IMPERIAL

Imperial College London

Department of Mathematics

Second-year Group Research Project

Explorations into PageRank: A Case Study on Cancer Driver Genes

Authors:
Student 1
Student 2
Student 3
Student 4
Thea Sotthivej

Supervisor:
Professor Darren Crowdy

Submitted 17th June 2024

Abstract

In the field of bioinformatics, there are many approaches to biological network analysis, some of which rely more heavily on biological data and others on the topological features of the network. Transcriptional Regulatory Networks (TRNs) are a type of biological network which gives us information about the regulatory interactions between genes. In this paper we modify the original PageRank algorithm, an algorithm used by Google to rank webpages according to their 'importance', for use on a TRN constructed using sample data from a breast cancer study in order to identify the most influential genes. We also verify mathematical properties of the algorithm, especially robustness under parameter changes. Comparing our results to existing literature, we show that this network-based approach has potential as a complementary tool to TRN analysis via biological data.

Keywords: PageRank, Cancer Driver Gene, Transcriptional Regulatory Network (TRN)

Contents

1 Introduction

PageRank is a method of ranking how important each *node* is in a *network*, based on the structure of the network. The algorithm evaluates nodes based on their connectivity, inferring the significance of a node by the quantity and quality of edges directed towards it.

The founders of Google, Brin and Page, created PageRank for the purpose of ranking *web pages* on the *internet*, connected by *hyperlinks*, allowing popularly referenced sites to appear first in search results and therefore giving birth to the modern search engine [1].

PageRank has found numerous scientific applications beyond the web, where the *nodes* and *edges* take on different meanings. For example:

- In social sciences and epidemiology, PageRank is used to identify well-connected *people* connected by *social interactions* [2].
- PageRank is used to identify influential *papers* connected by *citations*, as an alternative to citation counts [2].
- The application we will focus on is in bioinformatics, where PageRank can be used to rank *genes* connected by their *regulatory interactions.*

In biology, network-like structures are commonplace and can be seen in many contexts, such as protein-protein interaction (PPI) networks, metabolic networks, and transcriptional regulatory networks (TRNs) [3]. Here, we choose to analyse a TRN in order to identify cancer driver genes, as the function of the nodes of this type of network are integral to the development of a disease [4].

In this paper, a modified version of PageRank, known as 'Personalised PageRank', will be used to analyse a TRN. The relevant mathematical properties of this algorithm will also be explored. From this analysis, we get a ranked list of the most influential genes for *breast carcinogenesis* (formation of breast cancer) and compare it to existing literature.

While similar approaches have been taken previously with TRNs [5, 6], there are many different degrees of freedom in our graph interpretation in a biological context, and as such we propose a novel technique in our PageRank analysis, namely in the personalisation of the PageRank algorithm.

This differs from previous methods as gene expression data is used in the personalisation of the PageRank algorithm, with a protein-protein interaction (PPI) database used for the edge weighting. We find that this novel approach gives promising results for positive identification of cancer driver genes (CDGs), which opens up future avenues for exploration.

2 Background

2.1 Prerequisite Biology

Cells in the body respond to biological stimuli by producing special proteins called *transcription factors* [7]. Their function is to regulate genes in the cell in order to make sure that they are expressed in the appropriate manner according to the stimuli. We can consider the activities of the transcription factors, therefore, to be an internal representation of a compact description of the environment around the cell.

These transcription factors are encoded by genes, regulated by other transcription factors, which may be regulated by other transcription factors, and so on and so forth. These interactions can be presented in the form of a graph called a *transcriptional regulatory network* (TRN), which describes all of the regulatory transcription interactions in a cell. In a TRN, the *nodes* are *genes*, and *arrows* between genes X → Y signify that the protein product of gene X *is a transcription factor of* gene Y.

The reason we are analysing a TRN is, as previously mentioned, because the perturbation in the function of transcription factors plays an important role in the development of a disease, especially in cancer [4]. In this paper, we will specifically be looking at a TRN constructed using gene expression data from breast cancer, in order to identify genes important to *breast carcinogenesis* (formation of breast cancer).

2.2 PageRank as a Network Analysis Tool

As previously mentioned, PageRank is a method of ranking how important each node is in a network, based on the structure of the network, also known as a *centrality* measure. After running PageRank on our network, we have a list of centralities of the nodes, and rank these in order of magnitude. The final ranked list of nodes is what we want from using PageRank, without much regard necessarily for the actual centralities of the nodes.

We note that PageRank gives us the nodes with the highest quantity and quality of incoming edges, so for our purposes we 'reversed' the PageRank calculation by flipping all directed edges of the graph to consider the outgoing edges. We justify this as trying to find the genes with the most effect on other genes, i.e. the greatest number and quality of outgoing edges, although other mathematical interpretations are possible.

2.3 Network Science Approaches in Cancer Driver Gene Discovery

Cancer Driver Genes (CDGs) are genes which lead to tumour growth when mutated. There are many existing methods for detecting CDGs, many of which use other types of biological data. For example, Oncodrive-Fm [8] and CoMDP [9] are computational-based methods, which try to identify CDGs based on mutation and genomic data. DawnRank [10] and DriverNet [11] are methods which try to identify CDGs based on mutation data, prior knowledge of pathways and genetic interactions.

However, they still have some limitations, such as being dependent on high noise data, being computationally costly and ignoring the topology of the biological networks (e.g. TRNs) at hand [5]. Thus, in recent years, computationally light network-based methods have been suggested, which rely more heavily on the topology of the biological network. Generally, these use network analysis tools in conjunction with smaller datasets in order to identify nodes with high *centrality*. In this paper, we will be using *personalised PageRank* centrality in conjunction with gene expression data.

3 An Introduction to PageRank

3.1 Motivation

Consider a connected undirected graph with 4 nodes and 5 edges (Figure 1). Suppose, in this case, that this represents 4 webpages which a Web surfer navigates between using hyperlinks on each webpage, and that we wish to rank these pages by importance for a search engine.

Our task is therefore to assign each webpage a measure of its importance. Clearly, symmetry of the network structure implies that pages 1 and 3 should receive equal measure, as should pages 2 and 4.

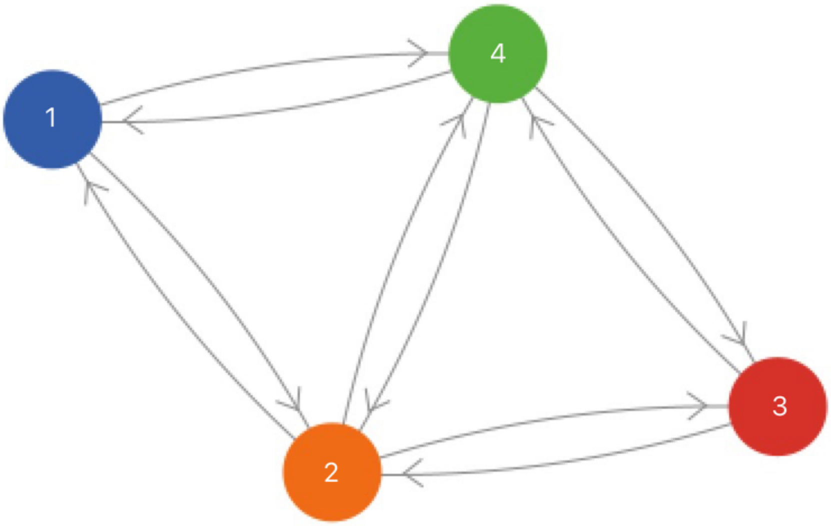

Figure 1 A connected undirected graph generated by NetLogo [12], with 4 nodes and 5 edges labelled. Each edge is substituted equivalently with a pair of directed edges.

To begin, we look at the model of a random Web surfer. From any node (webpage), the surfer chooses an outbound edge (hyperlink) with uniform probability to then travel to another one, and we simply use the traffic through each page as an indicator of importance.

Keeping this in mind, an intuitive ranking for the labelled nodes that we'd consider is the vector

$$\begin{pmatrix} \frac{2}{10} \\ \frac{3}{10} \\ \frac{2}{10} \\ \frac{3}{10} \end{pmatrix}$$

where we have counted up the number of edges connected to each node and scaled the vector such that the entries sum up to 1. We turn to the theory of random walks to verify this initial guess.

The Laplacian matrix K for this graph can be written as follows:

$$K = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}$$

K may be written in the form $\mathbf{K} = \mathbf{D} - \mathbf{W}$, where D, W are the degree and adjacency matrices respectively. By definition,

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}, \ W = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Of note is the fact that the vector of ones

$$\boldsymbol{x}_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

is a right null vector of the Laplacian matrix K, i.e. $K\boldsymbol{x}_0 = 0$.

Consider also the matrix of hopping probabilities T, i.e.

$$T = \begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

We call T a *stochastic matrix*, since the entries in each row are non-negative and sum to 1. Note, then, that $\boldsymbol{x_0}$ is an eigenvector of T with unit eigenvalue. That is, $T\boldsymbol{x}_0 = \boldsymbol{x}_0$. What can we do with all of this information about the graph? We must turn to probability theory for some insight.

The stationary distribution of this graph can be found by finding the transpose of the left eigenvector of T, which represents the relative amount of traffic through each node. Then, to find the stationary distribution, let $\hat{\boldsymbol{x}}^\top$ be the left eigenvector of T. We proceed as follows, noting that $T = D^{-1}W$:

$$\begin{aligned}
& & \hat{\boldsymbol{x}}^\top T &= \hat{\boldsymbol{x}}^\top & \\
\Longrightarrow & & T^\top \hat{\boldsymbol{x}} &= \hat{\boldsymbol{x}} & \\
& & &= (D^{-1}W)^\top \hat{\boldsymbol{x}} & \text{since } T = D^{-1}W \\
& & &= W^\top (D^{-1})^\top \hat{\boldsymbol{x}} & \\
& & &= WD^{-1}\hat{\boldsymbol{x}} & \text{by symmetry of W, D.}
\end{aligned}$$

By observation, we see that setting $\hat{\boldsymbol{x}} = D\boldsymbol{x}_0$ gives us that $T^\top \hat{\boldsymbol{x}} = T^\top D\boldsymbol{x}_0 = D\boldsymbol{x}_0$, by definition of K and since $K\boldsymbol{x}_0 = 0$.

Thus, $\hat{\boldsymbol{x}} = D\boldsymbol{x}_0$ is our stationary distribution, i.e.

$$\hat{\boldsymbol{x}} = \begin{pmatrix} 2 \\ 3 \\ 2 \\ 3 \end{pmatrix}$$

which we simply rescale by multiplying through by $\frac{1}{10}$, verifying our intuitive guess.

Finding this stationary distribution is the core idea of PageRank. In reality, there are many more modelling considerations to be taken. In this case, our model instead computes a measure known as degree centrality, with scores proportional to the node degrees. This was due to our artificial assumption that for any pair of webpages, hyperlinks exist in both directions, or in other words, the undirected nature of the graph.

We aim to rigorously improve this model in the following subsection by allowing directed edges and considering their implications. For example, nodes with no outgoing edges can now exist (called *dangling nodes*), which causes a problem in terms of describing the graph as a Markov chain. We also need to make sure that the Markov chain is *irreducible* to ensure uniqueness of the resulting PageRank vector.

3.2 The Original PageRank Model

Suppose we have a graph consisting of a set of nodes connected by edges which may be directed or undirected. Viewing the graph as a rudimentary Markov chain, we define the transition matrix $\mathbf{P}$, such that $P_{i,j} = \frac{1}{N_i}$ if there is an edge from i to j and $P_{i,j} = 0$ otherwise, where N_i is the number of outgoing edges from node i (we consider an undirected edge as both an incoming and an outgoing edge). This is not necessarily a stochastic matrix, since we could have nodes with no outgoing edges, referred to as *dangling nodes* [13], resulting in zero rows in matrix $\mathbf{P}$.

With no further modifications, this matrix does not define a Markov chain, because where the chain goes after moving to a dangling node is not defined.

