

BACHELORTHESIS ABOUT COMMUNITY STRUCTURE IN
GRAPHS

USING NETWORK PROPAGATION METHODS
AND APPLICATIONS IN PROTEIN FUNCTION
ANNOTATIONS

Yiftach Kolb

Instructor

Prof. Dr. Matin Vingorn

October 19, 2020

1 Introduction

This Thesis deals with non-negative matrices, propagation methods, clustering and community structure in Graphs. In scientific research graphs are often used as an abstraction of a complex interaction network. They constitute a simplified display of the information which is easier to work with yet hopefully still retains the essential properties of the network. When that is the case, then insightful information about the subject of research can be gained by analysis of the graphs which are constructed out of the raw data.

In the case of an unweighted undirected graph, the vertices represent particular units that are the subject of research, be they people, proteins or whatever; and the edges represent some sort of close proximity or interaction that is assumed to exist between the two edges it connects.

The direct inspiration to this thesis is the subject of function prediction of proteins in a protein-protein interaction network (PPI). These are graphs where the vertices represent a particular type of protein and an edge connects two proteins if they are found in close proximity to each other in a sufficient portion of the samples.

It may be the case that some of the proteins in the network are well researched already and their function is known. We are interested to the functionality of the unknown proteins in the network based on the known proteins and the network structure. We assume that proteins that take place in the same function will also cluster together in the graph as they do in practice (form protein complexes, bonds or whatever).

We may be interested in finding the most relevant associate of a certain known protein that has lots of neighbors. In the terminology of propagation we are going to search for the vertex that receives the most 'heat' from the known protein, or in terms of random walk, it is the most frequently visited vertex, by a random path starting from the known vertex.

One advantage of propagation methods is that they are relatively fast, in fact in linear complexity. The typical graph can be represented as a sparse matrix and The power method starting from any starting distribution converges exponentially fast to the greatest eigenvector which is the stationary distribution. There are methods that achieve that in about $O(n + m)$ (n vertices, m edges) complexity but that is beyond the scope of this thesis.

2 Linear algebra primer

Definition 2.1. A **Transition Matrix** is a real valued non negative square (n^2) matrix A s.t. each of its columns sums up to one:

$$A \geq 0, \forall j \sum_i A_{i,j} = 1$$

A acts from the left as a linear mapping: $A : v \rightarrow A \cdot v$. In the following script we treat matrices by default as acting from the left (Av).

A transition is **positive**, designated by $A > 0$ if all its entries are positive.

A transition is **primitive** if for some $k > 0$ A^k is positive. The same property is called **regular** in some other sources.

A transition is **irreducible** if for every entry $A_{i,j}$ there is some k such that $A_{i,j}^k > 0$.

Equivalently it can be shown that A matrix A is irreducible by if and only if it is NOT similar by permutations matrices to a block upper triangular matrix which means $\nexists P : A = P \begin{pmatrix} B & C \\ 0 & D \end{pmatrix} P^{-1}$

Where P is a permutation matrix and B, C are square matrices of size greater than 0.

Definition 2.2. A **State** is a non-negative vector $v \in \mathbb{R}^n$ s.t $\sum_i v_i = 1$.

Remark 1. If v is a state and A is a transition as defined in 2.1, then Av is also a state because:

$$\sum_i (Av)_i = \sum_j v_j \left(\sum_k A_{j,k} \right) = \sum_j v_j \cdot 1 = 1$$

Also its easy to confirm by multiplying with e_i , that if Av is a state for every state v each column Ae_i must sum to 1. Therefore this is an equivalent definition for a transition.

If A is a transition and x, y are two states such that $x \leq Ay$ then $x = y$.

Definition 2.3. Given $A \in \mathbb{C}^{n \times n}$ We let $|A| \in \mathbb{C}^{n \times n}$ be the resulting matrix from applying $|\cdot|$ element wise. Given a vector $v \in \mathbb{C}^n$ we let $|v| \in \mathbb{C}^n$ the corresponding non-negative vector.

We also let $A > 0, v > 0$ mean that it holds coordinate wise.

Here is a very useful lemma for non-negative matrices which we will need later:

Lemma 2.1. Let $0 \leq A \leq B \in \mathbb{R}^{n \times n}$ and let $0 < v \in \mathbb{R}^n$. If $Av = Bv$ then $A = B$.

Proof. Trivial. □

Remark 2. If $u \in \mathbb{C}^n$ is on the unit circle and T a transition then $|u|$ is a state, meaning $\| |u| \|_1 = 1$, so $T|u|$ is also a state so $\| T|u| \|_1 = 1$.

We have (component-wise) $|Tu| \leq T|u|$. If $T > 0$ and u has negative or non-real entries, then this inequality must be strict and then $\| Tu \|_1 < \| T|u| \|_1 = 1$.

Lemma 2.2. If T is a transition, then there is a state u such that $Tu = u$.

Proof. Because the columns of A all sum to 1, the columns of $A - I$ all sum to 0. Therefore $(1, 1, \dots, 1)$ is an (left) eigenvector of the rows with eigenvalue 1. Therefore there is also some real (right) column eigenvector with eigenvalue 1. (Also it follows from the Brouer fixed point theorem because T maps the l_1 sphere to itself).

Let $u \in \mathbb{R}^n$ be such vector: $Au = u$. Let $u = u^+ + u^-$ such that $u^- = \min(u_i, 0)$ and the other one defined similarly for the non-negative components.

Because A is non-negative $A(u^+) \geq (Au)^+ \geq 0$, and $A(u^-) \leq (Au)^- \leq 0$.

From A being non-negative and $(Au)^+ + (Au)^- = Au = u = u^+ + u^-$ And also $(Au)^+ = (u)^+$, so we must have $Au^+ \geq (Au)^+ = u^+$ (component wise). But if we had a strict inequality we would get: $\| A(u^+ / \|u^+\|_1) \|_1 > 1$ which is a contradiction to A being a transition matrix.

Then $Au^+ = u^+, Au^- = u^-$ and one of them must be non-zero. It follows that A has a non-negative eigenvector with eigenvalue 1 (one of $u^+, -u^-$ which is non-zero). If we l_1 -normalize that eigenvector it becomes a state. □

Lemma 2.3. *If a transition $A > 0$ (or primitive) , then it has exactly one eigenvector v with eigenvalue 1 and in addition it can be chosen so that $v > 0$ Any for any other eigenvalue λ it holds that $|\lambda| < 1$.*

If A is just irreducible then then again $v > 0$ and is unique but there may be additional eigenvalues on the unit circle.

Proof. Let $A > 0$ be a transition. We already know that there exists at least one such eigenvector. Let u, v s.t $Au = u, Av = v$. We can assume these are real vectors because A has only real entries. Therefore we can choose u, v to be states as we have already proven above.

Then let $w = u - v$. So $Aw = A(u - v) = u - v = w$. And $\sum_i w_i = 1 - 1 = 0$ by choice of u, v .

Like before $Aw^+ = w^+, Aw^- = w^-$ and because $w \neq 0$ but sums up to 0, both $w^+, -w^- > 0$. Because w^- is non zero exactly on entries where w^+ is zero and vice versa, each of them must have at least one 0 entry (and one none zero). But because A is strictly positive and w^+ is non-negative, Aw^+ must have ONLY positive entries, contradicting $Aw^+ = w^+$. It follows then that $u - v = 0$ is the only possibility, hence the eigenvector with 1 as eigenvalue is unique.

Suppose there is $Aw = \lambda w$ where $|\lambda| = 1$. Choose w so it is l_1 -normalized. Then $|w| = |Aw| \leq A \cdot |w|$ If w has any negative or complex coordinantes, then $|Aw| < A|w|$ and therefore $1 = ||Aw||_1 < \|A|w|\|_1 = 1$, a contradiction. Therefore there cannot be any other eigenvalues on the unit circle.

