

# Combinatorial Trace Method for Network Immunization

Muhammad Ahmad<sup>a</sup>, Sarwan Ali<sup>a</sup>, Juvaria Tariq<sup>a</sup>, Imdadullah Khan<sup>a,\*</sup>,  
Mudassir Shabbir<sup>b</sup>, Arif Zaman<sup>a</sup>

<sup>a</sup>*Lahore University of Management Sciences, Pakistan*

<sup>b</sup>*Information Technology University, Pakistan*

---

## Abstract

Immunizing a subset of nodes in a network - enabling them to identify and withstand the spread of harmful content - is one of the most effective ways to counter the spread of malicious content. It has applications in network security, public health policy, and social media surveillance. Finding a subset of nodes whose immunization results in the least vulnerability of the network is a computationally challenging task. In this work, we establish a relationship between a widely used network vulnerability measure and the combinatorial properties of networks. Using this relationship and graph summarization techniques, we propose an efficient approximation algorithm to find a set of nodes to immunize. We provide theoretical justifications for the proposed solution and analytical bounds on the runtime of our algorithm. We empirically demonstrate on various real-world networks that the performance of our algorithm is an order of magnitude better than the state of the art solution. We also show that in practice the runtime of our algorithm is significantly lower than that of the best-known solution.

*Keywords:* Network Immunization, Spectral Methods, Combinatorial Trace, Eigendrop, Closed Walks

---



---

\*Corresponding author

*Email addresses:* 17030056@lums.edu.pk (Muhammad Ahmad), 16030030@lums.edu.pk (Sarwan Ali), 14070004@lums.edu.pk (Juvaria Tariq), imdad.khan@lums.edu.pk (Imdadullah Khan), mudassir.shabbir@itu.edu.pk (Mudassir Shabbir), arifz@lums.edu.pk (Arif Zaman)

## 1. Introduction

Graphs or networks are used to model many practical scenarios involving pairwise interactions between entities. The entities could be humans, computers, mobile devices, power components, etc. while interactions can be face-to-face meetings, email and SMS communication and various kind of flows e.g. electric current in a power infrastructure network or fluid in pipelines. Many of the practical networks are very large with millions of nodes and edges.

Every interaction in such large networks can not be monitored and there is a possibility of undesired and potentially harmful communication taking place among entities in networks. Such undesired spread could be intentional or unintentional entailing various degrees of harms. The unintentional spread of flu-virus, for instance, may be life-threatening and may cause an epidemic. A rumor, on the other hand, may well be originated intentionally and its effect might be limited to a particular segment of a network. An effective way to safeguard a network against the spread of malicious content is to *empower* the nodes. The strengthening process may amount to vaccinating people, deploying surveillance systems at junctures and installing anti-virus software on computers depending on the underlying network. The nodes with these added capabilities will be referred to as the *immunized* nodes and the malicious content, as the *virus*. Effectively, when a node is immunized, it will neither get contaminated nor will it pass the contaminant to other nodes.

There is a cost associated with immunization, hence it is not feasible to immunize all nodes in large networks. The problem to select a subset of nodes (not exceeding a given budget) for immunization that will maximally hinder the virus spread is called the *Network Immunization Problem* and is abstractly formulated in [9] as follows:

**Problem 1.** *Given an undirected graph  $G = (V, E)$ ,  $|V| = n$  and an integer  $k < n$ , find a subset  $S$  of  $k$  nodes such that immunizing nodes in  $S$ , renders  $G$  the least ‘vulnerable’ to a virus attack over all choices of  $S$ .*

This requires a quantitative measure for the vulnerability of the graph. As in

the literature [9, 8, 3, 30], we use the largest eigenvalue of the adjacency matrix of the graph to quantify the graph vulnerability. The objective in Problem 1, therefore becomes that of immunizing a fixed-sized subset so as the remaining graph has the minimum largest eigenvalue. More precisely,

**Problem 2.** *Given an undirected graph  $G = (V, E)$ ,  $|V| = n$  and an integer  $k < n$ , find a subset  $S$  of  $k$  nodes such that the largest eigenvalue of the adjacency matrix of  $G - S$  (the matrix after removing the rows and columns corresponding to  $S$ ) is minimum over all choices of  $S$ .*

A score function was proposed in [3, 30], for a subset of nodes based on the number of small length closed walks a node is contained in. The number of fixed length closed walks containing a node co-relates with the node’s contribution towards the largest eigenvalue of the adjacency matrix of the graph. However, while the longer walks provide a better approximation, only shorter walks were considered due to time complexity. In this work, we propose a randomized approximation approach to address the time complexity issue, and extend to walks of length 8, resulting in considerable improvement in accuracy. Formally, the contribution of this work can be summarized as follows:

- We extend the score function based on the number of closed walks of length 8 for sets of nodes that quantify the importance of sets to reduce the graph vulnerability defined in [3]. This score function is monotonically non-decreasing and sub-modular that enables employing greedily constructing a set with improved approximation quality
- We derive a closed-form formula to compute the number of walks of length 8 passing through a node which may be of independent interest. We also give an approximate method that closely estimates the number of walks of length 8 passing through a node
- We evaluate the quality of our solution on several real-world graphs. We show that our approximate method is a close estimate of the exact solution. Results show that our approach maximally reduces the virus spread and

the vulnerability (the largest eigenvalue) of the immunized graph. Moreover, our algorithm is scalable on large graphs and has a lower runtime based on the approximation parameters used. Comparisons also demonstrate that our approach outperforms the state-of-art methods both in terms of quality and runtime

The rest of the paper is organized as follows. In Section 2, we discuss the related work and give the background of the problem in Section 3. In Section 4, we present our solution along with its analysis. We report experimental results and comparisons with the existing solution in Section 7.

## 2. Related Work

Information spread in networks is widely studied in epidemiology, sociology and information sciences. Researchers are usually interested in estimating the extent to which a contagion will affect the population, predicting the timeline of infection and methods for containing or limiting the effect. The spreading process is studied on a network: agents are represented by nodes and the potential spread of information between a pair of agents is modeled by the presence of an edge between the corresponding pair of nodes. Popular models assume the knowledge of an infection rate  $\beta$  (the rate at which an individual/agent accepts content from its neighbors) and a rate of recovery  $\delta$  (the rate at which an individual/agent loses content). A relation between spread rate of virus and the largest eigenvalue of adjacency matrix  $A$  of the graph,  $\lambda_{max}(A)$ , was established in [36, 15]. In particular, they showed that if  $\beta < \delta/\lambda_{max}(A)$ , then the infection dies out in sub-linear time with respect to the size of the population following a stochastic model. Similarly, an exponential lower bound on expected die-out time or time for full network recovery (i.e.  $\geq e^{cN}$  where  $c$  is a constant dependent on the infection rate and  $N$  is the size of population) is also known when  $\beta > \delta/\lambda_{max}(A)$  [32, 33]. Recent works of [4, 16, 17] established a similar relation of infection and recovery rates with  $\lambda_{max}$  for infection spread or die-out while approximating the stochastic model by a deterministic one.

Various studies have proposed preemptive methods to control virus spread and avoid a potential outbreak of contagion. These methods remove a subset of nodes or edges from the graph, so the remaining graph has the least  $\lambda_{max}$ . This problem has been shown to be NP-COMPLETE in [9, 34]. An approximation scheme to select nodes for immunization based on eigenvector corresponding to  $\lambda_{max}$  of the graph is devised in [9].

A combinatorial trace method is adopted in [3, 30, 2] to select a subset of nodes whose removal will result in the maximum reduction in the  $\lambda_{max}$ . The trace of a large power of an adjacency matrix,  $A^p$ , is closely related to the  $\lambda_{max}(A)$  (also known as the spectral radius of the graph) [1]. Trace of  $A^p$ , on the other hand, is just the count of the number of closed walks of length  $p$  in the graph. Approximation algorithms are given in [3, 2] to select nodes removing which will eliminate the most number of closed walks of length 4 from the graph. Approximation of the number of closed walks of length 6 containing a node using a randomly constructed summary of a graph is given in [30]. In this paper, we extend this work by considering walks of length 8 that leads to improved quality. We note that more sophisticated techniques from graph summarization literature [19, 25, 26, 6] could be utilized to improve this work.

Edge removal techniques are also devised to minimize graph vulnerability. Methods for selecting edges whose removal will reduce  $\lambda_{max}$  the most are devised in [18, 31]. In [18] virus spread is modeled by the dynamical system and the transition function which defines the interaction of a node with its neighbors and state of each node (healthy or infected) in order to reduce  $\lambda_{max}$ .

In another line of work, non-preemptive techniques are devised in [38, 39, 28] to immunize select nodes after the virus spread has started and the healthy and infected nodes are known. In this setting, methods are evaluated by *save ratio* ( $SR$ ): the ratio of the number of affected nodes in a graph when  $k$  nodes are immunized to the number of infected nodes in case of no immunization.

