Freie Universität Berlin and Max Planck Institute's MPIMG

Department of Bioinformatics

Partially Labeled Classification in Graphs

Using Network Propagation Methods and Applications in Protein Function Annotaions

Yiftach Kolb

yiftach.kolb@fu-berlin.de

a Bachelor Thesis

Supervisor:

Prof. Dr. Martin Vingron, vingron@molgen.mpg.de

November 21, 2020

Abstract

We use a RWR propagation to propose a prediction algorithm for a partially labeled protein-protein-interaction network. The methods seems to improve the prediction accuracy when compared to the neighbor counting method. Further experiment indicate that the strength or weakness of the prediction method relies on how the closely the labeling adheres to the inherent community structure of the network.

Contents

1 Introduction **2**

2 Linear algebra primer **4**

3 More on matrices, graphs and stochastics **8**

4 A simple propagation based heuristic for graph clustering **16**

5 A function prediction method **21**

6 Discussion **28**

A Appendix: Linear algebra primer **30**

B Appendix: More on matrices, graphs and stochastics **38**

C Appendix: Spectral clustering and other methods **44**

D Reference **48**

1 Introduction

In this paper we propose a protein function prediction algorithm, which in essence works similarly to the adage ”Tell me who your friend are, and I will tell you who you are”. Imagine that we can accurately predict the function of a protein in the cells of an organism bases on its biochemical interactions with the other proteins, and the interactions of the other proteins between themselves. This has in fact demonstrated to be true by Schwikowski, Uetz, and Fields [13]. But we are trying to improve on it. In the formulation of the adage above, We want to tell you who you are based on information on some friends of your friends. We want that because in the world of protein research, there are still many that are unresearched or mostly unresearched.

The specific problem we are dealing here, from the perspective of bioinformaticians, is of protein function prediction in a protein-protein interaction network (PPIN), yet the actual algorithm is more general and applies to any network really. For this reason why we call the problem ’Partially labeled classification’ (PLC) as it was named by Szummer and Jaakkola [16].

Our proposed algorithm uses the network propagation method of random walk with restart (RWR). These methods has been used most famously by Google for their search algorithm but also in bioinformatics for example for ’disease module identification’ by [17, 1].

We dedicate sections 2 and 3 of this Thesis to explain the underlying mathematical theories of stochastic matrices and random walk with restart in graphs, largely going according to textbooks from Meyer [10] and Herstein and Winter [6]. The purpose is to make this work somewhat self standing with the mathematical foundations it is based on, to give a ’feel’ for how RWR actually work, as well as presenting some lemmas that prove why we can use it in the networks we are dealing with. These two sections are exapnded by appendices 1 and 2.

In the following section 4 We deal with the subjects of community structure and spectral clustering in graphs. There is a great deal of overlap between the PLC problem and community structure. Essentially the function prediction algorithm is based on the paradigm that proteins of the same functional classification do tend to form ’communities’ within the PPI. We therefore first propose and test a heutistic for graph clustering, because if it doesn’t work there is no reason that our approach has any hope resolving the PLC. We also provide in appendix 3 some details about spectral clustering and compare it with the heuristic.

In section 6 we propose an algorithm for protein function prediction (or more generally for the partially labeled classification problem). We try and compare several clustering methods and then carry forward more test with the method of choice.

Finally in section 7 we draw some conclusions, , discuss other methods and approaches, and consider possible ways to carry with this research forward.

2 Linear algebra primer

Prologue

This section is mostly based on [10] and [6]. We are going to have to use allot of definitions anyway and cite big theorems like the Perron-Froebenius theorems, so we might as well provide an understanding of the important properties of stochastic matrices which we need for the RWR methods.

This chapter is accompanied by an appendix chapter, which containes more material and presents proofs or sketch of a proof to most theorems and lemmas. We try to keep the chapter intself short and just mention theorems and properties that are strictly necessary for the following chapters.

This chapter together with the appendix is also meant to intercept likely questions, quite possibly even by by the author himself within a few decades from now, such as, 'what if we consider complex vectors, maybe there is a complex eigenvalue who is greater than 1 ?', no there isn't, read here why.

Theory

Definition 2.1. A **Transition Matrix** (we call it a **Transition** in short), which is sometimes also called a **stochastic matrix**, is a real valued non negative square $(n \times n)$ matrix A such that 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: $A : v \to A \cdot v$. In this paper we use left multiplication convention (Av). There are many other publications that deal with random walk and Markov processes, where right multiplication is used $(v \cdot A)$, and accordingly the rows are normalized rather than the columns.

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

A transition is **primitive** if for some $k >$ 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 $A_{i,j}^k > 0$.

It can be shown that A matrix A is irreducible by 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 $v \in R^n$ 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:

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

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 $A \in \mathbb{R}^{n \times n}$ We let $|A| \in \mathbb{R}^{n \times n}$ be the resulting matrix from applying $|\cdot|$ element wise. Given a vector $v \in \mathbb{C}^n$ we let $|v| \in \mathbb{R}^n$ 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. In the appendix ☐

Lemma 2.3. *If a transition $A > 0$ (or primitive) , then it has exactly one eigenvector v with eigenvalue 1 and in addition v can be chosen to be strictly positive. Furthermore, 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. In the appendix ☐

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. In the appendix ☐

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 algebaic 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. In the appendix □

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, which is stated with a sketch of proof in the appendix.

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 2.3 (Perron-Frobenius [10])**.** *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 2.4 (Perron-Frobenius for irreducible matrices [10])**.** *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

Prologue

Here we are starting transition from matrix terminology into that of graphs and random walk. As in the previous section, there is an included appendix for this chapter. We show how to derive the properties of RWR from the stochastic matrix theory and provide some examples for propagation in graphs and setting the scene for the next section where we deal with clustering. One reason why we provided the relative expanded mathematical details on the previous section and in this one is to explain why the restart is essential to turning a connected graph where a random walk generally doesn't reach a steady state, into an acyclic graph (corresponding to a primitive adjacency matrix).

Theory

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 j to i (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 form 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 4. If G is strongly connected then the periods of all vertices are indeed equal and its easy to prove. The corresponding adjacency matrix A is irreducible so it too has a period h as defined in 10 and it is equal to the graph period (can be shown using A.4 and 10).

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 2.3 that there is a unique stationary distribution p and that the Markov process 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.

If the graph G is strongly connected, then 2.4 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 (see theorem A.4).

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.

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 starting distribution p_0 and has the form $p_{k+1} = Ap_k = \cdots = A^k p_0$, where p_k is the distribution after k steps. 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.

When we talk about **random walk with restart (RWR)** we set a restart state q and a restart parameter $\alpha \in [0, 1]$. At each step, we either restart over to the state q, with probability of $alpha$, or continue to walk using the normalized adjacency matrix A. The state sequence is therefore

$$p_{k+1} = \alpha q + (1 - \alpha)A \cdot p_k \qquad (1)$$

It turns out that this random walk with restart is actually a normal random walk but with an modified adjacency matrix (and respectively, an modified weighted directional graph).

Lemma 3.1. *Let q be any state, let $\alpha \in [0,1]$ and let $Q = (q|q|\dots|q)$ (The square matrix whose every column equals q). Then Q is a transition and for any state p we have $Qp = Q$.*

In addition if T is any transition then $W = \alpha Q + (1-\alpha)T$ is also a transition, And we can rewrite the RWR from 1 as

$$p_{k+1} = \alpha q + (1-\alpha)Ap_k = [\alpha Q + (1-\alpha)A]p_k = Wp_k$$

Proof. Trivial and uses B.2 □

The matrix W represents a graph G' where each edge of the original graph G is rescaled by a factor $1-\alpha$ (and if G is undirected we treat each undirected edge as 2 directed edges in G'. In addition from each vertex v we add edges to every other vertex and the weight of the additional edge is $\alpha q[u]$.