Extending this for primitive matrix is easy because for some sufficiently big k $A^k > 0$.

To prove the uniqueness of the 1-eigenvector for the irreducible case, we have $(\forall k \in \mathbb{N}) A^k w^+ = w^+$ and from that with some more work left undone it follows that $w^+ > 0$ or $w^+ = 0$. \square

\square

Remark 3. The lemmas and theorems in this section are phrased in term of transitions. They hold true in the more general case of positive/non-negative linear transformations and one just replaces 1 with the **spectral radius** $\rho = \rho(A)$.

In general a non-negative linear transformation has a **spectral radius** $\rho = \rho(A)$ which is the absolute value of its greatest eigenvalue. In the case of positive maps there is a unique single eigenvector with ρ as the unique greatest eigenvalue and so forth When we deal with a transition map, lemma 2.3 guaranties it has a spectral value of $\rho(A) = 1$.

Theorem 2.1. *Let T be a positive (or primitive) transition. Then*

1. 1 is the greatest eigenvalue of T and it has one unique eigenvector which is also positive, so there exists a unique stationary state.

2. All the other eigenvalues have absolute value strictly less than 1.

3. For every state u , $T^k u$ converges to the stationary state π . In particular the columns of T^k converge to π .

Proof. 1 and 2. We already know.

3. There is a Jordan decomposition $T = PJP^{-1}$, such that J is a Jordan matrix, $J_{1,1} = 1$ and the rest of the main diagonal $|J_{i,i}| < 1$. So now the convergence is clear $J^k \rightarrow (e_1 | 0 \dots | 0)$. For the matrix P it must hold that $P = (P_1 | \dots | P_n)$ and P_1 is the column eigenvector of T corresponding

to 1 which we are free to l_1 normalize and the first row of P^{-1} is the left eigenvector corresponding to 1 and so force ...

Some more work or literature check should confirm that $T^k \rightarrow (v|v \dots |v) = V$. Then one can verify $Vu = v$ for any state u . \square

Theorem 2.2. *Let T be an irreducible positive (or primitive) transition. Then:*

1. *Then 1 is the greatest eigenvalue of T and it has one unique eigenvector which is also positive, so there exists a unique stationary state.*
2. *If there are other other eigenvalues on the unit circle then their algebraic multiplicity is equal their geometric multiplicity.*
3. *For every state u , the **Cesaro sums** $\frac{1}{n} \sum_{k=1}^n T^k u$ converge to the stationary state π .*

Proof. 1 We already know.

2. There is a Jordan decomposition $T = PJP^{-1}$. such that J is a Jordan matrix, $j_{1,1} = 1$ and the rest of the main diagonal $|J_{i,i}| \leq 1$.

If we had a Jordan block of size greater than 1 for an eigenvalue λ , Then on the superdiagonal of J^k we would have $k\lambda$. If $|\lambda| = 1$ then J^k and hence T^k would be unbounded, but that is impossible since T^k is a transition. It follows that all eigenvalues on the unit circle must be semi simple (algebraic multiplicity equals geometric).

3. For the convergence, it follows from calculations on the Jordan blocks, which I omit. See Meyer [5] or Serre [8] for rigorous proof. \square

What differs irreducible non-primitive matrices from primitive is that they are periodical on their eigenvectors with complex eigenvalues on the unit cycle. There is, in fact a wonderful theorem from Wielandt which characterizes these Matrices:

Theorem 2.3 (Wielandt (1950)). *Let $A, B \in \mathbb{C}^{n \times n}$ such that $A \geq 0$ is irreducible and $|B| \leq A$ (component-wise). Then $\rho(B) \leq \rho(A)$. If $\rho(B) = \rho(A)$ then $|B| = A$, B has an eigenvalue of the form $\mu = \rho(A)e^{i\phi}$ and:*

$$B = e^{i\phi} D A D^{-1}$$

where D has the form:

$$D = \begin{pmatrix} e^{\theta_1} & & & \\ & e^{\theta_2} & & \\ & & \ddots & \\ & & & e^{\theta_2} \end{pmatrix} \quad (1)$$

And conversely any B of the form 1 has $\rho(B) = \rho(A)$, $|B| = A$ and μ is an eigenvalue of B which corresponds to the eigenvalue $\rho(A) = |\mu|$ the greatest eigenvalue of A .

Proof. To see a rigorous proof I suggest Meyer [5].

The keys for proving this theorem are: First WLOG assume A is a transition. This is possible because we may replace A with AW^{-1} , and B with BW^{-1} , where W is the diagonal matrix that has the column-sums of A . Since A is irreducible it cannot have a column or a row that is all 0 so

this diagonal is positive W is indeed invertible and later we can cancel out the W 's and return to the general case.

Let v be the μ -eigenvector $Bv = \mu v$, $\|\mu\| = 1$ and choose it so that $\|v\|_1 = 1$.

Then

$$|v| = |\mu v| = |Bv| \leq |B||v| \leq A|v| \quad (2)$$

Since A is a transition by remark 1 $A|v| = |v|$. Since A is irreducible and $A|v| = |v|$ we must have by 2.2 that $|v| > 0$, and then by lemma 2.1 $A = |B|$ so that proves the first part.

Now let $w = v/|v|$ (component-wise division) and let

$$D = \text{diag}(w) = \begin{pmatrix} e^{\theta_1} & & & \\ & e^{\theta_2} & & \\ & & \ddots & \\ & & & e^{\theta_n} \end{pmatrix}$$

Then $v = D|v|$, $|v| = D^{-1}v$ and we have:

$$\begin{aligned} A|v| &= |v| = D^{-1}v = \\ &= D^{-1}\mu^{-1}Bv = \mu^{-1}D^{-1}BD|v| \end{aligned} \quad (3)$$

We know already that $|v| > 0$. If $C := \mu^{-1}D^{-1}BD$ contains any negative or complex entries, then 3 cannot hold. It follows that

$$A = C = \mu^{-1}D^{-1}BD$$

This proves the harder direction of the claim, the other direction is easy \square .

\square

The amazing consequence from 2.3:

Theorem 2.4 (Corollary). *If A is irreducible transition with h eigenvalues on the unit circle then its eigenvalues are exactly the h th unit roots $\lambda_k = e^{2\pi i k/h}$, $k = 0 \dots h-1$ and A is similar to λA by a diagonal matrix for any such eigenvalue.*

Proof. Use theorem 2.3 with $B = A$. If $|\lambda| = 1$ is an eigenvalue then $A = \lambda D A D^{-1}$. Since similarity doesn't change the eigenvalues, A and λA must have the same eigenvalues with the same multiplicity. Since 1 is a simple eigenvalue of A and hence of λA , and its corresponding eigenvalue λ is simple in λA and therefore in A as well.

The matrices $\lambda_k A$, $k = 0 \dots h$ are all similar, with $\lambda_0 = 1$ and $|\lambda_k| = 1$, $k = 0 \dots h-1$ all simple eigenvalues. The only way for this to hold is if $\{\lambda_k\}_0^{h-1}$ form a multiplicative group of order h on the unit circle, in other words, the eigenvalues are exactly all the h th unit roots. \square

Remark 4. If A is irreducible transition with exactly h eigenvalues on the unit circle then h is called the **period** of A . $A \geq 0$ is primitive if and only if it is irreducible and aperiodic ($h = 1$).

If $\omega = e^{2\pi i/h}$ then 2.4 shows that $A = \omega D A D^{-1}$. So if λ is any eigenvalue not necessarily on the unit circle, then $\omega\lambda$ is also an eigenvalue with the same multiplicity and rotation with ω is an automorphism on the eigenvalues.

