**Course Project: Network Alignments**

**Karthik Rangaraj – 801135834**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Title (3 points) | PINALOG: a novel approach to align protein interaction networks—implications for complex detection and function prediction | | |
| 2. Authors (3 points) | Hang T. T. Phan and Michael J. E. Sternberg | | |
| 3. Institution and lab (3 points) | Division of Molecular Biosciences, Faculty of Natural Sciences, Imperial College, London SW7 2AZ, UK | | |
| 4. Problem formulation (11 points) | The omission of function information results in many pairs of equivalenced proteins having little or no similarity of function which in turn makes it less accurate in detecting conserved functional modules or predicting protein function, which thus leads the authors to question whether the inclusion of functional similarity might influence the resulting alignments and any subsequent applications. | | |
| 5. Categories in problem formulation of the paper (20 points in total) |  | Yes or No | Explanation |
| Local alignment | No | For local alignment methods, the difficulty is less severe as some local alignments can be evaluated by their agreement with known protein complexes. However, this approach is less useful to evaluate global alignment methods. This assessment measure would always penalize equivalences based on network topology when this conflicts with equivalences between orthologous pairs. The authors have used a series of other measures. |
| Global alignment | Yes | PINALOG forms the alignment between two PPINs based on the similarities of protein sequence and the protein function between the two networks. |
| One-to-one mapping | Yes | IsoRank and MI-GRAAL generate alignments with one-to-one mapping. |
| One-to-many mapping | Yes | Protein complexes and functional modules form highly connected components in the PPINs. Therefore, it would be more reliable and efficient to align two PPINs by first finding highly similar protein pairs (seed protein pairs) from these highly connected protein subnetworks (referred to as communities from now on) in the networks and then extending the alignment to other proteins in the neighbourhoods of seed protein pairs. |
| many-to-many mapping | Yes | CFinder identifies communities that can be overlap, community mapping can result in many-to-many mappings in seed protein pairs, thus in the final alignment. |
| Node-to-node mapping | Yes | Conserved interactions are interactions occurring in both species when two protein nodes forming an interaction in one species are equivalenced to two protein nodes which also form an interaction in the other species. |
| Edge-to-edge mapping | Yes | PINALOG and MI-GRAAL have similar values for NC. BLAST finds far less aligned pairs and conserved edges than PINALOG consistent with the objective of network alignment in establishing more equivalences than a purely sequence-based method. PINALOG finds far more pairs of aligned proteins with a functional similarity*>*0.5 than IsoRank, MI-GRAAL and BLAST. This is consistent with PINALOG including functional similarity in its equivalence. |
| Path-to-path mapping | Yes | The first step is to determine the highest score when equivalencing proteins in each community in species A with each community in species B. To this end, we define the similarity between two communities as the score F(CAi ,CBj ), the sum of similarities between protein pairs in the optimal equivalence (OptMap) of proteins in community CAi in species A with proteins in CBj in species B using the Hungarian method. The optimal equivalence is the mapping where the sum of protein pair similarities is largest. The Hungarian algorithm is a combinatorial optimization algorithm solving the assignment problem in polynomial time. |
| Module-to-module mapping | Yes | Protein complexes and functional modules form highly connected components in the PPINs. Therefore, it would be more reliable and efficient to align two PPINs by first finding highly similar protein pairs (seed protein pairs) from these highly connected protein subnetworks in the networks and then extending the alignment to other proteins in the neighbourhoods of seed protein pairs. |
| Allow graph editing operations | Yes | The Hungarian method is used to find the optimal equivalence of candidates. These candidates are then added to the equivalences in the core. The process is repeated until no more pairs are added. |
