arXiv:2002.12309v1 [cs.SI] 27 Feb 2020

Node Immunization with Non-backtracking Eigenvalues

Leo Torres
leo@leotrs.com
Network Science Institute,
Northeastern University

Kevin S. Chan
kevin.s.chan.civ@mail.mil
U.S. Army Research Lab

Hanghang Tong
htong@illinois.edu
Department of Computer Science,
University of Illinois at Urbana-Champaign

Tina Eliassi-Rad
tina@eliassi.org
Network Science Institute and
Khoury College of Computer Sciences,
Northeastern University

Abstract

The *non-backtracking matrix* and its eigenvalues have many applications in network science and graph mining, such as node and edge centrality, community detection, length spectrum theory, graph distance, and epidemic and percolation thresholds. Moreover, in network epidemiology, the reciprocal of the largest eigenvalue of the non-backtracking matrix is a good approximation for the epidemic threshold of certain network dynamics. In this work, we develop techniques that identify which nodes have the largest impact on the leading non-backtracking eigenvalue. We do so by studying the behavior of the spectrum of the non-backtracking matrix after a node is removed from the graph. From this analysis we derive two new centrality measures: *X-degree* and *X-non-backtracking centrality.* We perform extensive experimentation with targeted immunization strategies derived from these two centrality measures. Our spectral analysis and centrality measures can be broadly applied, and will be of interest to both theorists and practitioners alike.

Keywords. Non-backtracking matrix, epidemic threshold, perturbation analysis

1 Introduction

A *non-backtracking walk* in a graph is a sequence of pairwise adjacent edges such that no edge is traversed twice in succession, i.e., the walk does not contain *backtracks.* Non-backtracking walks are known to mix faster than standard random walks [1], whereas *non-backtracking cycles* (i.e. closed non-backtracking walks) contain important topological information from the so-called *length spectrum* of the graph [2]. The associated *non-backtracking matrix* is the unnormalized transition matrix of a random walker that does not trace backtracks, and it has many applications such as community detection [3, 4], influencer identification [5, 6], graph distance [2, 7], etc. Additionally, the *non-backtracking centrality* of nodes, defined in terms of the principal eigenvector of the non-backtracking matrix, has more desirable properties than the standard eigenvector centrality [8]. The non-backtracking framework has also been adapted to directed networks [9], weighted networks [10], and multi-layer networks [11]. In this paper, to avoid repetition we use the prefix "NB" to mean non-backtracking. For example, we refer to the non-backtracking matrix as the NB-matrix.

The eigenvalues of the NB-matrix (or NB-eigenvalues for short) are related to certain kinds of spreading dynamics. Karrer et al. [12] and Hamilton and Pryadko [13] showed that the percolation threshold is approximated by the inverse of the largest NB-eigenvalue λ_1. This implies that the epidemic threshold of susceptible-infectious-recovered (SIR) dynamics can also be approximated by λ_1 [14, 15]. Furthermore, Shrestha et al. [16] argued the same for susceptible-infectious-susceptible (SIS) dynamics, though Castellano and Pastor-Satorras [17] highlight that this may only hold for networks with certain amounts of degree heterogeneity, and propose a fully non-backtracking version of SIS dynamics where the NB-walks also play a large role. Whether one is talking about the percolation threshold or the epidemic threshold on SIR or SIS dynamics, λ_1 of the NB-matrix provides a better approximation to the true epidemic threshold than the largest eigenvalue of the adjacency matrix [16, 12].

Given the importance of the largest NB-eigenvalue in network dynamics, we ask: **which nodes have the largest influence on the largest NB-eigenvalue?** In the cases of SIR and SIS dynamics, answering this question will lead to targeted immunization strategies, as it is equivalent to asking which are the nodes whose removal from the network causes the epidemic threshold to increase the most. In the case of percolation, this is equivalent to determining which nodes' removal will put the network closer to splitting into many connected components. Operationally, we frame this question as follows.

Problem 1. Consider a graph G with largest NB-eigenvalue λ_1. Given an arbitrary node c, define $\lambda_1(c)$ as the largest NB-eigenvalue of the network after removing c. Define $\lambda_1 - \lambda_1(c)$ as the *eigen-drop induced by* c. **Which node c induces the maximum eigen-drop?**

The contributions of the present work are as follows:

- We develop the spectral theory of the NB-matrix to study the behavior of its eigenvalues under the removal of a node.

- For Problem 1, we propose two new centrality measures, *X-degree* and *X-non-backtracking (or X-NB) centrality.* Further, X-degree can be computed in approximately log-linear time in the number of nodes.
- Our experiments show that immunization strategies induced by X-degree and X-NB centrality are more effective than other methods.

In Section 2 we present the necessary background theory. In Section 3 we develop a spectral perturbation theory of the NB-matrix. We use this theory in Section 4 to introduce two new centrality measures and argue why they are effective at identifying nodes with largest eigen-drops. In Section 5 we review previous studies related to the present work. In Section 6 we provide experimental evidence for our claims. We conclude the paper in Section 7.

2 Background

Let G be a simple undirected graph with node set V and edge set E. We consider the set of directed edges $\overline{E}$ where each undirected edge $(i, j) \in E$ gives rise to two directed edges $i \to j \in \overline{E}$ and $j \to i \in \overline{E}$. A *walk* in G is a sequence of directed edges $i_1 \to j_1, \ldots, i_k \to j_k$, where $j_r = i_{r+1}$ for each $r = 1, \ldots, k-1$. Here, k is the *length* of the walk. A walk is *closed* if $j_k = i_1$. A *backtrack* is a walk of length 2 of the form $i \to j, j \to i$. A walk is a *non-backtracking walk*, or *NB-walk*, if no two consecutive edges in it form a backtrack. The *non-backtracking matrix*, or *NB-matrix*, B is the unnormalized transition matrix of a walker that does not perform backtracks. Concretely, B is indexed in the rows and columns by elements of $\overline{E}$. Let $m = |E|$, then B is of size $2m \times 2m$, and it is defined by

$$B_{k\to l, i\to j} = \delta_{jk} \left(1 - \delta_{il}\right), \tag{1}$$

where δ is the Kronecker delta. In words, $B_{k\to l,i\to j}$ is 1 iff $j = k$ and $i \to j, j \to l$ is not a backtrack. Notably, the powers of B count the number of NB-walks in G, i.e. $(B^r)_{k\to l, i\to j}$ is the number of NB-walks that start with $i \to j$ and end with $k \to l$ of length $r + 1$.

Among other applications, the NB-matrix has been used to define a notion of node centrality that has more desirable properties than the usual eigenvector centrality [8, 18]. Concretely, let λ be the largest eigenvalue of B and let $\mathbf{v}$ be the corresponding unit right eigenvector. By Perron-Frobenius theory, λ is positive, real, and has multiplicity one, while $\mathbf{v}$ can be chosen to be non-negative. The *non-backtracking centrality* of a node i is defined as

$$\mathbf{v}^i = \sum_j a_{ij} \mathbf{v}_{j\to i}, \tag{2}$$

where $A = (a_{ij})$ is the adjacency matrix of G. Now let D be the diagonal matrix with the degree of each node, and let $\mathbf{v}_{aux}$ be the left principal eigenvector of

$$B_{aux} = \begin{pmatrix} 0 & D - I_n \\ -I_n & A \end{pmatrix}, \tag{3}$$

where I_r is the identity matrix of size r. We have that $\mathbf{v}_{aux} = (\mathbf{f}, -\lambda\mathbf{f})$, where $\mathbf{f}$ is of size n, and it is known that $\mathbf{f}$ is parallel to the vector of NB-centralities, $\mathbf{f}^i \propto \mathbf{v}^i$ [8]. It is more efficient to use B_{aux} than B when computing the NB-centrality, since the former matrix is of size $2n \times 2n$, where $n = |V|$.

The NB-matrix is not symmetric and therefore its eigenvalues, other than the largest one, can be complex numbers. Even so, it contains a subtle structure, sometimes called PT-symmetry [3]. Indeed, let P be the matrix such that $P\mathbf{x}_{i\to j} = \mathbf{x}_{j\to i}$ for any vector $\mathbf{x}$ indexed by $\overline{E}$. It is readily checked that (i) the product PB is symmetric, and (ii) there exists a basis where P can be written as

$$P = \begin{pmatrix} 0 & I_m \\ I_m & 0 \end{pmatrix}. \tag{4}$$

3 Non-backtracking eigenvalues under node removal

We are interested in the behavior of the NB-eigenvalues when we remove a node from G. Suppose the target node we want to remove is $c \in V$, and partition the edges in $\overline{E}$ as those that are incident to c and those that are not. Sort the rows and columns of B accordingly, so that it takes the block form

$$B = \begin{pmatrix} B' & D \\ E & F \end{pmatrix}, \tag{5}$$

as shown in Figure 1. Here, B' is the NB-matrix of the graph after node c is removed, while F is the NB-matrix of the star graph centered at c; if d is the degree of c, then F is of size $2d \times 2d$. Further, D is indexed in the rows by directed edges not incident to c, and in the columns by directed edges incident to c, and vice versa for E.