We may choose D so that $D[1, 1] = 1$. If we reindex the dimension we can make D look like

$$D = [D_0 | \omega D_1 | \dots \omega^{h-1} D_{h-1}]$$

So Indexes corresponding to the same phase of the period appear sequentially.

Then use the identity $\forall k D^h A^k = A^k D^h$ and the fact that A is irreducible to show that $D^h = I$. Then use the identity $\omega D A = A D$ to show that in the new indexing A has the following periodical block structure (0 on the main diagonal):

$$A = \begin{pmatrix} 0 & M_1 & 0 & \dots & 0 \\ 0 & 0 & M_2 & 0 \dots & 0 \\ & & \ddots & \ddots & \\ 0 & \dots & 0 & 0 & M_{h-1} \\ M_h & 0 & \dots & 0 & 0 \end{pmatrix}$$

Now we will just present the Perron-Frobenius theorems. The main parts that are important to our work have appeared in the previous theorems.

Theorem 2.5 (Perron-Frobenius [5]). *Let $0 < A \in \mathbb{R}^{n \times n}$ with spectral radius $\rho := \rho(A)$, then the following are all true:*

- $\rho > 0$
- ρ is a simple root of the characteristic polynomial of A , in other words its algebraic multiplicity is 1.
- $(\exists v > 0) Av = \rho v$
- If $Au = \lambda u$ and $\|u\| = \rho$ then $\lambda = \rho$ namely, ρ is the unique eigenvalue on the spectral circle.
- (Collatz-Wielandt Formula) $\rho = \max_{x \in \Gamma} \min_{i: x_i \neq 0} [Ax]_i / x_i$ with $\Gamma = \{x | x \geq 0, x \neq 0\}$

Theorem 2.6 (Perron-Frobenius for irreducible matrices [5]). *Let $0 \leq A \in \mathbb{R}^{n \times n}$ be irreducible with spectral radius $\rho := \rho(A)$, then the following are all true:*

- $\rho > 0$
- ρ is a simple root of the characteristic polynomial of A , in other words its algebraic multiplicity is 1.
- $(\exists v > 0) Av = \rho v$
- There are no additional non-negative unit eigenvectors of A other than v .
- (Collatz-Wielandt Formula) $\rho = \max_{x \in \Gamma} \min_{i: x_i \neq 0} [Ax]_i / x_i$ with $\Gamma = \{x | x \geq 0, x \neq 0\}$

3 More on matrices, graphs and stochastics

A directed non-weighted graph G can be uniquely represented by its **adjacency matrix**, $A_{i,j} := 1$ if and only if there is a directed edge from i to j (if we want to use it for transitioning on columns as done above). It's possible to assign different edge weights rather than only 1 and 0. If the graph is undirected each edge would be counted in both directions and the matrix is symmetric. Relabeling the vertices of a graph yields an adjacency matrix that is similar by permutations (PAP^{-1} , where P is a permutation matrix) and vice versa.

To turn A into a transition, normalize each column by dividing it with its out going rank, so let $D_{i,i} = \text{out rank}(i)$, $T := AD^{-1}$ is the transition matrix of this graph (because right-multiplying by D normalizes each column by its rank). If the graph was stochastic to begin with then the adjacency matrix as we defined it is already column-normalized.

Definition 3.1. A graph is G **strongly connected** if there is a directed path from any edge to any edge. Equivalently G is strongly connected if and only if its adjacency matrix is irreducible.

We say that the **Period of a vertex** $v \in V(G)$ is the greatest common divisor of all closed paths on v . If the periods of all vertices are equal (spoiler: in the case that G is strongly connected they are), we call it the **Period of the graph** G .

Remark 5. If G is strongly connected then the periods of all vertices are indeed equal and it's easy to prove. The corresponding adjacency matrix A is irreducible so it too has a period h as defined in 4 and it is equal to the graph period (can be shown using 2.4 and 4).

So if the graph G is strongly connected and has period 1 then the adjacency matrix is **aperiodic** and hence primitive, and vice versa.

From all the above we have seen that a Markov process can be represented in two equivalent ways —as a transition matrix and as the corresponding weighted directed graph.

If the graph G is strongly connected and aperiodic, its corresponding adjacency matrix is primitive. We know from 2.5 that there is a unique stationary distribution p and that the Markov converges to p no matter from which distribution it starts. We may calculate p using the **power method** which is efficient because it can be done in a matrix-free method. We don't need to know the matrix itself just the dot product of it with a state vector.

If the graph G is strongly connected, then 2.6 assures us the existence and uniqueness of a stationary distribution p . But if G is not aperiodic, the corresponding adjacency matrix is not primitive. We cannot use the efficient power method to calculate p . Also the process itself doesn't stabilize on p . It is periodic and cycles between the h eigenvectors on the unit circle.

We are therefore interested to find how to convert an imprimitive matrix (= irreducible but not primitive) to a primitive matrix or equivalently to turn a strongly connected but periodic graph into an aperiodic graph.

Lemma 3.1. Let $A \geq 0$ be irreducible. Then $(A + I)^{n-1} > 0$, and therefore $A + I$ is primitive.

Proof. Notice that the I represents self loops and it absorbs all the lower powers so if $(A^k)_{i,j} > 0$ for some $k < n$ then so is $(A + I)^{n-1}$. Let $G := G_A$ be the corresponding graph to A . Then G is strongly connected. For every $i \neq j$ there is a directed **shortest path** σ in G from i to j . Its length must be $|\sigma| < n$. Otherwise σ would have to contain a cycle and not be of minimal length. This shows that we have for every $i \neq j$ some k such that $A_{i,j}^k > 0$ and therefore $(A + I)^{n-1} > 0$. \square

The properties of irreducibility, primitiveness and positivity only depend on the sign $(-, 0, +)$

of the entries and not on their size. So we use the following definition to test matrices for these properties:

Definition 3.2. Let $A \in \mathbb{R}^{n \times n}$, then its binary form is the matrix $\beta(A) := \text{sgn}(A)$ where sgn is applied element-wise.

The following trivial lemmas would help as to construct primitive transitions:

Lemma 3.2. A is positive/primitive/irreducibly if and only if $\beta(A)$ is.

Let $0 \leq \beta(A) \leq \beta(B)$. Then If A is positive/primitive/irreducibly so is B .

Let $0 < \alpha < 1$. If T, S are transitions the so is $W = (1 - \alpha)T + \alpha S$. If one of S, T is positive/primitive/irreducible so is W

Let G be any weighted graph and A its adjacency transition matrix. Some vertices may be unreachable from other vertices and there might not exist a single and unique stationary distribution. A random walk on this graph is generally dependent on the initial state. However if we allow the possibility of 'random restart' from any state, this graph becomes totally connected it is guarantied to have a unique stationary distribution to which any random walk converges regardless of the initial state.

In terms of matrices, we create a positive transition matrix $W = (1 - \alpha)A + \alpha \frac{1}{n}$ > 0 . Where $\frac{1}{n}$ is the $n \times n$ matrix whose entries are all $\frac{1}{n}$. The stationary distribution is called the **PageRank** for G with restart parameter α . It is used to order the vertices according to their 'relevance' in the network.

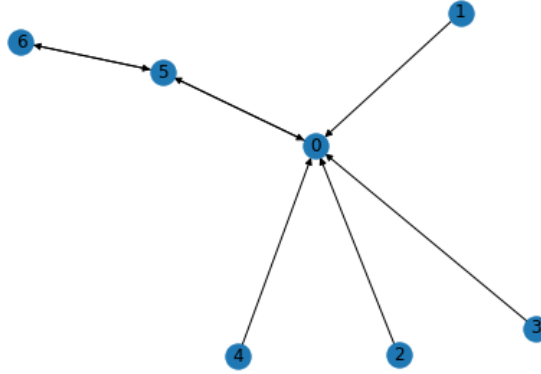


Figure 1: This is an example of a weakly connected graph. If we start walking from 0, 5 or 6 we can never reach vertices 1 – 4. It is also not immediately clear which vertex 0 or 5 is more 'important'. While 0 is directly connected to more vertices, 5 may get more 'flow' through it since every path of length 2 or more passes through it.

