions and missing interactions, and they are unable to accurately determine the probability of interactions.

PPI networks are modeled as undirected graphs in which vertices denote proteins and edges denote protein–protein interactions. Edge weights are used to describe the probability or certainty of an interaction. Psuedographs (which permit self-loops) allow for intramolecular interactions, and hypergraphs allow for multi-molecular interactions.

A number of algorithms have been developed to decrease noise in PPIs by identifying homologs. These algorithms solve a specific formulation of a *network alignment problem*. The objective of this class of

problems is, given two networks (n networks in the case of a multiple network alignment), to define a mapping between vertices in the networks that maximizes their overlap. Here, "overlap" can be given a number of definitions, but it usually incorperates both extent of homology between the aligned proteins and similarity of the network topology in the neighborhood of those proteins. Network alignment algorithms generally assume that homology is one-to-one in order to make the problem more tractable.

There are a number of limitations inherent to these approaches. First, homology is not one-to-one in real biological systems. Consequently, network alignment algorithms fail to identify many homology relationships that are biological and should be used to inform interactions. Second, these algorithms cannot identify or use homology relationships between proteins within the same species. Finally, all current network alignment algorithms rely on sequence alignments to identify homologous pairs. Since the data describes physical interactions between proteins, sequence alignment is ill-suited for deciding homology.

Approach

We propose an algorithm for improving PPI networks that differs from network alignment algorithms in three ways:

1. It does not require that homology is one-to-one.
2. It can identify homology relationships within the same species.
3. It is based on structural information rather than sequence information, making it more suitable for use with PPIs.

The specific objective is twofold. The primary objective is to assign a confidence to each interaction based on its conservation among any homologs of its participants, and a secondary objective is to merge homologous interactions to simplify the description of the network.

The method uses the Structural Classification of Proteins (SCOP) [10] and structural alignment based on Combinatorial Extension [14] to introduce edges in the graph that indicate homology. It then uses this information to update the probability of interactions that are shared between homologous proteins. Finally, it collapses degenerate vertices via edge contraction, where degeneracy is decided by the presence of a clique whose members share interactions. Clique-finding is performed naively via the Bron–Kerbosch algorithm [3], which was found to be sufficiently fast for biologically real networks.

Overview

Definitions

We define two graphs: $(V, Int(G))$ and $(V, Hom(G))$, where the vertex set V consists of proteins and is shared between both graphs, $(V, Int(G))$ is an interaction graph, and $(V, Hom(G))$ is a homology graph. Thus edges $e \in Int(G)$ denote interactions, and edges $e \in Hom(G)$ denote homology relations. The graph $G = (V, Int(G) \cup Hom(G))$ is a combined network consisting of both types of edges. For any $v \in Int(G)$, let $Int(v)$ be the set of neighbors of v in $Int(G)$. We define $Hom(v)$ in the same way.

Process

The algorithm consists of three major steps, which are each described more fully in the sections below.

Given an interaction network $G = (V, Int(G))$, build a network $G' = (V, Int(G') \cup Hom(G'))$ by using any available information (e.g. homology databases or alignment) to add edges to $Hom(G')$ and to increase the weights of those edges. At the end of this process, remove any edge from $Hom(G')$ with weight less than some threshold τ.

For this process, we let $\alpha(a, b)$ denote the weighted score of an alignment that has been performed between vertices a and b. Similarly, we let $\rho(a, b)$ denote the score of the most specific homology relation r identified between nodes a and b. For example, if a and b are from the same SCOP fold but not the same SCOP superfamily, $\rho(a, b)$ will be given by this relation.

Second, generate a network $G'' = (V, Int(G''))$ by using edges in $Hom(G')$ to update edge weights (probabilities) in $Int(G)$. Use both the probability of the shared interaction and the probability of the homology for both interaction participants in this process.

Remove any edge from $Hom(G')$ with weight less than some threshold ζ. Generate a network $G''' = (V', Int(G''') \cup Hom(G'''))$ by merging nodes contained in cliques of $(V, Hom(G''))$ in which every vertex in the clique is associated with the same set of neighbors in $(V, Int(G''))$.