| 6. Algorithm in paper (30 points) |  | Description | |
| Design strategy | **Cfinder** & **Hungarian** method are used to identify and map communities. | |
| Pseudocode | **Cfinder**:  Cfinder makes use of the clique percolation method which defines communities as percolation clusters of k-cliques. To do this it finds all k-cliques in a network, that is all the complete sub-graphs of k-nodes. It then defines two k-cliques to be adjacent if they share k-1 nodes, that is this is used to define edges in a clique graph. A community is then defined to be the maximal union of k-cliques in which we can reach any k-clique from any other k-clique through series of k-clique adjacencies. That is communities are just the connected components in the clique graph. Since a node can belong to several different k-clique percolation clusters at the same time, the communities can overlap with each other (Graph representation).  1. All cliques are found for different values of k.  2. A square matrix M = n×n , where n is the number of cliques found, is created. Each cell [i, j] contains number of nodes shared by cliques i and j.  3. All cliques of size equal or greater than k are selected and between cliques of the same size connections are found in order to create a k-clique chain.  **Hungarian algorithm**: Below are matrix and graph representations  **Matrix:**   1. For each row of the matrix, find the smallest element and subtract it from every element in its row. 2. Do the same (as step 1) for all columns. 3. Cover all zeros in the matrix using minimum number of horizontal and vertical lines. 4. *Test for Optimality:* If the minimum number of covering lines is n, an optimal assignment is possible and we are finished. Else if lines are lesser than n, we haven’t found the optimal assignment, and must proceed to step 5. 5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to step 3.   **Graph:**  Construct a subgraph graph G consisting of the "best cost edges";  Find a maximal matching M in subgraph G  repeat until M is a complete matching  {  Add the "next best cost edges" to G;    Find a maximal matching M in (modified) subgraph G;  }    **Note:** This Paper makes use of the graph representation. | |
| Parameters | **Cfinder:**  **Input:** The input of CFinder is a file containing strings and numbers ordered into three columns; in each row the first two strings correspond to the two end points of a link and the third item is the weight of this link.  **Output:** network of modules.  **Hungarian:**  **Input:** The input of the Hungarian algorithm is an n by n square matrix with only nonnegative elements (For matrix). This paper makes use of graph representation where the input is a weighted bipartite graph.  **Output:**  Lowest-cost way to assign the jobs (For Matrix). Minimum cost to complete matching in a weighted bi-partite graph(Graph representation). | |
| Time complexity analysis | **Cfinder:** O(nk) – Polynomial as it depends on the number of nodes in the weighted bipartite graph.  **Hungarian:** O(n3) | |
| Correctness analysis | **Cfinder:** The paper and the referenced papers do not mention correctness.  **Hungarian:**  1. The dual solution is always feasible. Note that u, v are clearly dual feasible in the beginning. Now, assume that u, v are dual feasible in the beginning of some iteration. They will only be changed during that iteration if there is no perfect matching in G0 . Feasibility simply means nonnegativity of reduced costs for all edges. Note that the only reduced costs that decrease are for edges (i, j) where i ∈ A ∩ L and j ∈ B\L. However, note that the reduced costs of those edges decrease by exactly δ, and by definition of δ equals the minimum of them; thus, none of them drop below 0 and dual feasibility is maintained.  2. Complementary slackness is maintained. Trivial: note that the matching is found only along edges with zero reduced cost. (Note that this is true at the point in iteration immediately after M is computed, which is what we need since this is the point where we will eventually exit the loop).  