The PageRank is the stationary distribution of the process when we use unbiased restart—A restart is equally likely from any vertex. But we want more. We want to find out what happens when we restart, for example, always from one single vertex u . We think of the stationary distribution p_u that results from such process as the heat (or flow) which propagates out of u . If we take another vertex v we think of $p_u[v]$ as a measure of how close v is to u , or how much heat u sends to v . Note that this is not symmetric $p_v[u] \neq p_u[v]$ in general.

Lemma 3.3. Let $A \geq 0$ be irreducible. Let $B \geq 0$ have a positive row (or column), then $A + B$ is primitive.

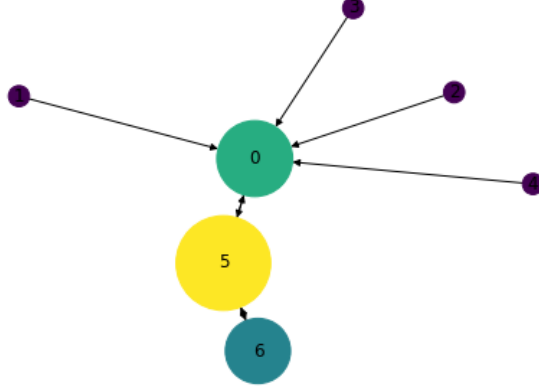


Figure 2: The same graph with vertex size and color indicating its PageRank with parameter $\alpha = 0.15$. We see that 5 is ranked first, followed by 0. The smaller the restart parameter, the more important 5 will get because we allow for longer paths with fewer restarts.

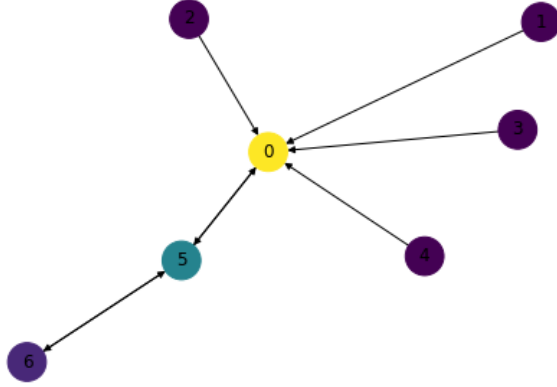


Figure 3: When the restart parameter is too large, the rank becomes almost uniform because the convex combination of the original graph with the n -clique graph of the restarts is weighted too heavily on the latter. It also shows that for high restart values 0 becomes hotter than 5. That is because paths of length > 2 (before restart) become very unlikely in this random walk.

Proof. Suppose WLG that $B_1 > 0$ (first row). Then $\forall k > (B^k)_1 > 0$. Fix i, j . Since A represents a strongly connected graph, there is a path from $i \leadsto 1 \leadsto j$ of some length k . So $A_{i,j}^k > 0$. Then $\forall l \geq 0 (A + B)^{k+l} > 0$ because we can go along this path, then self loop l times in 1 and keep going to j on that path.

So we have shown that $(A + B)_{i,j}^k$ stays positive once it becomes positive and it always turns positive at some point by irreducibility of A , so that means $A + B$ is primitive. \square

Remark 6. Lemma 3.3 proves that we can propagate from any arbitrary restart group including a single vertex and the combined transition matrix will be primitive if the adjacency matrix A is strongly connected.

Let q be a state column vector for example e_1 and let $Q = (q|q|\dots|q)$. So Q has a positive row and therefore $(1 - \alpha)AD + \alpha Q$ is a primitive transition.

Also worth noting that if $x \geq 0$ is any transition, then $Qx = q$.

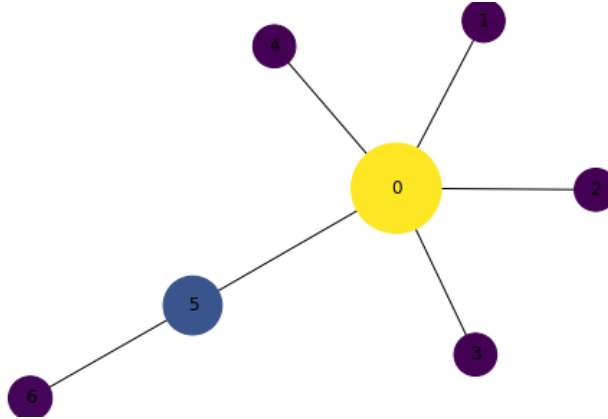


Figure 4: When we treat the graph as undirected and calculate the PageRanks ($\alpha = 0.15$), Vertex 0 comes this time on top. It is more central than 5 and more random paths intersect it than any other vertex.

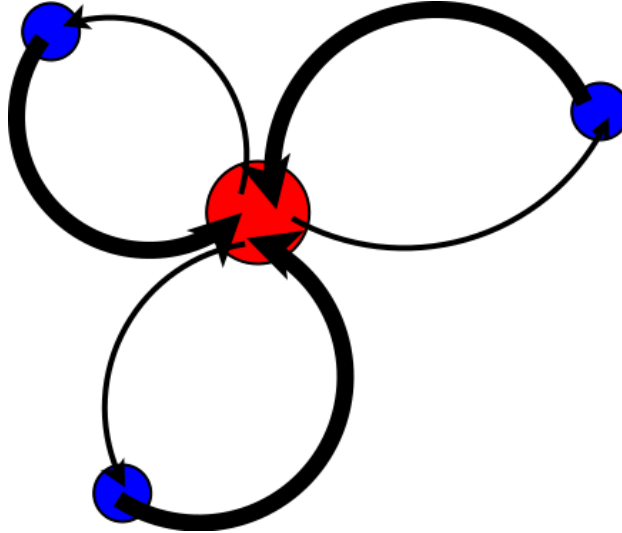


Figure 5: Stars have hot center because it spreads its heat on many sources, while each of the orbiting vertices sends all its heat into the star center.

If A is the adjacency matrix of G we defined the transition $T = AD^{-1}$. We define the transition matrix with restart parameter α and bias q as

$$T_{\alpha,q} := (1 - \alpha)T + \alpha Q$$

This matrix has a unique stationary distribution p so And use 6:

$$p = Ip = [(1 - \alpha)T + \alpha Q]p = (1 - \alpha)Tp + \alpha q$$

We can rearrange it now

$$(I - (1 - \alpha)T)p = \alpha q \tag{4}$$

We want to invert the matrix in 4 and use the following lemma to justify it (The proof is easy. See for example Serre [8]):

Lemma 3.4. *Let X be a contracting matrix, Then $(I - X)$ is invertible and the power sum of X converges to it:*

$$(I - X)^{-1} = \sum_{k=0}^{\infty} X^k$$

So now we may apply lemma 3.4 on equation 4 because $(1 - \alpha)A$ is contracting, and we have:

$$p = \alpha[I - (1 - \alpha)T]^{-1}q := Kq = \alpha \sum_{k=0}^{\infty} (1 - \alpha)^k T^k \quad (5)$$

K is called **diffusion kernel** of T with parameter α . It turns out that K itself is a transition because it maps the arbitrary transition q to the transition p . In addition, $K > 0$ because of 5.

What about the eigenvectors and eigenvalues of K ? If a matrix A is invertible then $Av = \lambda v \iff A^{-1}v = \lambda^{-1}v$. For any matrix $Av = \lambda v \iff (I + A)v = (1 + \lambda)v$.