3.1 The characteristic polynomial

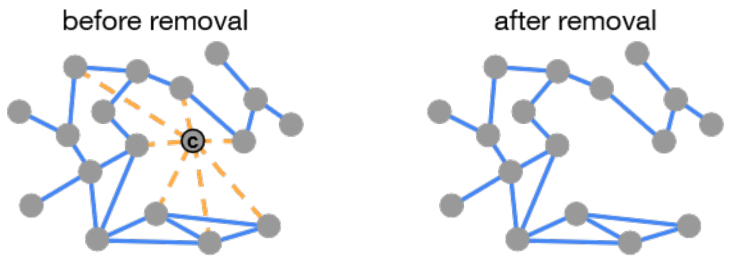
The NB-eigenvalues are the roots of the characteristic polynomial $\det(B - tI)$. The theory of Schur complements gives us an identity for the determinant of a block matrix,

$$\det(B - tI) = \begin{vmatrix} B' - tI & D \\ E & F - tI \end{vmatrix} \tag{6}$$

$$= \det(F - tI)\det\left(B' - tI - D(F - tI)^{-1}E\right), \tag{7}$$

where the size of I is given by context. This formula holds whenever $(F - tI)$ is invertible, i.e., whenever t is not an eigenvalue of F. To simplify this expression, we make the following observations.

Lemma 3.1. Let d be the degree of target node c. With D, E, F as in Equation (5), we have $DE = 0$ and $F^2 = 0$. Therefore, F is nilpotent, that is,



$$B = \left(\begin{array}{c|c} B' & D \\ \hline E & F \end{array}\right) \begin{array}{l} \}\, 2m-2d \\ \}\, 2d \end{array} \qquad \left(\; B' \; \right)$$

Figure 1: *Top:* Graph G with target node c before and after removal. G has m edges and c has degree d. Dashed yellow edges are incident to c, all other edges in solid blue. *Bottom:* Corresponding NB-matrices before and after removal.

all its eigenvalues are zero, and hence $\det(F - tI) = t^{2d}$. Finally, we have $(F - tI)^{-1} = -(F + tI)/t^2$ when $t \neq 0$.

Proof. Since D, E, F are sub-matrices of B, their element is given by Equation (1). Hence, computing DE and F^2 is straightforward, as long as care is placed in keeping track of the appropriate indices for the rows and columns of the involved matrices. Now, $F^2 = 0$ implies that all its eigenvalues are zero and that $\det(F - tI) = t^{2d}$. Finally, one can manually check that $(F - tI)(F + tI) = -t^2 I$. □

Now define $X = DFE$. One can manually check that

$$X_{k\to l, i\to j} = a_{ck} a_{cj} (1 - \delta_{kj}). \tag{8}$$

Per the Lemma, Equation (7) holds for $t \neq 0$ and

$$\det(B - tI) = t^{2d} \det\left(B' - tI + \frac{DFE}{t^2} + \frac{DE}{t}\right) \tag{9}$$

$$= t^{2d} \det\left(B' - tI + \frac{X}{t^2}\right). \tag{10}$$

Theorem 3.2. For a graph G and target node c, suppose the NB-matrix of G is B and the NB-matrix after removing c is B', and let X be as in Equation (10). If t is not an eigenvalue of B' then we have

$$\frac{\det(B - tI)}{\det(B' - tI)} = t^{2d} \det\left(I + \frac{1}{t^2}(B' - tI)^{-1} X\right). \tag{11}$$

Proof. Immediate from (10) by factoring out $B' - tI$. □

If c has degree 1, then X equals the zero matrix, and Equation (11) simplifies to show that removing c has no influence on the non-zero NB-eigenvalues. There are other nodes whose removal do not influence the non-zero NB-eigenvalues, which are characterized as follows. Let the 2-*core* of G be the graph that remains after iteratively removing nodes of degree 1. Let the 1-*shell* of G be the graph induced by the nodes outside of the 2-core.

Corollary 1. If c is in the 1-shell of G, removing it does not change the non-zero NB-eigenvalues.

Proof. If c has degree 1, Equation (8) gives $X = 0$. In this case, (11) becomes $\det(B - tI) = t^{2d} \det(B' - tI)$, which implies the assertion. In general, if c is in the 1-shell, then it must have degree 1 after iteratively removing some sequence of nodes each of which has degree 1 at the time of removal. Each of these removals has no effect on the non-zero eigenvalues, and therefore neither does the removal of c. □

Remark 1. Intuitively, since F is the NB-matrix of a star graph, which contains no NB-walks of length 3 or more, then immediately we have $F^2 = 0$. Following Figure 1, F^2 counts the number of NB-walks of length 3 whose edges are yellow-yellow-yellow, of which there are none. Similarly, DE counts the NB-walks whose edges are blue-yellow-blue, which also do not exist. Finally, $X = DFE$ counts the NB-walks of color blue-yellow-yellow-blue, which are precisely those that are destroyed when removing c. It is then no surprise that the rest of our analysis pivots fundamentally on the matrix X.

3.2 The largest eigenvalue

We now pivot to study the eigen-drop induced by removing c. The larger this eigen-drop, the more influential the target node is in determining the epidemic or percolation thresholds.

Theorem 3.3. With the same assumptions as in Theorem 3.2, let λ_1 be the largest eigenvalue of B and let $\mathbf{w}$ be a vector such that in Equation (10) we have

$$\left(B' - \lambda_1 I + \frac{X}{\lambda_1^2} \right) \mathbf{w} = 0. \tag{12}$$

Suppose $\{\mathbf{v}_i\}$ is a basis of right eigenvectors of B' and write $\mathbf{w}$ in this basis, $\mathbf{w} = \sum_i w_i \mathbf{v}_i$. Let $\mathbf{u}_1$ be the left eigenvector of B' corresponding to $\mathbf{v}_1$, and set $\alpha_i = \mathbf{u}_1^T X \mathbf{v}_i$. Finally, let λ_1' be the largest eigenvalue of B', so that the eigen-drop induced by c is $\lambda_1 - \lambda_1'$. Then, we have

$$\lambda_1 - \lambda_1' = \frac{1}{\lambda_1^2} \sum_i \frac{w_i}{w_1} \alpha_i. \tag{13}$$

Proof. If $\mathbf{v}_i$ corresponds to the eigenvalue λ'_i, then (12) gives

$$\sum_i w_i \left(B' - \lambda_1 I + \frac{X}{\lambda_1^2} \right) \mathbf{v}_i = \sum_i w_i \left(\lambda'_i \mathbf{v}_i - \lambda_1 \mathbf{v}_i + \frac{X \mathbf{v}_i}{\lambda_1^2} \right) = 0. \tag{14}$$

Let $\mathbf{u}_1$ be the left eigenvector corresponding to $\mathbf{v}_1$ normalized such that $\mathbf{u}_1^T \mathbf{v}_1 = 1$. Recall that $\mathbf{u}_1$ is orthogonal to every right eigenvector corresponding to a different eigenvalue. Since λ'_1 has multiplicity one, we have $\mathbf{u}_1^T \mathbf{v}_i = 0$ for each $i \neq 1$. Multiply by $\mathbf{u}_1$ on the left to get

$$w_1 \left(\lambda'_1 - \lambda_1 \right) + \sum_i w_i \frac{\mathbf{u}_1^T X \mathbf{v}_i}{\lambda_1^2} = 0. \tag{15}$$

Define $\alpha_i = \mathbf{u}_1^T X \mathbf{v}_i$ and rearrange to get Equation (13). □

Remark 2. We can reverse our argument and interpret (13) in terms of node addition rather than removal. Suppose the original graph does not contain c, and therefore its NB-matrix is B'. Then, the NB-matrix *after adding node c* is given by (5). All our arguments are valid in this setting, and (13) then says that the new largest NB-eigenvalue is the solution to a third-degree polynomial, the coefficients of which depend on the full eigendecomposition of B'.

