LEIDEN UNIVERSITY

MASTER THESIS

Using random rooted spanning forest for network immunization: Comparison and analysis of different approaches

Author:
Irina GUREWITSCH

First Supervisor:
Dr. Luca AVENA

Second Supervisor:
Dr. Michael EMMERICH

A thesis submitted in fulfillment of the requirements for the degree of Master of Mathematics and Business

in the

Mathematical Institute

July 11, 2022

LEIDEN UNIVERSITY

Abstract

Faculty of Science

Mathematical Institute

Master of Mathematics and Business

**Using random rooted spanning forest for network immunization:
Comparison and analysis of different approaches**

by Irina GUREWITSCH

We are interested in the so-called multiple-node immunization for complex networks under attack of a spreading agent. The latter is a hot topic in network science and epidemiology and it consists in identifying and removing a set of nodes of a given size in a graph to maximally impede the agent spread. Several approaches have been proposed in the literature based on numerical and theoretical insights on how classical models for virus spread (so-called compartmental models) evolve on graphs.

Based on the stability analysis of these compartmental models, the maximal eigenvalue of the adjacency matrix of the graph has been proposed as a measure of how resilient the network is. Thus one of the most common approach for immunization consists in identifying the set of nodes of a given cardinality, for which the reduced network, obtained by removing these nodes, has minimal largest eigenvalue. This spectral optimization problem turns out to be a computationally hard problem in the well-known NP class and the available exact or proxy algorithms offer good solutions only on certain graph instances.

We propose here a novel efficient randomized flexible method to identify these sets of nodes based on random walk kernels and random rooted forests. We explain the theoretical results behind the method, previously derived by Avena and Gaudilliere, and then test it via numerical experiments on classical synthetic and real-world benchmarks. Experimental results and related performances turn to be comparable to those obtained by implementing so-called Netshield, the currently best available algorithm. Yet, within the same running time order , for certain graph instances, our method can find better solutions than Nethsield.

The overall idea is that the most likely outputs of our randomized forest method concentrate on the optimal or near to optimal sets of nodes, and hence the method can be seen as a clever procedure to explore the space of solutions of the above mentioned optimization problem. Another advantage of our randomized method is that it outputs several good solutions.

Furthermore, the method its flexible because relies on a suitable choice of the graph-Laplacian, and as such, it could be adapted and used for other optimization problems.

Acknowledgements

This thesis work started around the same time as the COVID-19 pandemic in 2020 and was majorly influenced by its consequence. We had to find new ways of working and communicating as our lives were turned upside-down. I want to express my deepest gratitude to Luca Avena for his dedication and help, for the many calls and the feedback that got me where I am today. He was truly the heart of the project. My deepest appreciation to the whole team of researchers, Luca Avena, Michael Emmerich and Alexandre Gaudillière for your patience, commitment and knowledge that made this work possible despite all the challenges we faced.
Finally, I would like to mention the people, who were my biggest supporters and helped me keep my sanity, such as Julie, Maxine, Annie, Diana Regina, Martha, Lisa, my parents and grandparents and Fima. Thank you all!

Contents

Chapter 1

Introduction

Imagine a university account network that is attacked by a computer virus. The virus spreads when emails are sent from one account to the other and the university wants to stop the spread of the virus as soon as possible. The university does not want to stop all communication in the network as this can have big impact on education, collaborations and administrative matters, but the university also cannot let the virus spread unrestricted because sensitive data could be hacked. One way to approach this dilemma would be to select several accounts and deactivate them, such that the spread of the computer virus is slowed down and the university has more time to find a solution or to add a layer of protection to the accounts, without collapsing the whole network. But which accounts should be selected for temporary deactivation?

The above described situation is one example of the so-called k-node immunization problem for which there is no unique and easy answer to that question, see e.g. [14],[13], [11], [15]. There are many valid approaches, yet none of them offers perfect solutions except for trying all possible combinations, which gets infeasible for moderately-sized networks already. Our work aims to explore one new possible approach, namely using certain random rooted spanning forests to approach the k-node immunization problem. As a measure of immunization we will look at the biggest difference in the maximal adjacency-eigenvalue of the non-immunized graph (the original network) and the immunized graph (the reduced network obtained by removing the k identified spreader-nodes). Such a spectral immunization measure is by now considered a classical approach, which, as we will see, is justified by how certain compartmental models on networks behave.

We will argue that our randomised method based on rooted spanning forests is efficient and can perform better that existing deterministic algorithms used for this spectral k-node immunization problem.

Thesis overview: We will start in chapter 2 by shortly describing the history of epidemiological models, after which we will focus on to the key SIS model on networks and its relation to the maximal eigenvalue of the adjacency matrix. When the basics of the model are clear, we will move on to the k-node immunization problem and possible approaches. One of those approaches is based on the so-called Netshield algorithm, which we will look at in more detail. Netshield is indeed given special attention, as it is a fast and effective algorithm widely used in the literature, and we will use is for our later experiments as a benchmark.

In chapter 3, we will finally introduce the random object at the core of our new method: rooted spanning forests and related properties. After the definition of the key object, we will describe Wilson's algorithm and how we sample the random rooted spanning forest, such that we can get the desired number of roots, which we will call the forest method. In the last part of the chapter, we will look at more properties of random rooted spanning forest that make its study valuable in the context of the k-node immunization problem.

The fourth chapter will be dedicated to looking at the random rooted spanning forest as a tool for network immunization. We will discuss the root set and the graph Laplacian as parameters that can be tuned to make the forest method target network immunization. We will also highlight the results from the subsequent chapters 5 and 6 and discuss the outcomes, mostly without providing formal proofs, as empirical reasoning will follow in the subsequent chapters. Lastly, we will touch on the topic of batches and sample size, related to performance and efficiency analysis.

In the 5th chapter we present the first round of experiments that we performed to empirically determine which graph Laplacian should be used to maximize immunization. For that, we looked at two different graphs and compared the results of samplings using different Laplacians. We will start with a regular network and move on to the famous, irregular Karate Club network during these experiments and in the end formulate a conclusion based on our findings.

In the last chapter of this work, we will present more experiments, this time to determine how well the forest method performs. For that, we will compare it to the Netshield variants in two rounds of experiments. In the first round we just experimented with the forest measure to see if it performs well and compare it to Netshield. In the second round we introduce additional tuning in form of batches to see if we can improve the sampling results of the forest method. Additionaly, we also looked at weighted graphs for the first time. After the description of the experiments and results, we will sum up the results and formulate our empirical conclusions. To finalize this work, in the end of that chapter we will quickly sum up the results of this thesis and talk about further developments that happend after the research project was concluded.

Chapter 2

Epidemic spreading and the k-node immunization problem

This chapter is dedicated to introducing the k-node immunization problem. We will start by recalling the basics of epidemiological models. In particular the focus will lie on the SIS model on graphs, for which we will derive an epidemic threshold phrased in terms of the maximal eigenvalue of the graph adjacency matrix. Afterwards, the k-node immunization problem will be laid out and a solution based on the epidemic threshold will be presented in relation to a so-called maximal eigenvalue-drop. Lastly, we will introduce the Netshield algorithm (and Netshield+), an existing, deterministic algorithm that approximates the abovementioned drop in eigenvalue.

2.1 Epidemiological models

Epidemiology is the study of the patterns and effects of diseases in a certain population. The beginnings of epidemiology reach as far back, as the times of Ancient Greece, nevertheless the first mathematical models trace back to the 18th century [6].
In the early 20th century compartmental models were introduced and can now be considered the standard framework of modern epidemiology. The key idea behind compartmental models is, that the population is divided into compartments, such as 'susceptible' or 'infected', and individuals can 'transition' between these compartments. Such an evolution is described by differential equations [6]. More precisely, compartmental models describe deterministic densities of compartments and how they exchange mass.
Epidemiological models are not exclusively used to model diseases, but can be applied to processes with similar properties, such as the spread of computer viruses in a computer network or the spread of a rumor in a social network.
There are many different compartmental models that describe the spread of a virus, however in this thesis we will restrict ourselves to working with compartmental models on networks, a recent overview of which can be found here [2]. We will recall the three simplest ones tailored to a graph set up.
Before we can do that, we need to first introduce the central object of this thesis, namely the network.

Definition 2.1.1 (Network)**.** *We call a network a weighted and finite (directed) graph G, that is a triple*

$$G = (V, E, w)$$

where V is a finite set of n vertices, $E = \{(x, y) \in V \times V\}$ is the set of m directed edges and $w : V \times V \mapsto \mathbb{R}_0^+$ is a non-negative weight function, that assigns a weight $w(x, y)$ to each $(x, y) \in E$. [1]

For this thesis, we will use the terms graph and network interchangeably.

Definition 2.1.2 (Weighted Adjacency Matrix)**.** *Let $G = (V, E, w)$ be a network with $|V| = n$. Then the adjacency matrix is the $n \times n$ matrix $A = (a_{x,y})_{x,y \in V}$, with entries:*

$$a_{x,y} = \begin{cases} w(x,y), & if\ (x,y) \in E \\ 0, & otherwise \end{cases}$$

We next give an informal description of a compartmental model on an undirected (i.e. $a_{x,y} = a_{y,x}$) and unweighted (i.e. $w(x, y) \equiv 1$) graph $G = (V, E)$. The underlying idea is that a node $x \in V$ has one of two possible states: Susceptible (S) and Infected (I). A susceptible node can become infected by infected neighbors at rate β times the number of infected neighbors. Starting with this configuration, the most common compartmental models are:

1. **SI model**: Once a node is infected, it stays infected and keeps on infecting its neighbors. This model corresponds to e.g. the spread of a rumor in a social network. Once a person knows the information, they spread the information and don't forget it. In a connected network this will lead to all nodes being infected at some point. [2]
2. **SIS model**: Once a node is infected, it can recover spontaneously at rate δ and become susceptible again. This can be compared to the spread of diseases like the flue, that do not make the patient immune against another infection.
3. **SIR model**: There is a third state, Recovered (R), sometimes also referred to as removed. When a node is infected it can recover spontaneously at rate δ. A recovered node cannot infect anyone or become infected again and can be treated as if it were removed from the network. This models scenarios like e.g. the spread of lethal diseases or diseases that cause an immunity to further infections like mumps.

In this thesis, the main focus will lie on the SIS model on networks. The SIS model can reflect the early stages of virus spreading rather realistically which also makes it a relevant object for this applied work. We next describe the so-called epidemic threshold what will allow us to define a notion of robustness of a network under attack of a viral agent. The epidemic threshold will be derived for the SIS model,

[1] It is worth anticipating that in the novel experimental last chapters of this thesis, we will only consider undirected network datasets. Nonetheless, objects and results required to derive the underlying theoretical tools presented in Chapter 3 extend to the more general directed setting and it is convenient to present them in this more general framework.