It turns out then that T and K have the same eigenvectors and if $Tv = \lambda v$ then $Kv = \alpha[1 - (1 - \alpha)\lambda]^{-1}v$. And in particular if it turns out that all the eigenvalues are real (spoiler—they are), then they have the same order as eigenvalues of K or T for the same eigenvector. In particular we see that the choice of α , the restart parameter, NEITHER changes the eigenvectors NOR does it change the order of the eigenvalues.

To sum up the important facts that we would need later:

Theorem 3.1. *Let G be strongly connected undirected graph. Let A be its adjacency matrix and D the diagonal matrix which has the ranks of the vertices on its diagonal. Let $T = AD^{-1}$, let $0 < \alpha < 1$ and $K = \alpha[I - (1 - \alpha)T]^{-1}$. Then the following are all true:*

- $A \geq 0$ is symmetric therefore its eigenvalues are all real.
- $T = AD^{-1} = D^{1/2}[D^{-1/2}AD^{-1/2}]D^{-1/2}$ is similar to A . Therefore it has the same eigenvalues as A , which are all real: $\lambda_1 \geq \dots \lambda_n$.
- There exists for all $0 < \alpha < 1$ the invertible matrix: $K := \alpha \sum_{k=0}^{\infty} (1 - \alpha)^k T^k = \alpha[I - (1 - \alpha)T]^{-1}$. $K > 0$ is a transition. It has the same eigenvectors as T and their corresponding eigenvalues are all real and they have the same order as the corresponding eigenvalues for T have.

Finally as a side note the name diffusion kernel originates from the heat equation which associated with the graph, considering the graph as perfectly insulated system. We won't get deeper into that at this time.

4 A naive yet not useless propagation method for clustering a graph

Definition 4.1. Let G be a strongly connected graph and K the diffusion kernel as defined in 3.4, end let $u, v \in \{1 \dots n\}$ be two vertices and $e_u, e_v \in \mathbb{R}^n$ their corresponding index vectors.

$p_u := Ke_u$, the stationary distribution with random restart from u , is called the **influence vector** of vertex u . We say that $p_u[v]$ is the **influence of** u on v .

A p_u is a measure of how 'heat' which is pumped into u propagates in the graph. When we try to cluster the graph, it is natural to think that If vertex v is the top receiver from u , namely $v = \operatorname{argmax}(p_u)$ then maybe these 2 belong in the same cluster.

Informally we say that a vertex is 'hot' or 'cold' if it has a high (hot) or low (cold) PageRank. Remember that the PageRank is the unbiased stationary distribution $p = K \cdot (1/n, \dots, 1/n)^t$.

So when we start clustering a graph, I suggest that it makes sense to pick up a cold vertex and associate it with the vertex to which it sends most of its heat. If we start from a hot vertex, it usually has many neighbours and it doesn't send allot of heat down a single vertex.

The algorithm works as follows:

```

1 function coolWarmClustering(G, k)
2   # Input G = (V,E) : a directed strongly connected graph.
3   # Input m : The desired number of clusters
4   K <- diffusion_kernel(G)
5   H <- Disconnected_undirected_graph_on(V) # each vertex starts in its
        own cluster
6   p <- pageRank(G)
7   vlist <- arg_sort(p) # vertex-list sorted up by PageRank
8   While |connected_components(H)| > k do
9     x <- vlist.pop() # take the coldest remaining vertex and remove it
        from the
10          # list
11     p_x <- K * e_x # Influence vector of x
12     y <- argmax(p_x)
13     H.add_edge(x,y)
14   return H

```

The algorithm clusters the vertices by constructing a new graph on the same vertices, successively joining vertices. It picks the coldest remaining vertex and joins it with the vertex to which it propagates the most heat.

In every iteration of the algorithm adds an edge, which either reduces the number of connected components by 1 or keeps it unchanged. Because G is strongly connected algorithm will reduce the number of connected components in H until it reaches the desired number of components k .

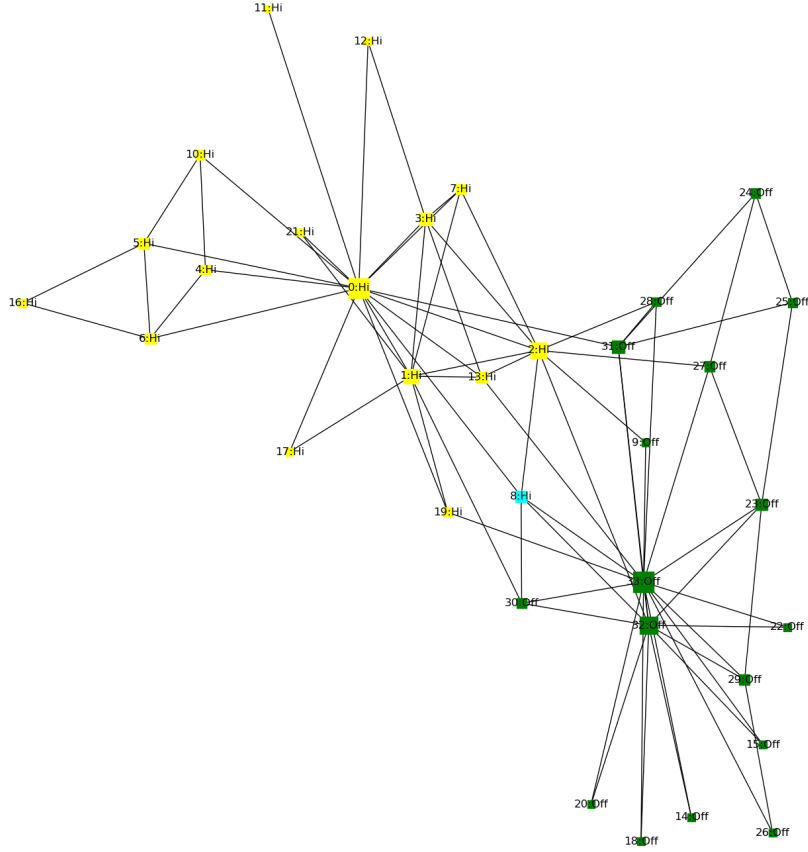
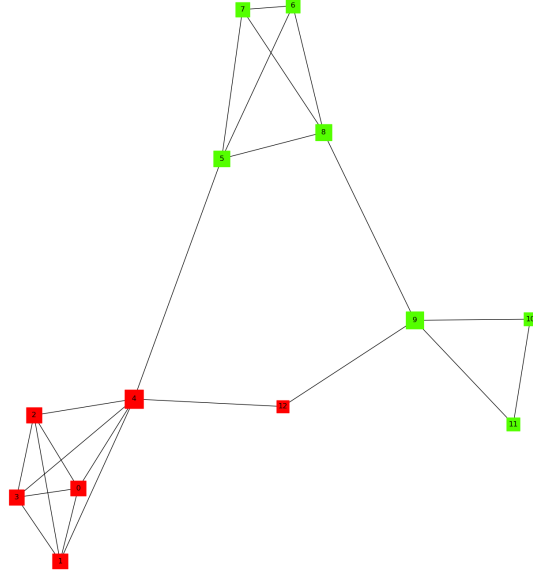
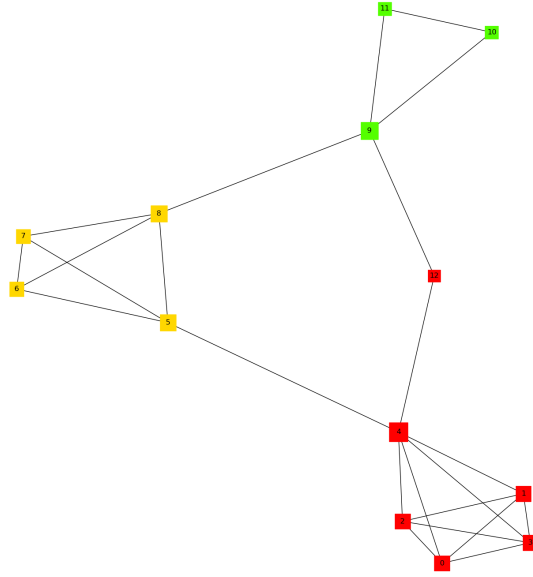