3.2.1 An approximation

Unfortunately, Equation (13) requires knowledge of all eigenvectors of B'. However, in our experience, the vector $\mathbf{w}$ is extremely closely aligned to $\mathbf{v}_1$ and therefore the coefficients $w_i / w_1 \ll 1$. In this case, all but one term in the right-hand side of Equation (13) can be neglected and we get

$$\lambda_1^2 \left(\lambda_1 - \lambda'_1 \right) - \alpha_1 \approx 0. \tag{16}$$

Here, the larger α_1, the larger the eigen-drop $\lambda_1 - \lambda'_1$. Therefore, we study the significance of α_1 next.

Proposition 3.4. Let $\mathbf{u}_1, \mathbf{v}_1$ be the left and right eigenvectors of B' normalized such that $\mathbf{u}_1^T \mathbf{v}_1 = 1$. Then we have

$$\alpha_1 = \mathbf{u}_1^T X \mathbf{v}_1 = \mathbf{v}_1^T P X \mathbf{v}_1 = \left(\sum_i a_{ci} \mathbf{v}_1^i \right)^2 - \sum_i a_{ci} \left(\mathbf{v}_1^i \right)^2, \tag{17}$$

where $\mathbf{v}_1^i$ is the NB-centrality of node i in the graph after removal (see Equation (2)). We call α_1 the *X-non-backtracking centrality*, or *X-NB centrality*, of c.

Proof. The first equality comes from the fact that $\mathbf{u}_1 = P \mathbf{v}_1$, by Lemma A.1. We can find $PX_{k \to l, i \to j} = a_{cl} a_{cj} \left(1 - \delta_{lj} \right)$ using Equations (8) and (4). The result then follows from manually computing $\mathbf{v}_1^T P X \mathbf{v}_1$ and applying Equation (2). □

The Proposition establishes that the behavior of the eigen-drop in (16) is governed by the X-NB centrality of c in (17), which is a function only of the NB-centralities of c's neighbors. Importantly, these centralities are measured *after* c is removed. We come back to this point in Section 4. Notably, the principal eigenvector is normalized by $\mathbf{u}_1^T \mathbf{v}_1 = \mathbf{v}_1^T P \mathbf{v}_1 = 1$, i.e. it does not have unit length.

3.2.2 An upper bound

An alternative way of studying the eigen-drop is by choosing $\mathbf{w}$ such that $w_1 = 1$, and bounding

$$q = q(c) = \sum_i w_i \alpha_i, \tag{18}$$

which drives the right-hand side of Equation (13).

Suppose that R is the matrix whose columns are the eigenvectors $\{\mathbf{v}_i\}$, and let $L = R^{-1}$ such that $B' = R\Lambda L$, where Λ is the diagonal matrix of the eigenvalues $\{\lambda_i'\}$. The rows of L are left eigenvectors of B', in particular, $\mathbf{u}_1^T$ is the first row of L. Then we have $\alpha_i = (LXR)_{1i}$, and q is the dot product between the first row of LXR and $\mathbf{w}$,

$$q = \mathbf{e_1^T} LXR\mathbf{w} = \operatorname{Tr}\left(LXR\,\mathbf{w}\mathbf{e_1^T}\right), \tag{19}$$

where $\mathbf{e_1} = (1, 0, \ldots, 0)$. Using the cyclic property of the trace, and the fact that $P^2 = I$, we now have

$$q = \operatorname{Tr}\left(LXR\,\mathbf{w}\mathbf{e_1^T}\right) = \operatorname{Tr}\left(XR\,\mathbf{w}\mathbf{e_1^T}\,L\right) = \operatorname{Tr}\left(PXR\,\mathbf{w}\mathbf{e_1^T}\,LP\right). \tag{20}$$

Applying the Cauchy-Schwarz inequality for the trace gives us

$$q \leq |PX|_F \left|R\,\mathbf{w}\mathbf{e_1^T}\,LP\right|_F, \tag{21}$$

where $|M|_F^2 = \operatorname{Tr}\left(M^T M\right)$ is the Frobenius norm. Finally, the fact that $\mathbf{w}\mathbf{e_1^T}$ is a matrix with rank one gives

$$q \leq |PX|_F \left(\mathbf{e_1^T} LPR\mathbf{w}\right). \tag{22}$$

As before, we have $w_i/w_1 \ll 1$ and since we chose $w_1 = 1$, the term $\left(\mathbf{e_1^T} LPR\mathbf{w}\right)$ is very close to 1. Therefore, we obtain $|PX|_F$ as an (approximate) upper bound for q. Observe that since PX is non-negative, we have $|PX|_F = \mathbf{1}^T PX\mathbf{1}$, where $\mathbf{1} = (1, 1, \ldots, 1)$.

Proposition 3.5. In Equation (13), let $\mathbf{w}$ be such that $w_1 = 1$, and define $q = \sum_i w_i \alpha_i$. The quantity $\mathbf{1}^T PX\mathbf{1}$ is an approximate upper bound for q, that is, $q \leq \mathbf{1}^T PX\mathbf{1}\left(\mathbf{e_1^T} LPR\mathbf{w}\right)$. Furthermore, we have

$$\mathbf{1}^T PX\mathbf{1} = \left(\sum_i a_{ci}\left(d_i - 1\right)\right)^2 - \sum_i a_{ci}\left(d_i - 1\right)^2, \tag{23}$$

where d_i is the degree of node i in the original graph, before removal. We call $\mathbf{1}^T PX\mathbf{1}$ the *X-degree centrality* of c.

Proof. The first claim was proved in the previous paragraphs. The second claim comes from direct evaluation of $\mathbf{1}^T PX\mathbf{1}$ using $PX_{k\to l,i\to j} = a_{cl}a_{cj}\left(1-\delta_{lj}\right)$, keeping in mind the degrees are measured after removal. $\square$

Remark 3. The fact that the X-degree of c, $\mathbf{1}^T PX\mathbf{1}$, bounds q only approximately merits further theoretical consideration. However, it will be immaterial in our exposition going forward, as our experiments will show that, in practice, the X-degree of nodes is an excellent predictor of the node's eigen-gap, regardless of the value of $\mathbf{e_1^T}LPR\mathbf{w}$.

3.3 X-centrality

In Section 3.2.1 we use the X-NB centrality, $\mathbf{v}_1^T PX\mathbf{v}_1$, while in Section 3.2.2 we use the X-degree centrality, $\mathbf{1}^T PX\mathbf{1}$, both for the purpose of studying the eigen-drop induced by c. The former is a function of the NB-centralities of the neighbors of c (Proposition 3.4), while the latter is a function of their degrees (Proposition 3.5). Importantly, both centralities are measured *after* c has been removed.

Consider a fixed target node c, which in turn fixes X and P. The matrix PX is capable of defining new node-level statistics given a vector of values for each directed edge. It does so by aggregating the edge values along NB-walks that go through c; following Figure 1, this aggregation is done along blue-yellow-yellow-blue walks. Recall that if G has m (undirected) edges and c has degree d, then X and P are of size $2m-2d$. Given an arbitrary vector $\mathbf{z}$ of size $2m-2d$, we have

$$\mathbf{z}^T PX\mathbf{z} = \left(\sum_i a_{ci}\sum_j \mathbf{z}_{j\to i}\right)^2 - \sum_i a_{ci}\left(\sum_j \mathbf{z}_{j\to i}\right)^2. \tag{24}$$

One can evaluate the right-hand side of (24) for any vector $\mathbf{z}$ of size $2m$, and use only the $2m-2d$ entries that correspond to edges not incident to c. In other words, we do not need to know X or P, but only who the neighbors of c are. Since c determines both X and P, the same vector $\mathbf{z}$ can be evaluated using different target nodes. Therefore the quantity in (24) naturally corresponds to whichever target node was used to evaluate it, and can be thought of as a node-level quantity derived from $\mathbf{z}$.

Now define $\mathbf{z}^i = \sum_j \mathbf{z}_{j\to i}$ and let $\mathrm{Var}_c\left(\mathbf{z}^i\right)$ be the variance of the $\mathbf{z}^i$ values corresponding to neighbors of c. Then we have

$$\mathrm{Var}_c\left(\mathbf{z}^i\right) = \frac{\sum_i a_{ci}\left(\mathbf{z}^i\right)^2}{d} - \left(\frac{\sum_i a_{ci}\mathbf{z}^i}{d}\right)^2, \tag{25}$$

which differs from (24) only in sign and a (non-linear) normalization. Accordingly, $\mathbf{z}^T PX\mathbf{z}$ will have large values when $\mathbf{z}^i$ has little variability among the neighbors of c.

Using this framework we could define, for example, *X-closeness centrality*, *X-betweenness centrality*, etc. Whether these concepts are as useful as the two studied here remains an open question.