In order to make $\mathbf{P}$ a stochastic matrix, we assign a probability distribution to each dangling node in the graph. We define $\overline{\mathbf{P}} = \mathbf{P} + \mathbf{a}\mathbf{v}^\top$, where $\mathbf{v}^\top$ is the probability distribution (as a row vector), and $a_i = 1$ if node i is a dangling node, and 0 otherwise. $\mathbf{v}^\top$ is generally chosen to be $\mathbf{e}^\top/n$ (where $\mathbf{e}$ is a vector of ones), corresponding to a uniform probability of teleporting to any node in the network. We call $\mathbf{v}^\top$ the teleportation vector, or the personalisation vector (especially when its elements are non-uniform, i.e. when we change $\mathbf{e}$ to some vector $\mathbf{x_0}$). We now have a stochastic matrix representing a Markov chain that is not necessarily irreducible.

However, this is not enough as we need to ensure that the Markov chain is irreducible (that every node is accessible from every other node) to guarantee the existence of a unique PageRank vector. Hence, we define $\overline{\overline{\mathbf{P}}} = \alpha\overline{\mathbf{P}} + (1-\alpha)\mathbf{e}\mathbf{v}^\top/n$ where $0 \leq \alpha \leq 1$ and $\mathbf{E} = \frac{1}{n}\mathbf{e}\mathbf{v}^\top$. We will later prove that this ensures that $\overline{\overline{\mathbf{P}}}$ is primitive (Prop. 3.3) and so a unique stationary PageRank vector $\pi^\top$ exists. Therefore, we arrive at the PageRank formula

$$\pi^\top\overline{\overline{\mathbf{P}}} = \pi^\top, \ \ \pi^\top\mathbf{e} = 1. \tag{1}$$

From the previous example of the random web surfer, the formula can be interpreted as the surfer moving to another webpage through hyperlinks with a probability of α, and with a probability of $1-\alpha$ choosing to "teleport" directly to a different webpage, bypassing hyperlinks (for instance, by directly entering a webpage's URL). The PageRank vector $\pi^\top$ therefore represents the probability distribution that the surfer is on a given webpage after a long period of time. That is, $\pi^\top$ is a stationary distribution for the Markov chain defined by $\overline{\overline{\mathbf{P}}}$.

3.3 Solving PageRank using the Power Method

3.3.1 The Power Method algorithm

By considering the PageRank formula (1), we see that the idea of getting a stationary state vector (PageRank) is equivalent to finding the eigenvector with unit eigenvalue of $\overline{\overline{\mathbf{P}}}$. One way to find this eigenvector is the Power Method.

The power method works by iteratively multiplying a starting probability vector $\pi^{(0)\top}$ with $\overline{\overline{\mathbf{P}}}$ [13]. Call the resulting vector after k multiplications $\pi^{(k)\top}$. By the definitions of $\mathbf{P}$, $\overline{\mathbf{P}}$, and $\overline{\overline{\mathbf{P}}}$ given above and the fact that $\pi^{(k-1)\top}$ is a probability vector (so $\pi^{(k-1)\top}\mathbf{e} = 1$), we can deduce with some algebra that

$$\pi^{(k)\top} = \alpha\pi^{(k-1)\top}\mathbf{P} + (\alpha\pi^{(k-1)\top}\mathbf{e} + (1-\alpha))v^\top.$$

We use the power method with this equation mostly in this form as we only need to use sparse $\mathbf{P}$ rather than dense $\overline{\mathbf{P}}$ or $\overline{\overline{\mathbf{P}}}$, saving computer memory and computation time.

3.3.2 Existence, uniqueness, and convergence of the Power Method

The irreducibility of the matrix $\overline{\overline{\mathbf{P}}}$, courtesy of the stochastic perturbation matrix $\mathbf{E}$, ensures the existence of the unique stationary distribution vector for the PageRank equation.

In order to prove the convergence of the power method to this unique solution, we introduce the following theorem [14] and proposition:

Theorem 3.1 (Perron-Frobenius for positive matrices). *Consider a positive matrix (i.e. all entries are positive)* $\mathbf{B}$*. There exists a real, positive eigenvalue* λ *such that:*

- $\lambda = \rho(\mathbf{B}) > 0$*, and all other eigenvalues are smaller in magnitude.*
- *This eigenvalue is simple, all elements of the corresponding eigenvector have the same sign, and there are no other eigenvectors where all elements have the same sign.*

Proposition 3.2. *Define* $\overline{\overline{\mathbf{P}}}$ *as above:* $\overline{\overline{\mathbf{P}}} = \alpha\overline{\mathbf{P}} + (1-\alpha)\mathbf{e}\mathbf{v}^\top/n$*, where* $\overline{\mathbf{P}}$ *is a stochastic matrix and* $0 \leq \alpha \leq 1$*. Let* $\pi^\top$ *be a left eigenvector of* $\overline{\overline{\mathbf{P}}}$ *with eigenvalue 1. Then* $\pi^\top$ *cannot contain two elements with different sign.*

Proof. Take the transpose of (1), then refer to [15]. □

We may now prove convergence by showing:

Proposition 3.3. *Define* $\overline{\overline{\mathbf{P}}}$ *as above.* $\overline{\overline{\mathbf{P}}}$ *is primitive - that is, it is a non-negative, irreducible matrix that has only one eigenvalue on its spectral circle.*

Proof. Clearly $\overline{\overline{\mathbf{P}}}$ is non-negative and irreducible by construction. By the Perron-Frobenius theorem (3.1), the eigenvector $\pi^\top$ with eigenvalue 1 (from Prop. 3.2) corresponds to a simple positive eigenvalue larger in magnitude than all other eigenvalues. Therefore, $\lambda = 1$ is the largest eigenvalue and its simplicity implies that $\overline{\overline{\mathbf{P}}}$ is primitive. □

The power method converges on the eigenvector with greatest eigenvalue, hence it will converge uniquely to the stationary vector $\pi^\top$.

3.4 Dangling nodes

There are indeed some cases where a page does not have outgoing links to other pages, for instance some images or static documents (e.g. PDFs). Notice that such pages are dangling nodes. As mentioned previously, we have to modify $\mathbf{P}$ to be a stochastic matrix by assigning a probability distribution $\mathbf{v}^\top$ to each dangling node in the graph, usually chosen to be uniform probability of teleporting to any node in the network. Then, we calculate the stationary vector $\pi^\top$ using the PageRank formula. Here, we take $\mathbf{v}^\top = \frac{1}{4}\mathbf{e}^\top$ and $\alpha = 1$.

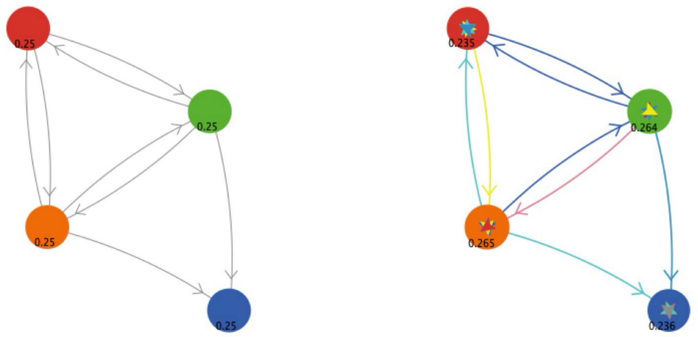

Figure 2 System with dangling node 1 (the blue node).

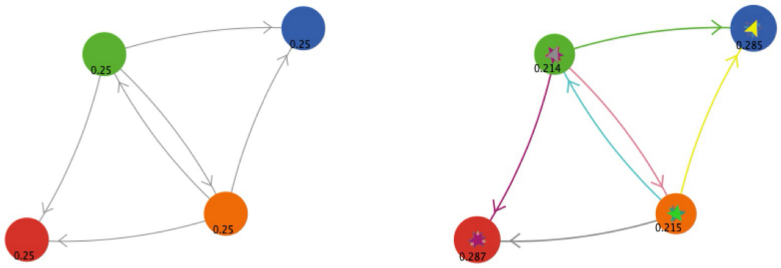

Figure 3 System with dangling nodes 1 and 3 (the blue and red nodes).

Returning to the connected undirected graph with 4 nodes and 5 edges, shown earlier in Figure 1, we get Figure 2 after removing the two outgoing links of node 1. After some calculation, we obtain the PageRank vector

$$\begin{pmatrix} \frac{8}{34} \\ \frac{9}{34} \\ \frac{8}{34} \\ \frac{9}{34} \end{pmatrix}.$$

Similarly, we could have two dangling nodes in our modified system shown in Figure 3. In this case we have the PageRank vector

$$\begin{pmatrix} \frac{4}{14} \\ \frac{3}{14} \\ \frac{4}{14} \\ \frac{3}{14} \end{pmatrix}.$$

Notice how in the former case, the PageRanks of the orange and green nodes are greater than that of the red and blue nodes, but in the latter case, we instead have that the PageRanks of the red and blue dangling nodes are greater. To understand this, we return to the stochastic matrix defining the Markov chain, $\overline{\mathbf{P}}$.

In the former case shown in Figure 2, we have

$$\overline{\mathbf{P}}_1 = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}.$$

Considering a random walker on this Markov chain, notice that once the walker reaches node 3, they can only choose to go to nodes 2 or 4, so intuitively, the walker will end up spending more time on nodes 2 and 4, giving these nodes higher PageRank.

However, in the latter case shown in Figure 3, we have

$$\overline{\mathbf{P}}_2 = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}.$$

Here, on reaching node 3, our random walker can move to any node in the network with equal probability, including staying on node 3. This change to the graph structure is enough to suggest that in the limit, the walker spends more time on the dangling nodes than the non-dangling nodes, because the walker cannot stay on a given non-dangling node; it must move to a different node.

3.5 Self-loops

Another common component in random graphs are self-loops. Self-loops arise when an edge directly connects a node back to itself. The existence of self-loops affects the PageRank vector; suppose node i has a self-loop and that the probability of the chain taking this self-loop is p. At node i, the Markov chain may stay at node i with probability αp, or leave either by moving to another node via an outgoing edge (probability $\alpha(1-p)$) or by teleporting to another node (probability $1-\alpha$). The larger αp is, the more time the chain will spend on node i in the limit, giving it a higher PageRank.

Thus, the presence of a self-loop on a node will inflate its PageRank compared to that without the self-loop. Note that multiple self-loops, with combined probability q of the chain taking any loop, can be reduced to one self-loop with probability q of the chain taking this loop.

Of note is the fact that if we have $\alpha p = 1$ for any number of nodes, such nodes will be an absorbing state for the Markov chain, resulting in the elements of the PageRank vector being 0 for all other nodes. This is why we required the perturbation matrix $\mathbf{E}$ and commonly disallow setting $\alpha = 1$ in order to ensure the irreducibility of the Markov chain.

4 Further Discussion of PageRank

4.1 The Linear System Formulation

An alternative method to compute the PageRank vector involves solving the following related linear system. We can rearrange the eigenvalue problem (1) into

$$\pi^\top(\mathbf{I} - \alpha\overline{\mathbf{P}}) = (1-\alpha)\mathbf{v}^\top. \tag{2}$$

Note that we still enforce the restriction that $\pi^\top\mathbf{e} = 1$ to ensure a valid stationary distribution.

This formulation admits a more efficient method to compute the PageRank of a graph, by exploiting dangling nodes [16].

4.2 Derivation of the Efficient PageRank Algorithm

If we have a large number of dangling nodes, $\overline{\mathbf{P}}$ becomes dense, increasing computation time. Using our definition $\overline{\mathbf{P}} = \mathbf{P} + \mathbf{a}\mathbf{v}^\top$ from the previous section, we can rewrite (2) as

$$\pi^\top(\mathbf{I} - \alpha\mathbf{P} - \alpha\mathbf{a}\mathbf{v}^\top) = (1-\alpha)\mathbf{v}^\top.$$

If we set $\pi^\top\mathbf{a} = \gamma$, the linear system becomes

$$\pi^\top(\mathbf{I} - \alpha\mathbf{P}) = (1-\alpha+\alpha\gamma)\mathbf{v}^\top$$

where γ is equal to the sum of the PageRanks of all the dangling nodes. Since we enforce $\pi^\top\mathbf{e} = 1$ only at the end, we can choose γ arbitrarily as the effect of this choice will be negated upon normalisation in the final step. Thus, let $\gamma = 1$. We now have:

$$\pi^\top(\mathbf{I} - \alpha\mathbf{P}) = \mathbf{v}^\top. \tag{3}$$

We claim that $\mathbf{I} - \alpha\mathbf{P}$ is invertible so long as $\alpha \neq 1$.

