On the Vulnerability of Large Graphs

Hanghang Tong
IBM T.J. Watson
htong@us.ibm.com

B. Aditya Prakash
Carnegie Mellon University
badityap@cs.cmu.edu

Charalampos Tsourakakis
Carnegie Mellon University
ctsourak@cs.cmu.edu

Tina Eliassi-Rad
Rutgers University
tina@eliassi.org

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

Duen Horng Chau
Carnegie Mellon University
dchau@cs.cmu.edu

Abstract

Given a large graph, like a computer network, which k nodes should we immunize (or monitor, or remove), to make it as robust as possible against a computer virus attack? We need (a) a measure of the 'Vulnerability' *of a given network, (b) a measure of the* 'Shield-value' *of a specific set of k nodes and (c) a fast algorithm to choose the best such k nodes.*

We answer all these three questions: we give the justification behind our choices, we show that they agree with intuition as well as recent results in immunology. Moreover, we propose NetShield*, a fast and scalable algorithm. Finally, we give experiments on large real graphs, where* NetShield *achieves tremendous speed savings exceeding 7 orders of magnitude, against straightforward competitors.*

1 Introduction

Given a graph, we want to quickly find the k best nodes to immunize (or, equivalently, remove), to make the remaining nodes to be most robust to the virus attack. This is the core problem for many applications: In a computer network intrusion setting, we want the k best nodes to defend (e.g., through expensive and extensive vigilance), to minimize the spread of malware. Similarly, in a law-enforcement setting, given a network of criminals, we want to neutralize those nodes that will maximally scatter the graph.

The problem boils down to three questions, which we address here:

Q1. *Vulnerability measure:* How to capture the '*Vulnerability*' of the graph, in a single number? That is, how likely/easily that a graph will be infected by a virus.

Q2. 'Shield-value'*:* How to quantify the '*Shield-value*' of a given *set* of nodes in the graph, i.e., how important are they in terms of maintaining the '*Vulnerability*' of the graph? Alternatively, how much less vulnerable will be the graph to the virus attack, if those nodes are immunized/removed?

Q3. *Algorithm:* How to quickly determine the k nodes that *collectively* exhibit the highest '*Shield-value*' score on large, disk-resident graphs?

We start by adopting the first[1] eigenvalue λ of the graph as the '*Vulnerability*' measurement (for Q1). Based on that, we propose a novel definition of the '*Shield-value*' score $\mathrm{Sv}(\mathcal{S})$ for a specific set of nodes (for Q2). Finally, we propose an *effective* and *scalable* algorithm (*NetShield*) to find a set of nodes with highest '*Shield-value*' score (for Q3). We conduct extensive experiments on several real data sets, illustrating the effectiveness and efficiency of the proposed methods.

The rest of the paper is organized as follows: Section 2 gives the problem definitions. We present the '*Vulnerability*' measurement in Section 3. The proposed '*Shield-value*' score is presented in Section 4. We address the computational issues in Section 5. We evaluate the proposed methods in Section 6. Section 7 gives the related work, and Section 8 gives the conclusions.

2 Problem Definitions

Table 1 lists the main symbols we use throughout the paper. In this paper, we focus on un-directed un-weighted graphs. We represent the graph by its adjacency matrix. Following standard notations, we use capital bold letters for matrices (e.g., $\mathbf{A}$), lower-case bold letters for vectors (e.g., $\mathbf{a}$), and calligraphic fonts for sets (e.g., $\mathcal{S}$). We denote the transpose with a prime (i.e., $\mathbf{A}'$ is the transpose of $\mathbf{A}$), and we use parenthesized superscripts to denote the corresponding variable after deleting the nodes indexed by the superscripts. For example, λ is the first eigen-value of $\mathbf{A}$, then λ^i is the first eigen-value of $\mathbf{A}$ after deleting its $i^{(\text{th})}$ row/column. We use $(\lambda_i, \mathbf{u}_i)$ to denote the i^{th} eigen-pair (sorted by the magnitude of the eigenvalue) of $\mathbf{A}$. When the subscript is omitted, we refer to them as the first eigenvalue and eigenvector respectively (i.e., $\lambda \triangleq \lambda_1$ and $\mathbf{u} \triangleq \mathbf{u}_1$).