Algorithm

Input

- A graph $G = (V, Int(G) \cup Hom(G))$
- Edge weights given by $Prob(e)$ for all $e \in Int(G)$
- Edge weights given by $S(e)$ for all $e \in Hom(G)$
- An homology scoring function $\Phi : V \times V \to [0, 1]$, where Φ is a metric on the space V
  Computationally, Φ will calculate alignments and perform database lookups as needed.
- A "pre-crossing" cutoff $\tau \in [0, 1]$
- A "pre-merging" cutoff $\zeta \in [0, 1]$, where $\zeta \geq \tau$
- A crossing search depth parameter $\xi \in \mathbb{N}$

StructNA(G, Φ, τ, ζ, ξ)

1. Parse network.
2. For every pair of vertices (a, b), add an edge (a, b) to $Hom(G)$, with label $\Phi(a, b)$.
3. For every edge $e \in G$ such that $S(e) < \tau$, remove e.
4. For every edge $(u, v) \in Int(G)$:
   (a) Let $I = \emptyset$, and let $Q_a = 0, X_a = Y_a = 0 \; \forall \; a \in V$.
   (b) Run breadth-first search on $(V, Hom(G))$ from u and v concurrently to generate vertex labelings X and Y, respectively. Stop after reaching depth ξ.
   (c) For both $R = X$ and $R = Y$, when visiting a vertex b via edge e:
      i. If $R_b = 0$, let $R_b := 1$.
      ii. Let $Q := Q + \log S(e)$.