Proposition 4.1. *$\mathbf{I} - \alpha\mathbf{P}$ is invertible for $0 \leq \alpha < 1$.*

Proof. We will show that $\mathbf{I} - \alpha\mathbf{P}$ with $0 \leq \alpha < 1$ satisfies property I_{27} from Theorem 2.3 of Chapter 6 from [16]. The property states that if there exists a vector $\mathbf{x} \gg 0$ such that $\mathbf{A}\mathbf{x} \gg 0$ for a real matrix $\mathbf{A}$ where $A_{i,j} \leq 0$ for all $i \neq j$, then $\mathbf{A}$ is invertible. (We say a vector $\mathbf{x} \gg 0$ if $x_i > 0$ for all i.)

Taking $\mathbf{A} = \mathbf{I} - \alpha\mathbf{P}$ and $\mathbf{x} = \mathbf{e}$, we can see that the property is satisfied because the rows of $\mathbf{P}$ sum to 1 for non-dangling nodes and 0 for dangling nodes, so the rows of $\mathbf{I} - \alpha\mathbf{P}$ must sum to $1-\alpha$ for non-dangling nodes and 1 for dangling nodes, giving $\mathbf{A}\mathbf{x} \gg 0$. □

We now have an even more efficient method to compute the PageRank vector, exploiting the zero rows of $\mathbf{P}$. We permute $\mathbf{P}$ such that the rows and columns corresponding to dangling nodes are at the bottom and right of the matrix respectively (this is equivalent to exchanging the rows then the columns of $\mathbf{P}$):

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

where $\mathbf{P_{11}}$ is the matrix corresponding to the hopping probabilities from non-dangling nodes to non-dangling nodes, and $\mathbf{P}_{12}$ is the matrix corresponding to hopping probabilities from non-dangling nodes to dangling nodes. Slightly abusing notation such that $\mathbf{I}$ is always of the required size, we now have:

$$\mathbf{I} - \alpha\mathbf{P} = \begin{pmatrix} \mathbf{I} - \alpha\mathbf{P}_{11} & -\alpha\mathbf{P}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

We wish to find the inverse of this matrix, which we know exists from above. Writing the inverse as $\left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$ and block multiplying:

$$\begin{pmatrix} \mathbf{I} - \alpha\mathbf{P}_{11} & -\alpha\mathbf{P}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} (\mathbf{I} - \alpha\mathbf{P}_{11})A - \alpha\mathbf{P}_{12}C & (\mathbf{I} - \alpha\mathbf{P}_{11})B - \alpha\mathbf{P}_{12}D \\ C & D \end{pmatrix}$$

For the RHS to be equal to $\mathbf{I}$, clearly we need $C = \mathbf{0}$ and $D = \mathbf{I}$, which implies that we need $A = (\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}$ and $B = \alpha(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}\mathbf{P}_{12}$. We note that $(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}$ also exists using a similar proof as in Prop 4.1. Hence, we have that

$$(\mathbf{I} - \alpha\mathbf{P})^{-1} = \begin{pmatrix} (\mathbf{I} - \alpha\mathbf{P}_{11})^{-1} & \alpha(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}\mathbf{P}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

Therefore, we can write the PageRank vector $\pi^\top = \mathbf{v}^\top(\mathbf{I} - \alpha\mathbf{P})^{-1}$ as

$$\pi^\top = \left(\mathbf{v}_1^\top(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1} \quad | \quad \alpha\mathbf{v}_1^\top(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}\mathbf{P}_{12} + \mathbf{v}_2^\top\right)$$

where $\pi^\top$ and $\mathbf{v}^\top$ have been reordered such that they can be partitioned into $\begin{bmatrix}\pi_1^\top & \pi_2^\top\end{bmatrix}$ and $\begin{bmatrix}\mathbf{v}_1^\top & \mathbf{v}_2^\top\end{bmatrix}$ respectively, where the former section corresponds to non-dangling nodes and the latter section corresponds to dangling nodes. With this, we now have a simple algorithm to calculate the PageRank vector:

Algorithm 1: How to more efficiently calculate PageRank using the Linear System Formulation

```
1 Apply the permutation described above to $\mathbf{P}$ and $\mathbf{v}^\top$.
2 Solve for $\pi_1^\top$ using $\pi_1^\top = \mathbf{v}_1^\top(\mathbf{I} - \alpha\mathbf{P}_{11})^{-1}$.
3 Calculate $\pi_2^\top = \alpha\pi_1^\top\mathbf{P}_{12} + \mathbf{v}_2^\top$.
4 Normalise $\pi^\top = \begin{bmatrix}\pi_1^\top & \pi_2^\top\end{bmatrix} / \| \begin{bmatrix}\pi_1^\top & \pi_2^\top\end{bmatrix} \|_1$.
5 Apply the inverse permutation to $\pi^\top$.
```

Note that in step 4, in order to ensure that the final PageRank vector is a valid probability distribution, we used the 1-norm for an n-dimensional vector defined as

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|.$$

Because we only need to solve a linear system on a submatrix of $\mathbf{P}$, this algorithm generally reduces computation time compared to solving the linear system in (3), and the savings are greater the more dangling nodes are present because the size of $\mathbf{P}_{11}$ gets smaller.

4.3 Changing α

α is the "damping factor" which determines the proportion of time that a random surfer surfs following the hyperlinks instead of teleporting. Below, we explore how changes in this parameter affect the PageRank.

4.3.1 Rate of convergence

We claim that higher values of α will result in a slower rate of convergence to the PageRank vector using the power method.

Proposition 4.2 (rate of convergence and damping factor α)**.** *Let $\bar{\bar{\mathbf{P}}}$ be an $n \times n$ primitive matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ satisfying $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$.*

The convergence rate of the power method is $\gamma \leq \left|\frac{\lambda_2}{\lambda_1}\right|$ with λ_1 the dominant eigenvalue of the primitive matrix and λ_2 the subdominant eigenvalue.

Proof. Let $u_1, u_2, \ldots, u_n$ be eigenvectors that form a basis of $\mathbb{R}^n$, with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ respectively, such that $Au_i = \lambda_i u_i$, for $A = \bar{\bar{\mathbf{P}}}^\top$. Expressing the initial vector as a linear combination of the eigenvectors, $\pi^{(0)} = \beta_1 u_1 + \beta_2 u_2 + \cdots + \beta_n u_n$ for some coefficients β_i, and applying power iterations yields

$$\begin{aligned}
\pi^{(k+1)} &= A\pi^{(k)} \\
&= A^{k+1}\pi^{(0)} \\
&= \sum_{i=1}^{n} \beta_i A^{k+1} u_i \\
&= \sum_{i=1}^{n} \beta_i \lambda_i^{k+1} u_i \\
&= \lambda_1^{k+1}\left[\beta_1 u_1 + \beta_2 \left(\frac{\lambda_2}{\lambda_1}\right)^{k+1} u_2 + \sum_{i=3}^{n} \beta_i \left(\frac{\lambda_i}{\lambda_1}\right)^{k+1} u_i\right].
\end{aligned}$$

By Prop. 3.3, we have $1 = |\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Now, there exists a positive integer m such that $\lambda_i = \lambda_2$ for all $2 \leq i \leq m$. For all the $\frac{\lambda_i}{\lambda_1}$ with $i > m$ in the last equation, these will tend to zero faster than $\frac{\lambda_2}{\lambda_1}$ as k increases. Thus, the rate of convergence γ will depend on $\frac{\lambda_2}{\lambda_1}$, since all other terms will vanish. Moreover, we have $\gamma \leq \left|\frac{\lambda_2}{\lambda_1}\right|$ if $\beta_2 = 0$ and $\gamma = \left|\frac{\lambda_2}{\lambda_1}\right|$ as long as $\beta_2 \neq 0$. □

The convergence is fast if the ratio $\left|\frac{\lambda_2}{\lambda_1}\right|$ is small. However, if λ_2 gets closer to λ_1, $\left|\frac{\lambda_2}{\lambda_1}\right|$ will get larger and convergence will thus be much slower. We note that it has been found that $|\lambda_2| = \alpha$ [17]. Therefore, as α increases, γ will increase, that is, the rate of convergence will be slower [18].

Another way to see how α affects the rate of convergence is to use NetLogo to see how many ticks (iterations) run in the PageRank model [19] before reaching a stationary distribution. We will illustrate this by building the connected, undirected graph shown earlier in Figure 1 in NetLogo, and then compute the PageRank whilst varying α. NetLogo's PageRank model calculates PageRank by simulating a number of random surfers and considering the frequency of visits over a number of iterations. The PageRank of a page at any iteration is calculated by dividing the total number of visits to that page from any surfer up to that iteration by the total number of visits to all pages from all surfers up to that iteration. The surfer will go to another page following a link with probability α, and with probability $1 - \alpha$ jump to another webpage directly. Essentially, we numerically compute the stationary distribution of the Markov chain.

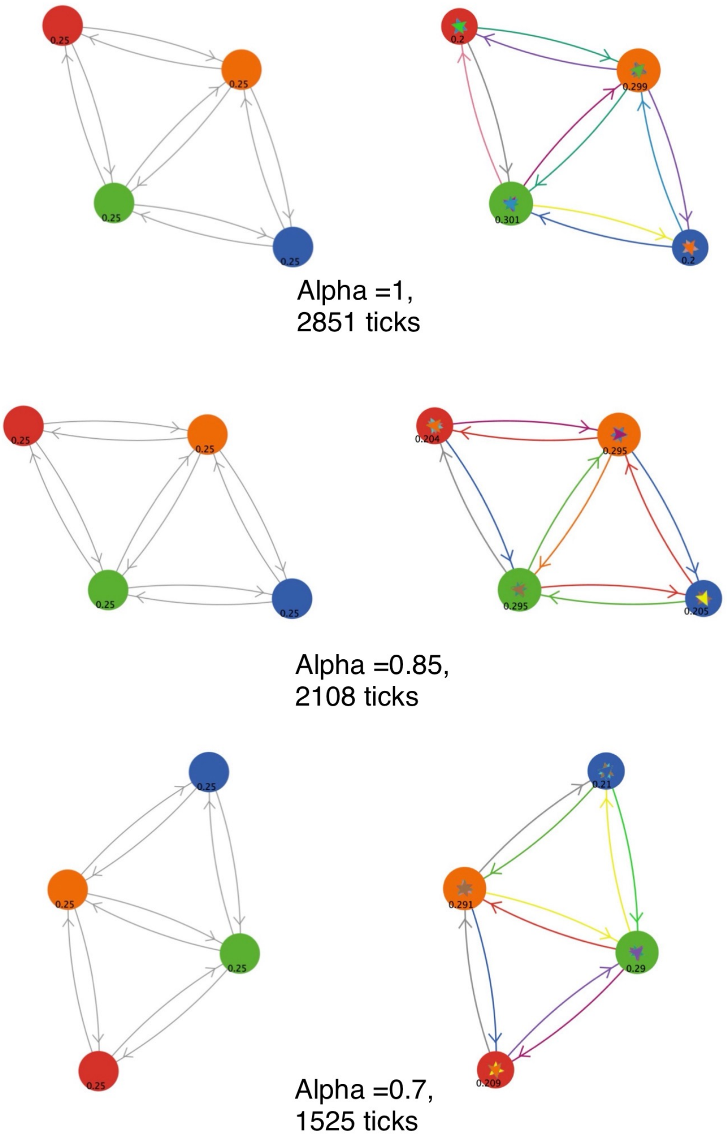

Figure 4 Different number of ticks (iterations) needed to converge to a stationary distribution for α equal to 1, 0.85 and 0.7 respectively.

Clearly from Figure 4, larger α values result in more iterations needed to reach the stationary distribution. This means that the rate of convergence to a stationary value becomes slower as α increases. One more thing to note is that since α is changing, the values that the stationary vector converges to are different. However, the order of the nodes for this sample graph (after sorting by their PageRank values) remains the same.

4.3.2 Changing distribution

In this analysis, we will select $\alpha = 0.85$ as the baseline, and show that different values of α result in different PageRank distributions.