3. The algorithm reaches an ending. First, note that δ always stores a strictly positive number. If δ gets the value 0 at some point, then there is an edge (i, j) for which i ∈ A ∩ L, j ∈ B\L, and wij = 0; however, since wij = 0, then (i, j) ∈ E0 , and j is reachable from A ∩ L or i was reached via (i, j) (if (i, j) ∈ M), so j ∈ L, which is not possible. Next, recall that after we run the bipartite (unweighted) matching algorithm, C = (A\L) ∪ (B ∩ L) is a vertex cover of the same cardinality as the matching M we find. Note that in every iteration, the dual increase is δ|A ∩ L| − δ|B ∩ L| = δ(|A ∩ L| + |A\L| − |A\L| − |B ∩ L|) = δ(|A| − |C|) = δ( n 2 − |M|) ≥ δ where we get n 2 − |M| ≥ 1 from the fact that M is not perfect. Thus, each iteration increases the dual by at least δ, which is at least 1 since the data is integral; since the primal problem is feasible (there exists a perfect matching in the original graph), the dual problem is bounded, and so the dual solution value can’t increase forever and the algorithm must terminate.  Now, since clearly the algorithm ends with a perfect matching (i.e. a primal feasible solution), the above discussion shows that we also have a dual feasible solution and that those two solutions maintain complementary slackness. Therefore, we have shown that the algorithm ends in finite time and correctly. | |
| Comments | - | |
| 7. Software provided by paper authors (24 points in total) | Link | http://www.sbg.bio.ic.ac.uk/~pinalog/ | |
| Used data in paper (including link and summary of data) | The link to the data which the authors use is http://www.sbg.bio.ic.ac.uk/~pinalog/downloads\_test.html. The data consists of PPIN data for different species extracted from the IntAct database and a BLAST data file.  **PPI data file format:**  Tab-delimited file containing two columns  ID1 ID2  ID3 ID2  ...  where ID1, ID2, ID3 are the string identifying the protein names in the species.  The authors have aligned different pairs of PPINs from human, yeast, fly, worm and mouse.  **BLAST data file:**  This file contains the result of the all-against-all BLAST results of proteins in the input species. This includes the BLAST results of the proteins within each species as well as with those in the other species. The self-BLAST score of each protein is required so that a normalization on the sequence similarity score to be accurately computed.  The format of the BLAST data file is as below:  ID1 ID2 BLAST-score  where ID1, ID2 are protein IDs of the input species, and BLAST-score is the score obtained from the BLAST sequence alignment result. | |
| Usage of the software | The software is used is the PINALOG web server for protein interaction network alignment. This software takes two PIN files and 1 BLAST file to perform network alignment. There is also an option to use the functional information in the alignment process. This can be achieved by providing the function annotation file for proteins in the network or choose the GOA file available for IntAct proteins. Providing all the input data and clicking on submit will schedule a job with a unique job id which can be referenced once the job has finished. The results generated are the list of aligned nodes and a list of conserved edges. Along with these results we also see PINALOG alignment statistics for both the networks. | |
| Programming language | The authors have made use of C++ in order to develop the system. | |
| Test experiment profile in paper | **Example:**  Here I upload 2 pin files of the mouse and rat species along with 1 mouse\_rat blast score file. I choose to not to use any function annotations for this job.      Following the link, we reach our results as seen below.      **List of aligned nodes:**    **List of conserved edges:** | |
| Application(s) in paper | Protein complex prediction and protein function prediction. Complex prediction is performed by direct inheritance of protein complex data from a known species to an unknown one. PINALOG has been tested for complex prediction power using a human-yeast alignment. Protein function prediction is via direct transfer to an un-annotated protein from an annotated protein. | |
| In paper, does it include visualization sub-module? How did authors visualize results? | No, the paper does not include a visualization sub module. The authors have instead plotted complex prediction results of PINALOG in comparison with other prediction methods based on recall, precision and F-measure.    The above figure represents the precision, recall and F-measure of yeast protein complex prediction by PINALOG-A. | |
| Comments | - | |
| 8. Statistical significance analysis in paper (6 points in total) | Method in paper | Explanation | |