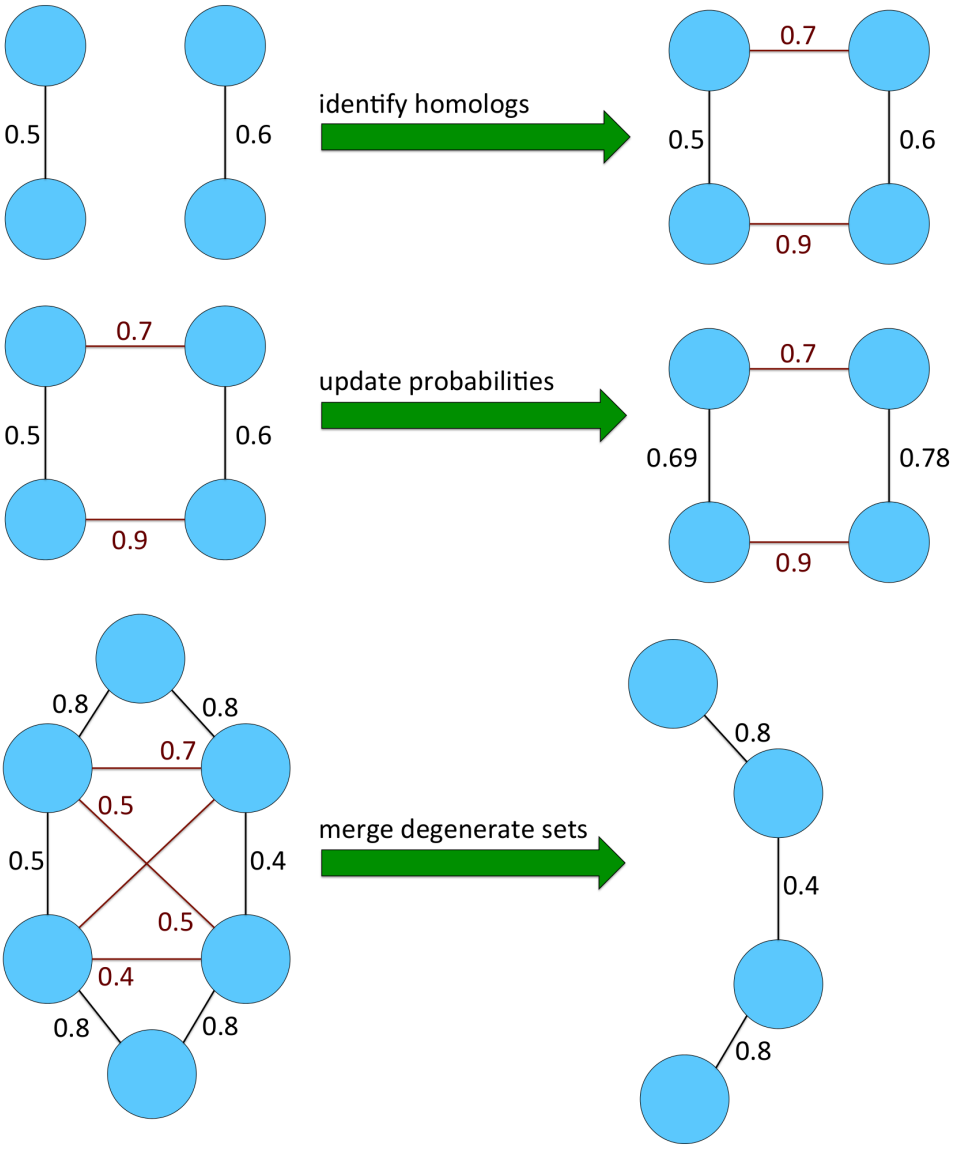

Figure 1: The three major steps of StructNA. 1. Identify homologous pairs (red lines) using homolog databases and structural and sequences alignments. 2. Increase the probabilities of interactions that are conserved across homologs. 3. Merge vertices in cliques whose interactions are equivalent.

iii. Let $R_b := R_b - exp(Q)$.

(d) For every edge $(x, y) \in Int(G)$ such $X_x \neq 0$ and $Y_y \neq 0$, let $Prob((u, v)) := Prob((u, v)) + 1 - X_x Y_y Prob((x, y))$.

5. For every edge $e \in G$ such that $S(e) < \zeta$, remove e.

6. For each connected component G_i in G:

   (a) Run the Bron–Kerbosch algorithm on G_i to enumerate all maximal cliques $C = \{c_1 \dots c_x\}$, for clique sizes $\gamma(C_j) > 1 \; \forall j$.

   (b) Partition C into sets $D = d_1 \dots d_z$ such that $\forall \; v \in d_i, u \in d_j$, $Int(v) \neq Int(u)$, $\forall \; i, j$, and $Int(v) = Int(u) \; \forall u, v \in d_i \; \forall i$.

   (c) For each $d \in D$, merge by edge contraction every vertex in d, modifying the reference graph G.

7. Output the network $(V, Int(G))$.

Steps 2–3 build the homology graph $(V, Hom(G))$. Steps 4 updates the probabilities of conserved interactions. Step 6 merges degenerate interactions and interactors.

Initial scoring of interactions

In a pre-processing step not listed in the above algorithm, interactions are given initial probabilities according to the number and type of experiment that identified them. In particular, we assign an interaction X a probability of:

$$Prob(X) = Prob\left(\bigvee_{x \in X} x\right) = 1 - \prod_{x \in X} [1 - \kappa(x) Prob(x)],$$

where each $x \in X$ is an experiment that determined interaction X, and $\kappa(x) \in [0, 1]$ is the quality of the experiment that determined x. κ ranged from 0.5 for yeast two-hybrid to 1.0 for protein complex co-immunoprecipitation, which is often considered a gold standard in protein–protein interaction detection [7]. The full listing for κ is available as a supplemental (aka SFP).

Establishing homology

In the first step, we identify homologs in the network. These are understood simply as proteins that are expected to have sufficient structural similarity to participate in some of the same chemistry and interact with other proteins in similar ways. In this way, homology is taken to be a species-independent property. A precise definition is decided by the alorithm based on the thresholds τ and ζ. We deliberately conflate probability of homology and degree of homology. This is because our model is concerned with determined interactions that are conserved among homologs, which is probabalistic by nature. Thus we cannot reasonably discern between the two.

A typical input network consists of a list interactions occurring between two polypeptide chains, where the chains are described by identifiers for UniProt [4]. In cases where the Protein Data Bank [1] contains at least one structure for that chain, a PDB Id and is found for that UniProt Id. From that, a SCOP

[10] domain corresponding to the PDB Id and chain are found. This selection process is problematic for multi-chain domains, for which we associate the chain with a domain arbitrarily.

We rely on both alignment and information from third-party databases to establish homology. Structural information is preferred over sequence information where it is available. First, scores are assigned using SCOP by using the most specific SCOP relationship found. The particular scoring system was not chosen in a rigourous way but reflects the frequency of relationships in SCOP 1.75B. It is listed below:

class	fold	superfamily	family	protein	species	domain
0.0	0.0045	0.010	0.029	0.23	0.75	1.0

In the case where a protein is not associated with a SCOP domain, the Pfam database is used instead.