[2] If a network is directed, it needs to be strongly connected to reach the state "all infected".

which we next introduce in more detail. However, this epidemic threshold can be extended to other compartmental models.

2.1.1 SIS model on arbitrary networks

Given a graph (V, E), consider the vector $\vec{\rho(t)} = (\rho_x(t))_{x \in V} \in [0,1]^{|V|}$, where we interpret each component $\rho_x(t) \in (0,1)$ as the probability that node x is infected at time t.

Definition 2.1.3 (SIS model on a network)**.** *Let $G = (V, E, w)$ be a weighted and undirected graph. The SIS model on the network G can be defined as the system of $|V| = n$ coupled equations:*

$$\frac{\partial \rho_x(t)}{\partial t} = -\delta \rho_x(t) + \beta \left[1 - \rho_x(t)\right] \sum_{y \in V} a_{x,y} \rho_y(t), \quad x \in V, \tag{2.1}$$

with given initial condition $\vec{\rho}(0) \in [0,1]^n$ and fixed parameters $\beta, \delta > 0$.

Let us interpret the evolution equation in (2.1). If a node is infected it can become susceptible with rate δ, which corresponds to a spontaneous healing event captured in the first term in the right hand side of (2.1). And if a node is susceptible, it can be infected by its neighbors, at rate β times the number of infected neighbors. This corresponds to the second term in (2.1) when $a_{x,y}$ are assumed to take values only 0 or 1 (i.e. unweighted case). More generally, for a weighted and directed graph, the interpretation would be that infections traverse edge (x, y) at rate $\beta \times a_{x,y}$.

By rescaling the time of the equations in (2.1) by the factor $1/\delta$ and setting $r := \beta/\delta$, we can rewrite the equations as follows:

$$\frac{\partial \rho_x(t)}{\partial t} = -\rho_x(t) + r \sum_{y \in V} a_{x,y} \rho_y(t) - r \sum_{y \in V} a_{x,y} \rho_x(t) \rho_y(t). \tag{2.2}$$

Starting from a given **configuration of states** $\vec{\rho}(0) := (\rho_1(0), ..., \rho_n(0))$, we want to solve the system of n coupled equations to understand, how the infection evolves. But due to non-linear terms $\rho_x(t)\rho_y(t)$, this system does not have an explicit closed solution. However, it is still possible to derive bounds on the value of r which guarantees that the infection will vanish exponentially fast.
Since the adjacency matrix A as well as the $\rho_x(t)$'s are non-negative, equation (2.2) implies the inequality:

$$\frac{\partial \rho_x(t)}{\partial t} \leq -\rho_x(t) + r \sum_{y \in V} a_{x,y} \rho_y(t). \tag{2.3}$$

For (2.3) with equality, this is a standard system of ordinary linear equations which admits a closed solution, leading to

$$\vec{\rho}(t) \leq \vec{\rho}(0) e^{(rA - Id)t}. \tag{2.4}$$

The adjacency matrix A is diagonizable according to the spectral theorem, since it is real and symmetric by assumption. It can be written in terms of its eigenvectors and eigenvalues as $A = UDU^{-1}$, where U is the matrix of A's eigenvectors and D is a diagonal matrix, that has the eigenvalues as entries. When we rewrite the exponential term, we see that the exponential factor is dominated by the largest eigenvalue of the adjacency matrix, which will be referred to as λ_{max}.

$$Ue^{(rD)t}U^{-1}e^{-t} \leq Ue^{(r\lambda_{max})t}U^{-1}e^{-t} \tag{2.5}$$

Therefore, for $r\lambda_{max} - 1 < 0$, the $\rho_x(t)$ decay to 0 at least exponentially fast for all $x \in V$. This proves the following theorem.

Theorem 2.1.4 (Epidemic threshold and lower bound)**.** *Let $\vec{\rho}(t)$ be the SIS dynamics in* (2.1) *for some given initial condition $\vec{\rho}(0)$. Set*

$$r_c := \sup \{r : t \to \rho_x(t) \text{ decays at least exponentially fast for all } x\}$$

and let λ_{max} be the largest eigenvalue of the adjacency matrix A. Then

$$r_c \geq \frac{1}{\lambda_{max}}. \tag{2.6}$$

We call r_c the epidemic threshold.

This means, that when $\lambda_{max} \leq \delta/\beta$, the virus will die out at least exponentially fast over time.

As shown in the next lemma, it is easy to see that λ_{max} can not exceed the maximal weighted degree $d_{max} = \max_{x\in V} \sum_{y\neq x} w(x,y)$ of the graph, and we recall that it is also possible to show that the square root of the maximal degree $\sqrt{(}d_{max})$ gives a bound from below.

Lemma 2.1.5 (Eigenvalues bound by maximal degree)**.** *The eigenvalues of the adjacency matrix of a given graph are smaller or equal than the maximal degree.*

Proof. Let $G = (V, E, w)$ be a weighted graph and A be the corresponding weighted adjacency matrix with entries $a_{x,y}$ for all $x, y < n \in \mathbb{N}$ and maximal weighted degree d_{max}. Let λ be an eigenvalue of A, with corresponding eigenvector $u = (u_1, ..., u_n)$. Rescale u, such that $|u_k| \leq 1$ for all k. We get:

$$|\lambda| = |\lambda u_x| = |\sum_{y=1}^{n} a_{x,y}u_y| \leq \sum_{y=1}^{n} |a_{x,y}||u_y| \leq \sum_{y=1}^{n} |a_{x,y}| \leq d_{max} \tag{2.7}$$

$\square$

The above theorem and these relations between λ_{max} and d_{max} confirm the intuition that high degree nodes are the main cause for the spreading of a virus, but as we will see in concrete examples, the picture is more subtle.

Let us also stress that the highest eigenvalue is also connected to the connectivity of a graph and being more strongly connected makes a graph more vulnerable,

thus lowering the epidemic threshold. It is also positively correlated with the so-called loop capacity and path capacity [8]. These measures describe the quantity of loops/ paths of length $k = 2, 3, ...$, hence when these measures are high, it indicates stronger connection and therefore less robustness against a virus.
In a modified and slightly simplified version of the SIS model described above, the authors in [7] show that for arbitrary undirected and unweighted graphs, such an epidemic threshold is eventually equal to $1/\lambda_{max}$.

2.1.2 SIS and the Contact Process

We used the SIS dynamics on networks to describe evolution of infections via a system of PDEs. From a probabilistic perspective, there is a well known stochastic process, referred to as contact process (see e.g. [12]), which is morally analogous to the SIS dynamics. Let us recall its definition and briefly clarify the connection to the SIS defined above.

Definition 2.1.6 (Contact Process)**.** *Given a graph $G = (V, E)$ and fixed parameters $\beta, \delta > 0$, the contact process is the continuous-time Markov chain $(\eta_t)_{t\geq 0}$ with state space $\{0,1\}^V$ and the following local transition rates:*

$$c(x,\eta) = \begin{cases} \beta \sum \eta(y) & \text{if } \eta(x) = 0 \\ \delta & \text{if } \eta(x) = 1, \end{cases}$$

to go from a given configuration $\eta \in \{0,1\}^V$ to the configuration η^x, obtained from η by flipping the coordinate at node x.

This Markov process corresponds to dependent Bernoulli random variables $\eta_t(x)$'s evolving over time, representing the state of individual x at time t (1 for infected and 0 for susceptible). Now, fixing an initial configuration η_0 and defining

$$\rho_x(t) := \mathbb{E}\left[\eta_t(x)\right] = P(\eta_t(x) = 1), \quad x \in V \tag{2.8}$$

it is possible to show that these expectations satisfy the following coupled equations

$$\frac{\partial \rho_x(t)}{\partial t} = -\rho_x(t) + r \sum_{y \in V} a_{x,y} \rho_y(t) - r \sum_{y \in V} a_{x,y} \mathbb{E}\left[\eta_t(x)\eta_t(y)\right], \quad x \in V. \tag{2.9}$$

Now, the Bernoulli random variables $\eta_t(x)$'s are correlated and thus it is not possible to factorize $\mathbb{E}\left[\eta_t(x)\eta_t(y)\right]$ into $\mathbb{E}\left[\eta_t(x)\right]\mathbb{E}\left[\eta_t(y)\right] =: \rho_x(t)\rho_y(t)$. If they were independnet, then the factorization would hold true and the coupled system in (2.9) would reduce to the system in (2.2), implying that the average evolution of the contact process would be exactly the same as the SIS dynamics defined in the previous section. Yet, these correlations are present even if they should typically be small, in this precise sense the average contact process and the SIS dynamics differ. Yet in practice one model approximate well the other and vice versa.

2.2 k-node immunization problem

In this thesis, we are interested in how to make a network more resistant to the outbreak of a virus. More precisely, we want to pick the best set of k nodes in a network and remove them, such that the resistance against virus outbreaks is maximized. This is called the **k-node immunization problem**.
For some synthetic topologies like a star graph the answer to this problem is obvious, but for most graphs it is not trivial. Especially real networks rarely exhibit regular properties and need to be examined further.
A natural suggestion would be to remove the highest degree nodes, but a simple example shows that this can be in general suboptimal. Take a graph that consists of two cliques that are connected by one node, that acts as the bridge between them. It is easy to see that in terms of degree, the nodes to be removed would be part of the cliques, but in terms of immunization it would be best to remove the bridging node. For another example see Figure 5.1
It is evident, that there are many different approaches to the k-node immunization problem, but they will not all be mentioned here. Instead, we will use the previously derived epidemic threshold to construct a method that works on all finite networks and makes them more robust, as described in the previous section.
Since the epidemic threshold is linked to the largest eigenvalue of the adjacency matrix of a graph, the obvious approach is to find the set of nodes, that achieves the most significant drop in the largest eigenvalue, which will be referred to as the **eigendrop** and denoted by $\Delta\lambda$.
This can be done by simply computing the eigendrop for all possible sets, an algorithm that we will call "Comp-EV". For very small graphs this is possible, but due to the costly eigenvalue computations, choosing a set of k nodes already gets problematic for say $n = 50$, since there are $\binom{n}{k}$ possibilities that need to be tested. The computational cost of this method is $O(\binom{n}{k} \cdot m)$, which especially proofs infeasible when considering that we want to work with large real-world networks. Therefore, there is a need for a different approach. For that, [8] propose the "Shield-value".