If we pick the restart state q in a way that makes W primitive, then 2.3 assures us that the RWR will converge, $\lim_{k\to\infty} p_k = \lim_{k\to\infty} W^k p_0 = p$, where p is the unique stationary distribution of W. This means in particular that we can use the power method to find out the distribution p by sequentially calculating $p_1, p_2, \dots$ until it sufficiently converges, and it will converge to p from any initial state p_0 which we choose.

Also we can take the limit $p = \lim_{k\to\infty} p_k$ and rewrite 1 as:

$$\begin{aligned} p = Wp = \alpha q + (1-\alpha)A \\ [I - (1-\alpha)A]p = \alpha q \end{aligned} \tag{2}$$

We will see later that we can invert the matrix in the second equation of 2 and use a direct solution for p instead of the power method. The power method has the advantage that we can use the matrix A implictly, because we only need to know $A \cdot p_k$ to compute p_{k+1}. A is usually a sparse matrix because each vertex usually only has few neighbors and so we can use matrix free methods efficiently to calculate p but that is beyong the scope of this thesis.

In the particular case of pageRank, we choose a uniform restart state $q = \frac{\mathbf{1}}{\mathbf{n}}$. The corresponding matrix $Q = (q|\dots|q) > 0$ is strictly positive, and therefore $W = \alpha Q + (1-\alpha)A > 0$ is positive and therefore primitive.

The stationary distribution which corresponds to this uniform restart state q is called the **PageRnak** for G with restart parameter α. It is used to order the vertices according to their 'relevance' in the network.

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

Proof. In the appendix □

Remark 5. Lemma 3.2 proves that we can propagate (namely do RWR) from any arbitrary restart state q, including a single vertex and the combined transition matrix will be primitive if the adjacency matrix A is irreducible, or equivalently, the corresponding graph G is strongly connected.

Assume that A is a normalized adjacency transition of a strongly connected graph G. Let q be any state column vector, for example e_1 if we restart from a single vertex, and let $Q = (q|q|\dots|q)$. So Q has a positive row and therefore $(1-\alpha)A + \alpha Q$ is a primitive transition according to 3.2.

Definition 3.2. Let G be a graph with adjacency matrix A, and let D be the diagonal matrix of the out ranks of the vertices of G. Then we can column normalize A and create the transition $T = AD^{-1}$. We define **the transition matrix with restart parameter α and bias q** as

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

In general, the matrix $T_{\alpha,q}$ may have more than one unique stationary distribution p (there is one for each strongly connected component of its corresponding graph). But if we require that

G be strongly connected (which means A is irreducible), then $T_{\alpha,q}$ is primitive by 3.2, and the stationary distribution p is unique.

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

We can rearrange it now

$$(I - (1-\alpha)T)p = \alpha q \tag{3}$$

We want to invert the matrix in 3 and use the following lemma to justify it (The proof is easy. See for example Serre [14]):

Lemma 3.3. *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 3.3 on equation 3 because $(1-\alpha)A$ is contracting, and we have:

$$p = \alpha[I - (1-\alpha)T]^{-1}q := Kq = \alpha\sum_{k=0}^{\infty}(1-\alpha)^k T^k \tag{4}$$

K is called **diffusion matrix** [9] of T (or of the graph G) 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 4 and since T is irreducible.

What about the eigenvectors and eigenvalues of K? If a matrix A is invertible then $Av = \lambda v \iff A^{-1}v = \lambda^{-1}$. 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 if all the eigenvalues are real (spoiler- they are), then they have the same linear 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.*

The diffusion matrix is the key for the next sections. What we saw here is that we can take any strongly connected graph G, set a restart value α and we generate the diffusion matrix K which is itself a stochastic matrix and it shows us how any restart distribution will propagate itself in the in the graph in a RWR process. In particular, the column $K[:, i] = K \cdot e_i$ gives us the stationary distribution which we will get if we do a RWR from node i. We interpret this as an indication that vertices which rank higher K_i are closer or more important to vertex i.

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. There is some confusion whether the diffsusion matrix K can be called diffsusion kernel ,which is usually used in the context of contituous processes. I use the term diffusion matrix as it was defined in Leiserson et al. [9] and in that paper's suppliamentary material the matrix is also referred to as kernel.

Examples

Figure 1 demonstrates a weakly connected directed graph. If we start walking from 0, 5 or 6 we would 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 though it.

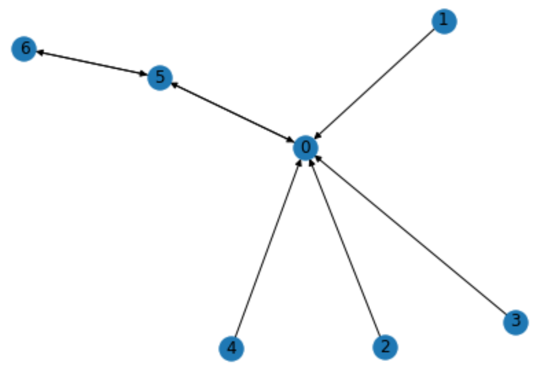


Figure 1: a weakly connected graph.

In figure 2 we have 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.

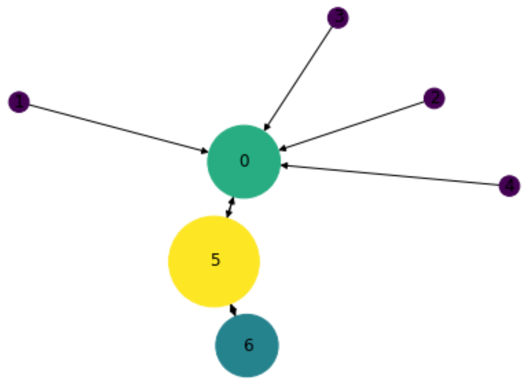


Figure 2: Pageranked graph.

If we pick a restart parameter that is too large, which is the case shown in figure 3, the ranks becomes almost uniform because the convex combination of the original graph with the n-clique graph of the restarts is weighted too heavily towards 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.

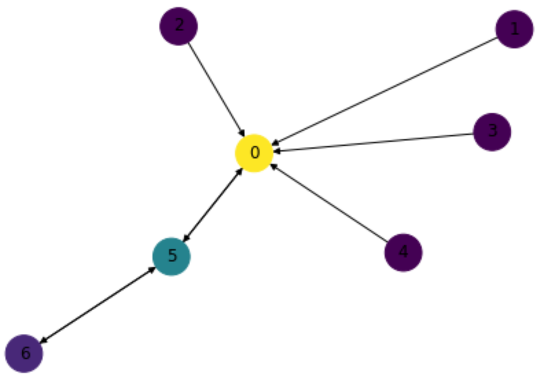


Figure 3: again the same graph, with node color and size indicating their PageRank, with a restart probability approaching 1.

When we treat the same graph as undirected (figure 4 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.

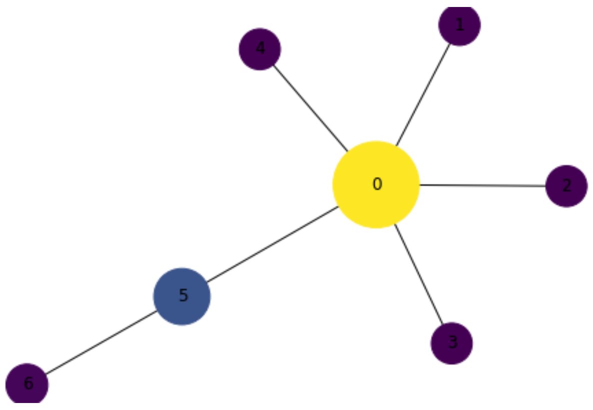


Figure 4: same graph, but undirected.

In undirected graph, starts have hot centers and cold periphery. a star has a hot center because it spreads its heat on many sources, while each of the orbiting vertices sends all its heat into the star center. The result is that hear accumulates at the center.