A reverse engineering technique is used to identify the nodes in a graph where the virus spread is initiated [24]. A related problem is to decontaminate the graph by deploying *cleaning agents* at certain nodes that travel along edges.

Monotonicity is assumed in [7, 13, 12, 14] that a node cleaned by the agent will not get affected again. Non-monotonic strategies are given in [10].

Some other problems related to graph immunization include the influence maximization [21], the filter placement [11] and the critical node detection problem (CNDP) [5, 20, 35]. In the influence maximization problem, the goal is to find a subset of nodes whose *activation* will lead to the maximal spread of information across the graph. The filter placement problem deals with minimizing the multiplicity of information flowing across the network. In CNDP, the goal is to identify nodes whose removal results in maximum graph fragmentation.

### 3. Preliminaries

In this section, we formulate the immunization problem. Given a simple graph  $G = (V, E)$ , the goal is to select a subset  $S$  of  $k$  nodes such that removing  $S$  from the graph maximally reduces the largest eigenvalue of the remaining graph denoted by  $\lambda_{\max}(A|_{-S})$ . Since  $\lambda_{\max}$  can be computed in  $O(|E|)$ , the optimal subset of nodes can be found by iterating through each of the  $\binom{n}{k}$  subsets. The overall runtime of this brute force algorithm is  $O(\binom{n}{k} \cdot |E|)$  rendering it computationally infeasible even for moderately large graphs.

Indeed, it turns out that Problem 2 is NP-HARD. A reduction from *Minimum Vertex Cover Problem* is as follows: if there exists a set  $S$  with  $|S| = k$  such that  $\lambda_{\max}(A|_{-S}) = 0$ , then  $S$  is a vertex cover of the graph. It follows from the following implication of famous *Perron-Frobenius theorem*:

**Fact 1.** *Deleting an edge from a simple connected graph  $G$  strictly decreases the largest eigenvalue of the corresponding adjacency matrix [27].*

Also, if there is a vertex cover  $S$  of the graph such that  $|S| = k$ , then deleting  $S$  will result in an empty graph which has eigenvalue zero.

Although Problem 2 is NP-HARD, its objective function is monotone and sub-modular. The greedy algorithm (GREEDY-1) guarantees  $(1+1/e)$ -approximation ( $e$  is the base of the natural logarithm) to Problem 2 by Theorem 1.

**Theorem 1.** [22] Let  $f$  be a non-negative, monotone and submodular function,  $f : 2^\Omega \rightarrow \mathbb{R}$ . Suppose  $\mathcal{A}$  is an algorithm, that chooses a  $k$  elements set  $S$  by adding an element  $u$  at each step such that  $u = \arg \max_{x \in \Omega \setminus S} f(S \cup \{x\})$ . Then  $\mathcal{A}$  is  $(1 + 1/e)$ -approximate algorithm.

---

**Algorithm 1 :** GREEDY-1 ( $G, k$ )

---

```

 $S \leftarrow \emptyset$ 
while  $|S| < k$  do
     $v \leftarrow \arg \min_{x \in V \setminus S} (\lambda_1(A_{- \{S \cup \{x\}\}}))$ 
     $S \leftarrow S \cup \{v\}$ 
return  $S$ 

```

---

We refer to the achieved benefit after immunizing subset  $S$  as *eigendrop* and is defined as  $\lambda_{max}(A) - \lambda_{max}(A|_{-S})$ . A score, termed as shield-value, is assigned to each subset  $S \subset V$ , which quantifies the approximated eigendrop achieved after removing  $S$ . Frequently used symbols in the paper are listed in Table 1.

Symbol	Definition & Description
$A$	adjacency matrix of the graph $G$
$G _{-S}$	subgraph after removing node set $S$ from the graph $G$
$A _{-S}$	adjacency matrix of the graph $G _{-S}$
$\lambda_i(A)$	$i^{th}$ largest eigen value of matrix $A$ on the basis of magnitude
$\lambda_{max}(A)$	the largest eigen value of matrix $A$ i.e. $\lambda_{max}(A) = \lambda_1(A)$
$\Delta\lambda(S)$	$\lambda_{max}(A) - \lambda_{max}(A _{-S})$ ; eigendrop achieved by immunizing node set $S$
$A^p$	$p^{th}$ power of (adjacency) matrix $A$
$\mathcal{CW}_p(v, G)$	the set of $p$ -length closed walks in $G$ containing $v$
$\mathcal{CW}_p(S, G)$	the set of $p$ -length closed walks in $G$ containing at least one vertex from $S$
$\mathcal{W}_p(v, G)$	number of $p$ -length closed walks in $G$ containing $v$
$\mathcal{W}_p(S, G)$	number of $p$ -length closed walks in $G$ containing at least one vertex from $S$
$d_G(v)$	degree of node $v$ in graph $G$

---

Table 1: List of Symbols

#### 4. Proposed Shield Value

In this section, we quantify the importance of a subset of nodes for immunization. We first derive a score for each set of size  $k$  that closely measures the value of the objective function of Problem 2. We prove that this score function is monotonically increasing and submodular. Using Theorem 1 we can greedily build up the set  $S$  by iteratively selecting nodes that are contained in the maximum number of closed walks of length  $p$ .

Let  $A$  be an  $n \times n$  matrix; the following two fundamental results from algebraic graph theory [29, 37, 23] relate the eigen spectrum and the trace of  $A$ .

**Fact 2.**

$$\text{trace}(A) = \sum_{i=1}^n A(i, i) = \sum_{i=1}^n \lambda_i(A)$$

**Fact 3.**

$$\text{trace}(A^p) = \sum_{i=1}^n \lambda_i(A^p) = \sum_{i=1}^n (\lambda_i(A))^p$$

From the theory of vector norms [29] and Fact 3 we know that

$$\begin{aligned} \lim_{\substack{p \rightarrow \infty \\ p \text{ even}}} (\text{trace}(A^p))^{1/p} &= \lim_{\substack{p \rightarrow \infty \\ p \text{ even}}} \left( \sum_{i=1}^n \lambda_i(A)^p \right)^{1/p} \\ &= \lim_{p \rightarrow \infty} \left( \sum_{i=1}^n |\lambda_i(A)|^p \right)^{1/p} = \max_i \{\lambda_i(A)\} = \lambda_{\max}(A) \end{aligned}$$

Using the above relation we establish that for the immunization problem, we want to find a subset  $S$  of nodes in graph  $G$  which, when removed, minimizes  $\text{trace}((A|_{-S})^p)$ . Next, we derive a combinatorial form of this objective function.

As described in Table 1, for a vertex  $v \in V(G)$ ,  $\mathcal{CW}_p(v, G)$  is the set of all closed walks of length  $p$  in the graph  $G$  containing  $v$  at least once and  $W_p(v, G) = |\mathcal{CW}_p(v, G)|$ . Similarly,  $\mathcal{CW}_p(S, G)$  denotes the set of closed walks of length  $p$  in  $G$  containing at least one vertex from  $S$  and correspondingly  $W_p(S, G) = |\mathcal{CW}_p(S, G)|$ . We use the following combinatorial definition of  $\text{trace}$ .



**Fact 4.** [37] Given a graph  $G = (V, E)$  with adjacency matrix  $A$ ,

$$\mathcal{W}_p(V, G) = \text{trace}(A^p)$$

From Fact 4 and definition of trace (Fact 2), we get that

$$\mathcal{W}_p(V, G) = \mathcal{W}_p(V \setminus S, G|_{-S}) + \mathcal{W}_p(S, G) \quad (1)$$

This is true because any walk in  $G$  either contains some vertex in  $S$  or it does not contain any vertex in  $S$ . The former type of walks are counted exactly once in the term  $\mathcal{W}_p(S, G)$ , while the first term counts closed walks of the latter type. Equation (1) can be equivalently rewritten as

$$\begin{aligned} \text{trace}(A^p) &= \text{trace}((A|_{-S})^p) + \mathcal{W}_p(S, G) \\ \implies \text{trace}((A|_{-S})^p) &= \text{trace}(A^p) - \mathcal{W}_p(S, G) \end{aligned}$$

Thus for a fixed graph  $G$  (since  $\text{trace}(A^p)$  is constant) minimizing  $\text{trace}((A|_{-S})^p)$  is equivalent to maximizing  $\mathcal{W}_p(S, G)$ . This implies that the set  $S$  with the largest value of  $\mathcal{W}_p(S, G)$  will yield the maximum eigendrop. Intuitively, we need to identify nodes contained in many closed walks of length  $p$  (nodes with high  $\mathcal{W}_p(v, G)$ ). We define the following shield value of a set  $S$ , that in addition to maximizing  $\mathcal{W}_p(S, G)$ , attempts to select those nodes which are far from each other i.e. having  $A(u, v) = 0$  in order to maximize the number of distinct closed walks going through nodes in a set  $S$ .

$$\text{score}_p(S) = \gamma \sum_{v \in S} \mathcal{W}_p(v, G)^2 - \sum_{u, v \in S} \mathcal{W}_p(v, G) A(u, v) \mathcal{W}_p(u, G), \quad (2)$$

where  $\gamma$  is a positive constant. Hence Problem 2 can be rephrased as follows.

