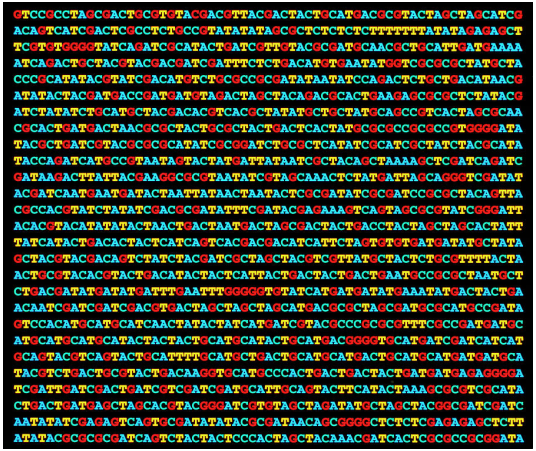
Community-based algorithms for protein function prediction

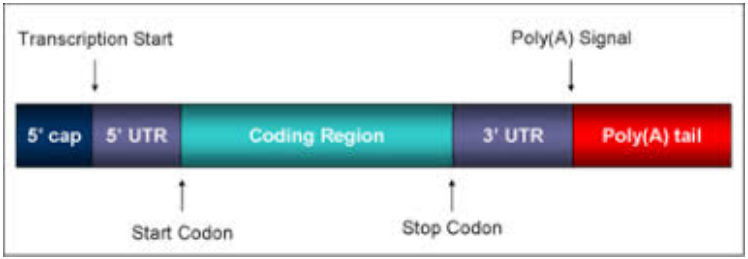
Nikola Kalábová

Faculty of Science, Charles university

10th May, 2021

From genome to protein




Transcription Start
Poly(A) Signal
5' cap
5' UTR
Coding Region
3' UTR
Poly(A) tail
Start Codon
Stop Codon

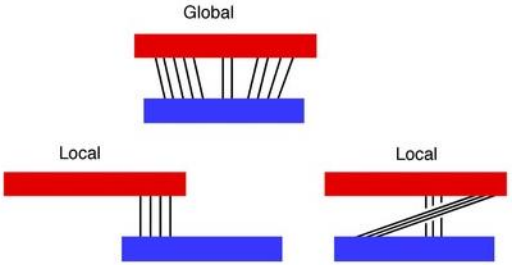
Protein function

Methods for function prediction

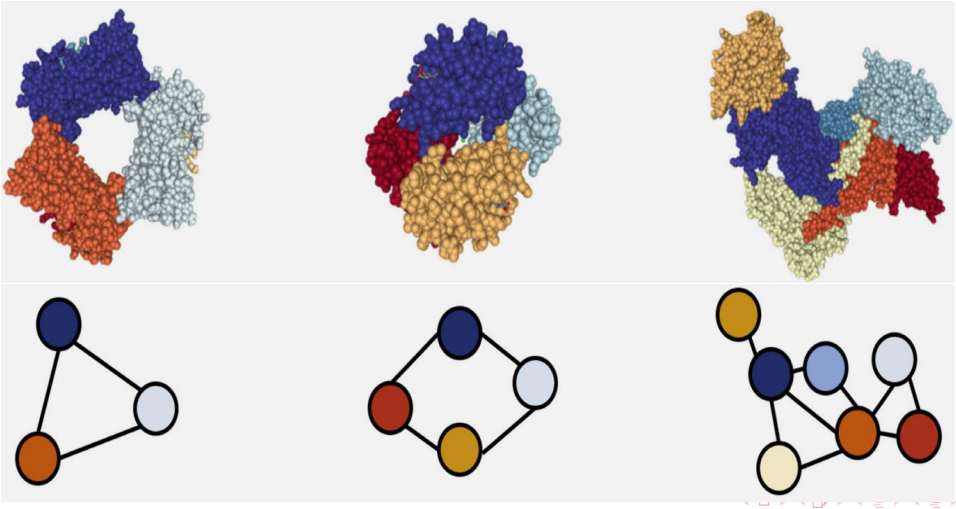
A C T C G C A A T A T G C T A G G C C A G C