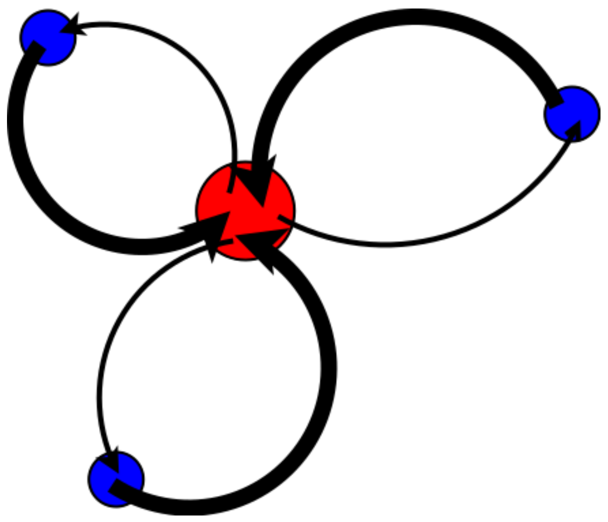


Figure 5: a star graph.

4 A simple propagation based heuristic for graph clustering

Prologue

In this section we come up with a very simple simple clustering algorithm which uses RWR to divide the well known Zachary's Karate Club graph. The results can be compared to the actual division of the subjects of the research into 2 clubs.

Theory

Definition 4.1. Let G be a strongly connected graph and K the diffusion matrix as defined in 3.3, 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 $p_u[v] = K[v, u]$ is the **influence of** u on v. In terms of random walk, $K[v, u]$ is the frequency that we visit v, if we do a RWR from u.

Remark 6. We use the definition of influence as presented in the Hotnet and Hotnet2 artticles [17] and [9].

Remember that we use in this paper $A \cdot v$ scheme while because that is the 'normal' way in linear algebra textbooks. However often in papers dealing with Markovian processes they use the other way. To add to the confusion, they sometime call these conventions left (repectively right) multiplication but whose left and whose right and why?

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 $v = \text{argmax}(p_u)$ 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 $p = K \cdot (1/n, \ldots, 1/n)^t$.

We 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:

```
function coolWarmClustering(G, k)
  # Input G = (V,E) : a directed strongly connected graph.
  # Input m : The desired number of clusters
  K <- diffusion_kernel(G)
  # each vertex starts in its own cluster:
  H <- Disconnected_undirected_graph_on(V)
  p <- pageRank(G)
  # vertex-list sorted up by PageRank:
   vlist <- arg_sort(p)
   While |connected_components(H)| > k do
    # take the coldest remaining vertex
    # and remove it from the list:
    x <- vlist.pop()
    # Influence vector of x:
    p_x <-K * e_x
    # find the max excluding the already visited nodes
    y <- argmax(p_x[i : i in vlist])
    H.add_edge(x,y)
  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, among the yet unvisited vertices.

In every iteration of the algorithm adds an edge from the currently selected vertex to one of the remaining vertices which haven't been selected yet.The number of connected components by 1. Because G is strongly connected the algorithm will reduce the number of connected components in H until it reaches the desired number of components k.

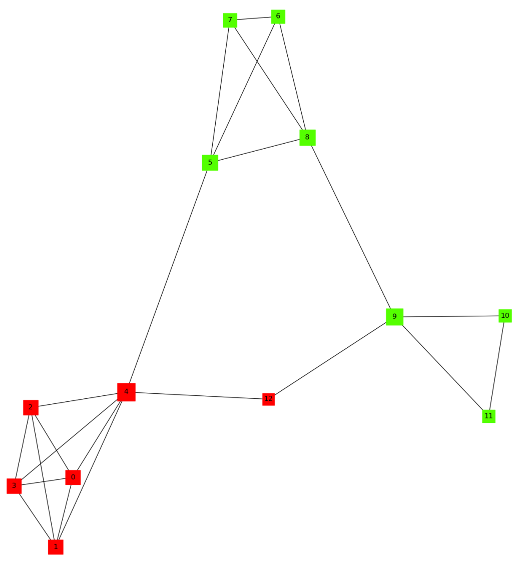
It is essential to run the algorith in the order from lower pageranked vertices upwards. If we start for example the other way around, from the warmest node downwards, the list of remaining nodes contains the coldest and least 'relevant' nodes.

Testing of the algorithm

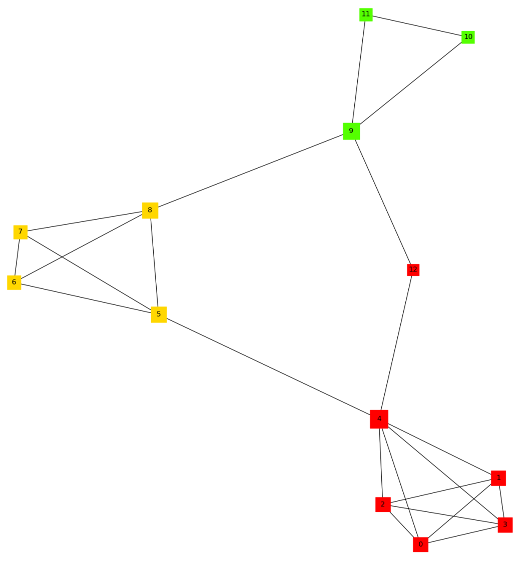
We tested the algorithm on a small 'toy graph', as well as on the famous 'Zachary's Karate Club' graph. In the appendix C, which deals with spectral clustering, we a tested a basic spectral clustering method on the same graphs.

Figures 6 are showing the results of running the algorithm set for finding a 2-partition and respectively a 3-partition on a 'toy graph'. 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

Figure 7 shows the results of running the algorithm on 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. Yellow indicates true positive for 'Mr Hi', green is a true positive for 'Officer', and cyan is a mismatch between the clustering algorithm and the actual division. There is only one 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).



(a) A 2 cluster partition