**Problem 3.** Let  $G = (V, E)$  be an undirected graph on  $n$  nodes and let  $k$  be an integer  $k < n$ , find a subset of nodes  $S \subset V$ , with  $|S| = k$  such that  $\text{score}_p(S)$  is the maximum over all  $k$ -subsets of  $V$ .

For fixed  $p$ , given  $\mathcal{W}_p(v, G), \forall v \in V$ ,  $\text{score}_p(S)$  can be evaluated in time  $O(k^2)$ . Selecting a set with maximum  $\text{score}_p(S)$  takes  $O(\binom{n}{k} k^2)$  time which

clearly is computationally prohibitive. Furthermore, note that for this we need to have the values of  $\mathcal{W}_p(v, G)$  pre-computed, which is not straight-forward.

We show that the objective function of Problem 3 is monotone and submodular. Given  $\mathcal{W}_p(v, G)$ , by Theorem 1, the greedy strategy for building up the set will yield  $(1 - 1/e)$ -approximation of the optimal subset.

**Theorem 2.** *For  $p \geq 1$ ,  $score_p(S)$  is monotonically non-decreasing.*

*Proof.* We prove that for any  $X \subset Y \subseteq V$ ,  $score_p(X) \leq score_p(Y)$ . Let  $E, F \subset V(G)$  and  $x \in V(G)$  such that  $F = E \cup \{x\}$ . Consider

$$\begin{aligned}
& score_p(F) - score_p(E) \\
&= \gamma \sum_{v \in F} \mathcal{W}_p(v)^2 - \sum_{u, v \in F} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) - \gamma \sum_{v \in E} \mathcal{W}_p(v)^2 \\
&\quad + \sum_{u, v \in E} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \\
&= \gamma \mathcal{W}_p(x)^2 - \sum_{v \in E} \mathcal{W}_p(v) A(x, v) \mathcal{W}_p(x) = \mathcal{W}_p(x) [\gamma \mathcal{W}_p(x) - \sum_{v \in E} \mathcal{W}_p(v) A(x, v)] \geq 0
\end{aligned}$$

Since  $\gamma > 0$ , for  $\gamma \geq k \max_{v \in V(G)} \{\mathcal{W}_p(v)\}$ , the last inequality is satisfied. Hence,  $score_p(S)$  function is monotonically non-decreasing.  $\square$

**Theorem 3.** *For  $p \geq 1$ ,  $score_p(S)$  is submodular.*

*Proof.* For any subsets  $X, Y$ , with  $X \subset Y \subseteq V$  and a subset  $Z \subset V$  such that  $Z \cap Y = \emptyset$ , we have  $score_p(X \cup Z) - score_p(X)$  is at least as large as

$score_p(Y \cup Z) - score_p(Y)$ . Let  $I, J, K \subset V(G)$  with  $I \subset J$ . We have

$$\begin{aligned}
& score_p(I \cup K) - score_p(I) - score_p(J \cup K) + score_p(J) \\
&= \left( \gamma \sum_{v \in I \cup K} \mathcal{W}_p(v)^2 - \sum_{u, v \in I \cup K} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) - \gamma \sum_{v \in I} \mathcal{W}_p(v)^2 \right. \\
&\quad \left. + \sum_{u, v \in I} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \right) - \left( \gamma \sum_{v \in J \cup K} \mathcal{W}_p(v)^2 - \sum_{u, v \in J \cup K} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \right. \\
&\quad \left. - \gamma \sum_{v \in J} \mathcal{W}_p(v)^2 + \sum_{u, v \in J} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \right) \\
&= \left( \gamma \sum_{v \in K} \mathcal{W}_p(v)^2 - \sum_{u, v \in K} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) - 2 \sum_{u \in K, v \in I} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \right) \\
&\quad - \left( \gamma \sum_{v \in K} \mathcal{W}_p(v)^2 - \sum_{u, v \in K} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) - 2 \sum_{u \in K, v \in J} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \right) \\
&= 2 \sum_{u \in K, v \in J} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) - 2 \sum_{u \in K, v \in I} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \\
&= 2 \sum_{u \in K, v \in J \setminus I} \mathcal{W}_p(v) A(u, v) \mathcal{W}_p(u) \geq 0
\end{aligned}$$

□

## 5. Computing Walks of Length 8

The proposed shield value,  $score_p(S)$ , quantifies the importance of set  $S$  based on the number of  $p$ -length closed walks containing nodes from  $S$ . Building  $S$  requires  $\mathcal{W}_p(v, G)$  for all  $v \in V$ . A closed-form of  $\mathcal{W}_p(v, G)$  depends on the actual value of  $p$ . In practice, the value of  $p = 8$  produces the set  $S$  with sufficient quality. We select nodes in a graph based on the number of closed walks of length 8 (referred to as 8-walks) for immunization purposes.

### 5.1. Justification for $p=8$

Recall that our aim is to find a set  $S$  that minimizes  $\lambda_{max}(A|_{-S})$ . From (1), we get that for large  $p$ ,  $trace(A^p)$  approaches  $\lambda_{max}(A)^p$ . Hence, we find a set  $S$  with minimum  $trace(A|_{-S}^p)$ . We show that in practice  $trace(A^8) = \sum_{i=1}^n \lambda_i(A^8)$  is sufficiently close to  $\lambda_{max}(A^8)$ . This is demonstrated by showing

that in real world graphs  $\frac{\lambda_{max}(A^8)}{\sum_{i=1}^n \lambda_i(A^8)} = \frac{\lambda_{max}(A^8)}{trace(A^8)}$  is close to 1 specially if there is significant *eigen-gap* ( $\lambda_{max}(A) - \lambda_2(A)$ ). In other words,  $\lambda_{max}(A^8)$  is the most dominant term in  $trace(A^8)$  and the combined effect of the other terms ( $\lambda_2(A^8) + \dots + \lambda_n(A^8)$ ) diminishes.

Graph	$ V $	$\lambda_{max}(A)$	$\lambda_2(A)$	$\frac{\lambda_{max}(A^8)}{\sum_{i=1}^n \lambda_i(A^8)}$
EngineeringApplicationofAI	4164	16	13.2	0.756
Facebook	4039	162.4	125.5	0.859
Email	1005	77.2	36.9	0.993
AICommunication	1203	33	12.1	0.999

Table 2: Ratio of  $\lambda_{max}(A^8)$  to  $\sum_{i=1}^n \lambda_i(A^8)$  is shown. Note that as relative eigen gap increases, the ratio approaches to 1. We show the ratio only for moderately large graphs because computing all  $n$  eigen values for very large graphs takes very long time.

## 5.2. Closed-Form Expression for $\mathcal{W}_8(v, G)$

We derive a closed-form expression for computing  $\mathcal{W}_8(v, G)$ . To the best of our knowledge, we are the first one to derive such expression.

### Theorem 4.

$$\begin{aligned} \mathcal{W}_8(v, G) = & 8A^8(v, v) - 8A^2(v, v)A^6(v, v) - 8A^3(v, v)A^5(v, v) - 4(A^4(v, v))^2 \\ & + 8A^2(v, v)(A^3(v, v))^2 + 8(A^2(v, v))^2A^4(v, v) - 2(A^2(v, v))^4 \end{aligned}$$

*Proof.* An 8-walk in  $G$  is represented as  $W = (a, b, c, d, e, f, g, h, a)$  and the goal is to compute the number of 8-walks containing a node  $v$ . Node  $v$  can occur at most four times in an 8-walk and we consider each case of the number of occurrences of  $v$  as follows.

Let  $T_{\{l_1, \dots, l_i\}}, 1 \leq i \leq 4$  be the collection of 8-walks containing  $v$  exactly  $i$  times. For  $W \in T_{\{l_1, \dots, l_i\}}$ , then  $W$  starts and ends at  $v$  and can be written as concatenation of walks of lengths  $l_1, \dots, l_i$ , each starting and ending at  $v$ . We note that  $2 \leq l_k \leq 8$ , for  $1 \leq k \leq 4$ , and  $\sum_{k=1}^i l_k = 8$ . For example

$T_{\{2,3,3\}}$  contains the walks of type  $(v, a, v, b, c, v, d, e, v)$  i.e. it is sequence of  $(v, a, v)$ ,  $(v, b, c, v)$  and  $(v, d, e, v)$ .