[1]In this paper, the first eigenvalue means the eigenvalue with the largest module.

Table 1. Symbols

Symbol	Definition and Description
$\mathbf{A}, \mathbf{B}, \ldots$	matrices (bold upper case)
$\mathbf{A}(i,j)$	the element at the i^{th} row and j^{th} column of matrix $\mathbf{A}$
$\mathbf{A}(i,:)$	the i^{th} row of matrix $\mathbf{A}$
$\mathbf{A}(:,j)$	the j^{th} column of matrix $\mathbf{A}$
$\mathbf{A}'$	transpose of matrix $\mathbf{A}$
$\mathbf{a}, \mathbf{b}, \ldots$	column vectors
$\mathcal{S}, \mathcal{T}, \ldots$	sets (calligraphic)
n	number of nodes in the graph
m	number of edges in the graph
$(\lambda_i, \mathbf{u}_i)$	the i^{th} eigen-pair of $\mathbf{A}$
λ	first eigen-value of $\mathbf{A}$ (i.e., $\lambda \triangleq \lambda_1$)
$\mathbf{u}$	first eigen-vector of $\mathbf{A}$ (i.e., $\mathbf{u} \triangleq \mathbf{u}_1$)
$\lambda^{(i)}$, $\lambda^{(\mathcal{S})}$	first eigen-value of $\mathbf{A}$ by deleting node i (or the set of nodes in $\mathcal{S}$)
$\Delta\lambda(i)$	eigen-drop: $\Delta\lambda(i) = \lambda - \lambda^{(i)}$
$\Delta\lambda(\mathcal{S})$	eigen-drop: $\Delta\lambda(\mathcal{S}) = \lambda - \lambda^{(\mathcal{S})}$
$\text{Sv}(i)$	'*Shield-value*' score of node i
$\text{Sv}(\mathcal{S})$	'*Shield-value*' score of nodes in $\mathcal{S}$
$\text{V}(\mathbf{G})$	'*Vulnerability*' score of the graph

With the above notations, our problems can be formally defined as follows:

Problem 1 Measuring '*Vulnerability*'

Given: *A large un-directed un-weighted connected graph G with adjacency matrix $\mathbf{A}$;*
Find: *A single number $V(\mathbf{G})$, reflecting the* 'Vulnerability' *of the whole graph.*

Problem 2 Measuring '*Shield-value*'

Given: *A subset $\mathcal{S}$ with k nodes in a large un-directed un-weighted connected graph $\mathbf{A}$;*
Find: *A single number $Sv(\mathcal{S})$, reflecting the* 'Shield-value' *of these k nodes (that is, the benefit of their removal/immunization to the vulnerability of the graph).*

Problem 3 Finding k Nodes of Best '*Shield-value*'

Given: *A large un-directed un-weighted connected graph $\mathbf{A}$ with n nodes and an integer k;*
Find: *A subset $\mathcal{S}$ of k nodes with the highest* 'Shield-value' *score among all $\binom{n}{k}$ possible subsets.*

In the next three sections, we present the corresponding solutions respectively.

3 Our Solution for Problem 1

Here, we focus on Problem 1. We suggest using the first eigenvalue λ as the solution.

3.1 '*Vulnerability*' Score

In Problem 1, the goal is to measure the '*Vulnerability*' of the whole graph by a single number. We adopt the first eigenvalue of the adjacency matrix $\mathbf{A}$ as such a measurement (eq. (1)): the larger λ is, the more vulnerable the whole graph is.

$$\text{V}(\mathbf{G}) \triangleq \lambda \tag{1}$$

3.2 Justifications

The first eigenvalue λ is a good measurement of the graph '*Vulnerability*', because of recent results on *epidemic thresholds* from immunology [2]: λ is closely related to the epidemic threshold τ of a graph under a flu-like SIS (susceptible-infective-susceptible) epidemic model, and specifically $\tau = 1/\lambda$. This means that a virus less infective than τ will quickly get extinguished instead of lingering forever. Therefore, given the strength of the virus (that is, the infection rate and the host-recovery rate), an epidemic is more likely for a graph with larger λ.