(b) A 3 cluster partition

Figure 6: 'toy graph' used for testing the algorithm

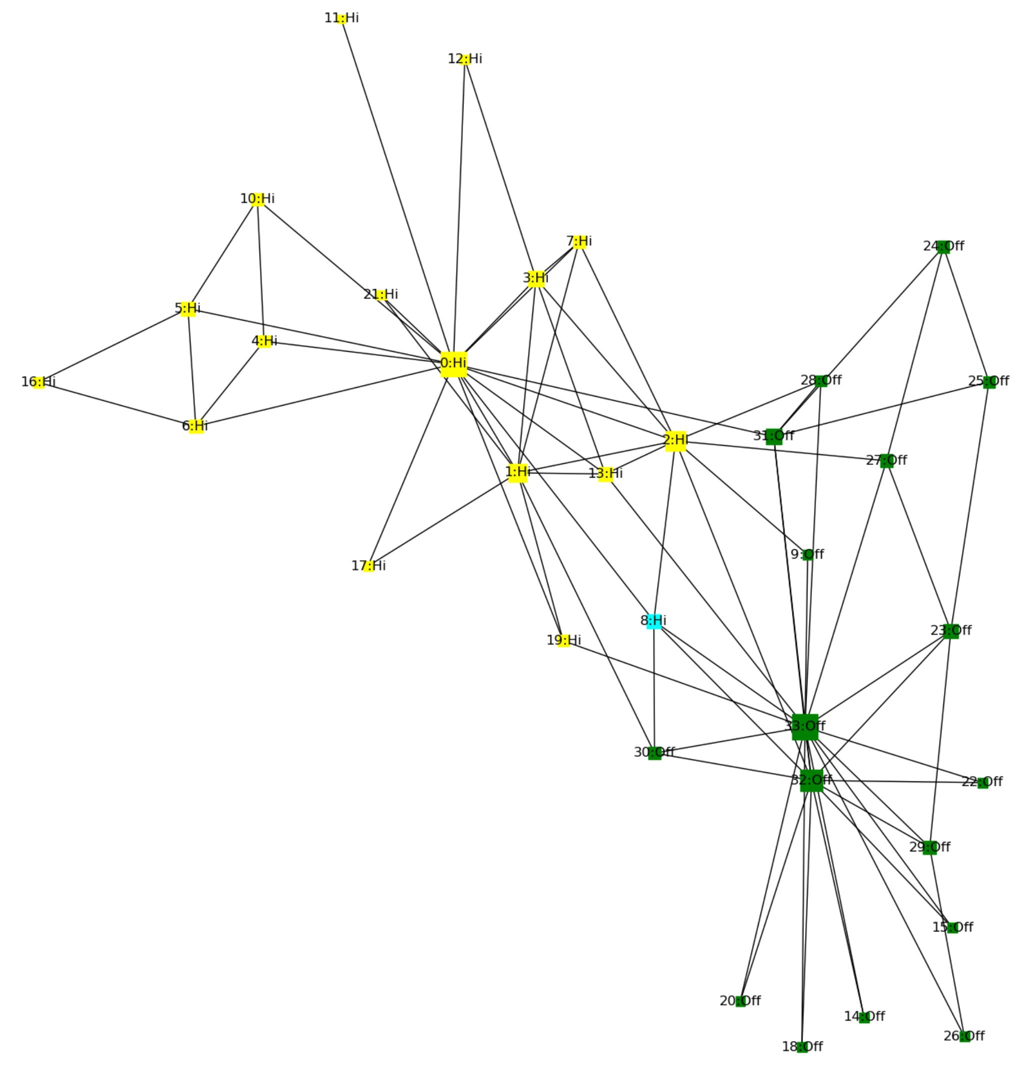


Figure 7: 'Zachary's Karate Club'. 'Mr Hi' (Yellow) and 'Officer' (Green) groups. The Cyan colored node indicates a mismatch between the clustering and the actual division.

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, vertex 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'.

5 A function prediction method

Prologue

As we mentioned in the introduction our main goal was to develop a a classification algorithm for the PLC probelm which can be used for the purpose of protein function prediction in a partially annotated PPI network.

What it means is we are going to take an input a strongly connected graph where the vertices represent protein types and the edges represent observed interaction between them. The some of the proteins are annotated with a label that specifies their function. The rest are unannotated.

The prediction algorithm is going to output for each unlabeled protein a proposed label. Our paradigm is that protein is that proteins of the same label should form, approximately, a community in the sennse of graph community structure which we discussed in the earlier sections. Since the algorithm is using propagation methods which are closely related to the community structure of the graph, if the paradigm doesn't hold then the prediction sohould be no better than a coin toss.

It is understood that proteins that interact are more likely than not to share the same function [13]. It's a weaker property than our paradigm because it doesn't speak about communities, only about adjacent proteins.

The diffusion matrix provides us with ranking such that for for every vertex i we can weigh the other vertices by 'importance' to that vertice (namely it's the i row of the diffusion matrix). We use that information to predict the label of the protein. We assume that members of the same label of the protein will have cummulatively the most 'influence' on it. We also assume that the distribution of labeled and unlabeled is the same across the network so by removing them from consideration, still the proteins that share the same label with the protein we test will have the most cummulative influence on it.

We tested and compare 5 different methods, 3 were diffusion based and 2 were neighbor based whih we used mainly as the reference point.

The source code for all the algorithms and the test can be found at the project's git repository [7].

The Algorithm

We tried 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 meas: 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 (see pseudocode for the decision function below) 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 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 was that the RWR as we have seen has the property that heat flows from the low ranks to the high ranks. So maybe we should pick a low ranked vertex and see where it flows to. If we start from a hot central vertex.

- 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 probably the simplest. This method is essentially the 'prediction function' from [13].

- 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.

  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 times (pageRank and for each group). However we suspect it will perform with less accuracy. The reason for that is that method 5 takes 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 but we can see that some subsets of these labels form clusters.

Listing 1: method 2, decision funciton