The rotations of nodes in a walk give different, and sometimes distinct, walks. Given a walk  $(a, b, c, d, e, f, g, h, a)$ , one vertex left rotation will produce another walk  $(b, c, d, e, f, g, h, a, b)$ . So recurrent, one vertex, rotations of walks in  $T_{\{l_1, \dots, l_i\}}$  can give up to  $8|T_{\{l_1, \dots, l_i\}}|$  different walks.

We count the walks of each type i.e. walks in  $T_{\{2,2,2,2\}}$ ,  $T_{\{2,2,4\}}$ ,  $T_{\{2,3,3\}}$ ,  $T_{\{2,6\}}$ ,  $T_{\{3,5\}}$ ,  $T_{\{4,4\}}$ ,  $T_{\{8\}}$  and their distinct rotations. In counting there are cases when walks in  $T_{\{2,3,3\}}$  are considered and these are different from walks in  $T_{\{3,2,3\}}$ , but  $|T_{\{2,3,3\}}| = |T_{\{3,2,3\}}|$ .

First, we count the number of walks containing  $v$  exactly 4 times. The walk  $(v, a, v, b, v, c, v, d, v)$ , where  $\{a, b, c, d\} \in N(v)$ , is represented as  $T_{\{2,2,2,2\}}$  as concatenation of 4 closed walks of length 2. The number of such walks is  $(A^2(v, v))^4$ . In this case, only one vertex rotation is possible which gives  $(a, v, b, v, c, v, d, v, a)$  because a second rotation gives the same original walk. Hence, the number of walks containing  $v$  exactly 4 times is  $2(A^2(v, v))^4$ .

The walks having  $v$  exactly 3 times are contained in  $T_{\{2,3,3\}}$  and  $T_{\{2,2,4\}}$ . The number of walks in  $T_{\{2,3,3\}}$  is  $A^2(v, v)(A^3(v, v))^2$  and for each walk in this set, 8 distinct walks are possible after rotations. The total number of walks containing  $v$  3 times is  $8[A^2(v, v)(A^3(v, v))^2]$ .

A walk in  $T_{\{2,2,4\}}$  is concatenation of  $(v, a, v)$ ,  $(v, b, v)$ ,  $(v, c, d, e, v)$ , where  $d \neq v$ . Number of all walks of form  $(v, a, v, b, v, c, d, e, v)$  is at most  $8(A^2(v, v))^2 A^4(v, v)$  but this number includes walks with  $d = v$  as well. To exclude those, we note that when  $d = v$ , walk is of type  $T_{\{2,2,2,2\}}$  which we have already counted in first case. Subtracting the instance when  $d = v$  in  $(v, a, v, b, v, c, d, e, v)$ , we get  $|T_{\{2,2,4\}}| = (A^2(v, v))^2 A^4(v, v) - (A^2(v, v))^4$ . All 8 vertex rotations of walks in  $T_{\{2,2,4\}}$  give distinct walks. The total number of 8-walks containing  $v$  thrice is

$$\begin{aligned} &= 8|T_{\{2,2,4\}}| + 8|T_{\{2,3,3\}}| \\ &= 8[(A^2(v, v))^2 A^4(v, v) - (A^2(v, v))^4] + 8[A^2(v, v)(A^3(v, v))^2] \end{aligned}$$

Walks containing  $v$  exactly twice are represented as  $T_{\{3,5\}}$ ,  $T_{\{2,6\}}$  and  $T_{\{4,4\}}$ . A walk in  $T_{\{3,5\}}$  is of the form  $(v, a, b, v, c, d, e, f, v)$  where  $d, e \neq v$ . The number of walks with  $d = v$  and  $e = v$  is  $|T_{\{3,2,3\}}|$  and  $|T_{\{3,3,2\}}|$ . So  $|T_{\{3,5\}}| = A^3(v, v)A^5(v, v) - 2A^2(v, v)(A^3(v, v))^2$ . In this case, vertex rotations give 8 distinct walks.

Walks in  $T_{\{2,6\}}$  are of the form  $(v, a, v, b, c, d, e, f, v)$  where  $c, d, e \neq v$ . There are maximum  $A^2(v, v)A^6(v, v)$  walks of type  $T_{\{2,6\}}$  but these include walks with  $c = v, d = v, e = v$  and  $c, e = v$ . For  $c = v$  and  $e = v$ , we get walks of types  $T_{\{2,2,4\}}$  and  $T_{\{2,4,2\}}$  respectively while if  $d = v$  then it is a walk of type  $T_{\{2,2,2,2\}}$ . For  $d = v$ , we get walk of type  $T_{\{2,3,3\}}$ .

$$\begin{aligned} |T_{\{2,6\}}| &= A^2(v, v)A^6(v, v) - 2|T_{\{2,2,4\}}| - |T_{\{2,3,3\}}| - |T_{\{2,2,2,2\}}| \\ &= A^2(v, v)A^6(v, v) - 2(A^2(v, v))^2A^4(v, v) - A^2(v, v)(A^3(v, v))^2 + (A^2(v, v))^4 \end{aligned}$$

In the case of  $T_{\{2,6\}}$ , rotations of vertices give 8 different walks.

The number of walks of type  $T_{\{4,4\}}$  in  $(A^4(v, v))^2$  but it also includes  $|T_{\{2,4,4\}}|$  and  $|T_{\{2,2,2,2\}}|$ . Therefore, we get

$$\begin{aligned} |T_{\{4,4\}}| &= (A^4(v, v))^2 - 2|T_{\{2,2,4\}}| - |T_{\{2,2,2,2\}}| \\ &= (A^4(v, v))^2 - 2(A^2(v, v))^2A^4(v, v) + (A^2(v, v))^4 \end{aligned}$$

In this case, only the first 4 vertex rotations give different walks and 5<sup>th</sup> rotation gives the original walk. The total number of walks containing  $v$  exactly twice is

$$\begin{aligned} &= 8|T_{\{3,5\}}| + 8|T_{\{2,6\}}| + 4|T_{\{4,4\}}| \\ &= 8[A^3(v, v)A^5(v, v) - 2A^2(v, v)(A^3(v, v))^2] + 8[A^2(v, v)A^6(v, v) \\ &\quad - 2(A^2(v, v))^2A^4(v, v) - A^2(v, v)(A^3(v, v))^2 + (A^2(v, v))^4] + 4[(A^4(v, v))^2 \\ &\quad - 2(A^2(v, v))^2A^4(v, v) + (A^2(v, v))^4] \\ &= 8A^3(v, v)A^5(v, v) + 8A^2(v, v)A^6(v, v) + 4(A^4(v, v))^2 - 24A^2(v, v)(A^3(v, v))^2 \\ &\quad - 24(A^2(v, v))^2A^4(v, v) + 12(A^2(v, v))^4 \end{aligned}$$

$T_{\{8\}}$  consists of walks containing  $v$  only once and are of the form  $(v, a, b, c, d, e, f, g, v)$ . The number of such walks is  $A^8(v, v)$ . But this includes walks with some combinations of  $b, c, d, e, f$  equal to  $v$  as well. Subtracting already counted walks from  $T_{\{8\}}$  gives

$$\begin{aligned} |T_{\{8\}}| = & A^8(v, v) - 2A^2(v, v)A^6(v, v) - 2A^3(v, v)A^5(v, v) - (A^4(v, v))^2 \\ & + 3A^2(v, v)^2A^4(v, v) + 3A^2(v, v)(A^3(v, v))^2 - (A^2(v, v))^4 \end{aligned}$$

In  $T_{\{8\}}$ , vertex rotations give 8 distinct walks so the number of walks containing  $v$  once is

$$\begin{aligned} 8|T_{\{8\}}| = & 8A^8(v, v) - 16A^2(v, v)A^6(v, v) - 16A^3(v, v)A^5(v, v) - 8(A^4(v, v))^2 \\ & + 24A^2(v, v)^2A^4(v, v) + 24A^2(v, v)(A^3(v, v))^2 - 8(A^2(v, v))^4 \end{aligned}$$

Combining all the four cases of occurrence of  $v$  in 8-walk gives

$$\begin{aligned} \mathcal{W}_8(v, G) = & 2|T_{\{2,2,2,2\}}| + 8|T_{\{2,2,4\}}| + 8|T_{\{2,3,3\}}| + 8|T_{\{3,5\}}| + 8|T_{\{2,6\}}| \\ & + 4|T_{\{4,4\}}| + 8|T_{\{8\}}| \\ = & 8A^8(v, v) - 4(A^4(v, v))^2 - 8A^2(v, v)A^6(v, v) - 8A^3(v, v)A^5(v, v) \\ & + 8A^2(v, v)(A^3(v, v))^2 + 8(A^2(v, v))^2A^4(v, v) - 2(A^2(v, v))^4 \end{aligned}$$

□

## 6. Proposed Algorithm

In this section, we give our algorithm to compute the number of 8-walks passing through each vertex and select nodes for immunization. Recall from Theorem 4 that computing number of 8-walks requires  $8^{th}$  power of the adjacency matrix  $A$ . Let  $f(n)$  be the running time for taking  $8^{th}$  power of  $A$ . Computing  $\mathcal{W}_8(v, G)$  for all  $v \in V$  using Theorem 4 takes  $O(n + f(n))$  time. Note that while for many real-world graphs  $A$  is sparse; this does not necessarily hold for  $A^2$  and higher powers of  $A$ . The above runtime, therefore is prohibitive

for real-world graphs, since best-known bounds on  $f(n)$  are super-quadratic.