Figure 6: The Famous 'Zachary's Karate Club' Graph. The labels show the actual division between the 'Mr Hi' and 'Officer' groups. Vertex size indicates its PageRank. The colors encode the result of the coolWarmClustering compared to the actual partition. It made 1 mistake: It associates wrongly vertex 8 to 'Officer'. This vertex is hard to resolve because it is connected to hubs in both clusters. In fact a further check confirms that the total sum of 'heat' it sends to members of 'Officer' is 0.45 vs 0.36 to members of 'Mr Hi' (excluding itself in both cases). So in some sense the algorithm is more correct then the actual partition in real life. Perhaps this is due to human irrationality? On the other hand, 8 receives more heat units from 17 members of 'Mr. Hi': 0.485 units, compared to 0.345 units from the 16 members of 'Officer'.



(a) A 2 cluster partition



(b) A 3 cluster partition

Figure 7: The result of running the algorithm set for finding a 2-partition and respectively a 3-partition. The interesting node is 12. In both runs 12 chooses to associate with the biggest clique because node 4 of that clique is the one which receives the most heat from 12

5 Spectral clustering and other methods

Spectral Partitioning [12, 6, 2, 9], is a common technique for partitioning graphs. The idea is to construct a Laplacian type matrix [3] for the graph G and analyze its eigenvalues and eigenvectors and construct a 2-partition of G . It is possible to create a hierarchical partitioning of G by repeating the process on each subdivision.

The problem of finding the best graph partition, also called the minimal cut problem is NP-hard. Spectral partitioning methods are heuristics that approximate the solution in 'real life' classes of networks.

Definition 5.1. Let G be a bidirectional graph with adjacency matrix A and diagonal degree matrix D . Then the following matrices are its **graph Laplacian**, and its **symmetric—, row-normalized—and column-normalized—Laplacians**:

$$\begin{aligned} L &= D - A \\ L_s &= D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2} \\ L_r &= D^{-1} L = I - D^{-1} A \\ L_c &= L D^{-1} = I - A D^{-1} \end{aligned} \tag{6}$$

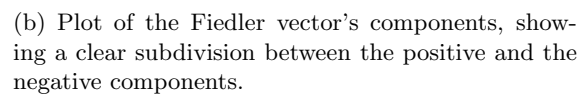
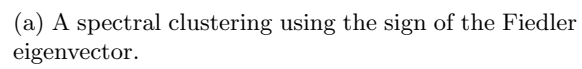
Consider the connected graph G and its adjacency matrix A . We create the transition matrix by normalizing it: $T = AD^{-1}$, and finally we choose a restart parameter α and create the diffusion kernel K . Theorem 3.1 assures us that K and T have the same eigenvectors and the orders of their corresponding eigenvalues are the same. The eigenvalues are all real and therefore the eigenvectors are real as well. And the eigenvalues of K are all non-negative.

The matrix $K^{-1} = \alpha^{-1}[I - (1 - \alpha)AD^{-1}]$ is closely related to the and the column-normalized Laplacian L_c . The added parameter $0 < \alpha < 1$ neither changes the eigenvectors, nor the order of the corresponding eigenvalues.

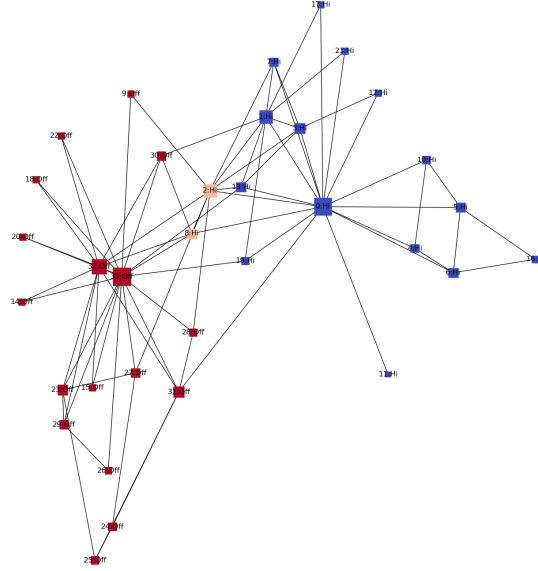
The smallest eigenvalue, 0 of L and L_s corresponds to the largest eigenvalue 1 of T and its eigenvector is (can be chosen as) all positive. Any other eigenvector of L (which corresponds to an eigenvector of T) must contain both positive and negative components due to 2.5.

It's the eigenvector of the second smallest eigenvalue L or L_s which is commonly used in They are commonly referred to as the **Fiedler** eigen-pair in the literature. for spectral partitioning [6]. The simplest method is to use the sign of the components of eigenvector of the second smallest eigenvalue of L or L_s . This eigen-pair corresponds to the second largest eigenvalue of T

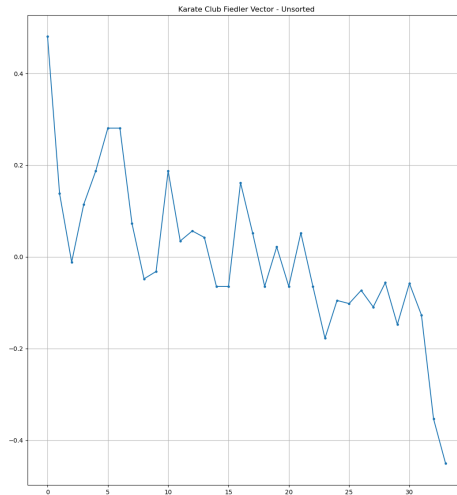
In Newman [7] there is a particular type of spectral partitioning which uses a matrix which the author called the modularity matrix. It has the advantage over the Laplacian in that its greatest eigenvalue may be positive or negative which is used to indicate or suggest the existence of community structure in the graph. However the normalized Laplacians L_r and L_c are more natural in the context of the random walk or propagation method and recommended for example in [12].



16



(a) Spectral clustering of the Karate Club network using the Fiedler eigenvector signs. There are 2 deviations from the actual partition, both are borderline nodes which are harder to resolve. The 8 node mistake is in common with the naive algorithm of the previous section.



(b) Plot of the Fiedler Vector of the Karate Club. The split between negative and positive values is much less dramatic but nonetheless still visible.

6 (Trying) function prediction method

In this section we are going to try to apply the accumulated knowledge of the previous sections and devise some simple function prediction algorithms for PPI networks. We are going to try several methods which use propagation as well as methods which just use direct neighbors for the predictions.

The network is going to be composed of around 500 vertices (proteins) which are divided into 4 different function groups. We are going to randomly choose half the proteins and mark them as unknown and try to predict their function based on the remaining known proteins.

6.a Methodology

We use the annotated yeast interactome which is a pure PPI network that comes prepackaged with the software tool 'Cytoscape'. The nodes represent yeast proteins and the edges represent protein-protein interactions.

This network comes with functional annotations in the 'MCs name' field. We select nodes in the network whose annotations contain **exactly** one of the following keywords: 'DNA Replication' (blue group), 'Golgi' (red), 'Meiosis' (yellow), and 'Stress response' (green). All other nodes are removed from the network. The result is shown on figure 10. We are going to test various annotation method on the largest connected component of this subnetwork.

Nodes whose annotations contain keywords of more then one group were discarded.