We can also show that the first eigenvalue λ is closely related to the so-called *loop capacity* and the *path capacity* of the graph, that is, the number of loops and paths of length l ($l = 2, 3, \ldots$). If a graph has many such loops and paths, then it is well connected, and thus more vulnerable (i.e., it is easier for a virus to propagate across the graph = the graph is less robust to the virus attack).

4 Our Solution for Problem 2

In Problem 2, the goal is to quantify the importance of a given set of nodes $\mathcal{S}$, and specifically the impact of their deletion/immunization to the '*Vulnerability*' of the rest of the graph. The obvious choice is the drop in eigenvalue, or *eigen-drop* $\Delta\lambda$ that their removal will cause to the graph. We propose to approximate it, to obtain efficient computations, as we describe later. Specifically, we propose using $\text{Sv}(\mathcal{S})$ defined as:

$$\text{Sv}(\mathcal{S}) = \sum_{i\in\mathcal{S}} 2\lambda\mathbf{u}(i)^2 - \sum_{i,j\in\mathcal{S}} \mathbf{A}(i,j)\mathbf{u}(i)\mathbf{u}(j) \tag{2}$$

Intuitively, by eq. (2), a set of nodes $\mathcal{S}$ has higher '*Shield-value*' score if (1) each of them has a high eigen-score ($\mathbf{u}(i)$), and (2) they are dissimilar with each other (small or zero $\mathbf{A}(i,j)$).

5 Our Solution for Problem 3

In this section, we deal with Problem 3. Here, the goal is to find a subset of k nodes with the highest '*Shield-value*' score (among all $\binom{n}{k}$ possible subsets).

5.1 Preliminaries

There are two obviously straight-forward methods for Problem 3. The first one (referred to as 'Com-Eigs'[2]) works

[2]To our best knowledge, this is the best known method to get the optimal solution of Problem 3.

as follows: for each possible subset $\mathcal{S}$, we delete the corresponding rows/columns from the adjacency matrix $\mathbf{A}$; compute the first eigenvalue of the new perturbed adjacency matrix; and finally output the subset of nodes which has the smallest eigenvalue (therefore has the largest eigen-drop). Despite the simplicity of this strategy, it is computational intractable due to its combinatorial nature. It is easy to show that the computational complexity of 'Com-Eigs' is $O(\binom{n}{k} \cdot m)$[3]. This is computationally intractable even for small graphs. For example, in a graph with 1K nodes and 10K edges, suppose that it takes about 0.01 second to find its first eigen-value. Then we need about 2,615 years to find the best-5 nodes with the highest '*Shield-value*' score!

A more reasonable (in terms of speed) way to find the best-k nodes is to evaluate Sv$(\mathcal{S})$, rather than to compute the first eigen-value $\lambda_{\mathcal{S}}$, $\binom{n}{k}$ times, and pick the subset with the highest Sv$(\mathcal{S})$. We refer to this strategy as 'Com-Eval'. Compared with the straight-forward method (referred to as 'Com-Eigs', which is $O(\binom{n}{k} \cdot m)$); 'Com-Eval' is much faster ($O(\binom{n}{k} \cdot k^2)$). However, 'Com-Eval' is still not applicable to real applications due to its combinatorial nature. Again, in a graph with 1K nodes and 10K edges, suppose that it only takes about 0.00001 second to evaluate Sv$(\mathcal{S})$ once. Then we still need about 3 months to find the best-5 nodes with the highest '*Shield-value*' score!

5.2 Proposed *NetShield* Algorithm

The proposed *NetShield* is given in Alg. 1. In Alg. 1, we compute the first eigenvalue λ and the corresponding eigenvector $\mathbf{u}$ in step 1. In step 4, the $n \times 1$ vector $\mathbf{v}$ measures the '*Shield-value*' score of each individual node. Then, in each iteration of steps 6-17, we greedily select one more node and add it into set $\mathcal{S}$ according to score(j) (step 13). Note that steps 10-12 are to exclude those nodes that are already in the selected set $\mathcal{S}$.

6 Experimental Evaluations

We present detailed experimental results in this section. All the experiments are designed to answer the following questions:

1: (*Effectiveness*) How effective is the proposed Sv$(\mathcal{S})$ in real graphs?