A C T _ _ _ _ T T A T G C T A T G C _ _ G C

Protein protein interactions



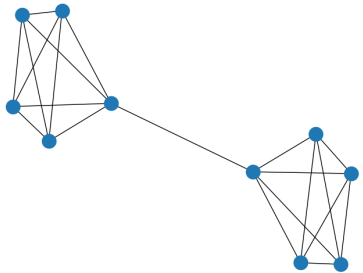
Protein protein interaction networks



Community networks

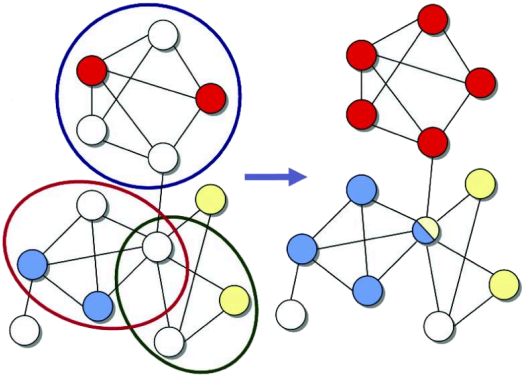
Definition (Community)

A *community* is a subset of vertices that are densely connected with each other and sparsely connected to the vertices outside of the community.



From PPIN to protein function

1. identify communities
2. annotate communitites



Problems

- Data reliability
- Overlapping of communities
- Specific structure of the communities
- Running time

Definition (Markov matrix)

The *Markov matrix* has on the index i, j the probability of selecting the edge $\{v_i, v_j\}$ from all edges directing from v_i, if there exists an edge between v_i and v_j, else 0. The probability of selecting a certain edge incident to one vertex is uniform for each edge incident to the vertex.

$$M_{i,j} = \begin{cases} \frac{1}{deg(v_i)} & \text{if } \{v_i, v_j\} \in E(G) \\ 0 & \text{if } \{v_i, v_j\} \notin E(G) \end{cases} \quad (1)$$

Paradigm

A random walk in G that visits a dense community will likely not leave the community until many of its vertices have been visited.

Steps

Expansion

$$M := M^e \tag{2}$$

Inflation

$$M_{i,j} := \frac{M_{i,j}^r}{\sum_j M_{i,j}^r} \tag{3}$$

Until convergence

Postprocessing

$$A_{i,j} = \begin{cases} 1 & \text{if } M_{i,j} > t \\ 0 & \text{if } M_{i,j} \leq t \end{cases} \tag{4}$$

Identify connected components

Improvements

Selfloops

- Problem: Odd powers - odd lenghts and vice versa

Line graph $L(G)$

- Reliability incorporation
- Information about larger neighborhood
- Overlapping communities

Phase one

1. Find all maximal cliques
2. Construct distace matrix D

Distance matrix

$$D_{i,j} = \frac{|C| - |C_i \bigcap C_j|}{|C|} \tag{5}$$

Where C is the set of maximal cliques

Dendrogram construction

Iterative merging of communities with minimal distance

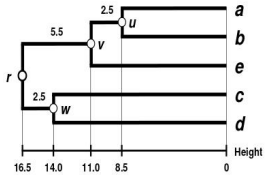
Matrix update

$$D_{A\cup B,C} = \frac{d(A,C)\cdot|A| + d(B,C)\cdot|B|}{|A|+|B|} \quad (6)$$

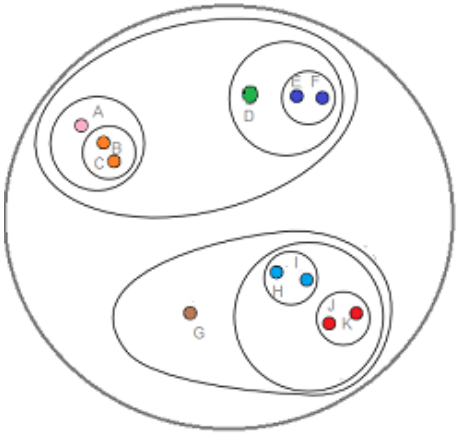
	a	b	c	d	e
a	0	17	21	31	23
b	17	0	30	34	21
c	21	30	0	28	39
d	31	34	28	0	43
e	23	21	39	43	0

	((a, b), e)	c	d
((a, b), e)	0	**30**	**36**
c	**30**	0	*28*
d	**36**	*28*	0

	(a, b)	c	d	e
(a, b)	0	**25.5**	**32.5**	**22**
c	**25.5**	0	*28*	*39*
d	**32.5**	*28*	0	*43*
e	**22**	*39*	*43*	0



From dendrogram to communities



Identify most probable communities

Modularity

$$Q = Tr(E) - ||E^2|| \tag{7}$$

Where $||E||$ is a sum is the sum of all elements of the matrix E

H-index

$$H = -\log \sum_{j=p}^{\min(M,n)} \frac{\binom{M}{j}\binom{F-M}{n-j}}{\binom{F}{n}} \tag{8}$$

Where p is the total number of direct intracommunity interactions, M is the maximum possible number of intracommunity direct interactions, F is the maximum possible number of edges, and n is the number of edges.