```
# vertices are enumerated 1 to n. labels are enumerated 1 to m.
# input v: vertex of unknown label, to be predicted
# input F: the diffusion matrix (so F is column-normalized)
# input labeledSubset: list of boolean vectors. so
#     if labeledSubset[l] = (1, 0, ...),
#     then vertex 1 has label l and vertex 2 doesn't.

def decisionFunction(v, F, labeledSubset):
  # get p, the influence vector of v, which is column v of F:
  let $p = F \cdot e_v$
  let labeledSums = [sum(p[x]) for x in labeledSubset]
  return argmax(labeledSums)
```

Listing 2: method 2, main funciton

```
# vertices are enumerated 1 to n. labels are enumerated 1 to m.
# Input G: a strongly connected graph
# input labeledSubset: list of boolean vectors. so
#     if labeledSubset[l] = (1, 0, ...),
#     then vertex 1 has label l and vertex 2 doesn't.

def predictionMethod2(G, labeledSubset):
  # calculate the diffusion matrix
  let F = diffusionMatrix(G)
  # calculate the pageRank order
```

```
let pr = $F \cdot \bar{1}/n$
let unknowOrderedList = [v : v is unlabeled].sort_up_by(pr)
for v in unknowOrderedList:
  predLabel <- decisionFunction(v,F,labeledSubset)
  # set v's labels to predLabel
  # it will participate in the subsequent predictions
  labeledSubset[predLabel][v] <- 1
return labeledSubset
```

Experimental Protocol

We first constructed a **fully** labeled subnetwork of the yeast interactome by choosing arbitrarily 4 labels, the only constrain was that each labeled subset be sufficiently large (approximately 100 or more nodes). We then removed from the network all nodes that didn't have one of those 4 labels: 'DNA replication' (blue), 'Golgi' (red), 'Meiosis' (yellow), and 'Stress response' (green). We also removed nodes that had more than one of these 4 labels (of which there were only a handful). The resulting subnetwork is shown in figure 8. We then selected the greatest connected component of that network, and this was our experimental graph. It contains around 500 vertices.

In the first set of tests, we tested the accuracy of each of the 5 methods. We randomly selected 50% of the nodes and assigned them to the known group, that is, they were the labeled subset. The rest were the unlabeled. We ran each of the 5 methods on this partially labeled networked and validated its accuracy by comparing its prediction with the fully labeled network. We repeated these tests twice, on 2 different randomly selected labeled subsets. The results are in table 1

We then chose method 2 which had the highest accuracy for further tests. We tested it with different parameters, repeating each settings 50 times with 50 different sets of known labels. The parameters we tested were the restart probability (α) and what we called the **labeled coverage**, which is the portion of the nodes that start with a known label.

We have conducted another test of method 2, with parameters $\alpha = 0.31$, and label coverage of 0.35. This time we calculated the sensitivity, specificity, accuracy and precision (PPV) for each label as well as the total.

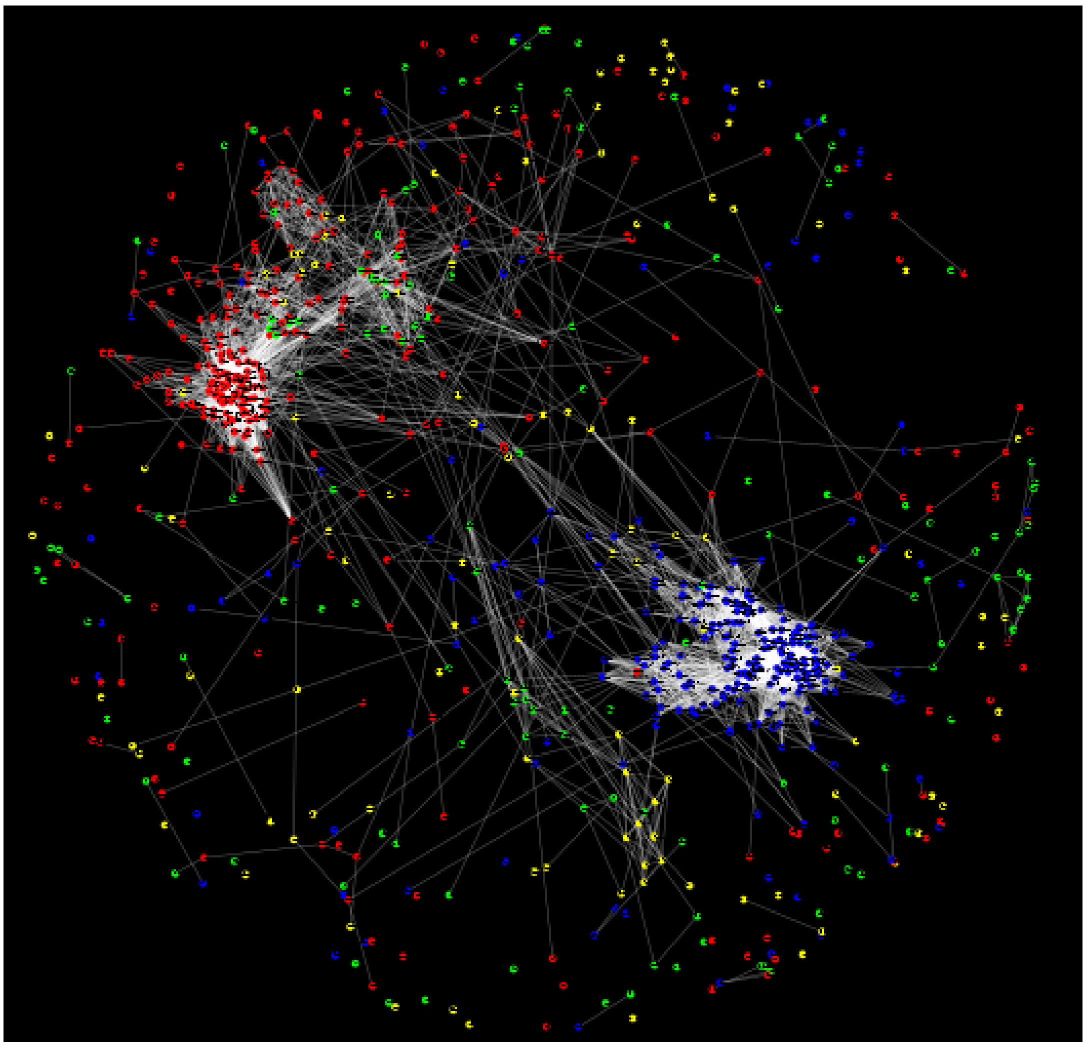


Figure 8: Subgraph of the yeast interactome showing nodes of 4 groups: 'DNA replication' (blue), 'Golgi' (red), 'Meiosis' (yellow), 'Stress response' (green)

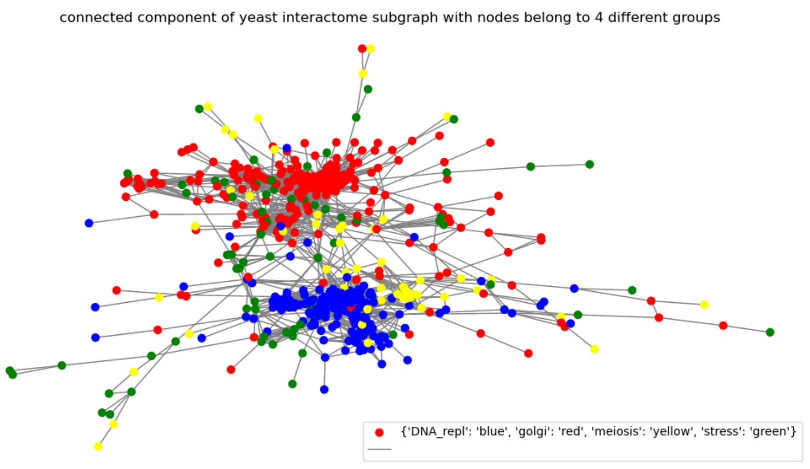
Results

Table 1 shows the results of the experimentally measured accuracy for each of the 5 method on 2 50% labeled networks (the networks are the same, the labeled vertices were randomly selected).

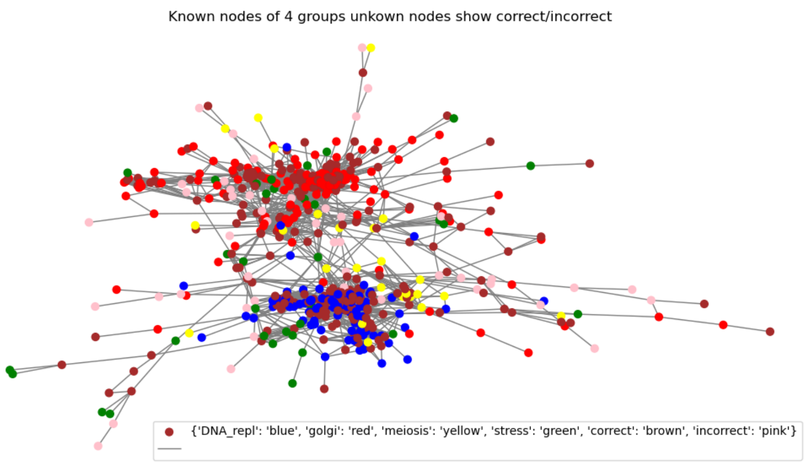
Table 1: Prediction Accuracy of the 4 Methods in 2 different random trials. Each trial with a different seed, which result in different randomly selected known/unknown nodes.

Seed / Method	1	2	3	4	5
42	0.795	0.854	0.765	0.791	0.701
6382020	0.774	0.832	0.751	0.751	0.755

We picked method 2 which was the best performer for further tests. Figure 9 shows the network and the hits and misses of the prediction algorithm method 2.


connected component of yeast interactome subgraph with nodes belong to 4 different groups
{'DNA_repl': 'blue', 'golgi': 'red', 'meiosis': 'yellow', 'stress': 'green'}

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


Known nodes of 4 groups unkown nodes show correct/incorrect
{'DNA_repl': 'blue', 'golgi': 'red', 'meiosis': 'yellow', 'stress': 'green', 'correct': 'brown', 'incorrect': 'pink'}

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

Figure 9: The test network

Upon further testing we conducted with method 2, the best mean result (for coverage of 0.5), was 0.8926 mean accuracy. It was obtained with a restart probability of 0.31. Restart parameters $\alpha \in [0.2 - 0.3]5$, lets call it mid-lower range, produce the best results, while with higher alphas accuracy falls towards 0.5 which is of course the accuracy of just randomly guessing.

Accuracy was about 0.8 when we start with only 10% labeled and it gradually increases with increased labeled proportions (0.85 accuracy at 0.3 label coverage). We plotted the relationship between accuracy and the parameters in figure 10.

We have tested method 2 for accuracy, precision, specificity and sensitivity, with parameters $\alpha = 0.31$, and label coverage of 0.35. The values were calculated for each label as well as the total. The results are the following:

	Label	P	TP	FP	N	TN	FN	Sens	Spec	Acc	PPV
0	golgi	165	149	25	209	184	16	0.90	0.88	0.89	0.86
1	DNA_repl	133	129	22	241	219	4	0.97	0.91	0.93	0.85
2	meiosis	30	17	4	344	340	13	0.57	0.99	0.95	0.81
3	stress	46	25	3	328	325	21	0.54	0.99	0.94	0.89
4	Total	374.00	320.00	54.00	1,122.00	1,068.00	54.00	0.86	0.95	0.93	0.86

We can see that the problem lies with the two small and spread out groups, 'meiosis' and 'stress'. It is hard to predict that an unlabeled vertex belong to these groups

Remarks and Conclusions

If we look at figure 9 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. We don't know if this is a good example of a real world scenario in the context of PPI networks.

When we look at figure 9b 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 bee 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 $O(n)$.

We suspect (but it requires testing) that the accuracy of methods 3 and 4 would decrease more rapidly with decreasing labeled coverage than it did with method 2. The reason is that methods 3 and 4 become completely arbitrary on a vertex which has no labeled neighbors. Method 1,2 rank all the vertices on a graph so they may have a better chance of finding the right label.

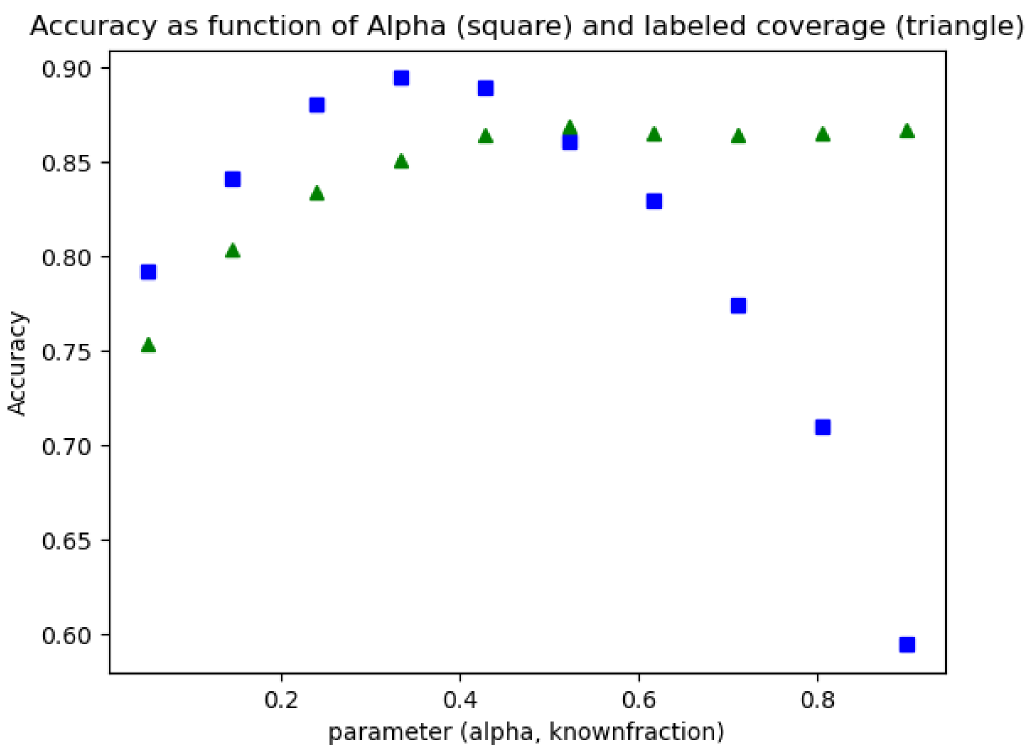


Figure 10: The accuracy as function of the resart parameter alpha, with fixed label coverage of 0.5, and of the label coverage with fixed alpha=0.2

