Struct-net-align

currently a very rough incoherent and partially incorrect sketch

Introduction

The idea is to use protein structural information to improve a noisy PPI network. The method would typically be applied after a more cononical, sequence- or network-structure- based, heuristic network alignment or degeneracy minimization algorithm.

The method uses the Structural Classification of Proteins (SCOP) and structural alignment based on Combinatorial Extension 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 because edges indicating homology are drawn relatively conservatively, resulting in a sparse graph.

Algorithm

Definitions

Let $G = (V, Int(G) \cup Hom(G))$ be a network whose vertices represent domains.

Let $Int(G)$ is a set of edges indicating interactions.

Let $Int(v)$ for any vertex $v \in Int(G)$ is the set of edges in $Int(G)$ that are incident to v. Also define $Hom(v)$ in the same way.

$Hom(G)$ is a set of edges indicating homology (similarity) between vertices.

An edge $e \in Int(G)$ is associated with a probability $Prob(e)$.

An edge $(a, b) \in Hom(G)$ is associated with a score denoted $s((a, b))$ ranging from 0 to ∞. Any edge with score $s \geq 1$ is defined as significant.

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.

Let $\alpha(a, b)$ denote the weighted score of an alignment that has been performed between vertices a and b.

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 $\iota^* = \max_{v \in V}\{\iota(v)\}$, $\eta^* = \max_{v \in V}\{\eta(v)\}$.

Overview

The algorithm consists of three major steps.

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 1.

Second, generate a network $G'' = (V, Int(G'')$ by using edges in $Hom(G')$ to update edge weights (probabilities) in $Int(G)$.

Third, define a equivalence relation between vertices, H, such that $H(u) \equiv H(v)$ iff v is reachable from v by traversing $Hom(G')$. 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 interaction edges in $(V, Int(G''))$, with respect to the equivalence relation H.

Algorithm

Pre- and post-processing steps are not included in the formal description but are listed below.

StructNetAlign(G, τ, ς, ξ):

1. Parse network.
2. For every pair of vertices (a, b), add an edge (a, b) to $Hom(G)$, with label $\rho(a, b)$.
3. For every pair of vertices (a, b) (optionally: not joined by an edge in $Hom(G)$), align a against b and add an edge to $Hom(G)$, with label $\alpha(a, b)$.
4. For every edge $e \in G$ such that $S(e) < \tau$, remove e.
5. 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 labellings X and Y, respectively. Visit an already-visited vertex, then stop there. Stop after reaching ξ depth.
   (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 $Q := Q + \log S(e)$.
      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 $Prob((u, v)) := Prob((u, v)) + 1 - X_x Y_y Prob((x, y))$.
6. Discard every vertex v such that $\iota(v) = 0$ and remove every edge incident to v.
7. For every edge $e \in G$ such that $S(e) < \varsigma$, remove e.
8. Run the Bron–Kerbosch algorithm on G_i to enumerate all maximal cliques $C = \{c_1 \ldots c_x\}$, for clique sizes $\gamma(C_j) > 1 \; \forall j$.
9. Partition C into sets $D = d_1 \ldots d_z$ such that $\forall \; v \in d_i, u \in d_j$, $Int(v) \neq Int(u)$, $\forall \; i, j$, and $Int_H(v) = Int_H(u) \; \forall u, v \in d_i \; \forall i$.
10. For each $d \in D$, merge by edge contraction every vertex in d, modifying the reference graph G.
11. Output the network $(V, Int(G))$.

Secondary steps

a) The algorithm should is run seperately for each connected component of a network.

b) If edge weights for interactions in the network are binary or otherwise determined to be unreliable, the interactions are scored.

Defining homology by relations

The input DIP network consists of a list interactions occuring between two polypeptide chains, where the chains are described by UniProt Ids. In cases where the PDB contains at least one structure for that chain, a PDB Id and is found for that UniProt Id. From that, a SCOP domain corresponding to the PDB Id and chain are found. This selection process is problematic for multi-chain domains, which are ignored for this step. Scores (ρ) are assigned by using the most specific SCOP relationship found.

Defining homology by alignment

In cases where homology is not established by SCOP relationships, homology can be derived by structural alignment using Combinatorial Extension.

Using homology to score interactions

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

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

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

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

Merging equivalent domains

To merge, we require:

1. Two subgraphs G_1, G_2 of G'' with edges in $Int(G')$
2. $\exists\, H \subseteq Hom(G)$ s.t. H defines an isomorphism between G_1 and G_2 (TODO: rigor).

Unfortunately, MAX-CLIQUE is NP-HARD. However, 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 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 $\gamma = 1, 2, \ldots, \eta^*$.

Note that a subset of $Hom(G'')$ nearly defines an isomorphism between two subgraphs of $(V', Int(G''))$ For a pair (u, v) where $(u, v) \in Hom(G')$, $Int(u)$ may not exactly equal $Int(v)$, but $Int(u)/H = Int(v)/H$, where $Int(u)/H$ is some kind of factor graph in the algebraic sense that I'll need to define (look up factor group or quotient ring).

Once we have a (maximal) clique C, we need to find which vertices in C share interactions. This results in a partitioning of C into disjoint subgraphs D. We can then merge every vertex in each D_i by edge contraction.

Initial scoring of interactions

Implementation

Results

Overview

Accuracy

Speed

Appendix 1. Formal proofs