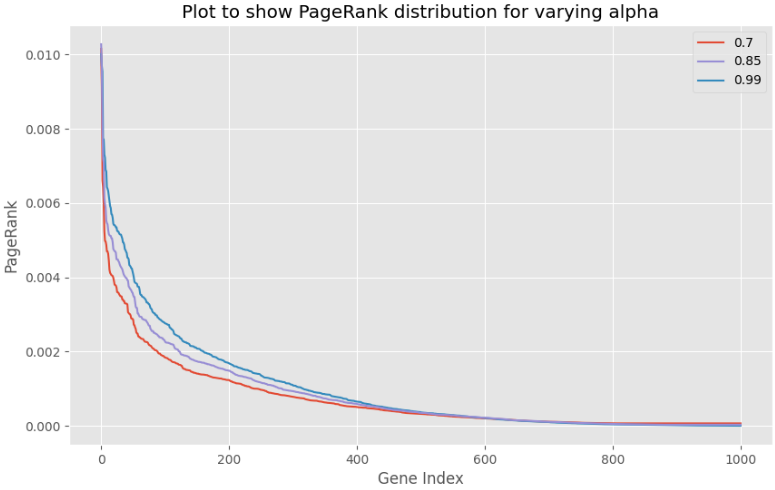

Figure 5 PageRanks of highest PageRank nodes.

We can use Kullback–Leibler divergence (K-L divergence) to show that the underlying probability distribution of the PageRank vector is different if we pick different α. The K-L divergence, denoted $D_{\text{KL}}(P\|Q)$, is a type of statistical distance: a measure of how one probability distribution P is different from a second, reference probability distribution Q. In the case of discrete distributions, we define relative entropy from Q to P as [20]

$$D_{\text{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right). \tag{4}$$

Subsequently, we compute the K-L divergence in increments of α of 0.05. Note here that K-L divergence jumps as $\alpha \to 1$, suggesting that the PageRank does not converge due to the unique structure of the biological graph, namely the large number of nodes ($\tilde{1}$0,000) that have no in-edges, and therefore have their PageRanks determined solely by α (and the personalisation vector $\mathbf{v}$).

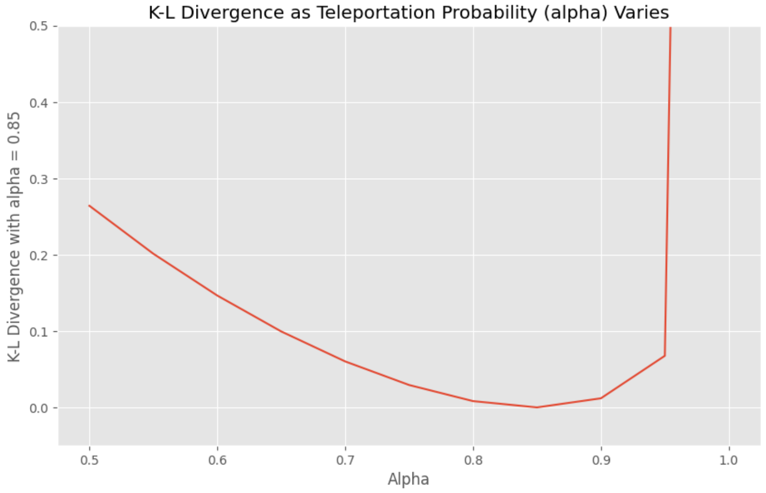

Figure 6 K-L divergence between alpha $= 0.85$ and given alpha.

4.3.3 Sensitivity and Stability

Consider the linear system form of the PageRank equation,

$$\pi^\top(\mathbf{I} - \alpha\overline{\mathbf{P}}) = (1-\alpha)\mathbf{v}^\top.$$

We introduce the condition number of the linear system $\kappa_\infty(\mathbf{I}-\alpha\overline{\mathbf{P}}) = \|(\mathbf{I}-\alpha\overline{\mathbf{P}})\|_\infty \|(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\|_\infty$, which measures the degree of ill-conditioning of $(\mathbf{I}-\alpha\overline{\mathbf{P}})$ [13]. We say that a matrix A is ill-conditioned if a small change in A causes a large relative change in A^{-1}. Supposing we have at least one non-dangling node, we have that $\|(\mathbf{I}-\alpha\overline{\mathbf{P}})\|_\infty = 1+\alpha$ and $\|(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\|_\infty = \frac{1}{1-\alpha}$ by considering the row sums of $(\mathbf{I}-\alpha\overline{\mathbf{P}})$ and $(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}$ respectively. Hence, we have that $\kappa_\infty(\mathbf{I}-\alpha\overline{\mathbf{P}}) = \frac{1+\alpha}{1-\alpha}$. If we choose α which is infinitely close to 1, then we find that a small change in the coefficient matrix will cause a massive change in the solution vector; the system seems to become more sensitive. However, because $\pi^\top$ is an eigenvector of $\overline{\overline{\mathbf{P}}}$, the direction of $\pi^\top$ may only change negligibly even though the elements of $\pi^\top$ change wildly in response to small perturbations to the coefficient matrix. In fact, upon normalisation of the solution vector to obtain a probability vector, the effect is very small.

On the other hand, using the eigenvector formulation below, we can deduce how changes in $\mathbf{P}$ affect $\pi^\top$ which is an eigenvector for the corresponding Markov chain. This will tell us about eigenvector sensitivity as opposed to the above linear system sensitivity.

$$\left[\alpha\overline{\mathbf{P}} + (1-\alpha)\mathbf{e}\mathbf{v}^\top\right]\pi^\top = \pi^\top$$

The proximity of a simple eigenvalue to the other eigenvalues of a matrix increases the sensitivity of its corresponding eigenvector to minor modifications in the matrix. More precisely, this concept examines the closeness of the subdominant eigenvalue λ_2 of matrix $\mathbf{P}$ to the dominant eigenvalue $\lambda_1 = 1$ within a Markov chain context. As a result, as the value of α increases, the PageRank vector becomes increasingly sensitive.

Here we provide a more precise version by evaluating the derivative $d\pi^\top(\alpha)/d\alpha$ which reflects the rate of changing in $\pi^\top$ along the change in α. To show such derivative is well-defined, we need to use the following proposition:

Proposition 4.3. *The PageRank vector is given by*

$$\pi^\top(\alpha) = \frac{1}{\sum_{i=1}^{n} D_i(\alpha)} \left(D_1(\alpha), D_2(\alpha), \dots, D_n(\alpha)\right)$$

where $D_i(\alpha)$ is the ith principal minor determinant of order $n-1$ in $\mathbf{I} - \overline{\overline{\mathbf{P}}}(\alpha)$. Because each principal minor $D_i(\alpha) > 0$ is just a sum of products of numbers from $\mathbf{I} - \overline{\overline{\mathbf{P}}}(\alpha)$, it follows that each component in $\pi^\top(\alpha)$ is a differentiable function of α on the interval $(0,1)$.

Proof. See [13]. □

Therefore, we can prove an upper bound on the norm of the derivative.

Proposition 4.4. *If $\pi^\top(\alpha) = (\pi_1(\alpha), \pi_2(\alpha), \dots \pi_n(\alpha))$ is the PageRank vector, then $\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \leq \frac{1}{1-\alpha}$ for each $j = 1, 2, \dots, n$. and*

$$\left\| \frac{d\pi^\top(\alpha)}{d\alpha} \right\|_1 \leq \frac{2}{1-\alpha}.$$

Proof. First, note that $\pi^\top(\alpha)\mathbf{e} = 1$ so we must have

$$\frac{d\pi^\top(\alpha)}{d\alpha}\mathrm{e} = 0.$$

Using the definition of $\overline{\overline{\mathbf{P}}}$ from section 3.2, we write

$$\pi^\top(\alpha) = \pi^\top(\alpha)\left(\alpha\overline{\mathbf{P}} + (1-\alpha)\mathbf{e}\mathbf{v}^\top\right).$$

We can then use implicit differentiation to obtain the following equation:

$$\frac{d\pi^\top(\alpha)}{d\alpha}(\mathbf{I} - \alpha\overline{\mathbf{P}}) = \pi^\top(\alpha)\left(\overline{\mathbf{P}} - \mathbf{e}\mathbf{v}^\top\right).$$

We need to be careful that $\mathbf{I} - \alpha\overline{\mathbf{P}}(\alpha)$ is non-singular before we right multiply by the inverse of it. Using a similar proof as in Prop 4.1, we find that the inverse of $\mathbf{I} - \alpha\overline{\mathbf{P}}(\alpha)$ is well-defined, such that we have

$$\frac{d\pi^\top(\alpha)}{d\alpha} = \pi^\top(\alpha)\left(\overline{\mathbf{P}} - \mathbf{e}\mathbf{v}^\top\right)(\mathbf{I} - \alpha\overline{\mathbf{P}})^{-1}. \tag{5}$$

Now, for every real $\mathbf{x} \in \mathbf{e}^\perp$ (the orthogonal complement of span$\{\mathbf{e}\}$), and for all real vectors $\mathbf{y}_{n\times 1}$,

$$\left|\mathbf{x}^\top\mathbf{y}\right| \leq \frac{1}{2}|\mathbf{x}\|_1\left(y_{\max} - y_{\min}\right). \tag{6}$$

This is derived from Hölder's inequality using the fact that for real α,

$$\left|\mathbf{x}^\top\mathbf{y}\right| = \left\|\mathbf{x}^\top(\mathbf{y}-\alpha\mathbf{e})\right\| \leq \|\mathbf{x}\|_1 \|\mathbf{y}-\alpha\mathbf{e}\|_\infty ,$$

and $\min_\alpha \|\mathbf{y}-\alpha\mathbf{e}\|_\infty = \frac{1}{2}(y_{\max}-y_{\min})$, where the minimum is attained at the 'mid-point' $\alpha = \frac{1}{2}(y_{\max}+y_{\min})$ [13]. Going back to (5) and right multiplying by $\mathbf{e}_j$ we find that

$$\frac{d\pi_j(\alpha)}{d\alpha} = \pi^\top(\alpha)\left(\overline{\mathbf{P}}-\mathbf{e}\mathbf{v}^\top\right)(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\mathbf{e}_j,$$

where $\mathbf{e}_j$ is the jth standard basis vector (i.e., the jth column of $\mathbf{I}_{n\times n}$). Since $\pi^\top(\alpha)\left(\overline{\mathbf{P}}-\mathbf{e}\mathbf{v}^\top\right)\mathbf{e} = 0$, by setting

$$\mathbf{y} = (\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\mathbf{e}_j$$

we can substitute it into (6) to obtain

$$\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \leq \left\|\pi^\top(\alpha)\left(\overline{\mathbf{P}}-\mathbf{e}\mathbf{v}^\top\right)\right\|_1 \left(\frac{y_{\max}-y_{\min}}{2}\right).$$

Now we wish to find an upper bound of the right hand side. Notice that $\left\|\overline{\mathbf{P}}-\mathbf{e}\mathbf{v}^\top\right\|_1 \leq 2$, hence we deduce the inequality $\left\|\pi^\top(\alpha)\left(\overline{\mathbf{P}}-\mathbf{e}\mathbf{v}^\top\right)\right\|_1 \leq 2$, implying

$$\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \leq y_{\max}-y_{\min}. \tag{7}$$

Observe that

$$(\mathbf{I}-\alpha\overline{\mathbf{P}})\mathbf{e} = (1-\alpha)\mathbf{e}$$

which implies [13] that

$$(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\mathbf{e} = (1-\alpha)^{-1}\mathbf{e} \geq 0.$$

Using this, we conclude that 0 is a lower bound of y and also

$$y_{\max} \leq \max_{i,j}\left[(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\right]_{ij} \leq \left\|(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\right\|_\infty = \left\|(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\mathbf{e}\right\|_\infty = \frac{1}{1-\alpha}.$$

Substituting these values back into (7), we get the first inequality of the proposition. We can get the second inequality using (5) and the fact that

$$\left\|(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\right\|_\infty = \left\|(\mathbf{I}-\alpha\overline{\mathbf{P}})^{-1}\mathbf{e}\right\|_\infty = \frac{1}{1-\alpha}.$$

□

Proposition 4.4 implies that the sensitivity of the PageRank vector as a function of α depends on the size of $(1-\alpha)^{-1}$. As α becomes smaller, the effect of the network structure on the PageRank vector $\pi^\top$ is decreased while the effect of the teleportation vector $\mathbf{v}^\top$ is increased. Since we would like to take advantage of the underlying link structure, we would prefer to choose α close to 1 . However, if α is too close to 1, then the PageRank vector will be much more sensitive, and the rate of convergence will be slower.

We will later determine a value of α through data, but for the following implementation, we make the standard choice for PageRank of $\alpha = 0.85$.