4 Node immunization

Targeted immunization works as follows. Given a graph G and an integer p, we want to remove from G the p nodes that increase the epidemic threshold the most (equivalently, decrease the largest NB-eigenvalue the most). Common strategies involve three steps: (i) the nodes are sorted by decreasing values of a certain statistic, for example degree; (ii) the node with the highest value of this statistic is removed from the graph; and (iii) the statistic has to be recomputed after each time a node is removed. These steps are repeated until the target number p has been removed. In this context, our framework presents two major obstacles:

a) Both the X-NB and X-degree centralities of a node must be computed *after* the node has been removed. So, to execute the step (i) above, we need to temporarily remove each node in turn before we decide which one to ultimately remove, which defeats the purpose of targeted immunization.

b) For step (iii), we must guarantee that recomputing the statistic of every node at each step is an efficient procedure.

4.1 Using X-NB centrality

Algorithm 1 naively follows the steps above to implement an immunization strategy based on X-NB centrality. We are tempted to think this strategy is the "right" one, as it approximates the true effect of a node's removal in the epidemic threshold. However, we must address the obstacles mentioned above.

```
Input: graph $G$, integer $p$
Output: removed, an ordered list of nodes to immunize
1  removed $\leftarrow \emptyset$
2  XNB[i] $\leftarrow 0$ for each node i
3  while length(removed) $< p$ do
4      foreach node c in $G$ do
5          $H \leftarrow$ RemoveNode($G$, c)
6          $v_H \leftarrow$ principal eigenvector of AuxNBMatrix($H$)
7          XNB[c] $\leftarrow$ XNBCentrality($v_H$, c)
8      node $\leftarrow \arg\max_i$ XNB[i]
9      $G \leftarrow$ RemoveNode($G$, node)
10     removed.append(node)
11 return removed
```

Algorithm 1: Naive X-NB immunization strategy.

To overcome obstacle (a), we propose to approximate Equation (17) by using the NB-centralities in the original graph before removing any node even temporarily. Algorithm 2 takes this approximation into account. The error incurred

```
Input: graph $G$, integer $p$
Output: removed, an ordered list of nodes to immunize
1  removed ← ∅
2  XNB[i] ← 0 for each node i
3  while length(removed) < $p$ do
4      $v_G$ ← principal eigenvector of AuxNBMatrix($G$)
5      foreach node c in $G$ do
6          XNB[c] ← XNBCentrality($v_G$, c)
7      node ← $\arg\max_i$ XNB[i]
8      $G$ ← RemoveNode($G$, node)
9      removed.append(node)
10 return removed
```

Algorithm 2: Approximate X-NB immunization strategy.

by this approximation is dampened by the fact that what we are ultimately interested in is the ranking of the nodes rather than the actual values of their centralities. For obstacle (b), one could use a strategy similar to [19], where they devise an algorithm to approximate the impact on a node's eigenvector centrality after the removal of a node without having to recompute the values again. However, doing so for X-NB centrality remains an open question.

Complexity Analysis

We assume that G is given in adjacency list format. In Algorithm 1, lines 2 and 8 take n operations each. Line 5 creates a copy H of the adjacency list and removes the target node c from it (but leaves G intact). Line 6 uses Equation (3) to compute the auxiliary NB-matrix, which takes $O(m)$ time, and it takes $O(m)$ to compute the principal eigenvector (using, e.g. the Lanczos algorithm with a number of iterations that does not depend on the parameters). Line 7 uses Lemma A.2 to compute the correctly normalized NB-centralities, and Equation (17) to compute X-NB centralities, both of which take n operations. The remaining lines take constant time. Accounting for loops, Algorithm 1 takes a total of $O(n + p(n(m+n)+n)) = O(pn(m+n))$. In Algorithm 2, the NB-centralities are computed outside of the inner loop, which gives a complexity of $O(p(m+n))$, or $O(m+n)$ for constant p.

4.2 Using X-degree

X-degree can be easily computed without temporarily removing any nodes, see Equation (23). Indeed, all we need to know about the graph after removal is the degree of each node. Hence, obstacle (a) is easily overcome in this case. Further, after each step we need not recompute the X-degree of all nodes, but only of those nodes two steps away from the target node. Indeed, removing c changes the degree of its neighbors, which in turn changes the X-degree of its

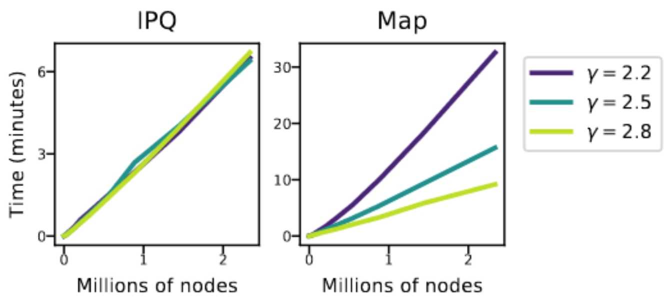


Figure 2: Average runtime scaling of Algorithm 3 on random power-law graphs with varying degree exponent γ. The runtimes are linear, with IPQ being faster than Map.

neighbors' neighbors. So obstacle (b) is also overcome. Algorithm 3 implements this strategy. Importantly, it does not involve the computation of any matrices or their eigenvectors.

Complexity Analysis

Lines $1, 6, 10, 11$ of Algorithm 3 take constant time, while line 2 takes $O(m)$. When using a standard map (or dictionary) to store the X-degree values, line 4 takes $O(n)$ operations, and line 9 takes $O(1)$. Now suppose that the nodes removed by Algorithm 3 are, in order, $i_1, \ldots, i_p$. At iteration j, the loop in line 5 takes d_{i_j} operations, and the double loop in lines $7-8$ takes as many iterations as the number of nodes two steps away from i_j, say D_{i_j}. This yields a total of $O\left(m + pn + \sum_{j=1}^{p} d_{i_j} + \sum_{j=1}^{p} D_{i_j}\right)$. We can also implement Algorithm 3 using an indexed priority queue (IPQ) to store the X-degree values instead of a map; see Appendix B. In this case the worst case scenario complexity is $O\left(m + p\log n + \sum_{j=1}^{p} d_{i_j} + \log n \sum_{j=1}^{p} D_{i_j}\right)$. In Appendix B we refine this analysis for networks with homogeneous or heterogeneous degree distributions, and show that the map or IPQ versions have better worst case scenario scalability for different values of network parameters.

Importantly, the average runtime of both versions is in fact close to linear, with the IPQ version being the fastest. Figure 2 shows the average runtime of both versions on random power-law configuration model graphs with varying degree exponent γ and constant p (see Appendix B for details). The reason the average runtime is considerably faster than the worst-case scenario is because graphs typically have very few large hubs. That is, roughly speaking, there are $O(1)$ many nodes that take $O(n)$ time to process, while there are $O(n)$ many nodes that take $O(1)$ time to process. This effect is intensified the closer γ is to 2, which counterbalances the exponent $\frac{2}{\gamma-1}$ in the worst case scenario.

```
Input: graph $G$, integer $p$
Output: removed, an ordered list of nodes to immunize
1  removed $\leftarrow \emptyset$
2  XDeg[i] $\leftarrow$ XDegree($G$, i) for each node i
3  while length(removed) $< p$ do
4      node $\leftarrow \max_i$ XDeg[i]
5      foreach i in $G$.neighbors[node] do
6          $G$.neighbors[i].remove(node)
7      foreach i in $G$.neighbors[node] do
8          foreach j in $G$.neighbors[i] do
9              XDeg[j] $\leftarrow$ XDegree($G$, j)
10     $G$.neighbors[node] $\leftarrow \emptyset$
11     removed.append(node)
12 return removed
```

Algorithm 3: X-degree immunization strategy.

5 Related work

Perturbation of NB-matrix Zhang [20] briefly treats the case of eigenvalue perturbation of a matrix derived from the NB-matrix in the case of edge removal, while Coste and Zhu [21] analyze the perturbation of quadratic eigenvalue problems, with applications to the NB-eigenvalues of the stochastic block model. Our theory is more general since it studies node removal (as opposed to single edge removal), and it applies to any arbitrary graph.

NB centrality Many notions of centrality based on the NB-matrix exist, for example NB-PageRank [22], NB-centrality [8, 18], and Collective Influence [6, 5]. The latter two have been proposed as solutions to the problem of "influencer identification". This problem aims to find nodes that determine the course of spreading dynamics, and is thus more general than our objective of increasing the epidemic threshold. Collective Influence in particular is similar to X-degree; see Appendix C.1. Also in this context, Kitsak et al. [23] propose to use the k-core index, and Poux-Médard et al. [24] highlight the importance of node degree. We compare our algorithms to all of these baselines in Section 6. Finally, Everett and Borgatti [25] study the influence of a node's removal in other nodes' centrality, which is reminiscent to our X-centrality framework.