In cases where homology is not established by SCOP [10] or Pfam relationships, homology can be derived either by structural alignment using Combinatorial Extension (CE) [14], or by Needleman–Wunsch [11] global sequence alignment. Structural alignment by Combinatorial Extension is preferred, so Needleman–Wunsch is only used in cases where a protein is not associated with a known structure in the Protein Data Bank [1].

For CE, we use the scoring matrix SDM of Prlic et. al. [12] added to BLOSUM62 [6] with a coefficient of 2. We then score homology relationships using TM-Score [16], which is essentially an RMSD that is normalized by the protein length. A TM-score of 1.0 is a perfect alignment, and a TM-score of 0.0 has no aligned regions.

For Needleman–Wunsch, we use BLOSUM62, a gap opening penalty of -12, and a gap extension penalty of -1. We then score homology relationships using percentage identity applied to a gamma distribution with shape $k = 25.54$, scale $\theta = 4.96$, and a linear shift of 0.2, which was found by Weber et. al. [15] to give the best results for global alignment.

Using homology to score interactions

Overview

The central idea here is very simple. Suppose we have four proteins a, b, c, and d in the situtation shown in the second step in the figure on page 4. a and b interact, and c and d interact. Since a is homologous to c and b is homologous to d, we now know with greater probability that a and b interact, and that c and d interact. (Indeed, we could also suggest that a and c interact.)

We wish to use y to increase $Prob(x)$. However, it would be inappropriate to let $x' := x \vee y$ and thereby $Prob(x)' := Prob(x) + Prob(y|\neg x)$, since this would not take into account the probabilities given by α and β. Instead, we let $x' := x \vee \alpha\beta y$ and thereby $Prob(x') := Prob(x) + Prob(y|\neg x)Prob(\alpha)Prob(\beta)$. This way, we take into account both the probablity of an interaction and the probability that the interaction is homologous.

Preferred treatment

We should favor being liberal when incorperating additional information into our model to alter probability. We can imagine a situtation in which a has some interaction x, and a is homologous to b which is then homologous to c, but due to some limitation in our method or an underlying algorithm or database used in the above section, we do not have an edge from a to c. If c has some interaction y that is similar to x,

we would like to use y to update the probability of x. This leads to the formulation below:

Define $R_{i,j}^{(u,v)} = Prob(i,j)\,(1 - Q_{u,i} - Q_{v,j} + Q_{u,i} \cdot Q_{v,j})$

Define $Q_{a,b} = \prod_{\text{paths } \pi \text{ from a to b}} \left(1 - \prod_k S(\pi_k)\right)$

Then we want to update the probability of an interaction (u, v) with:

$Prob((u,v))' = Prob((u,v)) + 1 - \prod_{i,j}(1 - R_{i,j}^{(u,v)})$

Unfortunately, this would require use to enumerate all possible paths.

Approximation

As a consequence of the failure above, we need to be satisfied with the assumption that our homology graph contains every edge that it should. Under this assumption, a case like the one described above is a violation of the idea of homology, since homology should be a metric on our graph and thus obey the triangle inequality.

With this in mind, we approximate the above by taking only the shortest path rather than all paths. This simplifies the definition of $Prob((u,v))$ to:

$$Prob((u,v))' = Prob((u,v)) + Prob((i,j)) \left(\prod_{\pi_k \in \pi^i} (S(\pi_k^i) \prod_{\pi_k \in \pi^j} (S(\pi_k^j) - Prob((x,y)) \prod_{\pi_k \in \pi^i} (S(\pi_k^i) \prod_{\pi_k \in \pi^j} (S(\pi_k^j) \right),$$

where (i, j) is the closest interaction corresponding to (u, v), and π^i, π^j are the shortest paths corresponding to $u \ldots i$ and $v \ldots j$, respectively.

Merging degenerate subgraphs

Overview

The secondary objective of StructNA is to remove unncessary information from the interaction network by merging equivalent interactors. Specifically, we wish to merge vertices and edges within subgraphs of the combined graph G such that the subgraph is a clique in the homology graph whose members share precisely the same interactions.

Formally, we call a subset $S \subseteq C$ for some clique $C \in (V, Hom(G'))$ *degenerate* if $\forall u, v \in S\ Int(u) = Int(v)$. If C is a maximal clique and $\nexists S_1 \subseteq C,\ S_1 \supset S$ for some degenerate set S_1, we say S is a *maximal* degenerate set.