We propose to approximately compute  $\mathcal{W}_8(v, G)$  from a summary of  $G$  [19, 26, 6]. Given a graph  $G = (V(G), E(G))$  on  $n$  nodes, a summary  $H$  of  $G$ ,  $H = (V(H), E(H))$  is a graph on  $t$  nodes with weights on both its nodes and edges.  $V(H) = \{V_1, \dots, V_t\}$  is a partition of  $V(G)$ , i.e.  $V_i \subset V(G)$  for  $1 \leq i \leq t$ ,  $V_i \cap V_j = \emptyset$  for  $i \neq j$  and  $\bigcup_{i=1}^t V_i = V(G)$ . Each  $V_i$  (called supernode) is associated with two integers  $n_i = |V_i|$  and  $e_i = |\{(u, v) | u, v \in V_i, (u, v) \in E(G)\}|$ . Weight of an edge  $(V_i, V_j) \in E(H)$  (called superedge), is  $e_{ij}$  : the number of edges in the bipartite subgraph induced between  $V_i$  and  $V_j$  i.e.  $e_{ij} = |\{(u, v) | u \in V_i, v \in V_j, (u, v) \in E(G)\}|$ . The original graph  $G$  is approximately reconstructed from  $H$  as the expected adjacency matrix,  $A'_{n \times n}$  with a row and column corresponding to each  $u \in V(G)$  given as:

$$A'(u, v) = \begin{cases} 0 & \text{if } u = v \\ \frac{e_i}{\binom{n_i}{2}} & \text{if } u, v \in V_i \\ \frac{e_{ij}}{n_i n_j} & \text{if } u \in V_i, v \in V_j \end{cases}$$

Let  $H$  be a summary graph of  $G$  on  $t$  supernodes and let  $C$  be its adjacency matrix. Clearly,  $C^p(i, j)$  is the number of walks of length  $p$  from nodes in  $V_i$  to nodes in  $V_j$ . We estimate the contributions of  $v \in V_i$  to  $C^p(i, i)$  by  $\alpha_p(v) \cdot C^p(i, i)$ , where  $\alpha_p(v) = \frac{d_G(v)^p}{\sum_{u \in V_i} d_G(u)^p}$ . Our estimate for  $\mathcal{W}_8(v, G)$  is

$$\begin{aligned} \mathcal{W}'_8(v, G) = & 8C^8(i, i)\alpha_8(v) - 8d_G(v)6C^6(i, i)\alpha_6(v) - 8C^5(i, i)\alpha_5(v)C^3(i, i)\alpha_3(v) - \\ & 4(C^4(i, i)\alpha_4(v))^2 + 8d_G(v)(C^3(i, i)\alpha_3(v))^2 + 8d_G(v)^2C^4(i, i)\alpha_4(v) - 2d_G(v)^4 \end{aligned} \quad (3)$$

This expression is same as that of Theorem 4 except for  $p \geq 3$ ,  $A^p(v, v)$  is substituted by  $\alpha_p(v) \cdot C^p(i, i)$  where  $V_i \ni v$ . Note that  $A^2(v, v) = d_G(v)$ .

We construct a summary  $H$  of  $G$  by randomly partitioning  $V(G)$  into  $t$  parts. There are better techniques [19, 26, 6] for graph summarization that might result in enhanced estimates.



### 6.1. Proposed WALK-8 Algorithm

We select a subset  $S$  that approximately maximizes  $score_8(S)$  as given in (2). In Algorithm 2, Line 3 computes  $W$  vector using (3) and  $W[i]$  is the estimated number of walks of length 8 containing vertex  $v_i$ . In each iteration of Lines 7-15, we greedily extend  $S$  by adding a node with the highest score (Line 11). Line 13 excludes nodes already selected in  $S$  from further consideration.

---

**Algorithm 2** : WALK-8( $A, k, t$ )

---

```

1:  $S \leftarrow \emptyset$ 
2:  $W_2, Score \leftarrow \text{ZEROS}(n)$ 
3:  $W \leftarrow \text{ESTIMATEWALKS}(A, t)$   $\triangleright$  compute approx. count of walks using
   super graph of order  $t$  based on Eq. (3)
4:  $\gamma \leftarrow \max_i W[i]$ 
5: for  $i = 1$  to  $n$  do
6:    $W_2[i] \leftarrow \gamma W[i]^2$ 
7: for  $i = 1$  to  $k$  do
8:    $\mathbf{u} \leftarrow A[:, S] * W[S]$ 
9:   for  $j = 1$  to  $n$  do
10:    if  $j \notin S$  then
11:       $Score[j] \leftarrow W_2[j] - 2\mathbf{u}[j]W[j]$ 
12:    else
13:       $Score[j] \leftarrow -1$ 
14:    $maxNode \leftarrow \arg \max_j Score[j]$ 
15:    $S \leftarrow S \cup \{maxNode\}$ 
16: return  $S$ 

```

---

### 6.2. Runtime Analysis of WALK-8

We derive analytical bounds on the runtime of Algorithm 2. Partitioning  $G$  into  $t$  supernodes takes  $O(n)$  time as it can be done with a linear scan on  $V(G)$  to put nodes in respective buckets (supernodes). Computing the summary graph (populating the weighted adjacency matrix,  $C$ ) requires traversing the edges  $E(G)$  and incrementing the appropriate entry of  $C$ . This takes a total of  $O(|E(G)|)$  time. The powers of  $C$  matrix can be computed in  $O(t^3)$  time. Thus ESTIMATEWALKS function takes  $O(n + |E(G)| + t^3)$  time. Line 4 and the first for loop (Lines 5-6) takes  $O(n)$  steps. An iteration of the inner for loop (Lines 9-13) takes  $O(n + nk)$  and Line 14 takes  $O(n)$  steps. This shows that the outer

loop (Line 7-15) takes  $\sum_{i=1}^k O(n + nk) = O(nk^2)$ . Therefore, Algorithm 2 takes total  $O(n + |E(G)| + t^3 + nk^2)$  time.

## 7. Experimental Evaluation

We present the results of the detailed experimentation of our proposed solution in this section. Experiments are performed on several real-world datasets to analyze the performance of our method and results are compared with NETSHIELD<sup>1</sup>, the state of art algorithm, to evaluate quality, scalability and efficiency. NETSHIELD computes the score of each node using the eigenvector corresponding to the largest eigenvalue  $\lambda_{max}$  of the original graph. WALK-6 and WALK-8 versions of our algorithm select nodes for immunization based on 6-walks and 8-walks respectively passing through each node.

We evaluate the performance of our algorithm across a range of budgets for the number of nodes to be immunized in the graphs and different counts of supernodes for approximation. First, we evaluate the quality of our approximation technique. To show that our approach maximally reduces the spread of the virus across the graph, we give results for the virus spread simulation on graphs immunized by NETSHIELD, WALK-6 and WALK-8. Furthermore, we measure quality in terms of the reduction in  $\lambda_{max}$  (*vulnerability*) of the graph after immunizing the set  $S$  of selected nodes. We report results using *eigendrop percentage*, which is  $\frac{\Delta\lambda(S)}{\lambda_{max}(A)} \times 100$ . Finally, we give runtime comparisons for the above-mentioned techniques.

We performed experiments on a standard desktop machine with 3.6 GHz Intel Core i7-7700 and 8 GB of main memory. The MATLAB code for our algorithm is available <sup>2</sup> for reproducibility and further experimentation.