Targeted immunization Pastor-Satorras et al. [14] review general immunization strategies and other generalities of spreading dynamics on networks. Chen et al. [19] propose `NetShield`, an efficient algorithm for immunization focusing on decreasing the largest eigenvalue of the adjacency matrix. We prefer to focus on decreasing the largest NB-eigenvalue instead since it provides a tighter bound to the true epidemic threshold in certain cases [12, 13, 16]. Lin et al. [26] study the percolation threshold in terms of so-called high-order non-backtracking matrices. Percolation thresholds are tightly related to epidemic thresholds of SIR dynamics [14, 15].

6 Experiments

6.1 Approximating the Eigenvalue

How close is the approximation in Equation (16)? We first compute the largest NB-eigenvalue λ_1 of a graph G. Then we fix a target node c and remove it from G and compute the new eigenvalue λ_c. (For ease of notation, in this section we use λ_c instead of λ_1', and α instead of α_1.) Finally, we use (16) to compute two approximations,

$$\widehat{\lambda}_c = \lambda_1 - \alpha/\lambda_1^2, \qquad \widetilde{\lambda}_c = \lambda_1 - \widetilde{\alpha}/\lambda_1^2, \tag{26}$$

where α is the true X-NB centrality of c, and $\widetilde{\alpha}$ is the approximate X-NB centrality used in Algorithm 2, i.e., it is computed using the NB-centralities before removing c. We now compare the approximations $\widehat{\lambda}_c$ and $\widetilde{\lambda}_c$ to the true value of λ_c for randomly selected nodes of synthetic graphs. We use different synthetic random graph models: Watts-Strogatz (WS) [27], Stochastic Block Model (SBM) [28, 29], Barabási-Albert (BA) [30], and Block Two–Level Erdős-Rényi (BTER) [31]. See Section C.2 for more details on the data sets, and Section C.3 for details on the experimental setup.

Fig. 3a shows that our approximation is extremely close for all graphs tested, though it tends to underestimate the eigen-drop in WS graphs. Fig. 3c shows the average relative error versus degree. Our approximation worsens as degree increases, though it is quite small for most degrees. In the worst case, the relative error is less than 10^{-4}, or 0.01%. Fig. 3b shows the eigen-drop computed using the approximate version of X-NB. This approximation is systematically overestimating the true eigen-drop. Fig. 3d shows that this systematic error is of the order of 10% in the worst case, though it is negligible for small degrees. In all, Figure 3 confirms the accuracy of our approximations, and it points to the fact that the terms neglected in (16) will become larger as degree increases.

6.2 Predicting the Eigen-drop

How well can X-NB centrality and X-degree predict a node's eigen-drop? Unlike in Experiment 6.1, here we do not approximate the eigen-drop, but only seek to predict its size. (In fact, we cannot use X-degree to approximate the eigen-drop at all.) See Section C.4 for experimental setup, and Section C.2 for details on data sets.

In Fig. 4a we measure how correlated the true value of X-NB, denoted by α, is to the true eigen-drop. For SBM, BA, and BTER graphs, the magnitude of α lines up extremely closely with the value of the eigen-drop, showing a correlation coefficient of $r = 1.00$. In all cases, α is better correlated to the

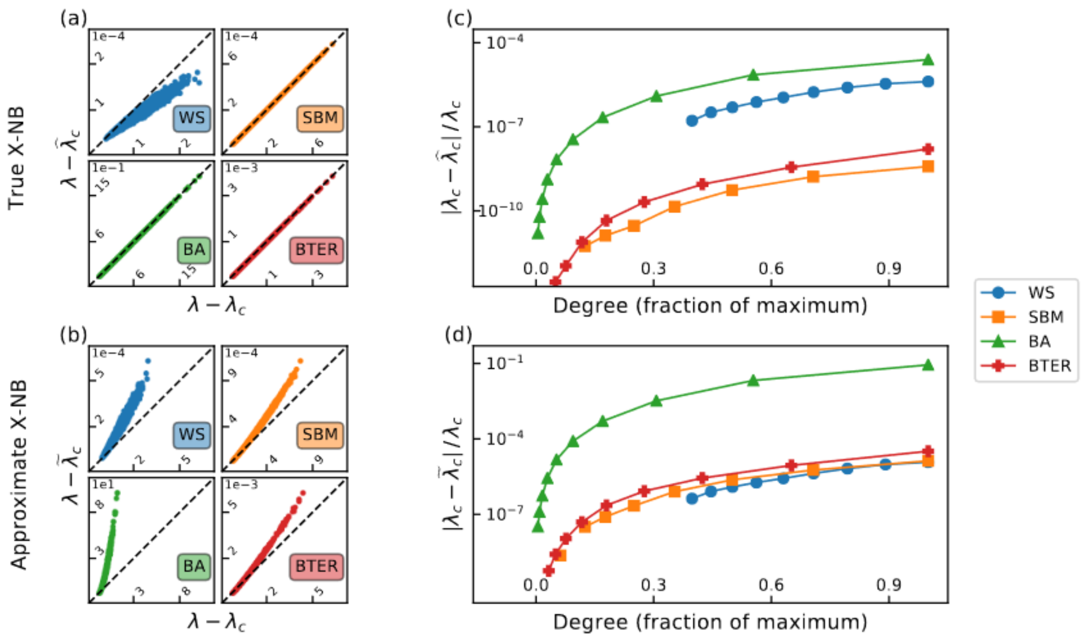


Figure 3: *Left:* True eigen-drop (vertical axis) vs approx. eigen-drop (horizontal axis), using *(a)* true, and *(b)* approx. values of X-NB. Dashed line is $y = x$. Each marker represents one node. *Right:* Relative error when predicting λ_c, as a function of degree, using *(c)* true, and *(d)* approx. values of X-NB. Degrees expressed as a fraction of the maximum degree among graphs in the same ensemble. Each marker is the average within log-binned values of degree; error bars too small to show at this scale. WS graphs (blue circles) have no nodes whose degree is less than 30% of the maximum. Our approximation of the eigen-gap is accurate.

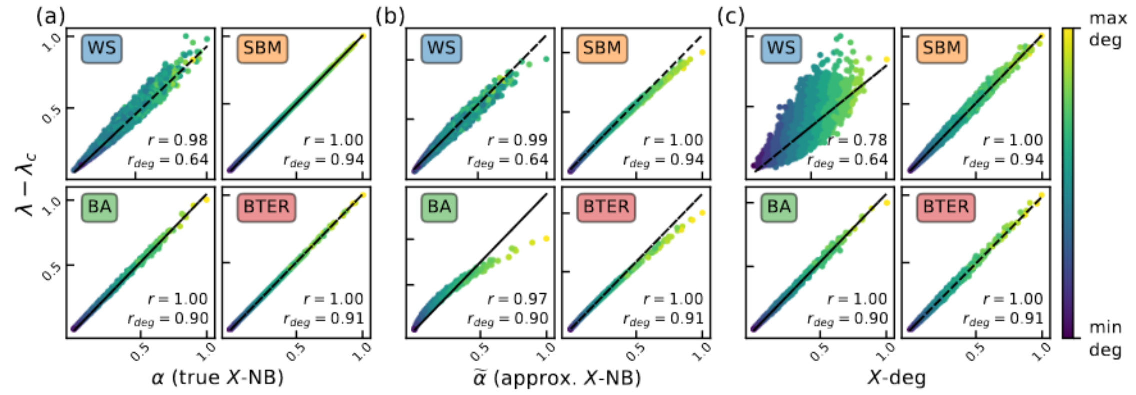


Figure 4: Predicting the eigen-drop using *(a)* true value of X-NB, *(b)* approx. value of X-NB, and *(c)* X-degree. Markers colored by degree, expressed as a fraction of the largest degree in the same ensemble. Each panel shows the correlation coefficient between the corresponding statistic and the eigen-drop (r), and the correlation between degree and the eigen-drop (r_{deg}). Dashed lines are linear regression lines. Our proposed node-level statistics accurately track eigen-drops in random graph models such as BA, SBM, and BTER. X-degree underestimates the eigen-drop in WS graphs.

eigen-drop than degree (as shown by the correlation coefficients r_{deg}). In WS we see considerably more variance than in other ensembles though α is still an excellent predictor of the eigen-drop, at $r = 0.98$. This picture repeats itself when using the approximate value of X-NB, $\widetilde{\alpha}$ (Fig. 4b), and X-degree (Fig. 4c). $\widetilde{\alpha}$ seems to slightly underestimate the eigen-drop, while X-degree has noticeably more variance than the other two statistics, especially in WS. All three statistics are better correlated to the eigen-drop than degree in all graph ensembles. We highlight that even when some of the panels in Fig. 4 are not precisely linear, they all show that the eigen-drop is an increasing function of all of α, $\widetilde{\alpha}$, and X-degree. These results encourage us to use X-NB and X-degree as immunization strategies. Further, using α has very little advantage over $\widetilde{\alpha}$, and therefore we are justified in using Algorithm 2 instead Algorithm 1 for computational reasons.