5 Methods

5.1 Network Construction Overview

As previously mentioned, we construct a network with nodes representing genes, and directed edges representing the regulatory interactions between them. In addition, we wish to preferentially include genes relevant to cancer in our network.

The rationale behind constructing this network is to leverage PageRank for ranking genes by their importance, hypothesising that applying PageRank to a reduced network of the genes that are expressed differently in cancerous (tumour) cells will identify genes correlated with carcinogenesis (formation of cancer). This idea has been applied in previous studies with a level of success [5, 6].

However, there are many choices to be made in how to construct the network and what data to filter for, and we will justify our approach and compare it to existing methods wherever possible.

5.2 Data Selection and Processing

5.2.1 Nodes

To obtain the genes of our regulatory interaction network, we query gene expression data for samples of breast cancer tissue from the GSE15852 dataset [21] on the Gene Expression Omnibus (GEO), selected as breast cancer is relatively well-studied compared to other types of cancers. This dataset contained data on gene expression values in tumour tissue samples compared to adjacent normal tissues' expression values. The use of GEO for this purpose is relatively well-established [5].

Using R and the Bioconductor package, we extracted these data and computed the mean gene expression values in both tumour and normal tissue samples. In Python, we then averaged the data where there were multiple samples of the same gene, obtaining a table (DataFrame) where each of the rows corresponds to a unique gene and the expression values of that gene in tumour tissue and its equivalent normal tissue. Thus, we obtained a collection of genes related to breast cancer, represented as the nodes of the network. We then computed the absolute value of the average difference to obtain the *difference in expression values* of each gene, therefore associating a numerical value to each node.

For a given gene, a difference in gene expression values of zero could suggest that the gene is not relevant to cancer, while a high difference in gene expression values could suggest some relevance, although it is distinct from the *importance* of that gene, which we wish to obtain.

5.2.2 Edges

To obtain directed edges between the genes in the sample, lists of known regulatory interactions in humans were downloaded from two transcriptional regulatory network (TRN) databases, namely TRRUST v2 [22] and TFLink [23]. TFLink incorporates multiple datasets including TRRUST, allowing us to conveniently use TRRUST, as the smaller dataset, to test our code and TFLink to verify and produce our final results.

We additionally downloaded the protein-protein interaction (PPI) database STRING, which has the benefit of assigning a confidence score $w \in (0, 1000)$ to each interaction, based on evidence in the literature. Since genes encode for proteins and thus can be considered essentially equivalent in the context of these interaction networks, we decided that this would be a valid method for obtaining edge weights.

Our method differs from some existing methods which used confidence scores on its own to analyse an unweighted network [6]; as PageRank was designed around directed networks [1], we preferred databases which included the direction of interactions in order to leverage the unique properties of PageRank (as the motivational example in 3.1 would suggest, PageRank on an undirected graph is simply equivalent to degree centrality for $\alpha = 1$). Some existing methods instead use TRRUST/TFLink and construct edge weights as a function of the existing data [5].

We selected the interactions that were present in the expression data of the GSE15852 dataset to get the interactions between the previously obtained collection of genes (sample genes).

We then filtered the data from the TRRUST v2 and TFLink databases to only include the interactions that also appeared in STRING. We note that up to 90% of interactions are discarded in this step. However, this is not a disadvantage as requiring an interaction to be listed in multiple reputable sources should increase the quality of edge information. These figures are summarised in Table 1 - we note that our networks had 6658 and 418832 edges when constructed from TRRUST and TFLink respectively.

	Edges			
	Described	Between sample genes	*And* with STRING data	Directed?
TRRUST v2	9296	8826	6658	Yes
TFLink	6739357	4039635	418832	Yes
STRING	13715404	8166626	-	No

Table 1 Comparison of biological interaction databases

5.3 Computation

We constructed a weighted, directed NetworkX [24] graph in Python; our weighting scheme is an impactful choice and will be discussed in a later section. We first normalised the weights in order to make this graph stochastic, then obtained its transition matrix.

Using the linear system formulation from section 4.2, we were able to calculate the PageRanks of the nodes using algorithm 1. We verify our results by comparison to the built-in approximate PageRank function of NetworkX using the power method; the maximum difference in any entry of the pagerank vectors is 2.343e-05, which falls within the default tolerance of 1e-06 given by NetworkX.

5.4 Exploratory Data Analysis

5.4.1 Gene Differential Expressions

From a biological standpoint, a large difference in expression value of a certain gene (in normal vs tumour tissue) does not necessarily imply its importance in carcinogenesis. Nevertheless, there is still some importance to this quantity, and as such we wish to normalise this quantity for later use. We will first look at the distribution of the differences in gene expression values, which we will henceforth refer to as *'differences in expression'*. Consider ordering these by magnitude.

Figure 7 shows that the differences in expression are *not uniformly distributed*. Indeed, by taking a log-log plot in Figure 8, we see the first 10,000 differences in expression approximately follow a power law distribution.

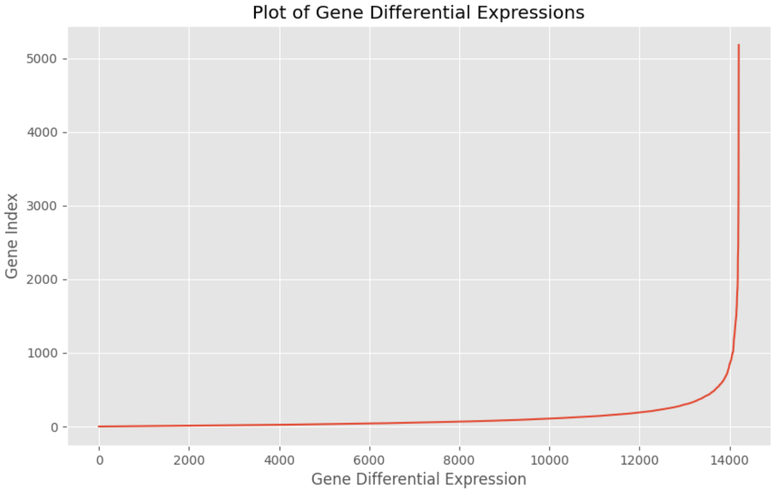

Figure 7 Plot of the differences in expression against the ordered index

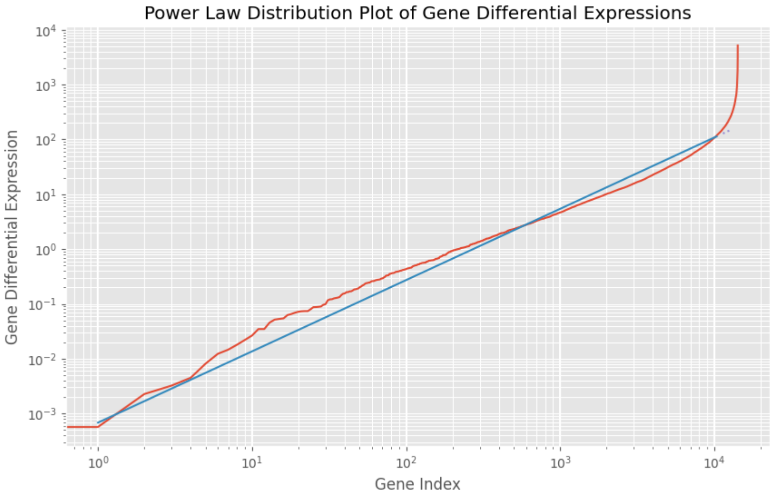

Figure 8 Log-log plot equivalent of Figure 7 with linear fit on $[0,\ 10^4]$

Truncating the data at 10,000 and using linear regression, we obtain a gradient of $1.31 \approx \frac{4}{3}$ (where we have used a rational approximation for computational reasons).

Suppose we would like to assign these genes re-scaled values in an approximately uniform fashion in the range $[0,\ M]$, for some $M > 0$. This is accomplished by taking the $\frac{3}{4}$th power of each difference in expression d, such that $\hat{d} := d^{\frac{3}{4}}$, for the first 10,000 genes. To deal with the extreme upper tail values, we assign each of these genes the maximum re-scaled value, M.

5.4.2 Edge Direction, Dangling Nodes and Self-Loops

In standard PageRank analysis, the PageRank value takes into account the quality and quantity of the *incoming* edges of each node as we consider nodes to be more important if they are linked to by many other nodes. However, in this setting, we are interested in the quality and quantity of the *outgoing* edges of each node, as biologically that represents finding the genes with the most and strongest effects on other genes, and thus will give us our cancer driver genes (CDGs). To this end, we have decided to simply *reverse the edge directions* as given by our databases, as the PageRank values obtained using this new graph should give us the required ranking.

We note that TFLink has no dangling nodes. There is a self-interacting gene in the network constructed from TRRUST (FOS → FOS in the dataset), which results in a self-loop in the network. As noted in section 3.5, this will result in FOS having a higher PageRank than if the self-loop did not exist. However, because this self-interaction could be important in a biological context, we decided to maintain the self-loop regardless.

5.5 Edge and Node Attributes

5.5.1 Initial Scheme

Recall that PageRank uses the weights of edges and the personalisation vector of the nodes. While weighting the edges of the network, we would like to consider:

- The confidence score $w \in (0, 1000)$ from STRING of the associated interaction, and
- The differences in expression x_1, x_2 of the two nodes the edge connects.

Our initial attempt was then to combine both these values by taking the product of the confidence score from STRING and the root mean square (RMS) of the two differences in expression:

$$\texttt{weight} = w \ \cdot \texttt{RMS}(x_1, x_2) \tag{8}$$

where $\texttt{RMS}(x_1, x_2) := \frac{1}{2}\sqrt{x_1^2 + x_2^2}$. In order to assess the performance of the model, we look at the rank of the gene TP53, which is the most commonly mutated cancer driver gene [25] and a well-documented tumour suppressor gene [26], hence has an association with almost all forms of cancer.

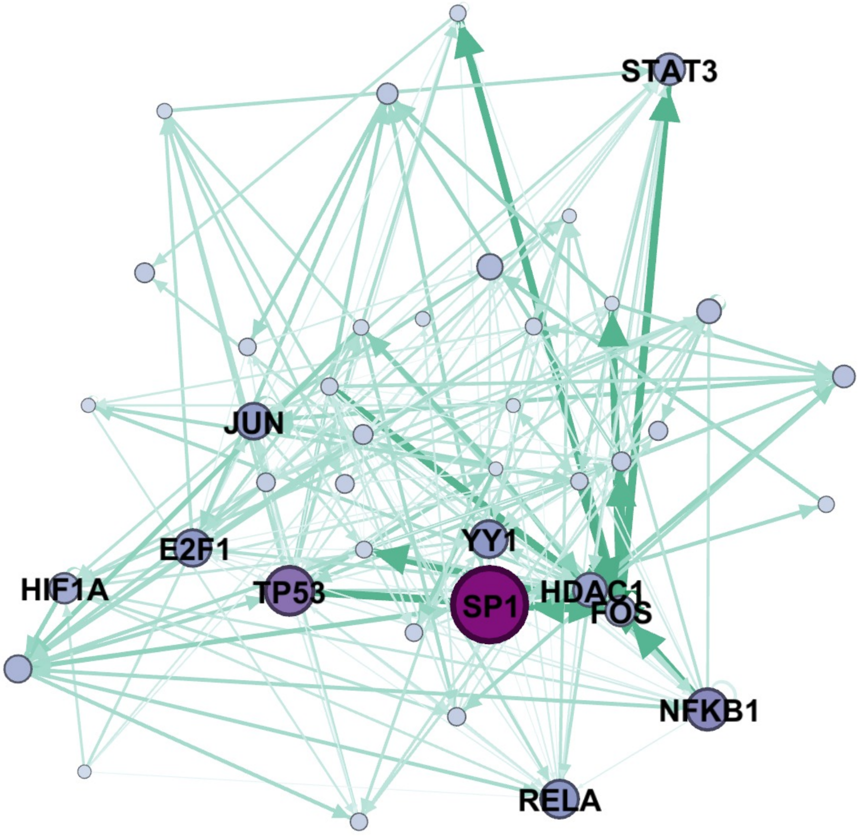

Figure 9 Subgraph of highest PageRank nodes. Node size and colour intensity corresponds to PageRank. Edge colour intensity and width correspond to weight, as described in (8).

However, this weighting yielded poor results, with TP53 receiving a low rank.