| The McNemar test | The McNemar test (McNemar, 1947) for the statistical significance of the difference in the performance was used based on the number of misclassifications in each method. The test indicates that PINALOG predictions are significantly different from PSI-BLAST in BP terms at the level of P=0.001 significance level. PINALOG predicts more BP terms with a recall twice that of PSIBLAST (14% versus 7%) at the same level of precision (∼28%). For the MF category, PINALOG and PSI-BLAST share very similar  levels of recall (∼28%) while the precision of PSI-BLAST is slightly better (43% versus 47%). The McNemar test suggests no significant difference between them at the P=0.001 significance level. On the other hand, PINALOG outperforms IsoRank in both  BP (P=0.001) and MF (P=0.001) categories. For example, in the BP category, PINALOG has almost twice the recall compared with IsoRank (14% versus 8%) at higher precision (28% versus 20%, Table 2). To summarize, in the challenging area where sequence similarity does not contribute substantially to the prediction of protein function, PINALOG enhances the ability to predict function of these proteins. | |
| 9. Validation in paper (10 points) | Approach in paper | | Results in paper |
| A 100-fold cross validation was performed. In each run of the cross validation, a set of proteins from the test set and had their GO terms hidden. The test proteins which were aligned with an annotated protein in the other species were identified and the function transferred. Over the 100 runs, a total of 169 proteins were equivalenced and their function predicted. Functions of mapped proteins in yeast represented by GO terms in two categories BP (biological process) and MF (molecular function) were directly transferred to the human proteins in the test set. | | The results were analyzed in terms of precision and recall, where precision is defined as tp/tp+fp and recall as tp/tp+fn ; tp, fp and fn being the number of true positives, false positives and false negatives of the predictions made. The method used in this paper is a binary predictor without a variable cut-off parameter and thus no precision/recall curve can be produced. |
| 10. Authors' contribution (10 points) | PINALOG is flexible in allowing the use of sequence only alignment or sequence-function alignment depending on the requirement of the user. For the sequence-function alignment, parameters are automatically calculated from the input species and this will help non-experts. When aligning two species where one or both species are poorly annotated, alignment might be biased towards aligning well-annotated proteins whose functions are similar. Then it is advisable that sequence and network topology only are used to align the networks to avoid bias. The authors are developing a version of PINALOG to perform alignment of PPINs from multiple species. The computing time of the alignment process depends on the size of the input networks. The typical computing time is <24 h with the longest run in our assessment being the human-yeast alignment which takes 24 h on a computer with 2.8 GHz processor with 8 GB memory. | | |
| 11. Propose your own approach (20 points) |  | Explanation | |
| Main idea | The improvement is mainly based on the PGM (Percolation Graph Matching) algorithm.  The approach is mainly divided into two steps:  1. At the first step, it uses the sequence similarities to generate a seed set for a PGM algorithm  2. Second step comprises of aligning the remaining couples.  Initially we have as inputs two PPI networks G 1(V 1,E 1) and G 2(V 2,E 2), the set of pairwise BLAST bit-score similarities for couples of proteins in V 1×V 2, and fixed thresholds ℓ,r>0, where ℓ and r are the sequence similarity and the local topological similarity thresholds, respectively.  BLAST bit-score is used as an indicator for detecting functional similarities between proteins. In the case of high similarity, we can make a functional inference with a high accuracy.  The main approach is to use such proteins with high sequence similarity in order to find global alignment.  The seed set A is generated from the pairwise similarities (the set S) in the following manner: Among all the couples of proteins with BLAST bit-score similarity above ℓ, couples [i,j] are matched in a descending order of sequence similarity, unless i or j is matched already. More precisely, (i) we add the couple [𝑖,𝑗]∈[i,j]∈S with the highest similarity to the seed set and match i to j; (ii) all the couples [ i,j ′] and [ i ′,j] are now forbidden and we remove them from S. We repeat the steps (i) and (ii) until there is no remaining couple in the set S with BLAST bit-score similarity at least ℓ. Note that, in the process of seed generation, when there are several couples with the same sequence similarity, we randomly pick one of them.  