6 Discussion

RWR propagation methods have been successfully used in bioinformatics mainly as a component in algorithms for 'disease module prediction', for example by [9, 1]. More generally diffusion based algorithms have been used in multiple applications [3].

The prediction method which was used by Schwikowski, Uetz, and Fields [13] is essentially method 3 but their test setup allowed multiple labels per proteins. Their experimental accuracy was 72%.

We adopted the PLC name and definition from Szummer and Jaakkola [16]. The proposed solution for the PLC on that paper was to maximize the posterior probability of the label for each unlabeled vertex. Essentially find label y which is the likeliest to be the group where the walk had began, considering it ended at vertex k after t steps. Their solution uses a fixed time scale but it doesn't have to. We think that method 5 is possibly the solution if we use RWR instead of time scale. That is because method 5 chooses for vertex the label from the group that if we start from it, it would be visited more frequently than if we started from any other labeled group. The fact that we deal with stationary distribution saves allot of headache inducing calculations but of course we need to justify why do we allow restart.

Another paper which deals with protein function prediction is Deng et al. [4]. Like in [13] they allowed multiple labels per protein. They measured sensitivities and specificities which were 'rougly the same' and depending on the setting were 45% to 65%. But we don't think those results are comparably with the experiment that we presented here.

An idea of improving method2 would be to assess the likelihood of the correctness of the prediction for a vertex on the fly, and only include the preicted labeling in the prediction of the follow up prediction if the likelihood is high enough. In this case we might alsop try to run it many times in different random orders rather than in a particular order.

Assessment of the likelihood of the prediction is very important in the cases where we don't want or need to predict the labels of all the unknown proteins, but rather need to select specific unlabeled proteins which are very likely to have a specific label we are interested in. Essentially this is what disease module prediction is about.

A Appendix: Linear algebra primer

Definition A.1. A **Transition Matrix** (we call it a **Transition** in short), which is sometimes also called a **stochastic matrix**, is a real valued non negative square ($n \times n$) matrix A such that 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: $A : v \to A \cdot v$. In this paper we use left multiplication convention (Av). There are many other publications that deal with random walk and Markov processes, where right multiplication is used ($v \cdot A$), and accordingly the rows are normalized rather than the columns.

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

A transition is **primitive** if for some $k >$ 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 $A^k_{i,j} > 0$.

It can be shown that A matrix A is irreducible by 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 A.2. A **State** is a non-negative vector $v \in R^n$ s.t $\sum_i v_i = 1$.

Remark 7. If v is a state and A is a transition as defined in A.1, then Av is also a state because:

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

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 A.3. Given $A \in \mathbb{R}^{n \times n}$ We let $|A| \in \mathbb{R}^{n \times n}$ be the resulting matrix from applying $|\cdot|$ element wise. Given a vector $v \in \mathbb{C}^n$ we let $|v| \in \mathbb{R}^n$ 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 A.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 8. 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 A.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 $(1, 1, \ldots, 1)$ 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 $u \in R^n$ 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 A.3. *If a transition $A > 0$ (or primitive) , then it has exactly one eigenvector v with eigenvalue 1 and in addition v can be chosen to be strictly positive. Furthermore, 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 $w^+ = 0$. ◻

◻

Remark 9. 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 A.3 guaranties it has a spectral value of $\rho(A) = 1$.

Theorem A.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 $J^k \to (e_1|0\dots|0)$. 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 $T^k \to (v|v\dots|v) = V$. Then one can verify $Vu = v$ for any state u. □

Theorem A.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 algebaic 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 $|J_{i,i}| \le 1$.

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. If follows that all eigenvalues on the unit circle must be semi simple (alebgaic multiplicity equals geometric).

3. For the convergence, it follows from calculations on the Jordan blocks, which I omit. See Meyer [10] or Serre [14] for rigorous proof. □

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 A.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:*

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

where D has the form:

$$D = \begin{pmatrix} e^{\theta_1} & & & \\ & e^{\theta_2} & & \\ & & \ddots & \\ & & & e^{\theta_2} \end{pmatrix} \tag{5}$$

And conversely any B of the form 5 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 [10].

The keys for proving this theorem are: First WLG 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

$$|v| = |\mu v| = |Bv| \leq |B||v| \leq A|v| \tag{6}$$

Since A is a transition by remark 7 $A|v| = |v|$. Since A is irreducible and $A|v| = |v|$ we must have by A.2 that $|v| > 0$, and then by lemma A.1 $A = |B|$ so that proves the first part.

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

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

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

$$\begin{aligned} A|v| = |v| = D^{-1}v = \\ = D^{-1}\mu^{-1}Bv = \mu^{-1}D^{-1}BD|v| \end{aligned} \tag{7}$$

We know already that $|v| > 0$. If $C := \mu^{-1}D^{-1}BD$ contains any negative or complex entries, then 7 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 □.

□

The amazing consequence from A.3:

Theorem A.4 (Corollarly). *If A is irreducible transition with h eigenvalues on the unit circle then its eigenvalues are exactly the hth unit roots $\lambda_k = e^{2\pi ik/h}, k = 0 \ldots n-1$ and A is similar to λA by a diagonal matrix for any such eigenvalue.*