6.3 Immunization with X-NB and X-degree

How effective are X-NB and X-degree at immunization? We remove $1, 2$ and 3 percent of nodes using different strategies and evaluate the resulting eigenvalue. We use the immunization strategies node degree (`degree`), k-core index (`core`), NetShield (`NS`), Collective Influence (`CI`), NB-centrality (`NB`), approximate X-NB (`XNB`), and X-degree (`Xdeg`). For computational reasons, we do not use the true value of X-NB; for more details on baselines see Section C.1.

		`degree`	`NS`	`CI`	`Xdeg`	`NB`	`XNB`
	1%	62.76	61.44	62.88	62.90	62.92	62.91
BA	2%	68.84	66.94	68.97	68.99	69.01	69.01
	3%	72.42	70.09	72.56	72.57	72.59	72.59
	1%	6.28	6.40	6.41	6.45	6.46	6.46
BTER	2%	10.60	10.72	10.80	10.85	10.86	10.86
	3%	14.31	14.40	14.55	14.61	14.63	14.63
	1%	3.31	3.41	3.40	3.43	3.44	3.44
SBM	2%	6.00	6.16	6.19	6.23	6.25	6.25
	3%	8.52	8.66	8.76	8.80	8.82	8.82
	1%	1.41	1.17	1.50	1.52	1.63	1.63
WS	2%	2.52	2.09	2.97	2.98	3.11	3.11
	3%	3.66	2.94	4.41	4.41	4.57	4.58

Table 1: Average percentage eigen-drop (larger is better) on synthetic graphs after removing 1%, 2%, and 3% of the nodes using different strategies. Strategies are (column) grouped in performance tiers. `NB` and `XNB` have the best performances.

In all data sets, `core` had the least performance and is therefore not shown in our results. We hypothesize this is because many nodes can have the same k-core index at the same time, so `core` cannot identify which is the best one among all of them.

Table 1 shows the percentage reduction of the eigenvalue after immunization, averaged over repetitions on synthetic graphs. We can arrange immunization strategies in tiers according to increasing performance: strategies within a tier have comparable performance across data sets. The third tier is made up of `NS` and `degree`. They perform similarly because `NS` targets the largest eigenvalue of the adjacency matrix, which is largely dominated by node degree. Strategies in this tier perform substantially better than `core` (not shown), and are very

	$p = 1$			$p = 10$			$p = 100$		
	`degree`	`CI`	`Xdeg`	`degree`	`CI`	`Xdeg`	`degree`	`CI`	`Xdeg`
`AS-1`	0.74	0.74	**2.35**	6.70	13.51	**15.43**	71.65	**78.26**	75.92
`AS-2`	2.02	2.02	**4.00**	17.09	22.36	**28.17**	87.60	**89.61**	87.02
`Social-Slashdot`	0.95	**1.02**	**1.02**	4.63	6.06	**6.94**	23.65	28.11	**30.30**
`Social-Twitter`	**2.18**	**2.18**	1.98	13.21	**13.97**	13.68	41.10	42.88	**43.39**
`Transport-California`	0.00	0.00	**0.65**	**2.65**	0.65	**2.65**	5.09	5.09	**7.80**
`Transport-Sydney`	**0.00**	**0.00**	**0.00**	0.00	0.00	**6.50**	0.00	7.37	**9.49**
`Web-NotreDame`	**9.34**	**9.34**	**9.34**	12.10	**13.79**	**13.79**	14.37	14.37	**19.22**

Table 2: Average percentage eigen-drop on real networks (larger is better) when removing $p = 1, 10$, or 100 nodes. `Xdeg` is effective and has log-linear time in the number of nodes. Details about the sizes of these datasets are in Table 3 of the appendix.

close to the strategies in the next two tiers, i.e. `degree` is a very strong baseline in this task. The second tier is comprised of `CI` and `Xdeg`, with `Xdeg` having a slight advantage over `CI`. Finally, the best performance was achieved by `NB` and `XNB`. Their performances were almost indistinguishable in most data sets, though they have a small margin over `CI` and `Xdeg`.

Strategies in the best two tiers, i.e. `CI`, `Xdeg`, `NB` and `XNB`, all showed standard deviations of similar magnitude across all data sets (not shown), and the ordering in increasing performance `CI ¡ Xdeg ¡ NB` $\approx$ `XNB` is statistically significant at $p \ll 10^{-10}$ (see Appendix C.5). Further, the best two (`NB` and `XNB)` use the principal NB-eigenvector, whereas `CI` and `Xdeg` depend only on node degree, and are therefore much more computationally efficient.

Table 2 shows the results on real data sets, where we have run only `degree`, `CI`, and `Xdeg` for computational reasons. We use social networks [32, 33], transportation networks, [34, 35, 33], Autonomous Systems (AS) of the Internet networks [36, 12], and web crawl networks [37]. See Section C.2 for data set descriptions. We remove from each network 1, 10, and 100 nodes at a time. Again, `degree` is a very strong baseline, but it is never better than both `CI` and `Xdeg` at the same time. All three strategies are able to drastically immunize the autonomous systems networks `AS-1` and `AS-2` at 100 nodes removed, probably owing to the fact that their degree distribution is extremely heterogeneous and thus the nodes with largest degree have a large eigen-drop. In all other networks, X-degree achieves the best performance. An interesting case is that of `Transport-Sydney`. The node identified by all three strategies has an eigen-drop of exactly 0.0. Following Corollary 1, this means that the chosen node lies outside of the 2-core of the graph and thus has no impact on non-zero NB-eigenvalues. After 10 nodes are removed, both `degree` and `CI` continue to achieve zero eigen-drop, while `Xdeg` already identifies the correct nodes and ahieves 6.50% decrease. Even at 100 nodes removed, `degree` cannot identify nodes that generate an eigen-drop. A similar case occurs on `Transport-California`, where the first node identified by `degree` and `CI` generates no eigen-drop, while `Xdeg` is able to correctly identify influential nodes.

We conclude that in cases where efficiency is of the essence, `Xdeg` is the best overall immunization strategy, as it has a slight advantage over `CI` and its performance is close to optimal. If effectiveness is more important than efficiency, either `XNB` or `NB` should be used.

7 Conclusion

We developed a theory of spectral analysis for the NB-matrix by studying what happens to its largest eigenvalue when one node is removed from the network. Our theory is independent of the structure of the graph, i.e. we make no assumptions of locally tree-like structure or density or length of cycles, as is usual in other studies. We find two new node-level statistics, or centrality measures, X-NB centrality and X-degree, which are excellent predictors of a node's influence on the largest NB-eigenvalue. Finally, we focus on the application of targeted immunization, where we propose two new algorithms that are shown to be more effective than other strategies for a variety of real and synthetic graph ensembles.

Our techniques open many possibilities for further research. For instance, the left-hand side of Equation (11) is reminiscent to certain quantities used in the theory of eigenvalue interlacing [38], while the matrix $(B' - tI)^{-1}$ on the right-hand side is known as the *resolvent* of B', which has many applications in random matrix theory [39]. On a different note, Cvetkovic et al. [40] highlight that most matrices associated to graphs are *linear* combinations of I, A, and D, whereas the NB-matrix is associated with a *quadratic* combination of I, A, and D, via Equation (3). In the future, we will explore which other matrices associated with graphs can be studied via quadratic, or higher order, combinations of I, A, and D.

We focused on the application to targeted immunization. However, other applications of NB-eigenvalues exist – e.g., community detection and graph distance. Further studying the behavior of NB-eigenvalues under small perturbations of the graph, using the framework presented here, has potential to affect those applications.

Acknowledgements

L.T. thanks Gabor Lippner for many invaluable discussions.

References

[1] Noga Alon, Itai Benjamini, Eyal Lubetzky, and Sasha Sodin. Non-backtracking random walks mix faster. *Commun. Contemp. Math.*, 9(4): 585–603, 2007.