The second step starts the alignment process from the seed couples (set A) obtained from the set of pairwise similarities S. It then incrementally generates the set π of matched couples among 𝑉1×𝑉2∖V1×V2∖A. In the MapPercolation step, the seed couples are added to the set of aligned couples π. Then, at each time-step, the goal of the PGM algorithm is to add a new couple to the set π so that structural similarity is maximized. | |
| Pseudo code | **Algorithm 1:** *Seed Generation Algorithm*  Input: BLAST bit-score similarities between S and L  Output: Seed set A  A = φ  for all couples [i,j] ∈ S from the highest similarity to the lowest do  If i ∉ V1(A), j ∉ V2(A) and BlastBit(i,j) >= l then  add the couple [i,j] to A:  end  end  return A  **Algorithm 2:** *Map Percolation Algorithm*  Input: G1(V1,E1) , G2(V2,E2), seed set A and threshold r  Output: The set of aligned couples π  π = A  while there exists an unmatched couple with score at least r do  among all the couples with the highest score select the unmatched couple [i,j] with the minimum |d1,i – d2,j|. If there is more than one couple with the minimum |d1,i – d2,j|, select the couple with the minimum d1,i + d2,j. Finally if there are still several candidates, randomly pick one of them.  add[i,j] to the set π | |
| Time complexity analysis | The Time Complexity of Seed Generation Algorithm is O (|S≥l| log |S≥l|) .  The Time Complexity of Map Percolation Algorithm is O (k2 (|E1| + |E2|) min (d1, d2) | |
| Correctness analysis | We now describe our PGM algorithm more formally. The input of the algorithm is the following:  • Two graphs G1 = (V1, E1) and G2 = (V2, E2)  • A seed set A0 of size a0, consisting of tuples (i, i) of known pairs of matched nodes.  The algorithm we propose and analyze simply maps any two nodes with at least r neighboring pairs already mapped. An equivalent description emphasizes the incremental nature of the process: we associate with every pair of nodes (i ∈ V1, j ∈ V2) a count of marks Mi,j . At each time step t, the algorithm uses exactly one unused but already mapped pair (it, jt). This pair adds one mark to each neighboring pair, i.e., to every pair in N1(it) × N2(jt). As soon as any pair gets r marks, it is added to the current map; if for some node i there are several nodes j such that all (i, j) have r marks, one pair is picked at random. The process iterates until there are no more unused pairs. The set A(t) consists of the map built until time t, and the set Z(t) ⊂ A(t) consists of mapped pairs that have been used until t, in the following way:  • At time t = 0, A(0) = A0 and Z(0) = ∅,  • At time step t the algorithm randomly selects a pair (it, jt) ∈ A(t − 1) \ Z(t − 1) and adds one credit mark to all pairs (i 0 , j0 ) ∈ V1 × V2 such that there exist (it, i0 ) ∈ E1 and (jt, j0 ) ∈ E2 (cf. Fig. 1). If a pair (i 0 , j0 ) has more than r marks then it is added to the map A(t); furthermore, all other candidates (i 00, j0 ) and (i 0 , j00) are permanently removed from consideration. Let ∆A(t) be the set of pairs with r marks, which are added to the map at time t. Then A(t) = A(t − 1) ∪ ∆A(t) and Z(t) = Z(t − 1) ∪ {(it, jt)}. Note that a(t) ≥ z(t) = t.  The process stops when A(t) \ Z(t) = ∅, which happens when all pairs from the map A(t) are used. Denote this time step by T = min(t ≥ 0 s.t. A(t) \ Z(t) = ∅). The final map is A ∗ = A(T) = Z(T) and its size is a ∗ = T. | |
|  |  | Experiment 1 | Experiment 2 |
| 12. Experiments based on using in paper's software data that we provide (10 points) | Graph 1 size | PPI Data Used:  SARS Coronavirus-Mouse (mus musculus) protein-protein interaction  Edges:15  Nodes:14 | PPI Data Used:  Human Coronavirus 229E-human (homo sapiens) protein-protein interaction  Edges:17  Nodes:18 |
| Graph 2 size | Edges:39  Nodes:11 | Edges:22  Nodes:23 |
| Edge correctness (EC) | 71.96% | 84.01% |
| Symmetric substructure score (SS) | NA | NA |
| Largest common connected subgraph (LCCS) | Edges:11  Nodes:11 | Edges:5  Nodes:5 |
| Functional consistency (FC) | NA | |
| Average of functional similarity (AFS) | 50% | |