Our goal is to enumerate all maximal degenerate sets. To do this, we require subgraphs with edges in $Int(G')$, and homology relations between the two subgraphs that result in a clique in $(V, Hom(G)$. A subset of $Hom(G'')$ defines an isomorphism between two subgraphs of $(V', Int(G''))$.

Our general process is: 1. enumerate all maximal cliques, and 2. enumerate all maximal degenerate sets from the maximal cliques.

Enumerating maximal cliques

Unfortunately, Max-Clique is NP-Equivalent and APX-Hard. The best known polynomial-time approximation algorithm has an error bound of $\epsilon = \mathcal{O}\left(\frac{n}{\log^2(n)}\right)$ [2]. This makes even approximating a solution to Max-Clique intractable according to worst-case time-complexity. However, Max-Clique is often solvable in practice for small n because, unlike many other NP-Hard problems, naive algorithms for Max-Clique do not require numerically intensive steps. The Bron–Kerbosch algorithm requires only incrementation and trivial set unions and subtractions. Consequentially, it has been found to perform well enough in practice for reasonable n such as those found in real biological networks.

Moreover, we can bound the time-complexity of a naive algorithm (such as Bron–Kerbosch) in terms of the maximum number of homology edges of any vertex in G.

Let $\iota(v)$ denote the number of edges (v, v') in $Int(G)$, $\forall\, v \in V$, and let $\eta(v)$ denote the number of edges (v, v') in $Hom(G)$, $\forall\, v \in V$. Let $\iota^* = \max_{v \in V}\{\iota(v)\}$, and $\eta^* = \max_{v \in V}\{\eta(v)\}$.

Let γ be the maximal clique size in $Hom(G)$. Then $\gamma^* \leq \eta^*$.
We can readily solve Max-Clique(G) in $\mathcal{O}(|V|^{\eta^*}(\eta^*)^3)$ time by enumerating all $|V|^\gamma$ subgraphs of size γ for every $\gamma = 1, 2, \ldots, \eta^*$.

Enumerating degenerate sets

Given a clique C, we need to find which vertices in C share interactions. This results in a partitioning of C into disjoint induced subgraphs D. We can then merge every vertex in each D_i.

We first identify all maximal cliques on $(V, Hom(G'))$ using the Max-Clique algorithm by Bron and Kerbosch [3], which has been found in practice to be faster than algorithms with superior worst-case time complexity. We then identify maximal degenerate sets by associating to each vertex of a clique C a string that uniquely identifies its interactions, then partitioning C by these strings.

If a vertex $v \in V$ belongs to two distinct maximal degenerate sets S_1 and S_2, we choose whether to include v in S_1 or S_2 by the following:

1. If $|S_1| > |S_2|$, choose S_1
2. If $|S_1| = |S_2|$, choose the set whose sorted vertex labels form the lexicographically smaller string

Finally, we perform edge contraction on each degenerate set with respect to both $(V, Int(G'))$ and $(V, Hom(G'))$. The vertex v_0 with the lexicographically smallest label for each degenerate set is chosen as the representative, and the labels of the other vertices are included as metadata in v_0.

Implementation

StructNA was implemented in Java and uses many Java 7 features. It is clearly documented and is under 7,000 SLOC. Version control and issue tracking are handled through GitHub. It is distributed under the Apache License version 2, and a distribution is available for download at `https://github.com/dmyersturnbull/network_merge`.

Speed

The method was implemented in Java and run on a JDK 7 platform on a single AMD Phenom II processor with 4GB of DDR3 memory. In general, the most time-consuming step was pairwise alignment in cases where a cached alignment result was unavailable. The second and third steps were found to require no more than 20 minutes for even extremely large networks of hundreds of thousands of proteins.

Benchmarks

Benchmark set

To determine the degree to which StructNA accurately assesses the confidence of each interaction in real biological networks, a benchmark set was derived from Interactome3D [9], which incorperates structural information into PPI networks. It is partially hand-curated and is of high quality. The particular dataset chosen was the "complete dataset" for the human proteome, which includes non-representative proteins.

Struct-NA was run on a data set from EBI IntAct [8] that contained only human proteins represented in SCOP, which was used to determine homology. The advantage of this choice of benchmark set and test set is a relatively close correspondence between the two.

Comparison

Interactions in the resulting network were compared against interactions in the benchmark set three times: once with binary confidences before initial scoring, once after initial scoring, and once after determining output confidences. In general, interactions to which StructNA assigns higher confidences should be more likely to be present in the benchmark set.