[2] Leo Torres, Pablo Suárez-Serrato, and Tina Eliassi-Rad. Non-backtracking cycles: length spectrum theory and graph mining applications. *Applied Network Science*, 4(1), 2019.

[3] Charles Bordenave, Marc Lelarge, and Laurent Massoulié. Non-backtracking spectrum of random graphs: Community detection and non-regular ramanujan graphs. In *FOCS*, pages 1347–1357, 2015.

[4] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proc. Natl. Acad. Sci. USA*, 110(52):20935–20940, 2013. ISSN 0027-8424.

[5] Flaviano Morone, Byungjoon Min, Lin Bo, Romain Mari, and Hernán A Makse. Collective influence algorithm to find influencers via optimal percolation in massively large social media. *Scientific Reports*, 6:30062, 2016.

[6] Flaviano Morone and Hernán A Makse. Influence maximization in complex networks through optimal percolation. *Nature*, 524(7563):65, 2015.

[7] Andrew Mellor and Angelica Grusovin. Graph comparison via the non-backtracking spectrum. *Phys. Rev. E*, 99(052309), 2019.

[8] Travis Martin, Xiao Zhang, and M. E. J. Newman. Localization and centrality in networks. *Phys. Rev. E*, 90(052808), 2014.

[9] Francesca Arrigo, Peter Grindrod, Desmond J. Higham, and Vanni Noferini. Non-backtracking walk centrality for directed networks. *J. Complex Networks*, 6(1):54–78, 2018.

[10] Mark Kempton. Non-backtracking random walks and a weighted iharas theorem. *Open Journal of Discrete Mathematics*, 6:207–226, 2016.

[11] Francesca Arrigo, Peter Grindrod, Desmond J. Higham, and Vanni Noferini. On the exponential generating function for non-backtracking walks. *Linear Algebra and its Applications*, 556:381 – 399, 2018.

[12] Brian Karrer, Mark EJ Newman, and Lenka Zdeborová. Percolation on sparse networks. *Phys. Rev. Lett.*, 113(208702), 2014.

[13] Kathleen E Hamilton and Leonid P Pryadko. Tight lower bound for percolation threshold on an infinite graph. *Phys. Rev. Lett.*, 113(208701), 2014.

[14] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Rev. Mod. Phys.*, 87(3), 2015.

[15] Mark EJ Newman. Spread of epidemic disease on networks. *Phys. Rev. E*, 66(016128), 2002.

[16] Munik Shrestha, Samuel V Scarpino, and Cristopher Moore. Message-passing approach for recurrent-state epidemic models on networks. *Phys. Rev. E*, 92(022821), 2015.

[17] Claudio Castellano and Romualdo Pastor-Satorras. Relevance of backtracking paths in recurrent-state epidemic spreading on networks. *Phys. Rev. E*, 98(052313), 2018.

[18] Filippo Radicchi and Claudio Castellano. Leveraging percolation theory to single out influential spreaders in networks. *Phys. Rev. E*, 93(062314), 2016.

[19] Chen Chen, Hanghang Tong, B. Aditya Prakash, Charalampos E. Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. Node immunization on large graphs: Theory and algorithms. *TKDE*, 28 (1):113–126, 2016.

[20] Pan Zhang. Nonbacktracking operator for the ising model and its applications in systems with multiple states. *Phys. Rev. E*, 91(042120), 2015.

[21] Simon Coste and Yizhe Zhu. Eigenvalues of the non-backtracking operator detached from the bulk. *CoRR*, abs/1907.05603, 2019.

[22] Francesca Arrigo, Desmond J. Higham, and Vanni Noferini. Non-backtracking pagerank. *J. Sci. Comput.*, 80(3):1419–1437, 2019.

[23] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. Identification of influential spreaders in complex networks. *Nature Physics*, 6(11):888, 2010.

[24] Gaël Poux-Médard, Romualdo Pastor-Satorras, and Claudio Castellano. Influential spreaders for recurrent epidemics on networks. *CoRR*, abs/1912.08459, 2019.

[25] Martin G. Everett and Stephen P. Borgatti. Induced, endogenous and exogenous centrality. *Social Networks*, 32(4):339–344, 2010.

[26] Yuan Lin, Wei Chen, and Zhongzhi Zhang. Assessing percolation threshold based on high-order non-backtracking matrices. In *WWW*, pages 223–232, 2017.

[27] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *Nature*, 393(6684):440, 1998.

[28] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99(12):7821–7826, 2002.

[29] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83(016107), 2011.

[30] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

[31] Comandur Seshadhri, Tamara G Kolda, and Ali Pinar. Community structure and scale-free collections of erdős-rényi graphs. *Phys. Rev. E*, 85 (056109), 2012.

[32] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. The anatomy of a scientific rumor. *Scientific Reports*, 3:2980, 2013.

[33] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[34] Tore Opsahl. Why anchorage is not (that) important: Binary ties and sample selection, 2011. URL `toreopsahl.com/datasets/#usairports`.

[35] Transportation Networks for Research Core Team. Transportation networks for research, 2013. URL `github.com/bstabler/TransportationNetworks/`.

[36] Beichuan Zhang, Raymond A. Liu, Daniel Massey, and Lixia Zhang. Collecting the internet as-level topology. *Computer Communication Review*, 35(1):53–61, 2004.

[37] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999.

[38] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[39] Terrence Tao. The semi-circular law, 2010. URL `terrytao.wordpress.com/2010/02/02/254a-notes-4-the-semi-circular-law`.

[40] Dragoš M Cvetkovic, Michael Doob, Horst Sachs, et al. *Spectra of graphs*, volume 10. Academic Press, New York, 1980.

[41] Albert-László Barabási. *Network science*. Cambridge University Press, 2016.

[42] Tamara G. Kolda, Ali Pinar, Todd D. Plantenga, and C. Seshadhri. A scalable generative graph model with community structure. *SIAM J. Sci. Comput.*, 36(5), 2014.

[43] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

A Technical Lemmas

In this subsection, B is the NB-matrix of a graph G, P is defined in Section 2, and $\lambda, \mathbf{v}$ are the Perron eigenvalue and corresponding unit right eigenvector of B. Let also $\mathbf{v}^i = \sum_j \mathbf{v}_{j\to i}$ as in Equation (2) and let $\overline{\mathbf{v}} = \left(\mathbf{v}^1, \ldots, \mathbf{v}^n\right)$.

Lemma A.1. $P\mathbf{v}$ is a left eigenvector of B corresponding to λ.

Proof. Since PB is symmetric and $P^2 = I$, we have $B = PB^TP$. Now $B\mathbf{v} = \lambda\mathbf{v}$ implies $B^TP\mathbf{v} = \lambda P\mathbf{v}$, which completes the proof. $\square$

Lemma A.2. Suppose $\mathbf{v}$ is such that $\mathbf{v}^TP\mathbf{v} = 1$. Let $(\mathbf{f}, -\lambda\mathbf{f})$ be the left unit eigenvector of B_{aux} corresponding to λ. Then, we have $\|\overline{\mathbf{v}}\| = \mu\|\mathbf{f}\|$, where

$$\mu = \sqrt{\frac{\lambda\left(\lambda^2 - 1\right)}{1 - \mathbf{f}^T D\,\mathbf{f}}}. \tag{27}$$

Proof. First, from $\mathbf{v}^TP\mathbf{v} = 1$ we get $\mathbf{v}^TPB\mathbf{v} = \lambda\mathbf{v}^TP\mathbf{v} = \lambda$, and we can expand $\mathbf{v}^TPB\mathbf{v}$ to find $\|\overline{\mathbf{v}}\|^2 - \|\mathbf{v}\|^2 = \lambda$. Second, since $(\mathbf{f}, -\lambda\mathbf{f})$ has unit length, we have $\|\mathbf{f}\|^2 = 1/\left(\lambda^2 + 1\right)$. Therefore,

$$\mu^2 = \left(\lambda^2 + 1\right)\left(\lambda + \|\mathbf{v}\|^2\right). \tag{28}$$

Now, $B\mathbf{v} = \lambda\mathbf{v}$ implies $\mathbf{v}_{j\to i} + \lambda\mathbf{v}_{i\to j} = \mathbf{v}^i$ for any i, j. Plug this identity in $\|\mathbf{v}\|^2 = \sum_{i,j} a_{ij}\mathbf{v}_{i\to j}^2$ to find

$$\|\mathbf{v}\|^2\left(\lambda^2 + 1\right) + 2\lambda = \sum_i \left(\mathbf{v}^i\right)^2 \deg i = \overline{\mathbf{v}}^T D\overline{\mathbf{v}} = \mu^2\mathbf{f}^T D\,\mathbf{f}. \tag{29}$$

Using (28) and (29) together finishes the proof. $\square$

Remark 4. Both $\overline{\mathbf{v}}$ and $\mathbf{f}$ determine the same node centrality ranking, though the latter is easier to compute. However, the normalization $\mathbf{v}^TP\mathbf{v} = 1$ is fundamental in our theory, which makes $\overline{\mathbf{v}}$ the more appropriate choice. Lemma A.2 allows us to compute $\overline{\mathbf{v}}$ only with the knowledge of $\mathbf{f}, \lambda$ and D, which is much more efficient than computing $\mathbf{v}$ and $\overline{\mathbf{v}}$ directly.