Definition 2.2.1 (Shield-Value)**.** *Let $G = (V, E, w)$ be a graph with adjacency matrix A and $\mathcal{S} \subset V$ be a set of nodes. Further, let u be the eigenvector belonging to λ. Then define the Shield-value as:*

$$Sv(\mathcal{S}) = \sum_{x \in \mathcal{S}} 2\lambda u(x)^2 - \sum_{x,y \in \mathcal{S}} a_{x,y} u(x)u(y) \tag{2.10}$$

This score gives a set of nodes a high Shield-value, when the $u(x)$ are high and the $a_{x,y}$ are low or zero at the same time. In this, the second sum acts as a sort of repellence that punishes nodes that are too similar. The Shield-value actually approximates the eigendrop $\Delta\lambda(\mathcal{S})$, which is the eigendrop, if the rows and columns of the $x \in \mathcal{S}$ are removed from the adjacency matrix A [8].
We call the algorithm that computes the Shield-values of all possible sets to optimize the eigendrop "Comp-Sv". Its computational cost is $O(\binom{n}{k} \cdot k^2)$, but is still very costly, since all possible sets need to be tested.

Lemma 2.2.2. *Let $\lambda^{(\mathcal{S})}$ be the eigenvalue of $\hat{A}$, the adjacency matrix A without the rows* *and columns of $\mathcal{S}$. Let α be the difference between the largest and the second-largest eigenvalue of A and d_{max} be the highest degree of the graph. If λ is the simple largest eigenvalue of A and $\alpha \geq 2\sqrt{2kd_{max}}$, then*

$$\Delta\lambda(\mathcal{S}) = Sv(\mathcal{S}) + O(\sum_{y \in \mathcal{S}} \|a_{:,y}\|^2)$$

This lemma states, that Comp-Sv is a good approximation for Comp-EV. The proof for this can be found in [8].

2.3 Netshield and Netshield+

Approximating the eigendrop using Comp-Sv is already more preferable than computing the eigenvalues for all possibilities, but is still too costly for actual application. To approximate the shild-value, [8] propose two algorithms that are significantly less costly than Comp-Sv, while providing a good approximation of the eigendrop.

Algorithm 1 Netshield

```
    Input: Adjacency matrix $A$ and a number of nodes to be removed $k \in \mathbb{N}$
    Output: Set $\mathcal{S}$ with $k$ nodes
 1: Compute the largest eigenvalue $\lambda$ of $A$ and corresponding eigenvector $u$.
 2: Set $\mathcal{S} = \emptyset$
 3: for $j = 1$ to $n$ do
 4:     $v(j) = (2\lambda - A(j,j)) * u(j)^2$
 5: end for
 6: for $iter = 1$ to $k$ do
 7:     $B = A(:, \mathcal{S})$
 8:     $b = B \cdot u(\mathcal{S})$
 9:     for $j = 1$ to $n$ do
10:         if $j \in \mathcal{S}$ then
11:             $score(j) = -1$
12:         else
13:             $score(j) = v(j) - 2b(j) \cdot u(j)$
14:         end if
15:     end for
16:     $i = argmax_j(score(j))$; add $i$ to the set $\mathcal{S}$
17: end for
18: return $\mathcal{S}$
```

The first algorithm is called **Netshield**. It starts by computing the largest eigenvalue and the associated eigenvector. Then it uses these to compute the individual Shield-values of each node. Then the algorithm does k iterations, in each adding a node to the set $\mathcal{S}$. The nodes are added in a greedy manner, based on the *score* of each vertex, which is comparable to the Shield-value of node sets. Finally, the algorithm outputs the identified node set.
[8] show that Netshield has the computational complexity $O(nk^2 + m)$, where n is the cardinality of the vertex set and m is the cardinality of the edge set of the underlying graph. Since for the k-node immunization problems usually the k is

much smaller than the n, the complexity can be seen as linear with respect to the size of the input graph. Especially, when comparing it to the complexity of Comp-Sv, this is a great improvement. Additionally, [8] prove that Netshield is close in effectiveness to Comp-Sv.

Theorem 2.3.1 (Effectiveness of Netshield)**.** *Let $\mathcal{S}$ be the node set chosen by Netshield and $\hat{\mathcal{S}}$ be the set chosen by computing all shield-values, and $\Delta\lambda(\mathcal{S})$ and $\Delta\lambda(\hat{\mathcal{S}})$ be their respective eigendrops and e Euler's number. Then,*

$$\Delta\lambda(\mathcal{S}) \geq (1 - 1/e)\Delta\lambda(\hat{\mathcal{S}})$$

This theorem implies that Netshield is close to optimal in comparison to Comp-Sv. On the other hand, Comp-Sv is an algorithm that approximates Comp-EV, which is the exact maximal eigendrop. Thus, Netshield should provide a good approximation of the actual maximal eigendrop $\Delta\lambda$. In our applications Netshield achieved really good results, as we will see in chapter 6.
Still, this algorithm also has its limitations. The conditions mentioned in lemma 2.2.2 might not always be satisfied. We have shown in lemma 2.1.5 that $\lambda_{max} \leq d_{max}$, so the conditions simplify to $k \leq d_{max}/8$, which can lead to bad approximations, when the highest degree of a graph is relatively low. For this purpose, [8] propose a variation of Netshield that solves this problem and can even improve the achieved eigendrop.

Algorithm 2 Netshield+

```
    Input:Adjacency matrix $A$, a number of nodes to be removed $k \in \mathbb{N}$ and a
    batch size $b \in \mathbb{N}$ with $b \leq k$
    Output: Set $\mathcal{S}$ with $k$ nodes
 1: compute number of iterations $t = \lfloor \frac{k}{b} \rfloor$
 2: Set $\mathcal{S} = \emptyset$
 3: for $j = 1$ to $t$ do
 4:     Set $\mathcal{S}' = \emptyset$
 5:     $\mathcal{S}' = Netshield(A, b)$
 6:     $\mathcal{S} = \mathcal{S} \cup \mathcal{S}'$
 7:     update $A$ by deleting the rows and columns of the nodes in $\mathcal{S}'$
 8: end for
 9: if $k > tb$ then
10:     $\mathcal{S}' = Netshield(A, k - tb)$
11:     $\mathcal{S} = \mathcal{S} \cup \mathcal{S}'$
12: end if
13: return $\mathcal{S}$
```

Netshield+ is an iterative version of the original Netshield algorithm, that finds the set $\mathcal{S}$ in batches of size b instead of finding the whole set at once. After finding a batch, it removes it from the vertex set and recomputes the eigenvalues and eigenvector. Then it computes the next b nodes and this is repeated until all k nodes are found. The available information is updated after each round and thus, better approximations of the eigendrop are possible.
A question that might arise at this point, namely how to choose b. The batch size b can make a big difference when it comes to precision, but also computation time.

The complexity of Netshield+ is of the order $O(mk/b + nkb)$, and highly depends on b. For $b = k$ both Netshield and Netshield+ have the same complexity, however also the same results.
To improve the precision, a smaller b need to be chosen at the cost of greater computational complexity. It would be ideal to compute batches of size $b = 1$, but this will make the computations very long, especially for bigger graphs, since the most time-consuming part is computing the largest eigenvalues and their corresponding eigenvectors. Experimental results from [8] suggest, that choosing $b \approx k/10$ has a good trade-off between computation time and achieved eigendrop. This is in particular a suggestion for large networks. For rather small networks, as we will see in the experiments presented in chapter 6, a smaller b should be chosen. Due to the small size of the network, the overall computation time is not that high and normally not that many nodes need to be removed.

Chapter 3

Random Rooted Spanning Forest

This chapter presents the theoretical framework of the novel method for the k-immunization. The core of the method is based on a certain probability measure on the space of spanning forests of the graph studied by Avena and Gaudilliere in some recent works, see [4].
Firstly, mathematical objects needed to construct the forest measure will be defined. We will define the rooted spanning forest and the forest measure and introduce Wilson's algorithm, an algorithm that allows us to sample a rooted spanning forest from the forest measure. Next, we will expand on the properties of the forest and its root set.
All together, this chapter will contain the basis of the experimental work that will be presented in chapters 5 & 6.

3.1 Key objects

Definition 3.1.1 (Strongly Connected or Irreducible Network). *Let $G = (V, E, w)$ be a (directed) network. We call G strongly connected or irreducible, if for every pair $x, y \in V$ there is a sequence $\{(x_i, y_i)\}_{i=1}^{l} \in E$ for some $l \in \mathbb{N}$, such that $x_1 = x$, $y_l = y$ and $y_i = x_{i+1}$.*

Thus, a directed graph is strongly connected, when there is a directed path connecting any two vertices.

Definition 3.1.2 (Graph Laplacian). *Given a graph G on n vertices, the **(weighted) graph Laplacian** is a matrix L defined as*

$$L = A - D$$

where A is the weighted adjacency matrix of G and D is a $n \times n$ diagonal matrix with entries $D(x,x) = \sum_{y \neq x} w(x,y) := w(x)$.

The weighted graph Laplacian[1] only differs from the weighted adjacency matrix by the diagonal. We notice in particular that all rows of a graph Laplacian sum up to zero.
The Laplacian allows us to define a random walk associated to a graph G.

Definition 3.1.3 (Random walk associated to a graph). *Let $G = (V, E, w)$ be an irreducible, finite, directed and weighted graph. We call the random walk associated to*

[1]In the literature, sometimes the matrix $-L$ is also called *combinatorial Laplacian*.

G the continuous-time Markov chain $X = \{X(t) : t \geq 0\}$ with state space V and infinitesimal generator the graph Laplacian L.

This stochastic process has a variable jump rate which depends on the current state of the chain. If the Markov process is in position x, the next jump will happen at an exponential time of rate $w(x) = \sum_{y \in V \setminus \{x\}} w(x,y)$
Because the network is irreducible and finite by definition, there exists a unique invariant measure μ for the Markov process X.

In what follows, we will use the so-called hitting times as defined next.

Definition 3.1.4 (Hitting time of a set). *For a subset $A \subset V$ the hitting time of A is defined as*

$$T_A = inf\{t \geq 0 : X(t) \in A\}$$

and when A is empty, the hitting time is $T_\emptyset = \infty$

The hitting time of a set is the time it takes the Markov process X to reach a state that is part of the set A.

3.2 Rooted forest measure and sampling

Coming to the more interesting part of this chapter, we will now define the rooted spanning forest and its root set.