---

<sup>1</sup><https://www.dropbox.com/s/aaq5ly4mcxhijmg/Netshieldplus.tar>

<sup>2</sup><https://www.dropbox.com/sh/n7hwjc4imh62pe6/AADCyHG7uMGX6o9xtr1pdH6Qa?dl=0>

Network	Number of Nodes	Number of Edges	$\lambda_{max}(A)$
HEP-TH	9,877	25,998	31.03
Facebook	4,039	88,234	162.37
Gowalla	196,591	950,327	170.94
Dblp	317,080	1,049,866	115.85
Amazon	334,863	925,872	23.98
AA	418,236	2,753,798	-
Youtube	1,134,890	2,987,624	210.40
Skitter	1,696,415	11,095,298	670.35

Table 3: Statistics of Datasets

### 7.1. Datasets

Experiments are performed on real-world graphs of order ranging from a few thousands to a few millions nodes. All graphs are undirected and unweighted. HEP-TH<sup>3</sup> is a collaboration network of High Energy Physics - Theory category extracted from the e-print arXiv. A node in the network represents an author and an edge between two authors shows collaboration between them. Facebook<sup>3</sup> graph shows the friendship network among users in which people are represented as nodes and relationships among two users are shown as edges.

To test our algorithm on large networks we use five different real-world graphs. Gowalla<sup>3</sup> dataset shows friendship relations in a location-based social network. Amazon<sup>3</sup> is a co-purchasing graph of products where each node is a product and there is an edge between two nodes if the products are purchased by a user in a single basket. Dblp<sup>3</sup> is a co-authorship network in which two authors are connected if they have co-authored at least one publication. Youtube<sup>3</sup> graph shows the friendship network of users in the Youtube social network. Skitter<sup>3</sup> is an internet topology network where nodes correspond to autonomous systems and communication between them constitutes edges.

The dataset AA<sup>4</sup> is a co-authorship network extracted from DBLP archive data. We select 4 different smaller co-authorship subgraphs each corresponding to manuscripts in a distinct journal. Node count goes up to a few thousands

---

<sup>3</sup><https://snap.stanford.edu/>

<sup>4</sup><http://dblp.uni-trier.de/xml/>

Network	Number of Nodes	Number of Edges	$\lambda_{max}(A)$
Applied Mathematics and Computing (AMC)	18,371	24,224	10.99
Decision Support Systems (DSS)	4,926	14,660	12.0
Ecological Informatics (EI)	1,990	4,913	16.68
Communication ACM	11,476	16,687	32.90

Table 4: Statistics of AA subgraphs

and edge count goes up to a few ten thousands for extracted subgraphs. Details of the subgraphs of AA data set are provided in Table 4.

### 7.2. Approximation Quality of WALK-8

In order to evaluate the goodness of our approximate method, we compare it with the exact solution as described in Theorem 4. The exact number of closed walks of length 8 can be computed using the original adjacency matrix  $A$  as given in Theorem 4 instead of using a summary graph. We analyze the quality of our approximation method by comparing the eigendrop percentages achieved using the exact and approximate method. We report comparison results of the exact solution with the summary graphs of order  $\{100, 500, 1000\}$ .

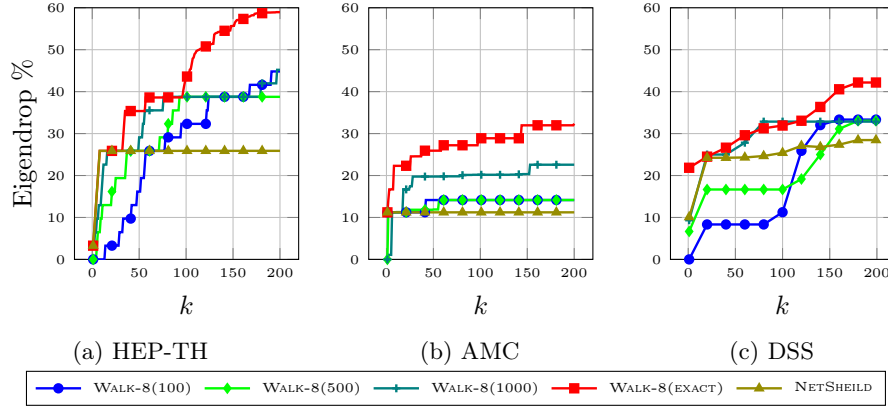


Figure 1: The effect of the order of summary graph on the quality of the approximation. Eigendrop percentages using different numbers of supernodes have been reported (WALK-8( $t$ ), where  $t$  is the number of supernodes). It is clear that as  $t$  increases, the quality of approximation tends to match with that of the exact solution.

It is clear from Figure 1 that the performance of our approximate method improves with the increase in the number of supernodes in the summary graph. As the order of the summary graph increases, the achieved benefit tends to match with that of the exact solution. Note that we compute the exact number of walks for small graphs having the order of a few thousands only as it is computationally infeasible to compute the exact solution for large graphs.

### 7.3. Virus Spread Simulation

Another criterion used for quality evaluation is to estimate the spread of virus propagation in the immunized version of the graph. We use SIR virus propagation model to observe the spread of contagion after immunizing a small subset ( $\sim 5\%$ ) of nodes in a graph. Let  $s = \lambda_{max} \times \beta/\delta$  be the virus strength (larger value of  $s$  corresponds to more strength of virus while the virus gradually dies out if  $s \leq 1$ ), where  $\beta$  and  $\delta$  denote the infection and recovery rate respectively. In our experimentation, we immunize  $k$  nodes in a graph and infect all the nodes in the immunized version of the graph. We then observe the spread of the virus under different virus strengths with varying values of  $\beta$  and  $\delta$ . Results in Figure 2 show that the graphs immunized by our approach have less number of infected nodes as compared to NETSHEILD. We report the average of 3 runs of experiments to mitigate the effect of randomness.

### 7.4. EigenDrop Percentage Comparison

We compare the quality of approximate versions of our algorithms with NETSHEILD in terms of eigendrop and results are shown in Figure 3. For smaller graphs and subgraphs of AA which consist of a few thousand nodes, a budget of up to 100 nodes is used and for large graphs with more than 100,000 nodes, we immunize up to 1000 nodes. We have used summary graphs with different orders (100, 500, 1000) to perform experiments. Time complexity increases as the number of supernodes increases but we observe that there is a proportionately minor improvement in the quality of solution for increasing order of graph

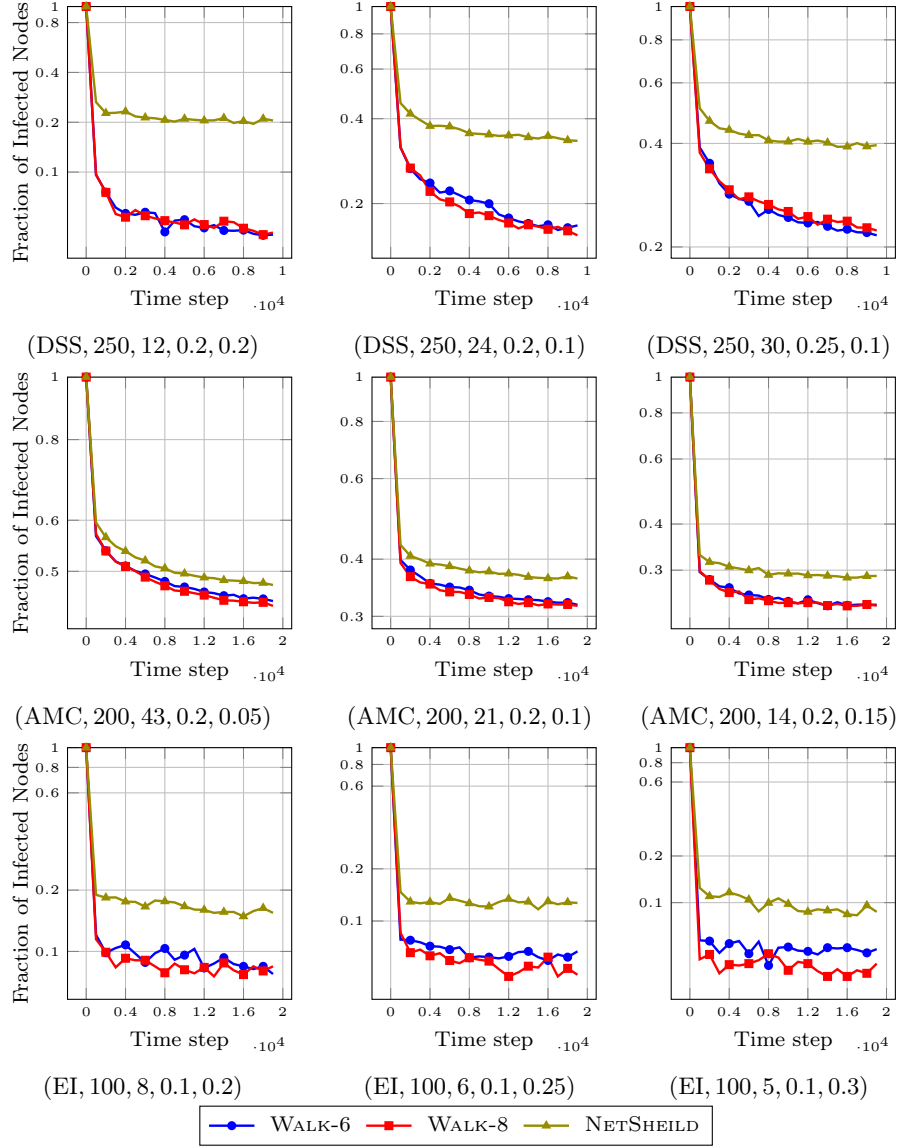


Figure 2: Virus propagation simulation for varying virus strength  $s$  on the immunized version of graphs. Caption of each plot represents (graph name, number of immunized nodes  $k$ ,  $s$ , infection rate  $\beta$ , recovery rate  $\delta$ ). Initially, all the nodes in the graphs were contaminated and the plots show the fraction of infected nodes ( $y$ -axis logged scale) as the time proceeds.

after a certain threshold is reached. For smaller graphs, we report results for supernode count of 500 and for large graphs, the number of supernodes is set to 1000.