We hypothesise that this arbitrary method of edge weighting did not work as the confidence score from STRING and differential expression essentially contain different categories of information.

To illustrate, suppose we have a critically important chain of genes as in Figure 10, along with numerous less significant interactions to/from other genes not drawn. Suppose additionally that this chain passes through two genes with coincidentally small differences in expression (**B**, **C**). Then the edge between them will have a similarly small `RMS` component, and block PageRank from being passed along the critical chain (to **D**), regardless of the confidence values of these interactions which cannot exceed 1000.

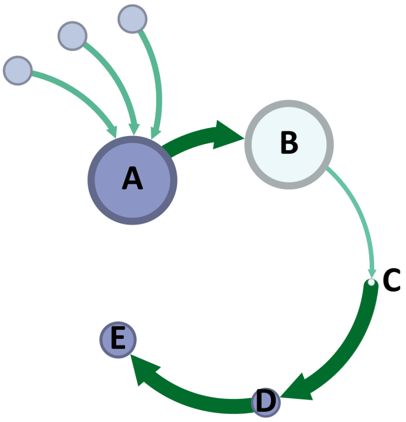

Figure 10 A constructed network with a PageRank source **A**, and genes with low difference in expression **B**, **C**, with a knock-on effect on the PageRanks of **D**, **E**.

5.5.2 Personalised PageRank Scheme

We finally bring our attention to the personalisation vector $\mathbf{v}$ used in PageRank, which we previously set to a uniform vector $\mathbf{e}$. Recognising that the differences in expression are values associated with each node, and that the entries of $\mathbf{v}$ are also values associated with each node, we instead assign edge weighting using STRING, and use confidence instead.

For $\alpha = 0.85$, this results in the network in Fig. 11.

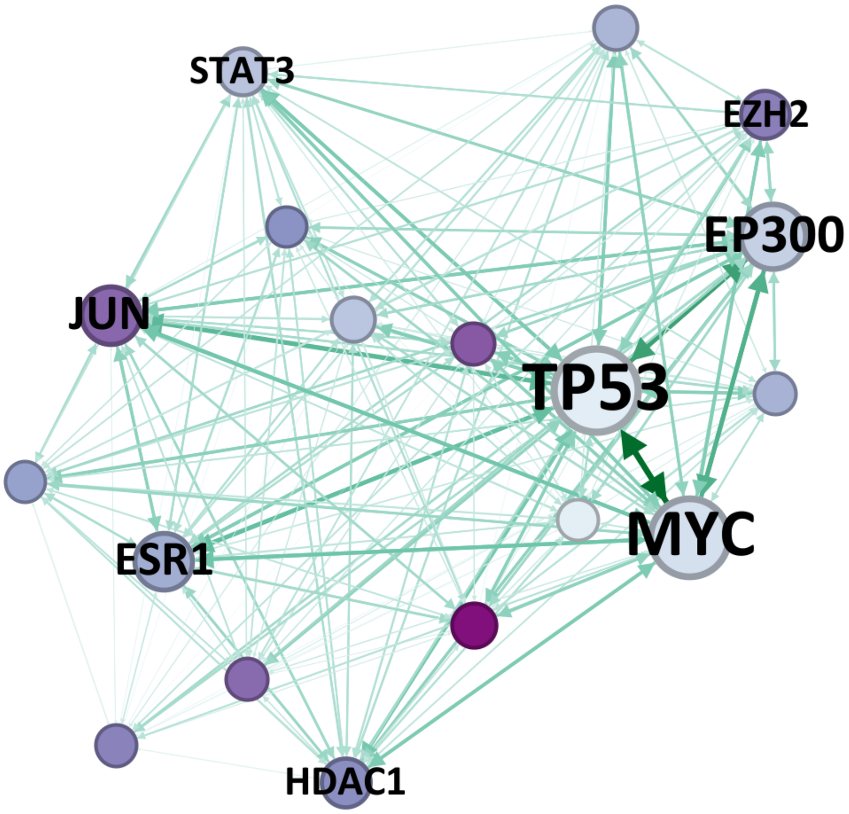

Figure 11 Subgraph of highest PageRank nodes. Node size corresponds to PageRank. Node colour intensity indicates personalised PageRank value. Edge colour intensity and width correspond to PageRank contribution.

6 Results

6.1 Robustness of PageRank

Robustness, in this context, refers to the consistency of the PageRank results when there is a change in α. A robust PageRank algorithm would exhibit minimal changes in rankings and PageRank values across different α values.

An important measure of robustness in rankings is Rank Biased Overlap (RBO), a similarity measure which can quantify the differences between incomplete and top-weighted rankings [27]. RBO score may vary from 0 to 1, where 1 indicates two rankings are identical and 0 indicates two rankings are completely distinct.

Conceptually, RBO considers the overlap between the top k elements of two lists, if k were chosen from a geometric probability distribution.

The concept of persistence in RBO is crucial because it determines the extent to which the similarity measure focuses on the agreement of items at the top of the ranked lists compared to the agreement throughout the entire lists. Persistence in RBO is controlled by a parameter, usually denoted by p, ranging from 0 to 1, and affects the rate of decay of each subsequent term's weighting. With p closer to 0, the series converges more quickly, making the measure focus on early ranks.

Given two infinite rankings S and T to depth d and the persistence p, the RBO is calculated with the following formula defined in [28]:

$$RBO(S, T, p) = (1 - p) \sum p^{d-1} \cdot A_d,$$

where $d = 1$ to ∞ is the depth of the ranking to be examined, $A_d = X_d/d$ is the agreement between S and T, i.e. the proportion of S and T that is overlapped, and $X_d = |S_{:d} \cap T_{:d}|$ is the size of overlap (intersection) between S and T up to depth d. The depth d can be considered as the top rankings we are interested in, i.e. to consider the differences in two rankings for the first 100 elements we simply set $d = 100$.

We compute the RBO for our specific PageRank rankings, with different values of α.

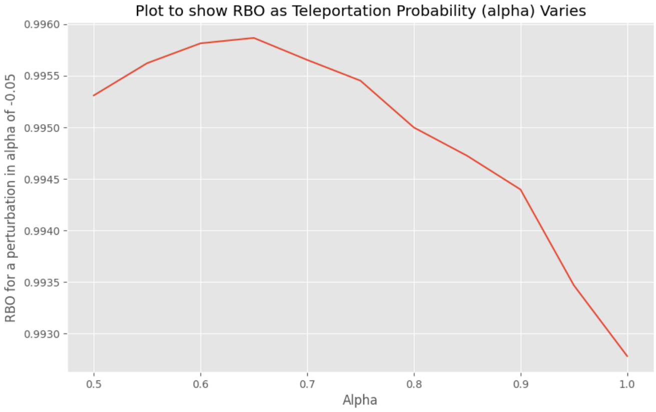

Figure 12 Robustness of PageRank.

We conclude that the PageRanks are most robust when $0.6 < \alpha < 0.65$, and therefore this provides a point of evidence to suggest that this may be a suitable range for α. This is in contrast to a small number of existing methods which have in some cases re-used the common choice of $\alpha = 0.85$ [1].

We also clearly identify that the rank order becomes more sensitive to perturbation as $\alpha \to 1$, confirming the results stated in [16].

7 Discussion

From the 11,165 genes in our final network, we chose to look at the top 50 PageRanked genes, and combed through existing literature to find their relevance to breast cancer. Indeed, out of our top 50, 42 of them have been identified as being involved in the development of various types of breast cancers.

Furthermore, 16 out of the top 50 genes are found in the highly selective Mut-driver verified gene list [29] (marked with *), defined as genes containing a sufficient number or type of driver gene mutations to unambiguously distinguish them from other genes. This source identifies 71 tumor suppressor genes and 54 oncogenes (cancer-causing mutation) out of 20,000 genes.

While the top TP53 gene (†) was used to verify and test our methods and its position is not surprising, the MYC gene (which is in fact a family of genes) contributes to the genesis of many human cancers [26] and therefore its second place ranking can be considered a success.

However, our list is not exhaustive; PageRank is not suited to pick up on all classes of cancer-related genes, and as such it misses an important gene, PIK3CA [25], which receives position 9434.

Two genes cited for their relevance specifically to breast cancer are BRCA1 and BRCA 2[26] (whose names stem from 'breast cancer'). While BRCA1 falls within the top 50 and takes position 33 (top 0.027%), BRCA2 takes position 735 (top 6.3%), and so for practical purposes this method may not be able to distinguish these genes.

Gene	Relevant?	Ref.	Score	Gene	Relevant?	Ref.	Score
TP53	YES*†	[26]	0.010276	CEBPB	YES	[48]	0.004484
MYC	YES	[26]	0.009606	TBP	NO		0.004484
EP300	YES*	[30]	0.007956	SMAD3	YES	[49]	0.004437
JUN	YES	[31]	0.007192	E2F1	YES	[50]	0.004412
ESR1	YES	[32]	0.007087	GATA2	YES	[51]	0.004352
HDAC1	YES	[33]	0.006140	KDM1A	YES	[52]	0.004307
EZH2	YES*	[34]	0.005981	CEBPA	YES*	[53]	0.004279
STAT3	YES	[35]	0.005876	BRCA1	YES*	[26]	0.004246
FOS	YES	[36]	0.005535	NR3C1	NO		0.004219
SMARCA4	YES*	[37]	0.005459	GATA1	YES*	[54]	0.004153
HDAC2	YES	[38]	0.005430	TRIM28	YES	[55]	0.004090
HIF1A	YES	[39]	0.005275	NCOR1	YES*	[26]	0.004053
PPARG	NO		0.005169	EGR1	YES	[56]	0.004029
CREB1	YES	[40]	0.005133	STAT1	YES	[57]	0.003998
CTCF	YES	[26]	0.005119	FOXA1	YES	[26]	0.003987
RUNX1	YES*	[26]	0.005118	KLF4	YES*	[58]	0.003968
NFKB1	YES	[41]	0.005052	SPI1	NO		0.003925
CREBBP	YES*	[42]	0.005048	SP1	YES	[59]	0.003912
BRD4	YES	[43]	0.004944	PTEN	YES*	[26]	0.003758
SOX2	YES	[44]	0.004762	NANOG	NO		0.003733
YY1	YES	[45]	0.004731	KDM6A	YES*	[60]	0.003697
POU5F1	NO		0.004706	SUZ12	NO		0.003678
AR	YES*	[46]	0.004698	MYCN	YES††	[26]	0.003650
RELA	YES	[47]	0.004662	FOXO1	NO		0.003604
GATA3	YES*	[26]	0.004486	SMAD2	YES*	[29]	0.003589

Table 2 Top 50 PageRanked genes, and their involvement in breast carcinogenesis

†† Although not explicitly mentioned in [26], MYCN (n-MYC) is a member of the MYC family, which is ranked in position 2.

7.1 Further Development

PageRank clearly has its limitations, for example problems with dangling nodes and self-loops. When considering random biological networks, these are a nuisance to deal with, and further data processing must be done before constructing the network. In future research, it may be a good idea to use another measure of centrality which does not have these problems, or possibly even several measures of centrality (and then aggregating the rankings), in order to obtain the ranked list.

8 Conclusion

In conclusion, we have shown that personalised PageRank is a mathematically viable algorithm for analysis of TRNs, and gives us satisfactory results for positive identification of cancer driver genes. It is clear that there is further work to be done in order to more accurately incorporate biological knowledge into the construction of the graph, but as a complementary analysis tool to other computational methods our procedure is adequate. Furthermore, as network-based biological analysis is a relatively new area of research, we do not currently know how best to model TRNs for use with centrality algorithms; as such, this approach and its justifications will hopefully open up new avenues for further exploration.

Acknowledgements

We would like to thank:
Prof. Darren Crowdy, Imperial College London, for offering his experience to supervise this project and for his advice.
Larissa Potapova, Imperial College London undergraduate, for help with prerequisite biology knowledge.

References

[1] Page L, Brin S, Motwani R, Winograd T. The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Library Technologies Project; 1998. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1768.

[2] Gleich DF. PageRank beyond the web. siam REVIEW. 2015;57(3):321-63.

[3] Mason O, Verwoerd M. Graph theory and networks in biology. IET Systems Biology. 2007.

[4] Vaquerizas JM, Kummerfeld SK, Teichmann SA, Luscombe NM. A census of human transcription factors: function, expression and evolution. Nature Reviews Genetics. 2009 Apr;10(4):252-63. Available from: https://www.nature.com/articles/nrg2538.