Definition 3.2.1 (Rooted Spanning Forest). *Given a (directed) graph $G = (V, E)$, a **rooted spanning forest** $\phi = (V_\phi, E_\phi)$ is a (directed) subgraph of G, that has the following properties:*

1. *$V_\phi = V$, so the set of vertices is identical for both graphs.*
2. *ϕ is acyclic.*
3. *For all $x \in V$, there is at most one $y \in V$, such that $(x, y) \in E_\phi$*

*The **root set** $R(\phi)$ of a rooted spanning forest ϕ is the subset of vertices $x \in V$, for which there is no $(x, y) \in E_\phi$.*

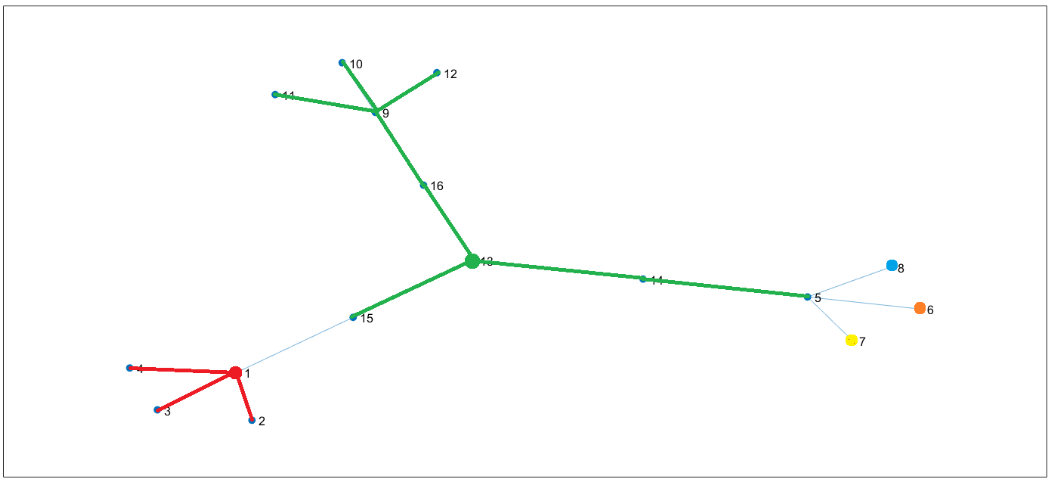


FIGURE 3.1: Random rooted spanning forest: The different rooted trees are colored in different colors with the big dots being the roots of each tree

The connected components of a rooted spanning forest are called rooted trees. The edges of each tree are directed towards the tree's root and every tree has exactly one root. The cardinality of the root set is thus equal to the cardinality of the trees. We stress that a rooted tree can be degenerate, i.e. it may consist of only one vertex, the root of the tree.
The **set of all spanning rooted forests** of a given graph G is denoted by $\mathcal{F}$. We also notice that since rooted spanning forests $\phi \in \mathcal{F}$ only differ by their edge set, within $\mathcal{F}$ they can be characterized by their respective (directed) edge sets E_ϕ.

In the context of the immunization problem, we will use certain random points (see Section 3.3) corresponding to the root set of a parametric random forest on $\mathcal{F}$ which we next describe.

Definition 3.2.2 (Random Forest of parameter q)**.** *Fix $q \geq 0$. We call random forest the random variable Φ_q with values in $\mathcal{F}$ characterized by the following distribution:*

$$\mathbb{P}(\Phi_q = \phi) = \frac{w(\phi) q^{|R(\phi)|}}{Z(q)}, \quad \phi \in \mathcal{F}, \tag{3.1}$$

with $w(\phi) = \prod_{e \in \phi} w(e)$, the weight of a given forest, $|R(\phi)|$ the cardinality of the root set, and $Z(q) = \sum_{\phi \in \mathcal{F}} w(\phi) q^{|R(\phi)|}$ the normalization constant referred to as partition function.

We note that even the degenerate case $q = +\infty$ can be included, which yields the deterministic degenerate forest $\Phi_\infty = \emptyset \in \mathcal{F}$, consisting of n roots and no edges (i.e. degenerate trees).

There exists a classical method that allows to sample a random rooted spanning forest according to the forest measure. This method is known as **Wilson's Algorithm**, see [16], and it is based on so-called loop-erasure of the random walk X associated to the graph. A version of this algorithm, that samples Φ_q for $q > 0$ is briefly described next in the spirit of [4].

Algorithm 3 Wilson's Algorithm

```
    Input: $q > 0$, $G = (V, E, w)$.
    Output: Random rooted spanning forest $\Phi_q$
 1: Set $i = 0$, $B_i = \emptyset$, $\phi_i = \emptyset$
 2: while $B_i \neq V$ do
 3:     Chose arbitrary $x \in V \backslash B_i$
 4:     Run Markov process $X : [0, T_q \wedge T_{B_i}] \to V$ starting in $x$
 5:     Erase the loops of the process $X$ to obtain the trajectory $(x_0, x_1, ..., x_k)$
 6:     $B_i = B_i \cup \{x_0, x_1, ..., x_k\}$                          ▷ add the nodes and
 7:     $\phi_i = \phi_i \cup \{(x_0, x_1), (x_1, x_2), ..., (x_{k-1}, x_k)\}$   ▷ the edges of the trajectory
 8:     $i = i + 1$
 9: end while
10: return $\Phi_q = \phi_i$
```

Wilson's Algorithm adds nodes and edges to a forest in an iterative manner. We begin with an empty set B_i of nodes and an empty forest ϕ_i. Starting with a random node $x \in V$ the algorithm runs the Markov process X associated to the graph and records the trajectory. In line 4 of the algorithm, T_q stands for an independent exponential random variable with parameter q and T_{B_i} is the hitting time of the set B_i. In the next step, the loops of the trajectory are erased, which gives a trajectory of the form:

$$\Gamma^x_{q,B_i} = (x_0, x_1, ..., x_k) \in \{x\} \times (V \backslash (B_i \cup \{x\}))^{k-1} \times (V \backslash \{x\}),$$

where all the nodes are different from each other. These nodes and the edges resulting from the trajectory are added to B_i and ϕ_i respectively. The steps in lines 3 - 8 are repeated with an $x \in V \backslash B_i$ until ϕ is a spanning forest, so until all nodes in V are also in B_i.
The roots in this algorithm can be obtained by examining the resulting Φ_q. The roots will correspond to the random walks that were terminated by T_q and the importance of the choice of q is evident once again.
It is possible to prove that this algorithm samples the random rooted spanning forest according to the right distribution, i.e. the distribution in equation (3.1). In particular, this algorithm is exchangeable, that is, does not depend on the specific choice of starting point in line 3. A proof of this can be found in [16] or in [3].

For our application it is important that the number of sampled roots can be controlled. Wilson's Algorithm is a randomized procedure, so the number of sampled roots is random. However, there is an iterative method presented in [4], that allows to sample the desired number of roots. In particular, this method is fast when one wants an approximate given number m with an error of size $\sqrt{m}$. By solving the equation

$$\sum_{j<n} \frac{q}{q + \lambda_j} = m,$$

where the λ_j are the eigenvalues of $-L$, we can find the desired q^*. But computing the eigenvalues is costly, especially for big networks. [4] proposes a fast algorithm, that allows to sample Φ_q with $m \pm 2\sqrt{m}$ roots.

Algorithm 4 Recursive Algorithm for root approximation

```
   Input: $q > 0, V$
   Output: Random rooted spanning forest $\Phi_{q_i}, q_i$
1: Set $i = 0, q_i = q$
2: repeat
3:     Sample $\Phi_{q_i}$ with Wilson's Algorithm
4:     if $|R(\Phi_{q_i})| \notin [m - 2\sqrt{m}, m + 2\sqrt{m}]$ then
5:         $q_i = mq_i / |R(\Phi_{q_i})|$
6:         $i = i + 1$
7:     end if
8: until $|R(\Phi_{q_i})| \notin [m - 2\sqrt{m}, m + 2\sqrt{m}]$
9: return $\Phi_{q_i}, q_i$
```

The non-negative input parameter q can be picked arbitrarily, since the algorithm converges rather fast. We also stress that in concrete examples Algorithm 4 returns the exact number of desired roots.

3.3 The set of roots as important nodes in a network

We have established what the forest measure is and also how to sample a random rooted spanning forest with the desired amount of roots. However, we still did not expand on the properties of the forest measure and its root set, which is actually the main object of interest for this research.

Theorem 3.3.1 (Partition function)**.** *For any $q > 0$, the normalizing constant $Z(q)$ in* (3.1) *is given by the characteristic polynomial of the graph Laplacian L evaluated at q, that is:*

$$Z(q) = det\left[q \cdot Id_n - L\right].$$

This is a version of the so called matrix-tree theorem, which in its origin can be traced back to Kirchhoff [10]. It states that the partition function can be characterized in terms of the eigenvalues of the Laplacian. See [4] for the proof of a more general version of this theorem.

We can now finally discuss the random root set $R(\Phi_q)$ which is the main tool we will use in the experiments to come. A crucial property of this set is that its distribution is explicit and give rise to a so-called determinantal process related to how the random walk X observes the graph on timescale $1/q$. This is described in the next theorem.

Theorem 3.3.2 (Determinantal roots with killed random walk kernel)**.** *For a fixed $q > 0$,*

$$\mathbb{P}(A \subset R(\Phi_q)) = \det\left[K_q\right]_A,$$

with

$$K_q(x,y) := P_x(X(T_q) = y), \;\; x, y \in V$$

for any $A \subset V$. Here $\mathbb{P}$ denotes the probability w.r.t. to the distribution of the forest and P_x denotes the probability w.r.t. to the random walk X starting in $x \in V$. Further, $\left[K_q\right]_A$ stands for the matrix K_q restricted to the rows and colums of the set $A \subset V$.

The above statement allows for explicit computations and its derivation can be found in [4]. In particular, the theorem implies that if $w(x,y) = w(y,x)$, then the roots repel each other and are pairwise negatively correlated. In fact, for all $x, y \in V$ we get:

$$\begin{aligned}\det[K_q]_{\{x,y\}} &= K_q(x,x)K_q(y,y) - K_q(x,y)K_q(y,x) \\ &= K_q(x,x)K_q(y,y) - K_q(x,y)^2 \leq K_q(x,x)K_q(y,y),\end{aligned}$$

and thus $\mathbb{P}(x, y \in R(\Phi_q)) \leq \mathbb{P}(x \in R(\Phi_q))\mathbb{P}(y \in R(\Phi_q))$, which shows the negative correlation.
For the next property, a new object needs to be introduced.