2: (*Efficiency*) How fast and scalable is the proposed *NetShield*?

6.1 Data sets

We used three real data sets, which are summarized in table 2. The first data set (*Karate*) is a unipartite graph, which describes the friendship among the 34 members of a karate club at a US university [17]. Each node is a member

[3]We assume that k is relatively small compared with n and m (e.g., tens or hundreds). Therefore, after deleting k rows/columns from $\mathbf{A}$, we still have $O(m)$ edges.

Algorithm 1 *NetShield*

```
Input: the adjacency matrix $\mathbf{A}$ and an integer $k$
Output: a set $\mathcal{S}$ with $k$ nodes
 1: compute the first eigen-value $\lambda$ of $\mathbf{A}$; let $\mathbf{u}$ be the corresponding eigen-vector $\mathbf{u}(j)(j = 1, ..., n)$;
 2: initialize $\mathcal{S}$ to be empty;
 3: for $j = 1$ to $n$ do
 4:   $\mathbf{v}(j) = (2 \cdot \lambda - \mathbf{A}(j,j)) \cdot u(j)^2$;
 5: end for
 6: for iter $= 1$ to $k$ do
 7:   let $\mathbf{B} = \mathbf{A}(:, \mathcal{S})$;
 8:   let $\mathbf{b} = \mathbf{B} \cdot \mathbf{u}(\mathcal{S})$;
 9:   for $j = 1$ to $n$ do
10:     if $j \in \mathcal{S}$ then
11:       let score$(j) = -1$;
12:     else
13:       let score$(j) = \mathbf{v}(j) - 2 \cdot \mathbf{b}(j) \cdot \mathbf{u}(j)$;
14:     end if
15:   end for
16:   let $i = \text{argmax}_j$ score$(j)$, add $i$ to set $\mathcal{S}$;
17: end for
18: return $\mathcal{S}$.
```

Table 2. Summary of the data sets

Name	n	m
Karate	34	152
AA	418,236	2,753,798
NetFlix	2,667,199	171,460,874

in the karate club and the existence of the edge indicates that the two corresponding members are friends. Overall, we have $n = 34$ nodes and $m = 156$ edges.

The second data set (*AA*) is an author-author network from DBLP.[4] *AA* is a co-authorship network, where each node is an author and the existence of an edge indicates the co-authorship between the two corresponding persons. Overall, we have $n = 418,236$ nodes and $m = 2,753,798$ edges. We also construct much smaller co-authorship networks, using the authors from only one conference (e.g., *KDD*, *SIGIR*, *SIGMOD*, etc.). For example, *KDD* is the co-authorship network for the authors in the 'KDD' conference. For these smaller co-authorship networks, they typically have a few thousand nodes and up to a few ten thousand edges.

The last data set (*NetFlix*) is from the Netflix prize.[5] This is also a bipartite graph. We have two types of nodes: user and movie. The existence of an edge indicates that the corresponding user has rated the corresponding movie. Overall, we have $n = 2,667,199$ nodes and $m = 171,460,874$ edges. This is a bipartite graph, and we convert it to a unipartite graph $\mathbf{A}$: $\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}' & \mathbf{0} \end{pmatrix}$, where $\mathbf{0}$ is a matrix with

[4]`http://www.informatik.uni-trier.de/~ley/db/`
[5]`http://www.netflixprize.com/`

Table 3. Evaluation on the approximation accuracy of $f(\mathcal{S})$. Larger is better.

k	'KDD'	'ICDM'	'SDM'	'SIGMOD'
1	0.9519	0.9908	0.9995	1.0000
2	0.9629	0.9910	0.9984	0.9927
5	0.9721	0.9888	0.9992	0.9895
10	0.9726	0.9863	0.9987	0.9852
20	0.9683	0.9798	0.9929	0.9772

all zero entries and $\mathbf{B}$ is the adjacency matrix of the bipartite graph.

6.2 Effectiveness

Here, we first test the approximation accuracy of the proposed $\mathrm{Sv}(\mathcal{S})$. Then, we compared the different immunization policies.