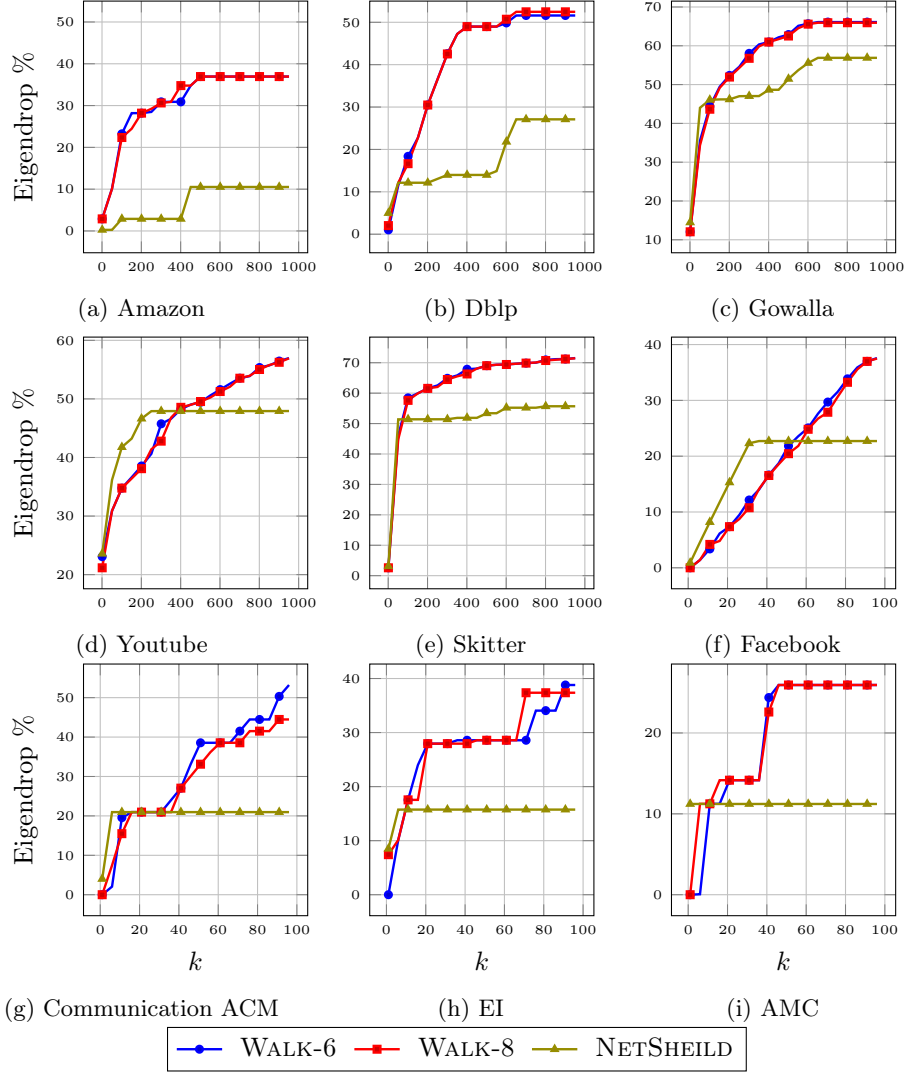


Figure 3: Comparison of NETSHEILD, WALK-6 and WALK-8 in terms of eigendrop percentages ( $y$ -axis) against budget  $k$ , number of nodes immunized, ( $x$ -axis). WALK-6 and WALK-8 achieve significantly higher eigendrop for increasing  $k$ . Results in (a)-(e) are computed using 1000 supernodes while in (f)-(i) experiments are performed using summary graph of order = 500. The range for  $k$  is chosen keeping in view the number of nodes in the host graphs.

We observe that the immunizing quality of our algorithm clearly outperforms NETSHEILD in terms of eigendrop. The improvement in quality of solution is particularly evident on large graphs Gowalla Figure 3c, Youtube Figure 3d, and Skitter Figure 3e. For reasonably large budget, WALK-8 outperforms both

NETSHEILD and WALK-6. Experiments also reveal that NETSHEILD performs better than our approach for very small values of budget  $k$  but as the count of nodes to be immunized increases, its effectiveness degrades.

### 7.5. Run Time Comparison

We also present comparable computational cost while achieving much better quality as one of the merits of our algorithm as discussed in the theoretical time complexity in Section 4. Comparison of runtimes of NETSHEILD, WALK-6 and WALK-8 is provided in Figure 4. Results show that the runtime of our algorithm matches with that of NETSHEILD. The results are reported with 1000 supernodes ( $t$ ) in summary graphs.

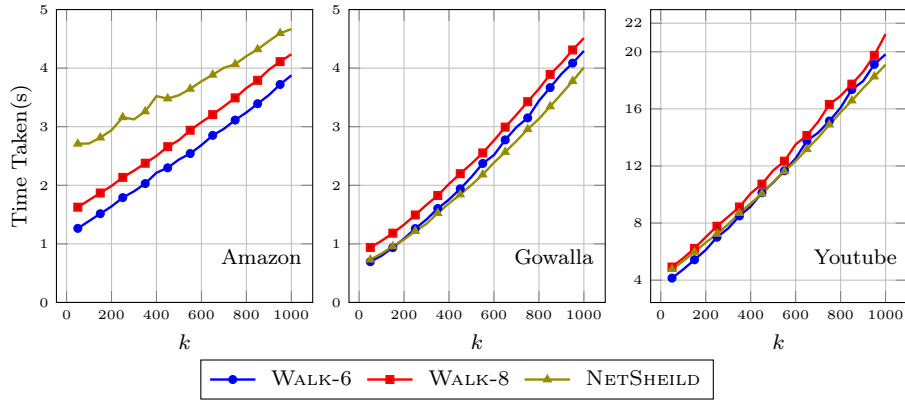


Figure 4: Comparison of time taken (in seconds) to immunize graphs using NETSHEILD, WALK-6 and WALK-8 approach against the number of nodes to be immunized ( $k$ ). Results are reported on summaries with 1000 supernodes.

Recall that runtime of our algorithm is  $O(n + |E(G)| + t^3 + nk^2)$ , where the first three terms comprise runtime of constructing a summary of order  $t$  and computing the  $\mathcal{W}_8(v, G)$  for all  $v \in V(G)$ , while the last term ( $nk^2$ ) is the runtime to select the best  $k$  nodes (NETSHEILD also requires  $O(nk^2)$  for this task). Hence runtime of our algorithm depends quadratically only on  $k$ , which generally is a small constant. We note that our runtime is superior to that of NETSHEILD in the sense that in relatively less time we achieve significantly more eigendrop even for a small value of  $t$  (see Figure 1).



## 8. Conclusion

In this work, we address the problem of finding a small subset of nodes in a network whose immunization results in a significant reduction in network vulnerability towards the spread of undesirable content. We explored the relationships between spectral and graph-theoretic properties of networks and exploit these relationships to design an efficient algorithm to find crucial nodes in the network. We select a subset of nodes for immunization based on the number of closed walks of length 8. With the use of easily computable local graph properties and approximation techniques, the running time of our technique is linear in the size of the graph. Thus, our method is scalable and can be applied to large graphs. Experiments on large real-world networks suggest that our algorithm provides better results than previously employed methods and is significantly faster in terms of time complexity. The approximation quality comparison shows that our method is a close approximation of the exact solution. Experimental results for various quality measures like virus spread simulation, reduction in network vulnerability and the run time comparison show that our method performs better than the state of the art solution.

Potential extensions of this work include i) utilizing specialized graph summarization methods, this will further reduce computational cost as well as improve immunization performance of the solution ii) extending this work to incorporate dynamic graphs. Dynamic graphs evolve with time and edges are added/removed iii) exploring non-preemptive graph immunization approaches, where the immunization process starts after the virus attack and the information of infected nodes is available.