B Complexity Analysis of Algorithm 3

In Section 4.2 we used a standard map (i.e. hash table, or dictionary) to store the X-degree values in line 2 of Algorithm 3. Alternatively, we can use an indexed priority queue (IPQ). An IPQ is a data structure that behaves like a priority queue except that, additionally, elements in the IPQ can be updated efficiently. The underlying data structure is a max-heap. An IPQ can find the maximum element in the heap, as well as update any element, in logarithmic time.

In this case, line 2 of Algorithm 3 takes m operations to compute the X-degree values plus n operations to heapify the IPQ. Further, lines 4 and 9 take $O(\log n)$ time, which yields a time complexity of

$$O\left(m + n + p\log n + \sum_{j=1}^{p} d_{i_j} + \log n \sum_{j=1}^{p} D_{i_j}\right). \tag{30}$$

B.1 Homogeneous degree distribution

In networks with a homogeneous degree distribution (e.g. Poisson) we can estimate $d_{i_j} \approx \langle k \rangle$ and $D_{i_j} \approx \langle k \rangle^2$, where $\langle k \rangle$ is the average degree. This yields $O\left(m + n + p\langle k \rangle^2 \log n\right)$ total complexity for the IPQ version, while the map version gives $O\left(m + pn + p\langle k \rangle^2\right)$. If $p = O(n)$ and $\langle k \rangle = O(1)$, the IPQ version scales better in the worst case scenario.

B.2 Heterogeneous degree distribution

In networks whose degree distribution is well approximated by a power law, the probability of finding a node of degree d scales as $d^{-\gamma}$, for some $\gamma > 0$. In this case, the first few nodes removed by Algorithm 3 will usually have large degree, comparable to the largest degree in the network, $d_{i_j} = O(d_{\max})$ for each j. Further, in the worst case scenario, each of their neighbors will also have a degree comparable to $d_{\max}$ and thus $D_{i_j} = O\left(d_{\max}^2\right)$ for each j. Using $d_{\max} = O\left(n^{\frac{1}{\gamma-1}}\right)$ [41] yields $O\left(m + pn + pn^{\frac{2}{\gamma-1}}\right)$ for the map version and $O\left(m + pn^{\frac{2}{\gamma-1}} \log n\right)$ for the IPQ version. In the typical case $2 \leq \gamma \leq 3$, the exponent $\frac{2}{\gamma-1}$ varies between 1 and 2.

B.3 Average runtime

We have provided the analysis of worst case scenario runtime. However, the average runtime of both the IPQ and map versions is close to linear, as shown in Figure 2. This figure was generated by first sampling a degree sequence from a power-law density $p_d \propto d^{-\gamma}$, and then generating a graph at random using the configuration model. Self-loops and multi-edges were removed and only the largest component was kept. Each marker is the average of 30 repetitions. We used $p = 100$.

C Experimental Setup

C.1 Base lines

Degree. The degree of a node i, denoted d_i is the number of neighbors it has in the graph. Nodes of degree 1 have zero Collective Influence, X-degree, NB-centrality, X-NB centrality.

	nodes	edges	n	m	λ_1	$d_{\max}$
`AS-1` [36]	AS	digital communication	34,761	107,720	151.442	2,760
`AS-2` [12]	AS	digital communication	22,963	48,436	64.678	2,390
`Social-Slashdot` [33]	users	friendships	77,360	469,180	128.550	2,539
`Social-Twitter` [32]	users	friendships	456,290	12,508,221	636.147	51,386
`Transport-California` [35]	intersections	roads	1,957,027	2,760,388	3.321	12
`Transport-Sydney` [35]	intersections	roads	32,956	38,787	2.266	10
`Web-NotreDame` [37]	websites	hyperlinks	325,729	1,090,108	175.657	10,721

Table 3: Real-world data sets. n: number of nodes, m: number of edges, λ_1: largest NB-eigenvalue, $d_{\max}$: largest degree. AS stands for autonomous systems.

k-core index. The k-core index of a node, also called *coreness*, is defined as follows. First, iteratively remove all nodes of degree 1 until there are none. All nodes removed in this step are assigned a value of k-core index of 1. Then, iteratively remove all nodes of degree 2; all nodes removed at this step have k-core 2. Repeat this process until there are no more nodes in the graph. Notably, following Corollary 1, all nodes with k-core value of 1 have zero NB-centrality.

NB-centrality. The NB-centrality of a node is defined in Equation (2). It was proposed in [18] as an indicator of influential spreaders on locally tree-like networks for the SIR model.

NetShield. NetShield is an efficient algorithm that identifies a subset of nodes with the highest "shield-value", which is defined as the impact a node, or set of nodes, has on the largest eigenvalue of the adjacency matrix [19].

Collective Influence. The Collective Influence (CI) of node i is

$$CI_i = (d_i - 1) \sum_j a_{ij} (d_j - 1), \tag{31}$$

though this definition can be generalized to include nodes in arbitrarily large neighborhoods around i [6]. Note that this is quite similar in nature to X-degree in Equation (23). We think of X-degree as a second-order aggregation of the values $(d_j - 1)$ of the neighbors of i, while CI is a first-order aggregation. Further, one can apply Algorithm 3 to perform targeted immunization based on CI instead of X-degree, and hence they have the same running time complexity (see Section 4.2 and Appendix B). Morone et al. [5] claim that the CI algorithm runs in $O(n \log n)$ time, though we were not able to reproduce this result. In any case, any efficient algorithm that computes CI can be used to compute X-degree as well.

C.2 Data sets

All synthetic graphs have $n = 10^5$ nodes and parameters were chosen so that the average degree was approximately 12. SBM graphs were generated with two blocks, or communities, so that the average within-block degree is 9 and the between-block degree is 3. WS graphs generated with rewiring probability 0.1. BTER graphs were generated with target average local clustering coefficient of 0.98, and target global clustering coefficient of 0.4. BTER graphs were generated with the authors' implementation [42]; all other graphs were generated using NetworkX [43] version 2.3. After generation, we extracted the largest connected component of each graph and converted all multi-edges to single edges and deleted self-loops. 100 graphs were generated from each ensemble. Table 3 describes the real data sets used. Directed networks were converted to undirected, and only the largest connected component of each data set was used.

C.3 Approximating the Largest Eigenvalue

Since nodes of large degree are bound to induce a larger eigen-drop than those of small degree, we chose target nodes at random by sampling 1% of nodes from each graph, proportionally to their degree. This was achieved by sampling one edge at random, with replacement, and then choosing one of its endpoints randomly. This yields a probability of sampling node i equal to $d_i/2m$.

C.4 Predicting the Eigen-drop

Nodes were sampled in the same way as in C.3. Figure 4 shows correlation coefficients, defined as the covariance divided by the product of the standard deviations of the two variables. We computed the correlation between the eigen-drop and each of the statistics: α, $\widetilde{\alpha}$, X-degree, and degree. No three-way correlation was computed.

C.5 Immunization with X-NB and X-degree

To confirm the ordering in increasing performance `CI` $<$ `Xdeg` $<$ `NB` $\approx$ `XNB`, we used a one-sided Wilcoxon signed-rank test, which is a non-parametric version of the paired T-test. In a paired sample setting, this test tests the null hypothesis that the median of the differences between the two samples is positive, against the alternative that it is negative. Therefore, a small p-value means that there is little probability that the first sample's median is smaller than the second's. For each graph ensemble and each percentage of removed nodes (1%, 2%, 3%), the ranking `CI` $<$ `Xdeg` $<$ `NB` was confirmed with $p \ll 10^{-10}$ in all cases. Further, we have `NB` $<$ `XNB` in WS networks ($p \ll 10^{-10}$) and BTER networks ($p < 0.05$), and `NB` $>$ `XNB` in BA networks ($p \ll 10^{-10}$) and SBM networks ($p < 0.05$). We summarize these results by writing `CI` $<$ `Xdeg` $<$ `NB` $\approx$ `XNB`.