The above described subnetwork is exported from cytoscape to graphml file, which we import and convert into a networkx object. We select the largest connected component from the above network.

We set a random seed (a process which we repeat 2 times for 2 different trials). Then we select randomly 50% of the nodes and mark them as 'unknown'. The other 50% are 'known'. We are going to try to determine the group affiliation of the unknown nodes based on the group affiliations of the known nodes.

We use 5 different prediction methods:

- Method 1 Determines the group affiliation of an unknown node by the group that has the maximal volume based on the propagation distribution from the unknown node. This means: we calculate the stationary distribution with restart to the unknown node. Among the known nodes, we calculate the weighted sums of each of the 4 groups according to that distribution. The group with largest volume is the affiliation we assign to the unknown node.
- Method 2 This method works like method 1, except that we first order the unknown nodes (decreasing order) according to their pageRank. We go over the nodes in this order and assign group affiliation according to method 1, but then we also update the list of the known nodes to include that node. So this node is going to be included in the function prediction of the lower ranked nodes.

The rational here is that the higher ranked nodes are probably going to be predicted with high accuracy and therefore be helpful in prediction of lower ranked nodes.

It is probably better to devise some simple rule to decide whether it is worth to include a node in the prediction of the rest of the nodes. For example, based on how many unknown neighbors in has vs known neighbors. But we tried to keep things simple at this stage.

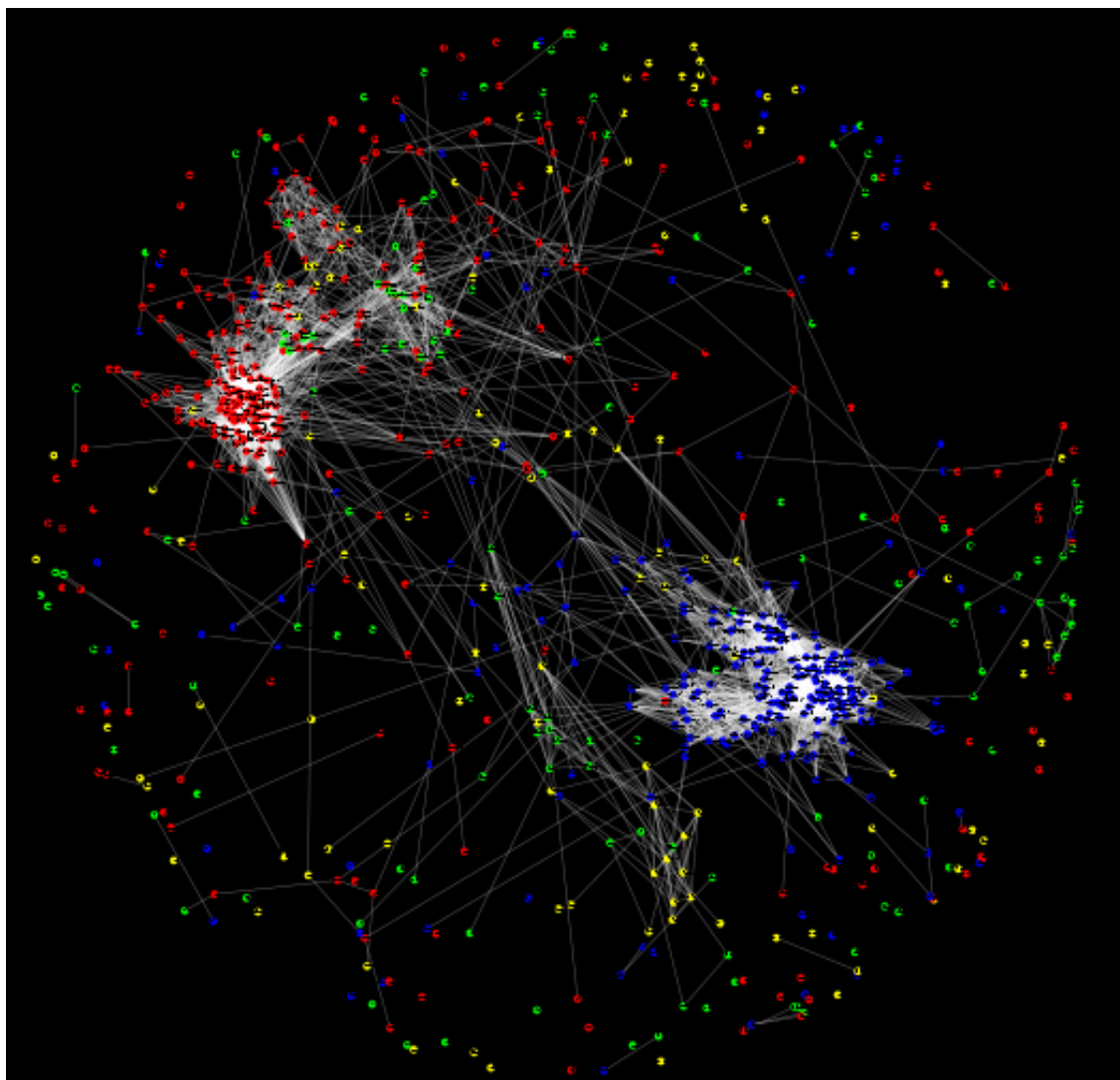


Figure 10: Subgraph of the yeast interactome showing nodes of 4 groups: 'DNA replication' (blue), 'Golgi' (red), 'Meiosis' (yellow), 'Stress response' (green)

- Method 3 We assign an unknown node to the group that has plurality among its neighbors with known group affiliation. So this method is fast and the probably the simplest.
- Method 4 Same as method 3, but again we go in the order of pagRanks and we update the list of known nodes on the fly, like method 2.
- Method 5 Here we take each group of the known nodes and propagate from it. So for example we calculate the stationary distribution when we restart in the known nodes that belong to group 'DNA Replication', and then for the next group and so forth. For each unknown node, we assign it to the group that propagates the highest probability to it.

So while in method 1 and 2, we start random walking from an unknown node and see what is the probability that we land at a certain group. In Method 5 we start walking randomly from one of the members of a group and check what is the probability that we visit a certain unknown node.

This method is an order of magnitude faster than method 1 and 2 because we only need to calculate the stationary distributions 4 time (pageRank and for each group). However we suspect it will perform with less accuracy. Because they take into account nodes that are far and not well connected to the unknown node whose function we try to predict. The function groups, in particular the yellow (meiosis) and green (stress) are not well clustered and spread all over the network.

6.b Somew code snippets

```

1  # The decision function which is used in methods 1 and 2 to predict a
   single
2  # node
3  def decision_function(v, G, group_membership_ar, known_unknown_ar):
4      """Input v: an node assumed to be an int and taken from the list
       of unkown
5      nodes. Input G: The graph.
6      Input group_membership_ar: Array of strings which specifies for
       each node
7      to which group belongs (including the 'unkonw' nodes) this is
       required for
8      thesting the correctness of the decision.
9      Input known_unknown_ar: boolean 1d array which
10     specifies for each node of the graph whether its membership is
       known or
11     unkown.
12     output guess_group: string the group that the function assigns the
       node to.
13     output correctness: True or false depending on the correctness of
       the
14     guess_group vs the real group_membership_list[v] value.
15     """
16     all_nodes = np.arange(len(G.nodes))
17     known_nodes = all_nodes[known_unknown_ar]
18     unkown_nodes = all_nodes[known_unknown_ar == False]
19     bias = np.zeros_like(G.nodes)
20     bias[v] = 1
21     bias
22     p = biasedPropagateGv2(G, bias=bias)
23     group_names = np.unique(group_membership_ar)

```

```

24     q = p * known_unknown_ar #0 on all unkown nodes
25     testscores = np.zeros_like(group_names)
26     for g in range(len(group_names)):
27         x = q[group_membership_ar == group_names[g]]
28         testscores[g] = x.sum()
29     decide = group_names[np.argmax(testscores)]
30     correctness = decide == group_membership_ar[v]
31     return decide, correctness
32
33 # Main part of method 2:
34 # Which is actually similar in methods 1-4
35 known_unknown2 = known_unknown.copy()
36 node_colors2 = node_colors.copy()
37 score2 = 0
38 for v in orderedUnkownNodeList:
39     _, test = decision_function(v, G, groups, known_unknown2)
40     score2 += test
41     known_unknown2[v] = True #mark v as 'known'
42     G2.nodes[v]['correctness2'] = "Correct"
43     node_colors2[v] = 'brown'
44     if not test: # mark as incorrect and colormark if mistake
45         node_colors2[v] = 'pink'
46         G2.nodes[v]['correctness2'] = "Mistake"

```

6.c Results

Table 1: Prediction Accuracy of the 4 Methods in 2 different random trials. Each trial with a different seed, which result in different randomly selected known/unknown nodes.