Proof. Use theorem A.3 with $B = A$. If $|\lambda| = 1$ is an eigenvalue then $A = \lambda DAD^{-1}$. 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 $|\lambda_k| = 1, k = 0\dot{h} - 1$ 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 hth unit roots. □

Remark 10. 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 A.4 shows that $A = \omega DAD^{-1}$. 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 identify $\forall k D^h A^k = A^k D^h$ and the fact that A is irreducible to show that $D^h = I$. Then use the identity $\omega DA = AD$ 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 A.5 (Perron-Frobenius [10]). *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 A.6 (Perron-Frobenius for irreducible matrices [10])**.** *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* $\Gamma = \{x | x \geq 0, x \neq 0\}$

B Appendix: 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 j to i (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 B.1. A graph is G **strongly connected** if there is a directed path form 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 11. If G is strongly connected then the periods of all vertices are indeed equal and its easy to prove. The corresponding adjacency matrix A is irreducible so it too has a period h as defined in 10 and it is equal to the graph period (can be shown using A.4 and 10).

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 A.5 that there is a unique stationary distribution p and that the Markov process 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.

If the graph G is strongly connected, then A.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 (see theorem A.4).

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 B.1. *[10] 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}$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 sucht that $A^k_{i,j} > 0$ and therefore $(A+I)^{n-1} > 0$. □

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 B.2. Let $A \in \mathbb{R}^{n \times n}$, then its binary form is the matrix $\beta(A) := \operatorname{sgn}(A)$ where sgn is applied element-wise.

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

Lemma B.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 starting distribution p_0 and has the form $p_{k+1} = Ap_k = \cdots = A^k p_0$, where p_k is the distribution after k steps.

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.

When we talk about **random walk with restart (RWR)** we set a restart state q and a restart parameter $\alpha \in [0, 1]$. At each step, we either restart over to the state q, with probability of *alpha*, or continue to walk using the normalized adjacency matrix A. The state sequence is therefore

$$p_{k+1} = \alpha q + (1 - \alpha)A \cdot p_k \tag{8}$$

It turns out that this random walk with restart is actually a normal random walk but with an modified adjacency matrix (and respectively, an modified weighted directional graph).

Lemma B.3. *Let q be any state, let $\alpha \in [0, 1]$ and let $Q = (q|q| \ldots |q)$ (The square matrix whose every column equals q). Then Q is a transition and for any state p we have $Qp = Q$.*

In addition if T is any transition then $W = \alpha Q + (1 - \alpha)T$ is also a transition, And we can rewrite the RWR from 8 as

$$p_{k+1} = \alpha q + (1 - \alpha)Ap_k = [\alpha Q + (1 - \alpha)A]p_k = Wp_k$$

Proof. Trivial and uses B.2 □

The matrix W represents a graph G' where each edge of the original graph G is rescaled by a factor $1 - \alpha$ (and if G is undirected we treat each undirected edge as 2 directed edges in G'. In addition from each vertex v we add edges to every other vertex and the weight of the additional edge is $\alpha q[u]$.

If we pick the restart state q in a way that makes W primitive, then A.5 assures us that the RWR will converge, $\lim_{k\to\infty} p_k = \lim_{k\to\infty} W^k p_0 = p$, where p is the unique stationary distribution of W. This means in particular that we can use the power method to find out the distribution p

by sequentially calculating $p_1, p_2, \dots$ until it sufficiently converges, and it will converge to p from any initial state p_0 which we choose.

Also we can take the limit $p = \lim_{k\to\infty} p_k$ and rewrite 8 as:

$$\begin{aligned} p = Wp = \alpha q + (1-\alpha)A \\ [I - (1-\alpha)A]p = \alpha q \end{aligned} \tag{9}$$

We will see later that we can invert the matrix in the second equation of 9 and use a direct solution for p instead of the power method. The power method has the advantage that we can use the matrix A implictly, because we only need to know $A \cdot p_k$ to compute p_{k+1}. A is usually a sparse matrix because each vertex usually only has few neighbors and so we can use matrix free methods efficiently to calculate p but that is beyong the scope of this thesis.

In the particular case of pageRank, we choose a uniform restart state $q = \frac{\mathbf{1}}{\mathbf{n}}$. The corresponding matrix $Q = (q|\dots|q) > 0$ is strictly positive, and therefore $W = \alpha Q + (1-\alpha)A > 0$ is positive and therefore primitive.

The stationary distribution which corresponds to this uniform restart state q is called the **PageRnak** for G with restart parameter α. It is used to order the vertices according to their 'relevance' in the network.

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 B.4. *Let $A \geq 0$ be irreducible. Let $B \geq 0$ have a positive row (or column), then $A + B$ is primitive.*

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 $i \curvearrowright 1 \curvearrowright j$ of some length k. So $A^k_{i,j} > 0$. 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 $(A+B)^k_{i,j}$ 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. □

Remark 12. Lemma B.4 proves that we can propagate (namely do RWR) from any arbitrary restart state q, including a single vertex and the combined transition matrix will be primitive if the adjacency matrix A is irreducible, or equivalently, the corresponding graph G is strongly connected.

Assume that A is a normalized adjacency transition of a strongly connected graph G. Let q be any state column vector, for example e_1 if we restart from a single vertex, and let $Q = (q|q|\dots|q)$. So Q has a positive row and therefore $(1-\alpha)A + \alpha Q$ is a primitive transition according to B.4.

Definition B.3. Let G be a graph with adjacency matrix A, and let D be the diagonal matrix of the out ranks of the vertices of G. Then we can column normalize A and create the transition $T = AD^{-1}$. We define **the transition matrix with restart parameter α and bias q** as