6.2.1 Approximation quality of $\mathbf{Sv}(\mathcal{S})$

The proposed *NetShield* is based on eq. (2). That is, we want to approximate the first eigen-value of the perturbed matrix by λ and $\mathbf{u}$. Here, let us experimentally evaluate how good this approximation is on real graphs. We construct an authorship network from one of the following conferences: *'KDD', 'ICDM', 'SDM' and 'SIGMOD'*. We then compute the linear correlation coefficient between $\Delta\lambda(\mathcal{S})$ and $\mathrm{Sv}(\mathcal{S})$ with several different k values ($k = 1, 2, 5, 10, 20$). The results are shown in table 3. It can be seen that the approximation is very good - in all the cases, the linear correlation coefficient is greater than 0.95.

6.2.2 Immunization by *NetShield*

The proposed '*Vulnerability*' score of the graph is motivated by the epidemic threshold [2]. As a consequence, the proposed *NetShield* leads to a natural immunization strategy for the SIS model (susceptible-infective-susceptible, like, e.g., the flu): quarantine or delete the subset of the nodes detected by *NetShield* in order to prevent an epidemic from breaking out. [6]

We compare it with the following alternative choices: (1) picking a random neighbor of a randomly chosen node[3] ('Aquaintance'), (2) picking the nodes with the highest eigen scores $\mathbf{u}(i)(i = 1, ..., n)$ ('Eigs')[7], (3) picking the nodes with the highest abnormality scores [15] ('abnormality'), (4) picking the nodes with the highest betweenness centrality scores based on the shortest path [6]('Short'), (5) picking the nodes with the highest betweenness centrality scores based on random walks [11]('N.RW'), (6) picking the nodes with the highest degress ('Degree'), and (7) picking the nodes with the highest PageRank scores [13]('PageRank'). For each method, we delete 5 nodes for immunization. Let $s = \lambda \cdot b/d$ be the normalized virus strength (bigger s means more stronger virus), where b and d are the infection rate and death rate, respectively. The result is presented in figure 1, which is averaged over 1000 runs. It can be seen that the proposed *NetShield* is always the best, - its curve is always the lowest which means that we always have the least number of infected nodes in the graph with this immunization strategy. Notice that the performance of 'Eigs' is much worse than the proposed *NetShield*. This indicates that by *collectively* finding a set of nodes with the highest '*Shield-value*', we indeed obtain extra performance gain (compared with naïvely choosing the top-k nodes which have the highest *individual* '*Shield-value*' scores).

[6]According to [2], for SIR (susceptible-infective-recovered) model, its epidemic threshold is also determined by λ. Therefore, we expect that our *NetShield* can also help with the immunization for the SIR model.

[7]For the un-directed graph which we focus on in this paper, 'Eigs' is equivalent to 'HITS'[9].

6.3 Efficiency

We will study the wall-clock running time of the proposed *NetShield* here. Basically, we want to answer the following two questions:

1. *(Speed)* What is the speedup of the proposed *NetShield* over the straight-forward methods ('Com-Eigs' and 'Com-Eval')?
2. *(Scalability)* How does *NetShield* scale with the size of the graph (n and m) and k?

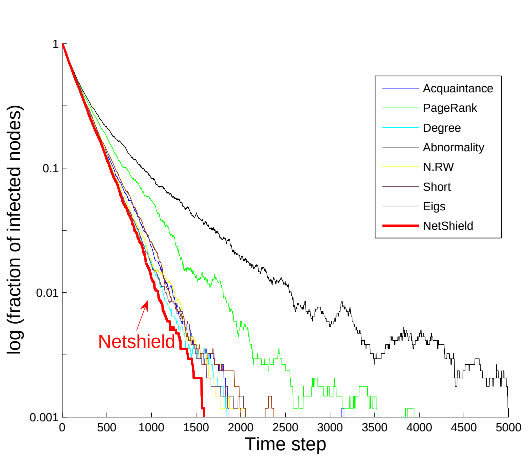
For the results we report in this subsection, all of the experiments are done on the same machine with four 2.4GHz AMD CPUs and 48GB memory, running Linux (2.6 kernel). If the program takes more than 1,000,000 seconds, we stop running it.