Seed / Method	1	2	3	4	5
42	0.795	0.858	0.765	0.784	0.701
6382020	0.774	0.832	0.752	0.748	0.755

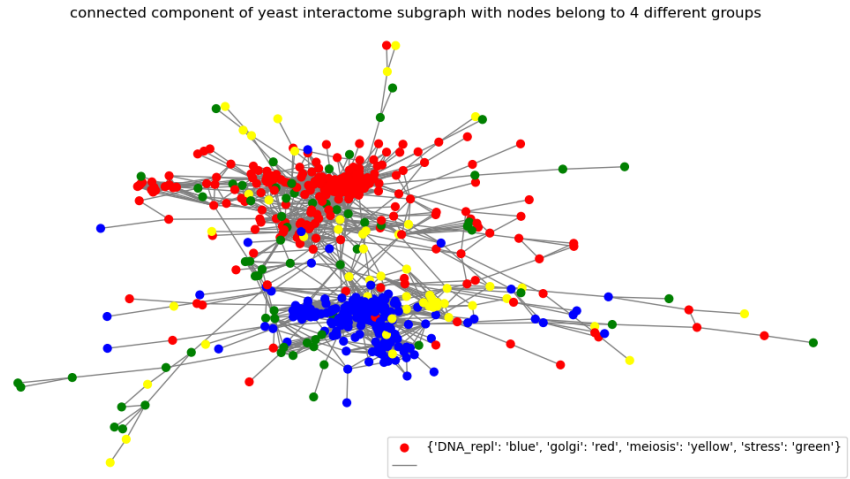
6.d Discussion

If we look at figure 11a which shows the connected component that we worked with, the red and blue are pretty nicely clustered. The greens and specially the yellows are sort of spread all over the network and not nicely clustered. I don't know if this is a good example of a real world scenario.

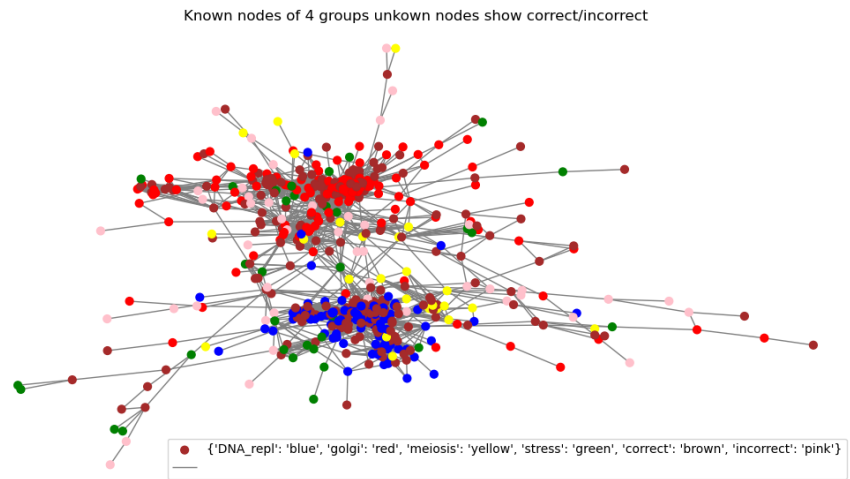
When we look at figure 11b it seems that most of the incorrect predictions are on nodes that seem 'peripheral' and are not well connected to known nodes. Another source of mistakes seems to be green nodes that are too close to the red cluster.

Method 2 shows somewhat higher accuracy over the other methods but on the flip side it is an $O(n^2)$ method when optimized methods are used whereas methods 3,4,5 are $O(n)$.

It's probably worth while to repeat the experiment a couple of hundred of times to estimate the mean accuracies of the methods.



(a) The largest connected component which was selected for the function prediction experiment. Here the Correct group affiliation of all nodes is color coded.



(b) The result of prediction method 2, which was the best performer. The known groups are color coded by RGBY. The unknown nodes are color coded: brown:correct prediction, pink:incorrect.

7 The Propagation Method In Bioinformatics

Propagation methods or as they are sometimes called, diffusion methods, have been used in Bioinformatic software tools mostly function prediction and 'disease characterization' [1].

Typically the diffusion method is used to screen for possible 'candidates' (i.e a disease causing genes) which is basically pageRanking, or for module detection which is essentially clustering/-community structure. In a second step these candidates are scored by a probabilistic model which may use additional information beyond the network structure itself.

The most simple application, which is briefly described in the introduction, is function prediction in a single PPI network. As an example application we bring the RIDDLE [13] functional prediction web application (www.functionalnet.org/RIDDLE).

(usw und sofort ...they use pagrRank, and then do a statistical test based on hypergeometric distribution to score the significance ...)

Hotnet [10] and its successor Hotnet2 [4] are an example of a more complex application used for module detection ...

(they have a system to set threshold influence, break the graph to small connected components as module candidates, then use additional information to score the significance)

8 Reference

References

- [1] Lenore Cowen et al. "Network propagation: a universal amplifier of genetic associations". In: *Nature Reviews Genetics* 18.9 (2017), p. 551.
- [2] *Graph Partition*. Wikipedia. 2020. URL: https://en.wikipedia.org/wiki/Graph_partition (visited on 09/11/2020).
- [3] *Laplacian Matrix*. Wikipedia. 2020. URL: https://en.wikipedia.org/wiki/Laplacian_matrix (visited on 09/12/2020).
- [4] Mark DM Leiserson et al. "Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes". In: *Nature genetics* 47.2 (2015), pp. 106–114.
- [5] Carl D Meyer. *Matrix analysis and applied linear algebra*. Vol. 71. Siam, 2000.
- [6] Maxim Naumov and Timothy Moon. *Parallel spectral graph partitioning*. Tech. rep. NVIDIA Technical Report, NVR-2016-001, 2016.
- [7] Mark EJ Newman. "Modularity and community structure in networks". In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.
- [8] D Serre. "Matrices theory and applications second edition". In: *Graduate Texts in Mathematics* 1.216 (2010), ALL–ALL.
- [9] *Spectral Clustering*. Wikipedia. 2020. URL: https://en.wikipedia.org/wiki/Spectral_clustering (visited on 09/11/2020).
- [10] Fabio Vandin et al. "Discovery of mutated subnetworks associated with clinical data in cancer". In: *Biocomputing 2012*. World Scientific, 2012, pp. 55–66.
- [11] Martin Vingron. "Propagation methods (Lecture Slides)".

- [12] Ulrike Von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4 (2007), pp. 395–416.
- [13] Peggy I Wang et al. “RIDDLE: reflective diffusion and local extension reveal functional associations for unannotated gene sets via proximity in a gene network”. In: *Genome biology* 13.12 (2012), R125.