[5] Akhavan-Safar M, Teimourpour B, Kargari M. GenHITS: A network science approach to driver gene detection in human regulatory network using gene's influence evaluation. Journal of Biomedical Informatics. 2021;114:103661. Available from: https://www.sciencedirect.com/science/article/pii/S1532046420302896.

[6] Rahimi M, Teimourpour B, Akhavan-Safar M. DGRanker: cancer driver gene detection in human transcriptional regulatory network. Iranian Journal of Biotechnology. 2022 Apr;20(2):e3066. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9583818/.

[7] Alon U. An introduction to systems biology: design principles of biological circuits. 2nd ed. CRC Press; 2019.

[8] Gonzalez-Perez A, Lopez-Bigas N. Functional impact bias reveals cancer drivers. Nucleic acids research. 2012;40(21):e169-9.

[9] Zhang J, Wu LY, Zhang XS, Zhang S. Discovery of co-occurring driver pathways in cancer. BMC bioinformatics. 2014;15:1-14.

[10] Hou JP, Ma J. DawnRank: discovering personalized driver genes in cancer. Genome medicine. 2014;6:1-16.

[11] Bashashati A, Haffari G, Ding J, Ha G, Lui K, Rosner J, et al. DriverNet: uncovering the impact of somatic driver mutations on transcriptional networks in cancer. Genome biology. 2012;13:1-14.

[12] Wilensky U. NetLogo; 1999. Available from: http://ccl.northwestern.edu/netlogo/.

[13] Langville AN, Meyer CD. Deeper Inside PageRank. Internet Mathematics. 2004 January.

[14] Briggs SF. Network Science Lecture Notes; 2024. Lecture notes for MATH50007 Network Science at Imperial College London.

[15] Briggs SF. Network Science Problem Sheet 2 Solutions, Question 3; 2024. Problem Sheet for MATH50007 Network Science at Imperial College London.

[16] Berman A, Plemmons RJ. 6. M-Matrices. In: Nonnegative Matrices in the Mathematical Sciences. Society for Industrial and Applied Mathematics; 1994. p. 132-64. Available from: https://epubs.siam.org/doi/abs/10.1137/1.9781611971262.

[17] Haveliwala TH, Kamvar SD. The Second Eigenvalue of the Google Matrix. Stanford University; 2003.

[18] da Silva Barros Rodrigues Mendes Pinto IC. PageRank: How to accelerate its computation. Handle Proxy. 2019 January 14. Available from: https://hdl.handle.net/10216/118321.

[19] Stonedahl F, Wilensky U. NetLogo PageRank model; 2009. Available from: http://ccl.northwestern.edu/netlogo/models/PageRank.

[20] MacKay DJC. Information Theory, Inference and Learning Algorithms. Cambridge University Press; 2003. Available from: https://books.google.co.uk/books?id=AKuMj4PN_EMC.

[21] Bong IP. GEO accession viewer; 2009. Accessed on June 17, 2024. Available from: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE15852.

[22] Han H, Cho JW, Lee S, Yun A, Kim H, Bae D, et al. TRRUST v2: an expanded reference database of human and mouse transcriptional regulatory interactions. Nucleic Acids Research. 2018 Jan;46(D1):D380-6.

[23] Liska O, Bohár B, Hidas A, Korcsmáros T, Papp B, Fazekas D, et al. TFLink: an integrated gateway to access transcription factor–target gene interactions for multiple species. Database. 2022 09;2022:baac083. Available from: https://doi.org/10.1093/database/baac083.

[24] Hagberg A, Swart P, S Chult D. Exploring network structure, dynamics, and function using NetworkX. Los Alamos National Lab.(LANL), Los Alamos, NM (United States); 2008.

[25] Mendiratta G, Ke E, Aziz M, Liarakos D, Tong M, Stites EC. Cancer gene mutation frequencies for the U.S. population. Nature Communications. 2021 Oct;12(1).

[26] Rajendran BK, Deng CX. Characterization of potential driver mutations involved in human breast cancer by computational approaches. Oncotarget. 2017 Apr;8(30):50252-72. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5564847/.

[27] Webber W, Moffat A, Zobel J. A similarity measure for indefinite rankings. ACM Transactions on Information Systems. 2010 Nov;28(4):1–38.

[28] Sarica A, Quattrone A, Quattrone A. Introducing the Rank-Biased Overlap as Similarity Measure for Feature Importance in Explainable Machine Learning: A Case Study on Parkinson's Disease. In: Mahmud M, He J, Vassanelli S, van Zundert A, Zhong N, editors. Brain Informatics. Cham: Springer International Publishing; 2022. p. 129-39.

[29] Vogelstein B, Papadopoulos N, Velculescu VE, Zhou S, Diaz LA, Kinzler KW. Cancer genome landscapes. Science. 2013 Mar;339(6127):1546–1558.

[30] Asaduzzaman M, Constantinou S, Min H, Gallon J, Lin ML, Singh P, et al. Tumour suppressor EP300, a modulator of paclitaxel resistance and stemness, is downregulated in metaplastic breast cancer. Breast Cancer Research and Treatment. 2017 Jun;163(3):461-74.

[31] Lukey MJ, Greene KS, Erickson JW, Wilson KF, Cerione RA. The oncogenic transcription factor c-Jun regulates glutaminase expression and sensitizes cells to glutaminase-targeted therapy. Nature Communications. 2016 Apr;7:11321. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4837472/.

[32] Dustin D, Gu G, Fuqua SA. Esr1 mutations in breast cancer. Cancer. 2019 Nov;125(21):3714-28. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6788940/.

[33] Tang Z, Ding S, Huang H, Luo P, Qing B, Zhang S, et al. HDAC1 triggers the proliferation and migration of breast cancer cells via upregulation of interleukin-8. Biological Chemistry. 2017 Nov;398(12):1347-56.

[34] Gan Y, Lo Y, Makower D, Kleer C, Lu J, Fineberg S. Ezh2 protein expression in estrogen receptor positive invasive breast cancer treated with neoadjuvant endocrine therapy: an exploratory study of association with tumor response. Applied immunohistochemistry & molecular morphology : AIMM. 2022 Oct;30(9):614-22. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9577480/.

[35] Ma Jh, Qin L, Li X. Role of STAT3 signaling pathway in breast cancer. Cell Communication and Signaling : CCS. 2020 Feb;18:33. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7048131/.

[36] Chang D, Li L, Xu Z, Chen X. Targeting FOS attenuates malignant phenotypes of breast cancer: Evidence from in silico and in vitro studies. Journal of Biochemical and Molecular Toxicology. 2023 Jul;37(7):e23358.

[37] Kim J, Jang G, Sim SH, Park IH, Kim K, Park C. Smarca4 depletion induces cisplatin resistance by activating yap1-mediated epithelial-to-mesenchymal transition in triple-negative breast cancer. Cancers. 2021 Oct;13(21):5474. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8582548/.

[38] Jo H, Shim K, Kim HU, Jung HS, Jeoung D. HDAC2 as a target for developing anticancer drugs. Computational and Structural Biotechnology Journal. 2023 Mar;21:2048-57. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10030825/.

[39] Ebright RY, Zachariah MA, Micalizzi DS, Wittner BS, Niederhoffer KL, Nieman LT, et al. HIF1A signaling selectively supports proliferation of breast cancer in the brain. Nature Communications. 2020 Dec;11(1):6311. Available from: https://www.nature.com/articles/s41467-020-20144-w.

[40] Hu PC, Li K, Tian YH, Pan WT, Wang Y, Xu XL, et al. Creb1/lin28/mir-638/vasp interactive network drives the development of breast cancer. International Journal of Biological Sciences. 2019 Oct;15(12):2733-49. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6854368/.

[41] Concetti J, Wilson CL. Nfkb1 and cancer: friend or foe? Cells. 2018 Sep;7(9):133. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6162711/.

[42] Peck B, Bland P, Mavrommati I, Muirhead G, Cottom H, Wai PT, et al. 3D functional genomics screens identify CREBBP as a targetable driver in aggressive triple-negative breast cancer. Cancer research. 2021 Feb;81(4):847-59. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7611219/.

[43] Lu L, Chen Z, Lin X, Tian L, Su Q, An P, et al. Inhibition of BRD4 suppresses the malignancy of breast cancer cells via regulation of Snail. Cell Death and Differentiation. 2020 Jan;27(1):255-68.

[44] Liu P, Tang H, Song C, Wang J, Chen B, Huang X, et al. Sox2 promotes cell proliferation and metastasis in triple negative breast cancer. Frontiers in Pharmacology. 2018 Aug;9:942. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6110877/.

[45] Wan M, Huang W, Kute TE, Miller LD, Zhang Q, Hatcher H, et al. Yin yang 1 plays an essential role in breast cancer and negatively regulates p27. The American Journal of Pathology. 2012 May;180(5):2120-33. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3828504/.

[46] Anestis A, Zoi I, Papavassiliou AG, Karamouzis MV. Androgen receptor in breast cancer—clinical and preclinical research insights. Molecules. 2020 Jan;25(2):358. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7024330/.

[47] Kanzaki H, Chatterjee A, Hossein Nejad Ariani H, Zhang X, Chung S, Deng N, et al. Disabling the nuclear translocalization of rela/nf-κb by a small molecule inhibits triple-negative breast cancer growth. Breast Cancer : Targets and Therapy. 2021 Jul;13:419-30. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8275049/.

[48] Qi L, Sun B, Yang B, Lu S. CEBPB regulates the migration, invasion and EMT of breast cancer cells by inhibiting THBS2 expression and O-fucosylation. Human Molecular Genetics. 2023 May;32(11):1850-63.

[49] Singha PK, Pandeswara S, Geng H, Lan R, Venkatachalam MA, Dobi A, et al. Increased Smad3 and reduced Smad2 levels mediate the functional switch of TGF-β from growth suppressor to growth and metastasis promoter through TMEPAI/PMEPA1 in triple negative breast cancer. Genes & Cancer. 2019;10(5-6):134-49. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6872668/.

[50] Hollern DP, Swiatnicki MR, Rennhack JP, Misek SA, Matson BC, McAuliff A, et al. E2F1 drives breast cancer metastasis by regulating the target gene fgf13 and altering cell migration. Scientific Reports. 2019 Jul;9(1):10718. Available from: https://www.nature.com/articles/s41598-019-47218-0.

[51] Yu S, Jiang X, Li J, Li C, Guo M, Ye F, et al. Comprehensive analysis of the GATA transcription factor gene family in breast carcinoma using gene microarrays, online databases and integrated bioinformatics. Scientific Reports. 2019 Mar;9(1):4467. Available from: https://www.nature.com/articles/s41598-019-40811-3.

[52] Li L, Wang Y, Mou Y, Wu H, Qin Y. KDM1A identified as a potential oncogenic driver and prognostic biomarker via multi-omics analysis. The Canadian Journal of Infectious

Diseases & Medical Microbiology = Journal Canadien des Maladies Infectieuses et de la Microbiologie Médicale. 2021 Dec;2021:4668565. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8677413/.

[53] Wang JL, Ji WW, Huang AL, Liu Z, Chen DF. CEBPA restrains the malignant progression of breast cancer by prompting the transcription of socs2. Molecular Biotechnology. 2024 May.

[54] Li Y, Ke Q, Shao Y, Zhu G, Li Y, Geng N, et al. GATA1 induces epithelial-mesenchymal transition in breast cancer cells through PAK5 oncogenic signaling. Oncotarget. 2015 Jan;6(6):4345-56. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4414194/.

[55] Wei C, Cheng J, Zhou B, Zhu L, Khan MA, He T, et al. Tripartite motif containing 28 (Trim28) promotes breast cancer metastasis by stabilizing TWIST1 protein. Scientific Reports. 2016 Jul;6(1):29822. Available from: https://www.nature.com/articles/srep29822.

[56] Shajahan-Haq AN, Boca SM, Jin L, Bhuvaneshwar K, Gusev Y, Cheema AK, et al. EGR1 regulates cellular metabolism and survival in endocrine resistant breast cancer. Oncotarget. 2017 May;8(57):96865-84. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5722529/.

[57] Bailey SG, Cragg MS, Townsend PA. Role of STAT1 in the breast. JAK-STAT. 2012 Jul;1(3):197-9. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3670245/.