We define a correspondence score B for each step as:

$B = \frac{1}{N} \sum_I B(I)$, where

$$B(I) = exp(Prob(I)) \cdot \begin{cases} 1 & \text{if in benchmark set and known by structure} \\ 0.5 & \text{if in benchmark set and known by model} \\ -0.2 & \text{if not in benchmark set} \end{cases}$$

N is a normalization constant defined by

$N = \sum_I exp(Prob(I))$

We expect this score to increase after each step in the process, to reflect greater correspondence with Interactome3D.

Results

The parameters used were $\tau = \zeta = 1$, $\xi = 2$, and Φ defined exclusively by the SCOP weight described in *establishing homology*. The choice of $\tau = \zeta = 1$ is preferable, as $\tau < 1$ and $\zeta < 1$ are chosen only to decrease running times. The results were as follows:

B_{input}	B_{init}	B_{output}
0.0195	0.0249	0.0285

The results indicate that running StructNA increases the correspondence between the network and the high-quality benchmark set from Interactome3D. Similar results were found for several other IntAct PPI networks.

Unfortunately, the statistical significance of the difference could not be determined without a reliable model for a random biological network containing both interactions and homology relationships, or a large number of similar input networks. However $B_{input} < B_{init} < B_{output}$, and the difference between the three appears significant.

In addition to the above benchmark, interactions whose probabilities were increased substantially (more than 0.5) were investigated manually for many networks. Generally, these interactions were associated with many homologous interactions in the network. These interactions were typically updated (had their probabilities increased) many times with relatively small update values, rather than fewer times with high update values.

References

[1] HM Berman, T Battistuz, TN Bhat, WF Bluhm, PE Bourne, K Burkhardt, Z Feng, GL Gilliland, L Iype, S Jain, P Fagan, J Marvin, D Padilla, V Ravichandran, B Schneider, N Thanki, H Weissig, JD Westbrook, and C Zardecki. The protein data bank. *Acta, Crystallogr. D. Biol. Crystallogr.*, 58:899–907, May 2002.

[2] R Boppana and M Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32:180–196, 1992.

[3] C Bron and J Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *ACM*, 16:575–577, 1973.

[4] The UniProt Consortium. Reorganizing the protein space at the universal protein resource (uniprot). *Nucleic Acids Res.*, 40, 2012.

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

[6] S Henikoff and JG Henikoff. Amino acid substitution matrices from protein blocks. *PNAS*, 89:10915–10919, 1992.

[7] B Kaboord and M Perr. Isolation of proteins and protein complexes by immunoprecipitation. *Methods Mol. Bio.*, 424:349–364, 2008.

[8] Samuel Kerrien, Bruno Aranda, Lionel Breuza, Alan Bridge, Fiona Broackes-Carter, Carol Chen, Margaret Duesbury, Marine Dumousseau, Marc Feuermann, Ursula Hinz, Christine Jandrasits, Rafael Jimenez, Jyoti Khadake, Usha Mahadevan, Patrick Masson, Ivo Pedruzzi, Eric Pfeiffenberger, Pablo Porras, Arathi Raghunath, Bernd Roechert, Sandra Orchard, and Henning Hermjakob. The intact molecular interaction database in 2012. *Nucleic Acids Res.*, 40:D841–D846, Jan 2012.

[9] R Mosca, A Céol, and P Aloy. Interactome3d: adding structural details to protein networks. *Nature Methods*, 10:47–53, 2013.

[10] AG Murzin, SE Brenner, TJP Hubbard, and C Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[11] B Needleman and C Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–53, 1970.

[12] A Prlic, FS Domingues, and MJ Sippl. Structure-derived substitution matrices for alignment of distantly related sequences. *Protein Eng.*, 13:545–550, 2000.

[13] G Rigaut, A Shevchenko, B Rutz, M Wilm, M Mann, and B Séraphin. A generic protein purification method for protein complex characterization and proteome exploration. *Nature Biotechnology*, 17:1030–1032, Oct 1999.

[14] IN Shindyalov and PE Bourne. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Protein Eng*, 11:739–47, 1998.

[15] C Webber and Geoffrey Barton. Estimation of p-values for global alignments of protein sequences. *Bioinformatics*, 17:1158–1167, 2001.

[16] Y Zhang and J Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins*, 57:702–710, 2004.