Basic algorithm

1. give every $v \in V(G)$ different label
2. while $\exists v : \ell(v) \notin \ell_{max}(v)$:
   1. for all $v \in V(G)$ assign v the most occuring label among its neighbors

Problems

- iteration in random order
- more labels with maximal frequency
- non-overlapping communities

Link label propagation algorithm

Edge similarity

$$S(e_{i,j}, e_{j,k}) = \frac{\sum_{x,y \in \{i,j,k\} \wedge x \neq y} |N(x) \bigcap N(y)|}{\sum_{x,y \in \{i,j,k\} \wedge x \neq y} |N(x) \bigcup N(y)|} \tag{9}$$

Where $N(x)$ is the neighborhood of a vertex x.

Edge weight

$$W(e_{i,j}) = \frac{\sum_{e_{x,y} \in N(e_{i,j})} S(e_{i,j}, e_{x,y})}{|N(e_{i,j})|} \tag{10}$$

Determines the order of iteration

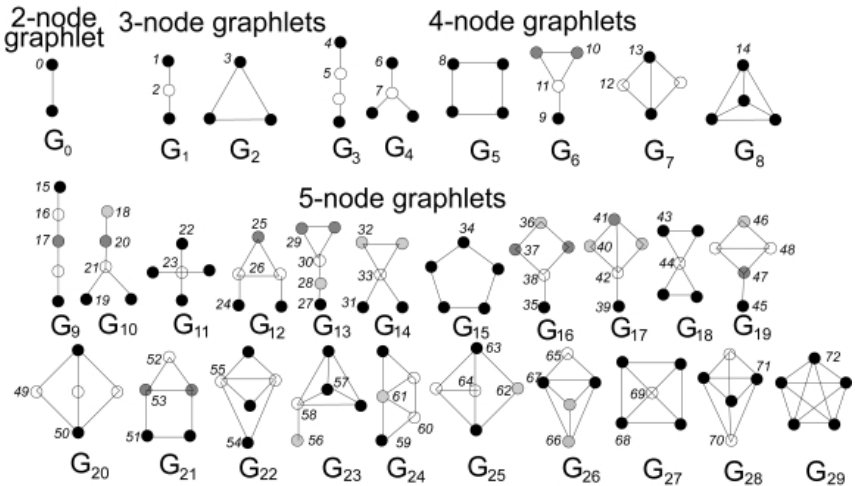
Label selection

$$\ell(e_{i,j}) = \text{argmax}_{\ell} \sum_{e_{x,y} \in N(e_{i,j}) \wedge \ell(e_{x,y}) = \ell} S(e_{i,j}, e_{x,y}) \quad (11)$$

Algorithm

1. assign a unique label for each edge
2. calculate the similarity of each pair of edge
3. calculate the weight of each edge
4. update the label of each edge in descending order of their weight until the maximum number of iterations is reached.
5. assign a label to each vertex according to the labels of incident edges
6. identify communities of vertices with the same label

Graphlets


2-node graphlet
3-node graphlets
4-node graphlets
5-node graphlets
G0
G1
G2
G3
G4
G5
G6
G7
G8
G9
G10
G11
G12
G13
G14
G15
G16
G17
G18
G19
G20
G21
G22
G23
G24
G25
G26
G27
G28
G29

Orbits

- importance of the position in the graphlet - automorphism orbits
- 73 orbits - vector of 73 elements

Weight of a graphlet

The less orbits the orbit contains, the bigger weight.

$$w_i = 1 - \frac{\log(o_i)}{\log(73)} \tag{12}$$

Where o_i is the number of orbiths a orbit i contains.

Algorithm

1. for every $v \in V(G)$ create a vector by calculating for every orbit the number of such orbits the vertex is involved in
2. calculate a vector similarity for every two vectors
3. for every $v \in V(G)$ identify all vertices with a similary above a threshold

Vector distance

Distance for one orbit

$$D_i(u, v) = w_i \cdot \frac{|\log(v_i + 1) - \log(u_i + 1)|}{\log(\max\{u_i, v_i\} + 2)} \tag{13}$$

Distance for a whole vector

$$D(u, v) = \frac{\sum_{i=0}^{72} D_i}{\sum_{i=0}^{72} w_i} \tag{14}$$

Vector similarity

$$S(u, v) = 1 - D(u, v) \tag{15}$$

Similarity threshold about $0.9 - 0.95$

Evaluation

Method	Overlap	Reliability incorp.	Running time	Specific structure
MCL	no	no	good	no
Jerarca	no	yes	slow	no
LLPA	yes	yes	good	yes
Graphlet	yes	no	fair	yes