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

In general, the matrix $T_{\alpha,q}$ may have more than one unique stationary distribution p (there is one for each strongly connected component of its corresponding graph). But if we require that G be strongly connected (which means A is irreducible), then $T_{\alpha,q}$ is primitive by B.4, and the stationary distribution p is unique.

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

We can rearrange it now

$$(I - (1-\alpha)T)p = \alpha q \tag{10}$$

We want to invert the matrix in 10 and use the following lemma to justify it (The proof is easy. See for example Serre [14]):

Lemma B.5. *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 B.5 on equation 10 because $(1 - \alpha)A$ is contracting, and we have:

$$p = \alpha[I - (1 - \alpha)T]^{-1}q := Kq = \alpha \sum_{k=0}^{\infty} (1 - \alpha)^k T^k \tag{11}$$

K is called **diffusion matrix** [9] of T (or of the graph G) 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 11 and since T is irreducible.

What about the eigenvectors and eigenvalues of K? If a matrix A is invertible then $Av = \lambda v \iff A^{-1}v = \lambda^{-1}$. 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 if all the eigenvalues are real (spoiler- they are), then they have the same linear 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 B.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.*

C Appendix: Spectral clustering and other methods

Spectral Partitioning [19, 11, 5, 15], is a common technique for partitioning graphs. The idea is to construct a Laplacian type matrix [8] for the graph G, 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 C.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**:

$$\begin{aligned} L &= D - A \\ L_s &= D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2} \\ L_r &= D^{-1} L = I - D^{-1} A \\ L_c &= L D^{-1} = I - A D^{-1} \end{aligned} \tag{12}$$

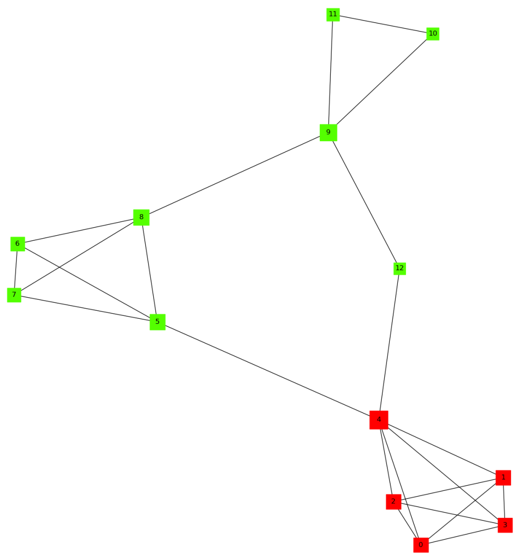
Consider the connected graph G and its adjacency matrix A. We create the transition matrix by normalizing it: $T = AD^-1$, and finally we choose a restart parameter α and create the diffusion matrix $K = \alpha[I - (1 - \alpha)T]^{-1}$. Theorem 3.1 assures us that K and T have the same eigenvectors and 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 $K^{-1} = \alpha^{-1}[I - (1 - \alpha)T]$ is closely related to the symmetric Laplacian L_s and thecolumn-normalized Laplacian L_c. The added parameter $0 < \alpha < 1$ neither changes the eigenvectors, nor the order of the corresponding eigenvalues.

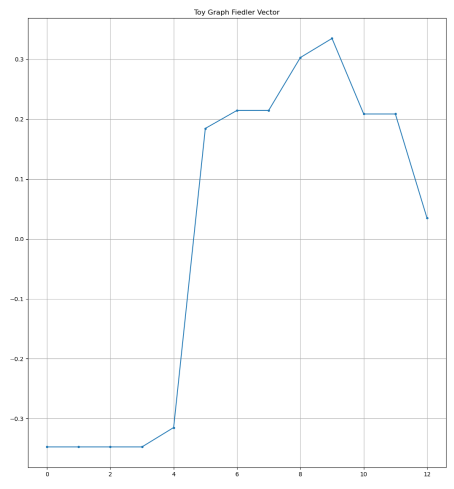
The smallest eigenvalue ,0 of L and L_s corresponds to the largest eigenvalue 1 of and T and its eigenvector is (can be chosen as) all positive. Any other eigenvector of L (which corresponds to an eigenvector of T) must contain both positive and negative components due to 2.3.

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 [11]. 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 [12] 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 [19].

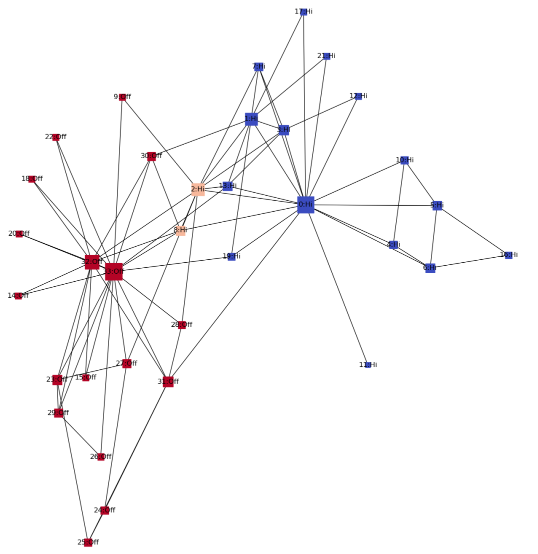


(a) A spectral clustering using the sign of the Fiedler eigenvector.

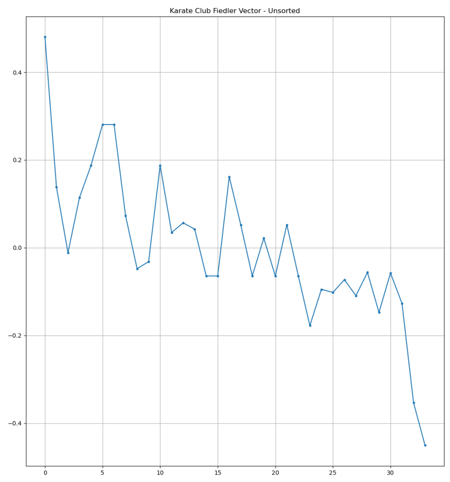


(b) Plot of the Fiedler vector's components, showing a clear subdivision between the positive and the negative components.

Figure 11



(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.

D Reference

References

[1] Gal Barel and Ralf Herwig. “NetCore: a network propagation approach using node coreness”. In: *Nucleic Acids Research* 48.17 (2020), e98–e98.

[2] Sarvenaz Choobdar et al. “Assessment of network module identification across complex diseases”. In: *Nature methods* 16.9 (2019), pp. 843–852.

[3] Lenore Cowen et al. “Network propagation: a universal amplifier of genetic associations”. In: *Nature Reviews Genetics* 18.9 (2017), p. 551.

[4] Minghua Deng et al. “Prediction of protein function using protein-protein interaction data”. In: *Proceedings. IEEE Computer Society Bioinformatics Conference*. IEEE. 2002, pp. 197–206.

[5] *Graph Partition*. Wikipedia. 2020. URL: https://en.wikipedia.org/wiki/Graph_partition (visited on 09/11/2020).

[6] Israel Nathan Herstein and David J Winter. *Matrix theory and linear algebra*. Macmillan Publishing Company, 1989.

[7] Yiftach Josef Kolb. *Project’s source code*. 2020. URL: https://github.com/zelhar/baproject.

[8] *Laplacian Matrix*. Wikipedia. 2020. URL: https://en.wikipedia.org/wiki/Laplacian_matrix (visited on 09/12/2020).

[9] 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.

[10] Carl D Meyer. *Matrix analysis and applied linear algebra*. Vol. 71. Siam, 2000.

[11] Maxim Naumov and Timothy Moon. *Parallel spectral graph partitioning*. Tech. rep. NVIDIA Technical Report, NVR-2016-001, 2016.

[12] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[13] Benno Schwikowski, Peter Uetz, and Stanley Fields. “A network of protein–protein interactions in yeast”. In: *Nature biotechnology* 18.12 (2000), pp. 1257–1261.

[14] D Serre. “Matrices theory and applications second edition”. In: *Graduate Texts in Mathematics* 1.216 (2010), ALL–ALL.

[15] *Spectral Clustering*. Wikipedia. 2020. URL: https://en.wikipedia.org/wiki/Spectral_clustering (visited on 09/11/2020).

[16] Martin Szummer and Tommi Jaakkola. “Partially labeled classification with Markov random walks”. In: *Advances in neural information processing systems*. 2002, pp. 945–952.

[17] Fabio Vandin et al. “Discovery of mutated subnetworks associated with clinical data in cancer”. In: *Biocomputing 2012*. World Scientific, 2012, pp. 55–66.

[18] Martin Vingron. “Propagation methods (Lecture Slides)”.

[19] Ulrike Von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4 (2007), pp. 395–416.

List of Figures

1 a weakly connected graph. 14

2 Pageranked graph. 14

3 again the same graph, with node color and size indicating their PageRank, with a restart probability approaching 1. 15

4 same graph, but undirected. 15

5 a star graph. 16

6 'toy graph' used for testing the algorithm 19

7 'Zachary's Karate Club'. 'Mr Hi' (Yellow) and 'Officer' (Green) groups. The Cyan colored node indicates a mismatch between the clustering and the actual division. 20

8 Subgraph of the yeast interactome showing nodes of 4 groups: 'DNA replication' (blue), 'Golgi' (red), 'Meiosis' (yellow), 'Stress response' (green) 25

9 The test network 26

10 The accuracy as function of the resart parameter alpha, with fixed label coverage of 0.5, and of the label coverage with fixed alpha=0.2 28

11 46

List of Tables

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. 25