Definition 3.3.3 (Partition induced by the forest)**.** *For a spanning forest ϕ, $\mathcal{P}(\phi)$ is the disjoint partition of V associated with the spanning forest ϕ. Two elements $x, y \in V$ are in the same block of the partition, if they are part of the same tree.*

Theorem 3.3.4 (Roots at restricted equilibria). *Let $[A_1, ..., A_m]$ be an arbitrary partition of V with $m \leq n$ subsets and and fix $r_i \in A_i$, $i = 1, ..., m$. Then*

$$\mathbb{P}(R(\Phi_q) = \{r_1, ...r_m\} | \mathcal{P}(\Phi_q) = [A_1, ..., A_m]) = \prod_{i=1}^{m} \mu_{A_i}(r_i)$$

given that the event, that is conditioned on, has a non-zero probability. Here, μ_{A_i} is the invariant measure of the continuous-time random walk associated to the graph restricted to the points in A_i.

This theorem implies that if we condition on the induced partition $\mathcal{P}(\Phi_q)$, where the vertices are partitioned into k blocks that correspond to the spanning trees, we can conclude that within the spanning trees, the k roots are distributed according to the invariant measures μ of the random walk X restricted to these blocks. That also means that if conditioning on having only one tree, which is the case for sufficiently small q, the sampled root will be distributed according to the invariant measure of the random walk X on the whole graph.
Furthermore, we can express the cardinality of the root set as a sum of independent Bernoulli distibuted variables, given that all eigenvalues of $-L$ are real.

Theorem 3.3.5 (Number of roots). *For a fixed $q > 0$. Set*

$$p_j(q) = \frac{q}{q + \lambda_j}, \; 0 \leq j \leq n - 1,$$

Split the eigenvalues in

$$J = \{j \leq n - 1 : \lambda_j \in \mathbb{R}\},$$

and define independent random variables B_j with distribution:

$$P(B_j = 1) = p_j(q), \; P(B_j = 0) = 1 - p_j(q), \quad j \in J$$

When $q > 0$ then $|R(\Phi_q)|$ is distributed as

$$S_q = \sum_{j \in J} B_j$$

Since in this thesis we work with symmetrical adjacency matrices and thus a real spectrum, we restricted the message to real eigenvalues only. However, this theorem, including the proof, can be found in [4] in a more general form for both, real and complex, eigenvalues.

We notice that since $\lambda = 0$ is always an eigenvalue of the Laplacian, the last theorem is consistent with the fact that at least one root is present. More generally, since the quantity of zero eigenvalues of Laplacians equals the number of connected components, every connected component will have at least one root.
Also, by differentiating the normalization partition function $Z(q)$ with respect to q, the momenta of $|R(\Phi_q)|$ can be obtained and we get the mean number of roots

$$\mathbb{E}\left[|R(\Phi_q)|\right] = \sum_{j<n} \frac{q}{q + \lambda_j}$$

and the variance

$$Var\left[|R(\Phi_q)|\right] = \sum_{j<n} \frac{q}{q+\lambda_j} - (\frac{q}{q+\lambda_j})^2$$

Chapter 4

Using the rooted spanning forest to maximize the eigendrop

After the formal introductionin to the forest method in the previous chapter, we will take an applied standpoint. In particular, we will deepen the understanding of the root set and then continue with the choice of Laplacian, that is crucial for the outcome of the sampling. We will close the chapter with results and conclusions about the forest method that were gained in the course of experiments on artificial and real networks.

The guiding question of the research that lead to this thesis was: Can the forest measure and in particular, the root set be used for network immunization problems? This question arose before we researched other methods of network immunzation or how this immunzation can be measured, but was purely motivated by the remarkable features of the root set.
In Chapter 2 we already established, that the eigendrop is a meaningful characteristic when immunizing a network and presented a fast algorithm that works by approximating the maximal eigendrop. Now, we will return to the random rooted spanning forest and dive into why it is meaningful for network immunization and how it can be tuned to deliver good results in the context of maximal eigendrop. For that we will inspect the forest from a more applied perspective.

4.1 The root set and its complement

In the previous section we described the main properties of the root set $R(\Phi)$ that make it an interesting object per se. However, from a heuristic perspective, what characterizes a good set of nodes to be removed from a network to increase robustness? These nodes need to be well-distributed throughout the network. Additionally, usually these nodes should have a relatively high degree such that they eliminate many links and paths when removed.
Indeed, when looking at theorem 3.3.2 we obtain that the roots in $R(\Phi)$ are well-distributed and even repel each other under the right conditions. Also, theorem3.3.4 indicates that with the right choice of graph Laplacian well-connected nodes are more likely to be a root than nodes of relatively low degree.
These properties and the versatility of the forest measure were the reason that initiated this research project. However, after experimenting with the roots we also

considered another set, namely the complement of the root set. The properties of the root set described in the previous section can be extended to the complementary set: As well as the actual root set $R(\Phi)$, its complement $R(\Phi)^C = V \backslash R(\Phi)$ is also determinantal, which is a known property of determinantal point processes. In addition to that, the complementary root set is also well-distributed within the graph and its distribution can be explicitly characterized , see section 3.4.2 in [5] for details. These properties make it a valuable object to consider for the experiments. Ultimately, we found that either set can be relevant, depending on the choice of Laplacian.

4.2 Manipulating the random rooted forest through choice of a graph Laplacian

One property that makes the forest method so versatile is the possibility to adjust the measure to the problem by choosing a graph Laplacian. The graph Laplacian assigns a weight structure to the network and thus has a significant influence on the outcomes of the sampling and the related random walk (see 3.1.3). Therefore, making the right choice is crucial. This will be even further illustrated in the next chapter.
Throughout the experiments, we tried three different Laplacians, out of which two delivered good results under the right conditions. We will first introduce the two promising variants and lastly, shortly describe the last option and why we decided to discard it in the context of network immunization.

4.2.1 (Original) Graph Laplacian

We started the experiments with the original (weighted) graph Laplacian $L = A - D$ where A is the adjacency matrix with entries $a_{x,y}$ and D is a diagonal matrix with entries $D(x,x) = \sum_{y=1}^{n} a_{x,y}$. As captured in the next lemma, if the original network is undirected then the correspondent random walk has uniform invariant measure on the vertex set of the graph (and hence so it is distributed the unique root of the random spanning tree obtained for q sufficiently small with the rooted forest).

Lemma 4.2.1. *Consider an undirected weighted connected network $G = (V, E, w)$, that is, with $w(x,y) = w(y,x)$ for any $x, y \in V$. Then the associated random walk with infinitesimal generator given by the graph Laplacian has uniform invariant measure.*

Proof. We first observe that the Markov chain is finite and irreducible, hence the classical Perron-Frobenious theorem guarantees that there exists a unique non-degenerate invariant measure. Further, it is easy to see that the random walk satisfied the detailed balanced equation with respect to the uniform measure $\pi(x) = 1/|V|, \forall x \in V$, in fact:

$$\pi(x)w(x,y) = w(y,x)\pi(y) \quad \text{for all } (x,y) \in E$$

if and only if $\pi(x) = \pi(y)$, since $w(x,y) = w(y,x)$. We have thus showed that $\pi(x) = 1/|V|$ is a reversible measure for the random walk. Due to the uniqueness of the invariant measure and the fact that a reversible measure is invariant, we get the claim $\square$

The first experiments using the graph Laplacian were discouraging, since for small k (with respect to the graph size), the most likely sets were indeed well spread within the graph and the nodes were repelling each other, but their eigendrop was always very low.
For the purpose of network immunization small k are far more interesting, since in practice immunization often comes with some kind of cost function and the cost needs to be kept at a minimum. However, for large k close to n we found that the forest measure was choosing some sets with significantly higher frequency than others. This encouraged us to do the same experiments, but instead of sampling k roots, we sampled a set of $k^C = |V| - k$ roots and looked at the non-root nodes. The latter would correspond to consider $R(\Phi_q)$ as the set of susceptible individuals to be kept in the network and $R(\Phi_q)^C$ as the set of "dangerous-spreader" nodes to be removed. This variant indeed produced more relevant results, as the algorithm started to discriminate between sets and the most chosen sets were often corresponding to the set with the highest eigendrop of the example graphs.
In particular, when using the complement set $R(\Phi_q)^C$ associated to the graph Laplacian to sample a single vertex, has captured in the next lemma, this node is distributed proportionally to the (weighted) degrees.

Lemma 4.2.2. *Let $G = (V, E, w)$ be an undirected weighted connected network and Φ_q, for some $q > 0$, be the rooted forest associated to the graph Laplacian of this graph. Then*

$$\mathbb{P}(R(\Phi_q)^C = \{x\}) = \frac{w(x)}{\sum_{y \in V} w(y)}, \quad \forall x \in V.$$

Proof. The probability that x is the only element in the complementary of the root set coincides with the event that all roots are degenerate, except x, and there is a single edge in the whole spanning tree. From the distribution of the forest we can thus compute

$$\mathbb{P}(R(\Phi_q)^C = \{x\}) = \sum_{F \in \mathcal{F} : R(\Phi_q)^C = \{x\}} \mathbb{P}(\Phi_q = F)$$

$$= \sum_{y \sim x} \frac{q^{n-1} w(x,y)}{Z(q)} = \frac{w(x) q^{n-1}}{Z(q)}.$$

The second equality is according to definition 3.2.2 and the last equality is because the sum of all weights connecting x to its neighbours is exactly $w(x)$. $\frac{w(x)q^{n-1}}{Z(q)}$ is then distributed proportional to the outgoing weights of the $x \in V$. □

4.2.2 Degree-Biased Laplacian

As in the beginning the experiments with the simple graph Laplacian showed little prospect, we started thinking about a different Laplacian that would put a stronger focus on the degrees. Therefore, a Laplacian which leads to degree biased sampling of the roots was included into the experiments.
This graph Laplacian is of the form $\tilde{L} = D^{-1}A - Id$, where as usual A is the adjacency matrix and D the diagonal matrix with entries $D(x,x) = \sum_{y \in V} a_{x,y} = w(x)$ and Id is the identity matrix of size $V \times V$. In the case that the graph is unweighted, we notice that the matrix D^{-1} has the inverse degrees on the diagonal.