[58] Yu F, Li J, Chen H, Fu J, Ray S, Huang S, et al. Kruppel-like factor 4 (Klf4) is required for maintenance of breast cancer stem cells and for cell migration and invasion. Oncogene. 2011 May;30(18):2161-72. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3088782/.

[59] Gao Y, Gan K, Liu K, Xu B, Chen M. Sp1 expression and the clinicopathological features of tumors: a meta-analysis and bioinformatics analysis. Pathology and Oncology Research. 2021 Jan;27:581998. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8262197/.

[60] Hua C, Chen J, Li S, Zhou J, Fu J, Sun W, et al. KDM6 demethylases and their roles in human cancers. Frontiers in Oncology. 2021 Dec;11. Available from: https://www.frontiersin.org/journals/oncology/articles/10.3389/fonc.2021.779918/full.

A Appendix - Code

Network visualisations up to and including section 4 were produced in NetLogo: http://ccl.northwestern.edu/netlogo/. Network visualisations from section 5 onwards were produced in Gephi: https://gephi.org/. Graphs were produced in Matplotlib using the ggplot style.

The complete source code of this project is available on GitHub: https://github.com/ageneric/applied-pagerank.

A.1 Selected Routines

The PageRank algorithm, implemented in linear system form:

```
import networkx as nx
import numpy as np


P = np.array([[0.7, 0.7, 0. , 0.6],
              [0. , 0. , 0. , 0. ],
              [0.3, 0.3, 0. , 0.4],
              [0. , 0. , 0. , 0. ]]).transpose()


def permute_dangling_rows(P, v):
    # Find zero & non-zero rows
    non_dangling_rows = np.any(P, axis=1)  # 1d boolean array

    # Sort the rows / relabel nodes - row-wise
    P_1 = P[non_dangling_rows]

    # Column-wise
    P_11 = P_1[:, non_dangling_rows]
    P_12 = P_1[:, ~non_dangling_rows]

    # Permute v
    v_1 = v[non_dangling_rows]
    v_2 = v[~non_dangling_rows]
    return P_11, P_12, v_1, v_2, non_dangling_rows


def linear_system_pagerank(P, v=None, alpha=0.85):
    node_count = P.shape[0]

    # Set the default choice of v - uniform probability per outbound edge
    if v is None:
        v = np.array([1/node_count for _ in range(node_count)])

    P_11, P_12, v_1, v_2, non_dangling_rows = permute_dangling_rows(P, v)
    permutation = np.arange(node_count)[non_dangling_rows]
    counter_permutation = np.arange(node_count)[~non_dangling_rows]

    # Set up the system of linear equations for PageRank
    I = np.identity(P_11.shape[0])
    X = (I - alpha*P_11)
    # Solving pi_1^T X = v_1^T <= or equivalently => X^T pi_1 = v_1
    pi_1 = np.linalg.solve(X.transpose(), v_1).transpose()

    pi_2 = alpha * pi_1.transpose() @ P_12 + v_2.transpose()

    # Invert the permutation
    ret = np.empty(node_count)
    ret[permutation] = pi_1
    ret[counter_permutation] = pi_2

    # Normalise using the 1-norm as standard
    return ret / np.linalg.norm(ret, 1)


def get_personalisation_vector_by_deg(nodes, gene_deg):
    base = np.array([min(gene_deg[node], 298.45) ** 0.75 for node in nodes])
    v_unit_vector = base / np.linalg.norm(base, 1)
    return v_unit_vector
```

Graph construction:

```
import pandas
import pandas as pd
import networkx as nx
from matplotlib import pyplot as plt

from gene_data import get_gene_differential_expressions
from weighting import WeightVectorMethod, expressions, get_TRRUST_subset, \
    get_TFLink_subset, get_STRING_subset, TF, TARGET
from pagerank import linear_system_pagerank, get_personalisation_vector_by_deg, \
    format_pagerank
from graphic import draw_network_pagerank


# filter for the top differentially expressed genes
# 6.9 = keep top ~90%. 14.6 = keep top ~80%.
# don't recommend dropping more than 20% as TP53 (important) has low expression
THRESHOLD_DEG = -1


def unzip(iterable):
    return zip(*iterable)

def generate_networkx_graph(_df, _weighting):
    network = nx.DiGraph()

    # Since few genes have out-links, we filter down to only the genes with
    # out-links and iterate through them. This is faster than enumerating
    # through all genes by a significant margin.
    genes_with_out_links = _df[TF].unique()
    num_genes_with_out_links = len(genes_with_out_links)

    for i, gene in enumerate(genes_with_out_links):
        neighbours = _df[_df[TF] == gene]
        if not neighbours.empty:
            weights = _weighting(gene, neighbours)
            network.add_weighted_edges_from((neighbour, gene, weight)
                                            for neighbour, weight in zip(neighbours[TARGET],
        if i % 100 == 99:
            print(f'Generating NetworkX graph: {i/num_genes_with_out_links:.2%} complete')

    return network


if __name__ == '__main__':
    print('Imported modules.')

    gene_deg = get_gene_differential_expressions(expressions)
    filter_list = [g for (g, differential) in gene_deg.items()
                   if abs(differential) > THRESHOLD_DEG]
    tflink_df = get_TFLink_subset(filter_list)
    string_df = get_STRING_subset(filter_list)
    print('Imported datasets.')

    df = tflink_df.merge(string_df, on=(TARGET, TF))
    weighting = WeightVectorMethod(gene_deg, df).STRING
    print('Merged datasets.')

    print(f'''Genes with known expressions               {len(gene_deg)}
Genes passing threshold expression         {len(filter_list)}
 -- and their interactions in TFLink       {len(tflink_df)}
 -- and their interactions in STRING       {len(string_df)}
Interactions in TFLink + STRING merged     {len(df)}
Weighting method                           {weighting.__name__}''')

    # Free up memory from the separate datasets
    del tflink_df
    del string_df

    network = generate_networkx_graph(df, weighting)
    print('Generated NetworkX graph.')

    stochastic_network = nx.stochastic_graph(network)
    matrix_P = nx.adjacency_matrix(stochastic_network, stochastic_network.nodes,
                                   weight='weight').todense()
    personalisation = get_personalisation_vector_by_deg(network.nodes, gene_deg)
    print('Formulated matrix problem.')

    pagerank = linear_system_pagerank(matrix_P, v=personalisation, alpha=0.85)
    print('Computed PageRanks.')

    pagerank_dict, top_genes = format_pagerank(pagerank, network)
    top_gene_names, top_pageranks = unzip(top_genes)

    # Visualise 'top 50' genes
    draw_network_pagerank(network, pagerank_dict, top_gene_names, top_pageranks)
```

Miscellaneous routines used to generate the graphics or statistics:

```
"""1. graph of differential expression values of top genes"""
plt.plot([gene_deg[g] for g in top_gene_names[:1500]])

"""2. linear system pagerank is close to networkx pagerank"""
pagerank_dict2 = nx.pagerank(stochastic_network)

np.mean(np.array(list(pagerank_dict.values())) - np.array(list(pagerank_dict2.values())))
Out[100]: 1.0458723819493994e-20
np.max(np.array(list(pagerank_dict.values())) - np.array(list(pagerank_dict2.values())))
Out[101]: 2.343220024556003e-05

"""3. number of selfloops"""
nx.number_of_selfloops(stochastic_network)

"""4. the Nth lowest gene differential expression value"""
sorted(gene_deg.values())[N]
```

```
sum(v < 18.664 for v in gene_deg.values())

"""5. write dataframe to file"""
df.write_csv()

"""6. pageranks of a network for different alpha"""
sorted([v for k, v in nx.pagerank(network).items()], key=lambda x: -x)

"""7. Power Law distribution for degs plot"""
# Sort the gene_deg values
sorted_values = sorted(gene_deg.values())

# Create the plot with a logarithmic y-axis
plt.figure(figsize=(10, 6))
plt.yscale('log'); plt.xscale('log')
plt.xlabel('Gene Index')
plt.ylabel('Gene Differential Expression')
plt.title('Power Law Distribution Plot of Gene Differential Expressions')

plt.plot(sorted_values)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.show()

# 1.337
# coefficients = np.polyfit(np.log(np.array(range(1, len(gene_deg)+1))),
    np.log([g for g in gene_deg.values()]), 1)

"""8. Side by side list of genes"""
list(zip(top_genes, top_genes2))
len(nx.cycle_basis(nx.Graph(network)))

"""9. graph of node degrees"""
plt.plot(sorted([val for (node, val) in G.degree()]))

"""Code used to generate plots"""
import networkx as nx
import numpy as np
from matplotlib import pyplot as plt


plt.style.use('ggplot')


def draw_network_pagerank(network, pagerank_dict, top_gene_names, top_pageranks, top_n=40):
    view_network = nx.subgraph(network, top_gene_names[:top_n])
    scale = 900 / max(0.01, top_pageranks[0])
    sizes = np.array([pagerank_dict[node] for node in view_network.nodes]) * scale
    nx.draw(view_network, node_size=sizes, with_labels=True)
    plt.show()

def draw_power_law(sorted_values, figure_mode=0, text_values=None, tex=False):
    """0 - Standard plot; 1 - Truncated plot with x^1.333 line"""
    if text_values is None:
        xl = 'Gene Index'
        yl = 'Gene Differential Expression'
        title = 'Power Law Distribution Plot of Gene Differential Expressions'
    else:
        xl, yl, title = text_values

    plt.figure(figsize=(10, 6))
    plt.yscale('log')
    plt.xscale('log')
    plt.xlabel(xl)
    plt.ylabel(yl)
    plt.title(title)

    if figure_mode % 5:
        for i in range(0, 5):
            line = plt.axvline(x=10**i, linestyle='-')
            line.set_color("white")

    if figure_mode % 2:
        sorted_values = sorted_values[:10000]

    plt.plot(sorted_values)

    if figure_mode % 3:
        space = 1.1 ** np.arange(0, 98)
        plt.plot(space, space ** 1.32 * sorted_values[0] * 1.2)
        cont_space = 1.1 ** np.arange(98, 100)
        plt.plot(cont_space, cont_space ** 1.32 * sorted_values[0] * 1.2,
                 linestyle='dotted')

    plt.grid(True, which='both')
    plt.show()

def draw_gene_deg_plot(gene_deg, top_gene_names):
    plt.plot([gene_deg[g] for g in top_gene_names[:1500]])
    plt.show()
```

```
def draw_sorted_gene_deg_plot(gene_deg):
    plt.figure(figsize=(10, 6))

    plt.xlabel('Gene Differential Expression')
    plt.ylabel('Gene Index')
    plt.title('Plot of Gene Differential Expressions')
    plt.grid(True, which='both')
    space = 1.1 ** np.arange(0, 98)
    plt.plot(space, (space ** 1.33) / 1000)
    plt.plot(sorted(gene_deg.values()))
    plt.show()

def draw_pagerank_hist(pageranks):
    plt.hist(sorted(pageranks, key=lambda x: -x), bins=100)
    plt.xlabel('PageRank')
    plt.ylabel('Frequency Density')
    plt.title('Histogram of Gene Differential Expressions (PageRank < 0.0001)')
    plt.show()



def write_to_graphml(network, pagerank_dict, gene_deg, name='graph'):
    import pathlib

    nx.set_node_attributes(network, gene_deg, name='differential_expression')
    nx.set_node_attributes(network, pagerank_dict, name='pagerank')

    attrs = {}
    ws = nx.get_edge_attributes(network, 'weight')
    ws3 = {k: v*v for k, v in ws.items()}

    for i, edge in enumerate(network.edges):
        attrs[edge] = 1000 * pagerank_dict[edge[0]] * pagerank_dict[edge[1]] * ws3[edge]
    nx.set_edge_attributes(network, attrs, 'pagerank_weight')

    nx.set_edge_attributes(network, ws3, 'rescaled_weight')
    nx.write_graphml(network, pathlib.Path.cwd() / (name + '.graphml'))


def alpha_plot(alphas, values, text_values=None, tex=False):
    if text_values is None:
        xl = 'Alpha'
        yl = 'K-L Divergence with alpha = 0.85'
        title = 'K-L Divergence as Teleportation Probability (alpha) Varies'
    else:
```

```
    xl, yl, title = text_values

plt.figure(figsize=(10, 6))
plt.ylim([-0.05, 0.5])
plt.xlabel(xl)
plt.ylabel(yl)
plt.title(title)

plt.plot(alphas, values)
```