First, we compare *NetShield* with 'Com-Eigs' and 'Com-Eval'. Figure 2 shows the comparison on three real data sets. We can make the following conclusions: (1) Straight-forward methods ('Com-Eigs' and 'Com-Eval') are computationally intractable even for a small graph. For example, on the *Karate* data set with only 34 nodes, it takes more than 100,000 and 1,000 seconds to find the best-10 by 'Com-Eigs' and by 'Com-Eval', respectively. (2) The speedup of the proposed *NetShield* over both 'Com-Eigs' and 'Com-Eval' is huge - in most cases, we achieve *several (up to 7) orders of magnitude* speedups! (3) The speedup of the proposed *NetShield* over both 'Com-Eigs' and 'Com-Eval' quickly increases wrt the size of the graph as well as k. (4) For a given size of the graph (fixed n and m), the wall-clock time is almost constant - suggesting that *NetShield* spends most of its running time in computing λ and $\mathbf{u}$.

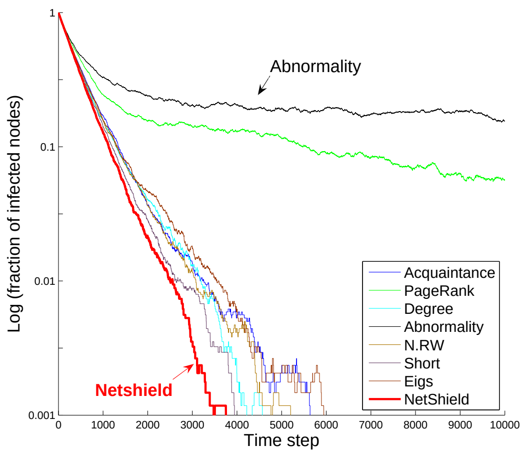
Next, we evaluate the scalability of *NetShield*. From figure 3, it can be seen that *NetShield* scales linearly wrt both n and m, which means that it is suitable for large graphs.

7 Related Work

In this section, we review the related work, which can be categorized into 3 parts: measuring the importance of nodes on graphs, immunization, and spectral graph analysis.

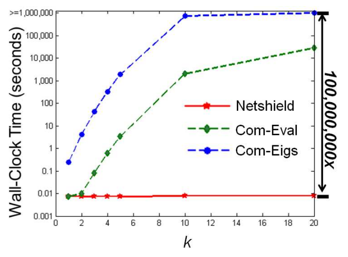


(a) $s = 1.4$

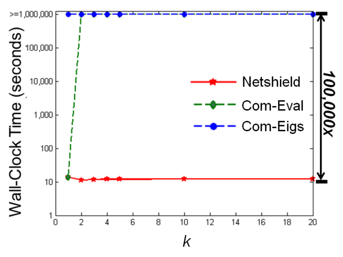


(b) $s = 2.9$

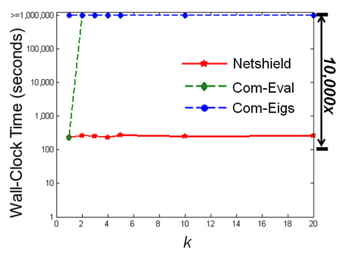
Figure 1. *Evaluation of immunization of* NetShield *on the* Karate *graph. The fraction of infected nodes (in log-scale) vs. the time step. s is normalized virus strength. Lower is better. The proposed* NetShield *is always the best, leading to the fastest healing of the graph. Best viewed in color.*



(a) *Karate*



(b) *AA*



(c) *NetFlix*

Figure 2. *Wall-clock time vs. the budget k for different methods. The time is in the logarithmic scale. Our* NetShield *(red star) is much faster. Lower is better.*

Measuring Importance of Nodes on Graphs. In the literature, there are a lot of node importance measurements, including betweeness centrality, both the one based on the shortest path [6] and the one based on random walks [11], PageRank [13], HITS [9], and coreness score (defined by k-core decomposition) [10].

Immunization. There is vast literature on virus propagation and epidemic thresholds: for full cliques (eg., Hethcote [8]), for power-law graphs [1], and studies of heuristics for immunization policies [3]. The only papers that study *arbitrary* graphs focus on the epidemic threshold (Wang et al. [16] and its follow-up work [7],[2].

In short, none of the above papers solves the problem of optimal immunization for an arbitrary, given graph.