We will call this normalized Laplacian $\tilde{L}$ the *degree-biased Laplacian*. In particular, as the next lemma shows, if the given weighted network is undirected, then the invariant distribution of the associated random walk with generator $\tilde{L}$ is proportional to the weighted degree

Lemma 4.2.3. *Let $G = (V, E, w)$ be an undirected weighted connected network and let $\tilde{L}$ denote the degree-biased Laplacian of this graph. Then the random walk on V with infinitesimal generator given by $\tilde{L}$ has invariant measure proportional to the weighted degree.*

Proof. We can argue as in Lemma 4.2.1 and check that the unique invariant measure is characterized by a reversible measure given the degree-biased distribution. In fact, detailed balanced with respect to this generator $\tilde{L}$ reads as

$$\pi(x)\tilde{L}(x,y) = \tilde{L}(y,x)\pi(y) \quad \text{for all } (x,y) \in E.$$

By definition of $\tilde{L}$, this is equivalent to

$$\pi(x)\frac{w(x,y)}{w(x)} = \frac{w(y,x)}{w(y)}\pi(y),$$

which due to the assumption that $w(x,y) = w(y,x)$ can be re-written as

$$\frac{\pi(x)}{\pi(y)} = \frac{w(x)}{w(y)},$$

from which, using that π must be a non-degenerate probability measure, we conclude that $\pi(x) = w(x)/\sum_{y\in V} w(y), \forall x \in V$. □

We notice that in view of this lemma, when sampling a unique node with root set associated to $\tilde{L}$, it is distribution is proportional to the weighted degree of the network. As we saw in Lemma 4.2.2 this is exactly the same as when using the complementary set associated to L, yet this equivalence in distribution is true only for $k = 1$. As we will see in the experiments, the results of the sampling for small k have in fact similarities to the results of *the complement root set associated to the original graph Laplacian*, but they are not the same. For the example graphs that we used for the experiments, both degree-biased Laplacian and the complement of the simple graph Laplacian lead to comparable decisions in terms of node immunization.
Out of curiosity, we also performed experiments with the complementary root set associated to $\tilde{L}$, just like we did for the graph Laplacian. Related experiments performed badly for the eigendrop immunization problem, so this variant was not pursued any further.

4.2.3 Eigenvalue-Biased Graph Laplacian

The last graph Laplacian was the most experimental. The other variants were already producing good results, but it was not directly evident why they found the highest eigendrop sets with higher frequency than others. The thought behind the *eigenvalue-biased Laplacian* was to construct a Laplacian that was directly connected to the eigenvalues. This had the goal to tailor the Laplacian to the problem

to be solved, but also to determine if we correctly understand the mechanisms of the forest method.
The idea was to have a Laplacian with two absorbing states. It is set up the following way: First, we construct a Sublaplacian of the form $L' = A - d_{max} \cdot Id$, where d is the maximal out degree of the graph. To make this Sublaplacian into a Laplacian we need to find b_i for all rows, such that the row sums are all equal to zero. Adding the column vector b and a zero row vector to L' makes it into a Laplacian $\hat{L}$, with the b_i being an absorbing state.
Now, by subtracting a diagonal matrix with q on the diagonal , and linking a column vector with entries q and a zero row vector to the Laplacian $\hat{L}$, we get the final Laplacian $\dot{L}$.
Contrary to the theory, this Laplacian did not produce good results. It seemed to prefer the high eigendrop sets, but they were relatively only slightly more likely than bad sets. The results were thus not very clear and for real application this method would be a lot more risky than the simple graph Laplacian and degree-biased Laplacian. In the end, this variant is more complicated and less reliable than the other options, thus we decided to discard it.
Of course, we also tested Option 3 complement, but the results were even worse and always somewhat close to uniform sampling and therefore useless in the context of maximal eigendrop.

4.3 Applying the forest method to node immunization problems

To get the best results when applying the forest method to eigendrop maximization problems, it is necessary to implement it the right way. There are several choices that need to be made, e.g. about the Laplacian, but also how big to choose the sample and other factors that can influence the end result. We stress that we report here the first experimental findings of the forest approach and that improvements of the method are in progress in a joint collaboration with Luca Avena, Alexandre Gaudillière and Michael Emmerich.

4.3.1 Complement of the simple graph Laplacian vs. degree-biased Laplacian

Both, the complement of the simple graph Laplacian and degree-biased Laplacian showed good results during the initial experiments. However, the complement of the simple graph Laplacian finally had more advantages than the degree-biased Laplacian and subsequent research conducted by Luca Avena and Alexandre Gaudillière confirmed the value of the complement of the simple graph Laplacian. The differences between the two Laplacians will be further elaborated in the next chapter and a detailed comparison will be presented for better understanding.

4.3.2 Sample size

For some of the experiments in this thesis, a sample size of 500.000 was used. This is of course ideal if you have a small graph and want to gain a lot of insight about the method, but for real world application this would be simply infeasible.

The dilemma is, that with a bigger graph and especially a bigger k you need a bigger sample, purely because there are so many possibilities. The forest method is a random algorithm, so having more samples is more preferable to increase the probability of sampling a 'good' set.
But also, the bigger the graph, the more complex it gets in a computational sense. In addition to that, for all the samples the eigenvalue needs to be computed or at least approximated, which adds vastly to the computation time. This is a big limitation when it comes to this method and finding a balanced sample size is necessary to achieve good results.
For the small graphs in chapter 5 we used very big samples, since the purpose of the experiments was to explore the mechanisms of the forest method and to find a graph Laplacian that can be used for eigenvalue maximization problems.
Taking such big samples was however not an option for the experiments in chapter 6. We wanted to see how the algorithm performs on bigger, real networks and also to compare its performance to the Netshield algorithms. Netshield and Netshield+ are computationally relatively fast and the challenge was to have comparable results with comparable or even better computation times.
To find a balance between sample size and precision, the idea was to use a sparsity parameter to decide about the size of the sample. The sparsity of a network is

$$c = \lceil m/n \rceil,$$

and the number of samples should be equal to c. Using this approach it was possible to achieve computation times similar to the Netshield algorithms.

4.3.3 Batches

Inspired by Netshield+, we also tried sampling the nodes in batches. Instead of sampling all k nodes at once, only b nodes are sampled in each round and the graph gets updated after each iteration. Just as it is with Netshield+, this way the algorithm works with the most current information when choosing nodes, which can lead to better results.
The experimentation with batches in the course of this research project was only limited, however it was later further explored by Luca Avena and Alexandre Gaudillière.

Chapter 5

Experiments on small graphs for different graph Laplacians

In this chapter the behaviour of the forest method on small regular and irregular graphs will be explored. Different variants of graph Laplacians were compared to each other in order to better understand their mechanisms and find a Laplacian that produces the best results in the context of network immunization.
In the first section the focus is on a very regular graph. The results of the first experiments will be used to form assumptions about the mechanisms of the different Laplacians. Thereafter, in the second section of this chapter, the previously made assumptions will be tested using an irregular social network. After analysing the results of both experiments, we will form a heuristic conclusion about which of the proposed Laplacians-root-sample combinations is most relevant for network immunization. This Laplacian will then be used for further experiments on bigger complex networks in chapter 6. We will use abbreviations for the different variants, so the simple graph Laplacian becomes SGL, the degree-biased graph Laplacian becomes DBGL and the eigenvalue-biased graph Laplacian becomes EBGL. When we take the complement of the root set, these abbreviations will start with a "C", e.g. CSGL for taking the complement of the root set sampled using the simple graph Laplacian.
The analysis of the data and in particular the eigenvalues was performed using MatLab.

5.1 The Synthetic Graph

For the first experiments we chose a graph that is easy to comprehend, but yet complex enough to be a valuable test subject. The graph that will be referred to as the *Synthetic Graph* has 16 vertices, 15 undirected edges and is unweighted. The maximal eigenvalue of this graph is 2.3028.

The *Synthetic Graph* is small enough to compute all possible combinations of node sets and their respective eigendrops. This makes the *Synthetic Graph* ideal to illustrate, how the forest method works and to explore different graph Laplacians. For all variants of the experiment discussed here, we performed 500.000 experiments for the roots and the complement of the roots. The large sample size is mostly due to eigenvalue-biased graph Laplacian and to find out if the frequencies would eventually converge, while the other variants converged for samples

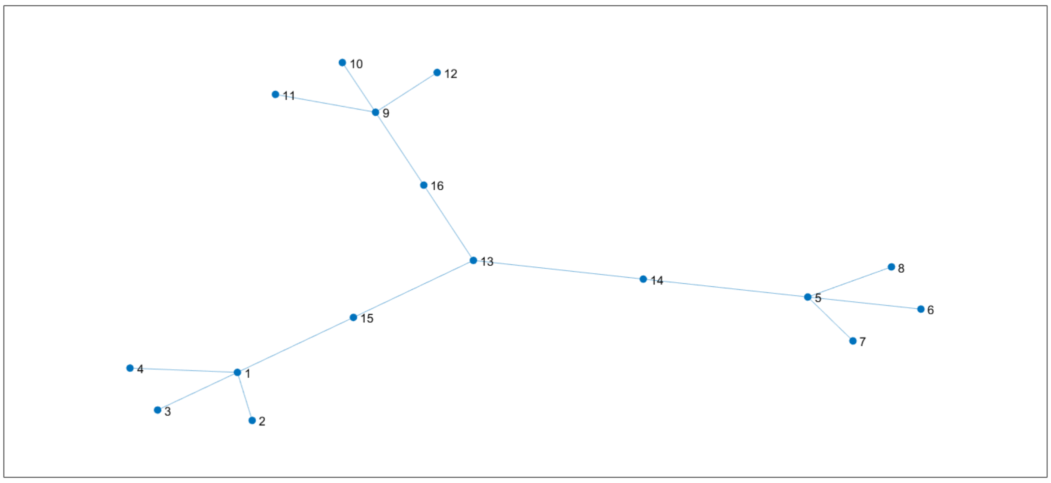
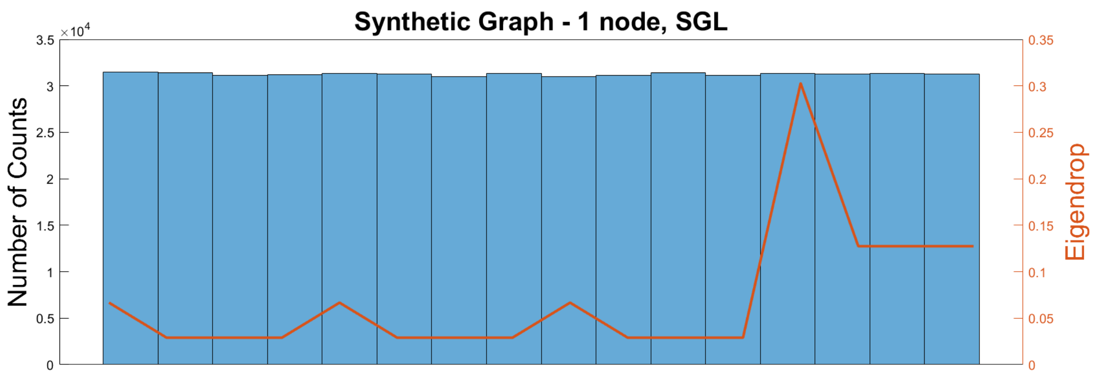