## References

- [1] Abbas, S., Tariq, J., Zaman, A., & Khan, I. (2017). Sampling based efficient algorithm to estimate the spectral radius of large graphs. In *IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW* (pp. 175–180). doi:10.1109/ICDCSW.2017.71.

- [2] Ahmad, M., Tariq, J., Farhan, M., Shabbir, M., & Khan, I. (2016). Who should receive the vaccine ? In *14th Australasian Data Mining Conference, AusDM*. ACS volume 170 of *CRPIT*.
- [3] Ahmad, M., Tariq, J., Shabbir, M., & Khan, I. (2017). Spectral methods for immunization of large networks. *Australasian Journal of Information Systems*, 21. doi:<http://dx.doi.org/10.3127/ajis.v21i0.1563>.
- [4] Ahn, H. J., & Hassibi, B. (2013). Global dynamics of epidemic spread over complex networks. In *IEEE Conference on Decision and Control, CDC* (pp. 4579–4585). doi:10.1109/CDC.2013.6760600.
- [5] Arulselvan, A., Commander, C. W., Pardalos, P. M., & Shylo, O. (2007). Managing network risk via critical node identification. *Risk Management in Telecommunication Networks*, Springer, .
- [6] Beg, M. A., Ahmad, M., Zaman, A., & Khan, I. (2018). Scalable approximation algorithm for graph summarization. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD* (pp. 502–514). doi:10.1007/978-3-319-93040-4\_40.
- [7] Bienstock, D., & Seymour, P. (1991). Monotonicity in graph searching. *Journal of Algorithms*, 12, 239–245. doi:10.1016/0196-6774(91)90003-H.
- [8] Chakrabarti, D., Wang, Y., Wang, C., Leskovec, J., & Faloutsos, C. (2008). Epidemic thresholds in real networks. *ACM Transactions on Information and System Security*, 10, 1:1–1:26. doi:10.1145/1284680.1284681.
- [9] Chen, C., Tong, H., Prakash, B., Tsourakakis, C., Eliassi-Rad, T., Faloutsos, C., & Chau, D. (2016). Node immunization on large graphs: Theory and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 28, 113–126. doi:10.1109/TKDE.2015.2465378.
- [10] Daadaa, Y., Jamshed, A., & Shabbir, M. (2016). Network decontamination with a single agent. *Graphs and Combinatorics*, 32, 559–581. doi:10.1007/s00373-015-1579-5.

- [11] Erdős, D., Ishakian, V., Lapets, A., Terzi, E., & Bestavros, A. (2012). The filter-placement problem and its application to minimizing information multiplicity. *PVLDB*, 5, 418–429. doi:10.14778/2140436.2140439.
- [12] Flocchini, P., Huang, M. J., & Luccio, F. (2007). Decontaminating chordal rings and tori using mobile agents. *International Journal of Foundations of Computer Science*, 18, 547–563. doi:10.1142/S0129054107004838.
- [13] Flocchini, P., Huang, M. J., & Luccio, F. L. (2008). Decontamination of hypercubes by mobile agents. *Networks*, 52, 167–178. doi:10.1002/net.20240.
- [14] Fraigniaud, P., & Nisse, N. (2008). Monotony properties of connected visible graph searching. *Information and Computation*, 206, 1383–1393. doi:10.1016/j.ic.2008.09.002.
- [15] Ganesh, A., Massoulié, L., & Towsley, D. (2005). The effect of network topology on the spread of epidemics. In *IEEE Annual Joint Conference of the Computer and Communications Societies, INFOCOM* (pp. 1455–1466). doi:10.1109/INFOCOM.2005.1498374.
- [16] Khanafer, A., Basar, T., & Gharesifard, B. (2014). Stability properties of infected networks with low curing rates. In *American Control Conference, ACC* (pp. 3579–3584). doi:10.1109/ACC.2014.6859418.
- [17] Khanafer, A., Başar, T., & Gharesifard, B. (2014). Stability properties of infection diffusion dynamics over directed networks. In *IEEE Conference on Decision and Control, CDC* (pp. 6215–6220). doi:10.1109/CDC.2014.7040363.
- [18] Kuhlman, C. J., Tuli, G., Swarup, S., Marathe, M. V., & Ravi, S. (2013). Blocking simple and complex contagion by edge removal. In *IEEE International Conference on Data Mining, ICDM* (pp. 399–408). doi:10.1109/ICDM.2013.47.

- [19] LeFevre, K., & Terzi, E. (2010). GraSS: Graph structure summarization. In *SIAM International Conference on Data Mining, SDM* (pp. 454–465). doi:10.1137/1.9781611972801.40.
- [20] Li, J., Pardalos, P. M., Xin, B., & Chen, J. (2019). The bi-objective critical node detection problem with minimum pairwise connectivity and cost: theory and algorithms. *Soft Computing*, (pp. 1–16).
- [21] Morone, F., & Makse, H. A. (2015). Influence maximization in complex networks through optimal percolation. *Nature*, 524, 65. doi:10.1038/nature14604.
- [22] Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions. *Mathematical programming*, 14, 265–294. doi:10.1007/BF01588971.
- [23] Neri, F. (2019). *Linear Algebra for Computational Sciences and Engineering*. Springer.
- [24] Prakash, B. A., Vreeken, J., & Faloutsos, C. (2012). Spotting culprits in epidemics: How many and which ones? In *IEEE International Conference on Data Mining, ICDM* (pp. 11–20). doi:10.1109/ICDM.2012.136.
- [25] Riondato, M., García-Soriano, D., & Bonchi, F. (2014). Graph summarization with quality guarantees. In *IEEE International Conference on Data Mining, ICDM* (pp. 947–952). doi:10.1109/ICDM.2014.56.
- [26] Riondato, M., García-Soriano, D., & Bonchi, F. (2017). Graph summarization with quality guarantees. *Data Mining and Knowledge Discovery*, 31, 314–349. doi:10.1007/s10618-016-0468-8.
- [27] Serre, D. (2002). *Matrices*. Springer.
- [28] Song, C., Hsu, W., & Lee, M. L. (2015). Node immunization over infectious period. In *ACM International Conference on Information and Knowledge Management, CIKM* (pp. 831–840). doi:10.1145/2806416.2806522.

- [29] Strang, G. (1988). *Linear Algebra and its Applications*. Academic Press.
- [30] Tariq, J., Ahmad, M., Khan, I., & Shabbir, M. (2017). Scalable approximation algorithm for network immunization. In *Pacific Asia Conference on Information Systems, PACIS* (p. 200).
- [31] Tong, H., Prakash, B. A., Eliassi-Rad, T., Faloutsos, M., & Faloutsos, C. (2012). Gelling, and melting, large graphs by edge manipulation. In *ACM International Conference on Information and Knowledge Management, CIKM* (pp. 245–254). doi:10.1145/2396761.2396795.
- [32] Van Mieghem, P. (2014). Exact markovian Sir and Sis epidemics on networks and an upper bound for the epidemic threshold. *arXiv preprint arXiv:1402.1731*, .
- [33] Van Mieghem, P., Sahnehz, F. D., & Scoglio, C. (2014). An upper bound for the epidemic threshold in exact markovian sir and sis epidemics on networks. In *IEEE Conference on Decision and Control, CDC* (pp. 6228–6233). doi:10.1109/CDC.2014.7040365.
- [34] Van Mieghem, P., Stevanović, D., Kuipers, F., Li, C., Van De Bovenkamp, R., Liu, D., & Wang, H. (2011). Decreasing the spectral radius of a graph by link removals. *Physical Review E*, 84, 016101. doi:10.1103/PhysRevE.84.016101.
- [35] Ventresca, M., & Aleman, D. (2015). Efficiently identifying critical nodes in large complex networks. *Computational Social Networks*, 2, 6. doi:10.1186/s40649-015-0010-y.
- [36] Wang, Y., Chakrabarti, D., Wang, C., & Faloutsos, C. (2003). Epidemic spreading in real networks: An eigenvalue viewpoint. In *Symposium on Reliable Distributed Systems, SRDS* (pp. 25–34). doi:10.1109/RELDIS.2003.1238052.
- [37] West, D. (2001). *Introduction to graph theory*. Prentice Hall.

- [38] Zhang, Y., & Prakash, B. A. (2014). Dava: Distributing vaccines over networks under prior information. In *SIAM International Conference on Data Mining, SDM* (pp. 46–54). doi:10.1137/1.9781611973440.6.
- [39] Zhang, Y., & Prakash, B. A. (2014). Scalable vaccine distribution in large graphs given uncertain data. In *ACM International Conference on Conference on Information and Knowledge Management, CIKM* (pp. 1719–1728). doi:10.1145/2661829.2662088.