Spectral Graph Analysis. Pioneering works in this aspect can be traced back to Fiedler's seminal work [5]. Representative follow-up works include [14, 12, 18, 4], etc. All of these works use the eigen-vectors of the graph (or the graph Laplacian) to find communities in the graph.

8 Conclusion

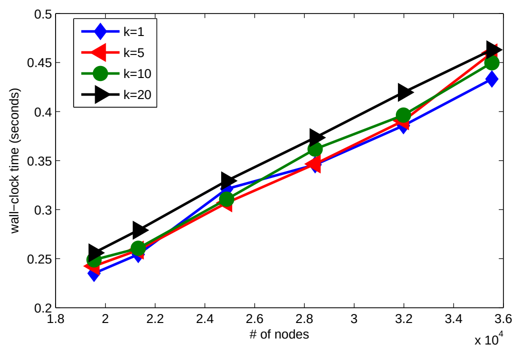
We studied the '*Vulnerability*' of large real graphs. Besides the problem definitions, our main contributions are

1. A novel definition of '*Shield-value*' score $\text{Sv}(\mathcal{S})$ for a set of nodes $\mathcal{S}$.
2. A *effective* and *scalable* algorithm (*NetShield*) to find a set of nodes with the highest '*Shield-value*' score.
3. Extensive experiments on several real data sets, illustrating both the effectiveness as well as the efficiency of our methods.

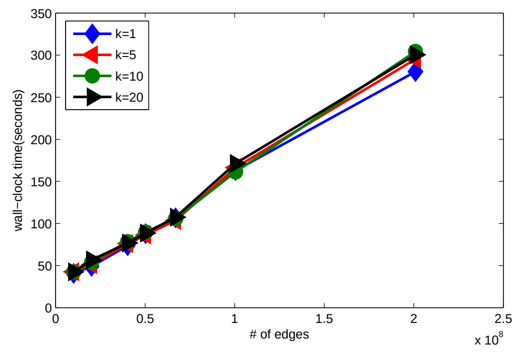
Specifically, the proposed method (a) gives an effective immunization strategy (b) scales linearly with the size of the graph (number of edges) and (c) outperforms competitors by several *orders of magnitude*.

A promising research direction is to parallelize the current method (e.g., using Hadoop[8]). Another one is to study extensions for additional virus propagation models, like SIR [8] etc.

[8]http://hadoop.apache.org/



(a) changing n (fix $m = 119,460$)



(b) changing m (fix $n = 2,667,119$)

Figure 3. *Evaluation of the scalability of the proposed* NetShield *wrt.* n *(number of nodes) and* m *(number of edges), respectively. The wall-clock time of our* NetShield *scales linearly wrt* n *and* m*.*

9 Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants No. DBI-0640543 IIS-0705359 CNS-0721736 IIS0808661 IIS1017415 iCAST and under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. DE-AC52-07NA27344.

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

This work is also partially supported by an IBM Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

References

[1] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. *WORM 2003*, Oct. 27 2003.

[2] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (ACM TISSEC)*, 10(4), 2007.

[3] R. Cohen, S. Havlin, and D. ben Avraham. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 91(24), Dec. 2003.

[4] C. H. Q. Ding, T. Li, and M. I. Jordan. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *ICDM*, pages 183–192, 2008.

[5] M. Fiedler. Algebraic connectivity of graphs. 1973.

[6] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[7] A. Ganesh, E. Massouli, and D. Towsley. The effect of network topology on the spread of epidemics. In *INFOCOM*, 2005.

[8] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42:599–653, 2000.

[9] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[10] J. Moody and D. R. White. Social cohesion and embeddedness: A hierarchical conception of social groups. *American Sociological Review*, pages 1–25, 2003.

[11] M. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27:39–54, 2005.

[12] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).

[14] J. Shi and J. Malik. Normalized cuts and image segmentation. In *CVPR*, pages 731–737, 1997.

[15] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*, pages 418–425, 2005.

[16] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. *SRDS*, 2003.

[17] W. W. Zachary. An information flow model for conflict and fission in small groups. pages 452–473, 1977.

[18] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In *NIPS*, pages 1057–1064, 2001.