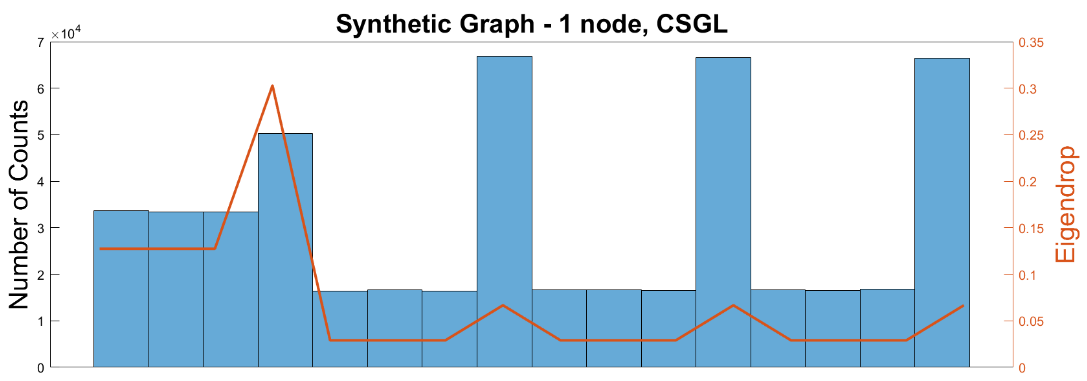
FIGURE 5.1: The Synthetic Graph

of significantly smaller size.
The sets are not labeled in the graphs, since for $k > 1$ that would make the plots unreadable. However, the sets are in increasing order, e.g. first $\{(1, 2, 3)\}$, then $\{(1, 2, 4)\}$ and so on, for the regular cases, and in decreasing order, e.g. first $\{(10, 9, 8)\}$, then $\{(10, 9, 7)\}$ and so on, for the complementary case. On the left axis, the number of counts is displayed, which corresponds to the height of the bars and represents how often a set was sampled. On the right axis, the achieved eigendrop is displayed, which corresponds to the orange line in the plot.
Here, the results of the experiments for $k = 1, 3, 4$ will be presented. We performed the experiments for all k, but these cases were the most insightful and meaningful. In particular, for $k \geq 4$, removing the optimal sets results in a graph with no edges and the eigendrop is equal to the maximal eigenvalue. Therefore, there is no necessity to look at $k > 4$ in the context of immunization.

5.1.1 Removing 1 node

In the case of $k = 1$ the best node to be removed is node 13, which achieves an eigendrop of **0.3028**. In total, there are 16 different sets that the algorithm can sample.

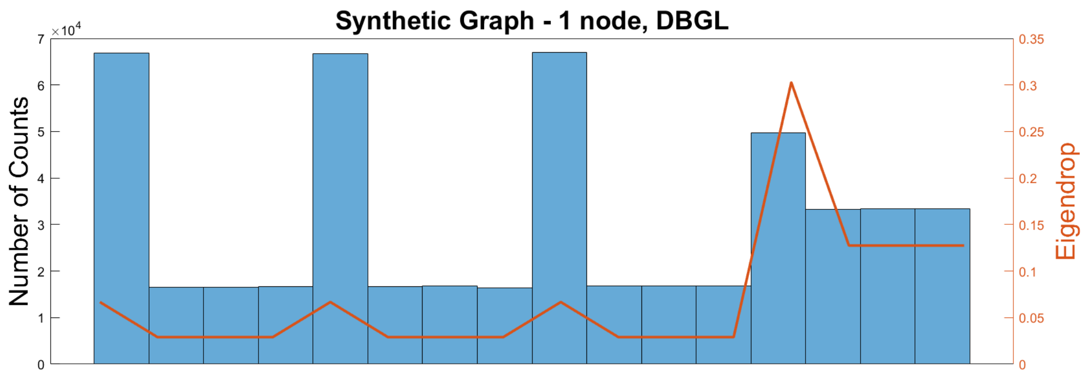


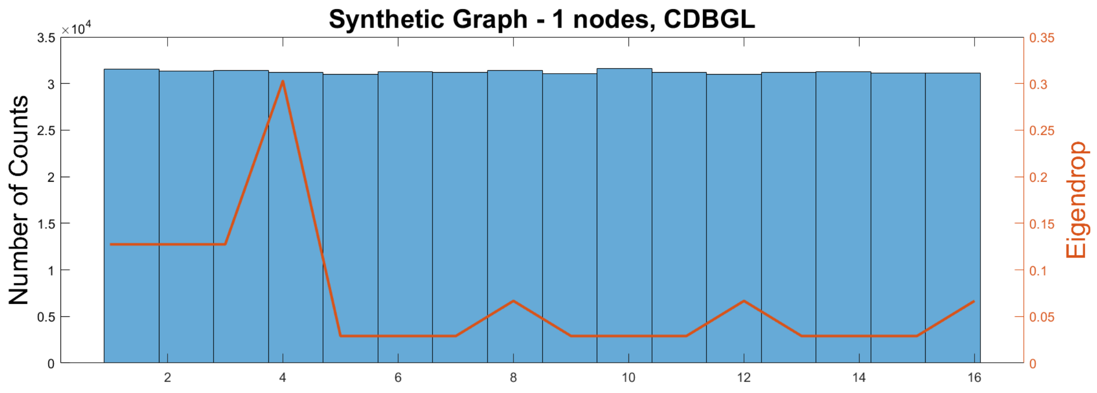


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1	31485	0.0667076602322045	9	66895	0.0667076602322054
2	31428	0.0289932192437967	5	66617	0.0667076602322054
11	31417	0.0289932192437976	1	66457	0.0667076602322045
8	31356	0.0289932192437976	13	50231	0.302775637731994
15	31347	0.12744789057092	16	33681	0.127447890570919
13	31335	0.302775637731994	15	33388	0.12744789057092
5	31323	0.0667076602322054	14	33327	0.12744789057092
6	31287	0.0289932192437976	2	16744	0.0289932192437967
14	31268	0.12744789057092	4	16725	0.0289932192437967
16	31262	0.127447890570919	8	16674	0.0289932192437976

TABLE 5.1: 10 most removed sets for SGL (left) and CSGL (right)

We can see that for the SGL all sets are chosen with approximately the same frequency and that the algorithm does not discriminate between sets. For node immunization this is a bad factor.
However, for the CSGL the plot is very different. We can clearly see that the magnitude of the different bars is proportional to their degrees. When also taking into account Table 5.1., the frequencies of the different sets are approximately $w(x) / \sum_{x \in V} w(x)$, just as stated in theorem 4.2.2.

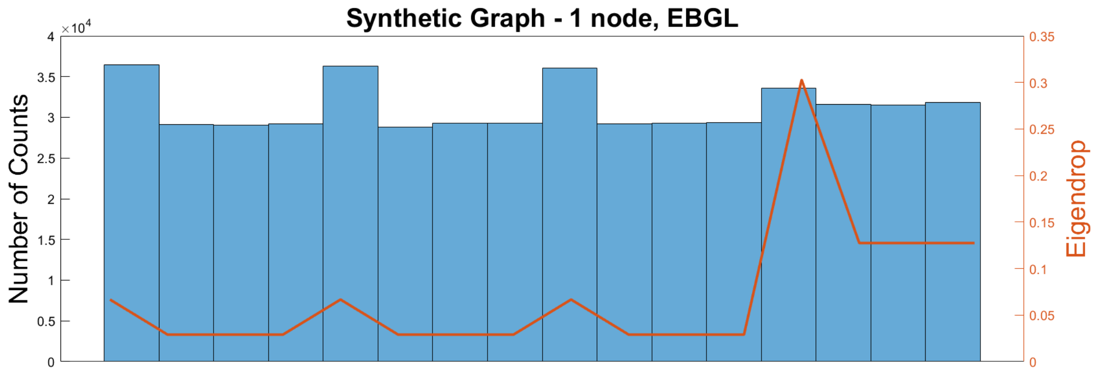


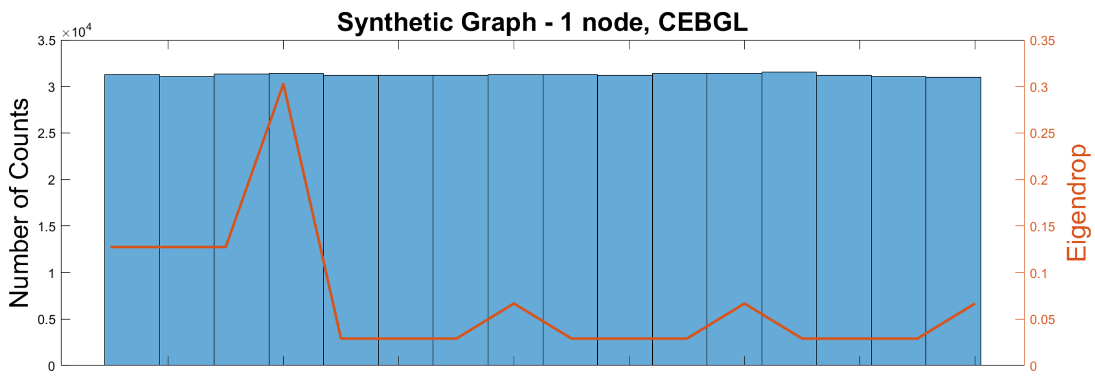


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
9	66939	0.0667076602322054	7	31611	0.0289932192437976
1	66822	0.0667076602322045	16	31557	0.127447890570919
5	66682	0.0667076602322054	14	31386	0.12744789057092
13	49701	0.302775637731994	9	31384	0.0667076602322054
16	33437	0.127447890570919	15	31356	0.12744789057092
15	33381	0.12744789057092	11	31282	0.0289932192437976
14	33251	0.12744789057092	3	31259	0.0289932192437967
7	16856	0.0289932192437976	4	31219	0.0289932192437967
12	16781	0.0289932192437976	10	31206	0.0289932192437976
11	16744	0.0289932192437976	13	31172	0.302775637731994

TABLE 5.2: 10 most removed sets for DBGL (left) and CDBGL (right)

Just like the SGL, the CDBGL samples all sets with approximately equal frequency and is not very interesting to look further into in this context. The samples of the DBGL are very similar to the ones of the CSGL, even in frequencies of the different sets. In the case of $k = 1$ both these Laplacians produce the same degree biased results, with approximately the same relative frequencies.





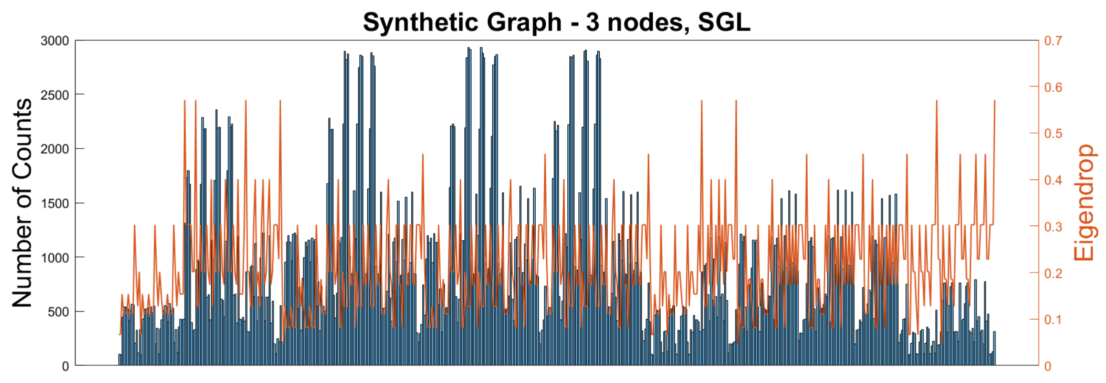
Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1	36457	0.0667076602322045	4	31537	0.0289932192437967
5	36313	0.0667076602322054	13	31422	0.302775637731994
9	36075	0.0667076602322054	6	31413	0.0289932192437976
13	33607	0.302775637731994	5	31407	0.0667076602322054
16	31801	0.127447890570919	14	31314	0.12744789057092
14	31566	0.12744789057092	8	31298	0.0289932192437976
15	31490	0.12744789057092	16	31264	0.127447890570919
12	29383	0.0289932192437976	9	31245	0.0667076602322054
11	29315	0.0289932192437976	3	31231	0.0289932192437967
7	29298	0.0289932192437976	7	31211	0.0289932192437976

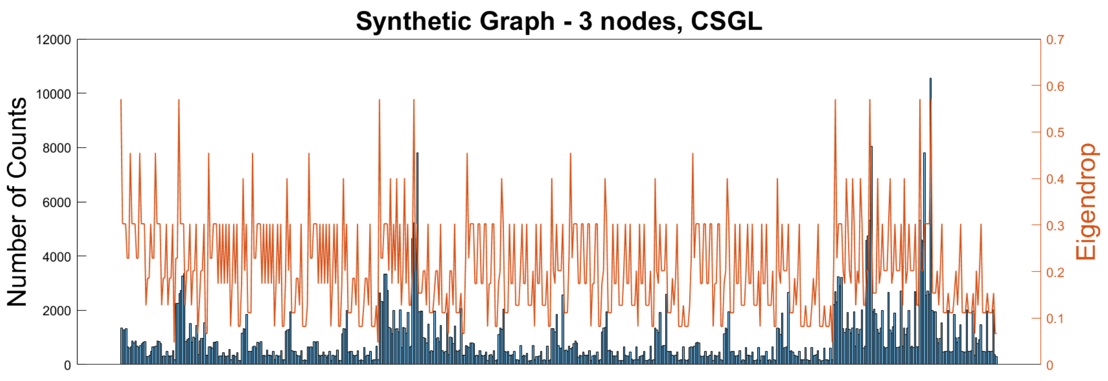
TABLE 5.3: 10 most removed sets for EBGL (left) and CEBGL (right)

The CEBGL exhibits the same uniform sample as the SGL and the CDBGL. However, the EBGL is very interesting. It does some discrimination based on the degree, but it is not proportional. We tried increasing the sample to see if the frequencies would converge towards a specific ratio, but even for samples bigger than 500.000 the outcomes looked similar. It seems like this Laplacian is manipulating the algorithm towards a degree biased sample, but there also seems to be a lot of noise.

5.1.2 Removing 3 nodes

In the case of $k = 3$ there are eight sets that achieve an eigendrop of **0.5707**. The best sets to be removed are all combinations of the peripheral center nodes 1, 5 and 9 and the bridge nodes 14, 15 and 16, where each node is in a different arm of the graph. In theory, the forest measure should prefer the set $\{(1, 5, 9)\}$, since the roots repel each other. In total, there are 560 different possibilities to pick a set.



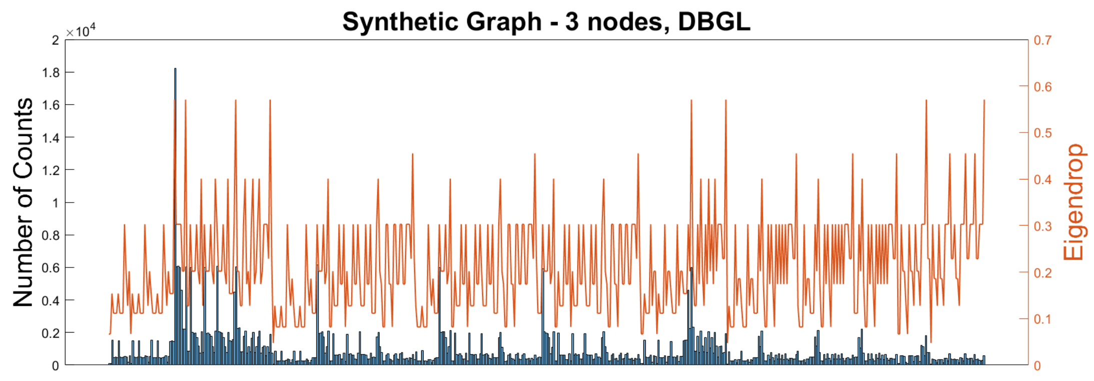


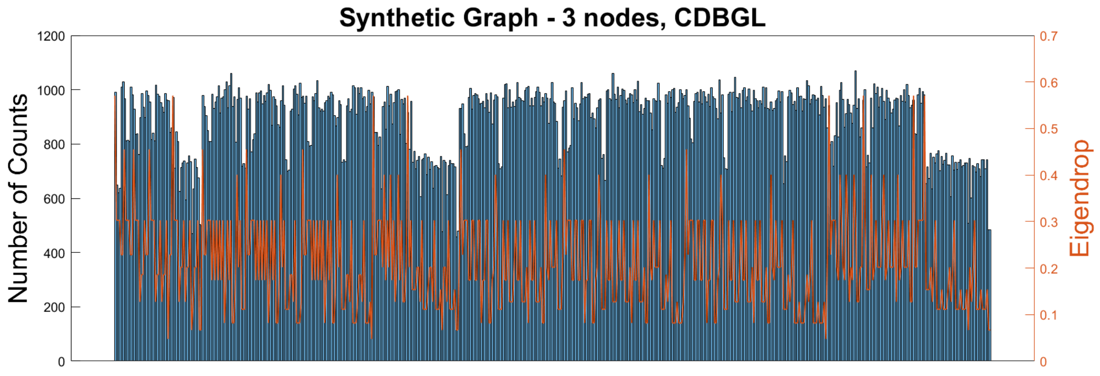
Sets	Counts	Eigendrop	Set	Counts	Eigendrop
3 6 11	2929	0.127447890570919	1 5 9	10552	0.570724830163117
3 7 10	2929	0.127447890570919	1 9 13	8043	0.302775637731995
3 6 12	2915	0.127447890570919	5 9 13	7808	0.302775637731995
4 7 11	2905	0.127447890570919	1 5 13	7800	0.302775637731995
4 7 10	2898	0.127447890570919	1 9 14	5328	0.570724830163117
2 6 10	2897	0.127447890570919	1 5 16	5316	0.570724830163117
4 8 11	2897	0.127447890570919	5 9 15	5230	0.570724830163117
2 8 10	2885	0.127447890570919	1 9 15	4752	0.201772648116536
3 7 11	2880	0.127447890570919	5 9 16	4645	0.201772648116536
2 6 12	2869	0.127447890570919	5 9 14	4637	0.201772648116536

TABLE 5.4: 10 most removed sets for SGL (left) and CSGL (right)

The most chosen sets of the SGL all exclusively consist of peripheral nodes with low eigendrop, as is clear from the table. However, this also beautifully illustrates the mechanisms of the forest method, since the algorithm preferred nodes with the highest distance possible. In the plot we can also see that sets with a high eigendrop have mostly low frequencies and the counts vs. the eigendrops seem negatively correlated.
On the other hand, when looking at the CSGL, the most likely set is the predicted set $\{1, 5, 9\}$, which is also one of the highest-eigendrop sets. The three next most likely sets can be grouped together, since they consist of two peripheral centers and the central node. This indicates that the forest method indeed prefers high degree sets. That pattern also repeats for the next three sets, that consist of two peripheral centers and one bridging node. Also, just like in the SGL, for more

likely sets the nodes are distributed well within the graph.



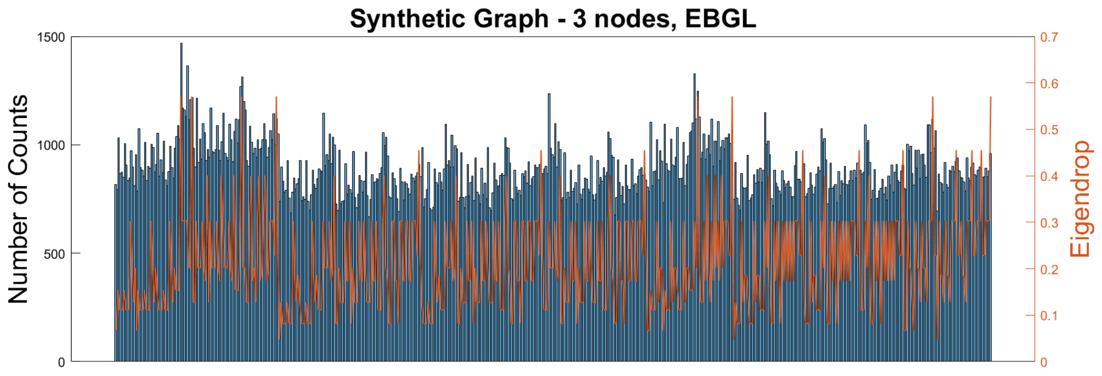


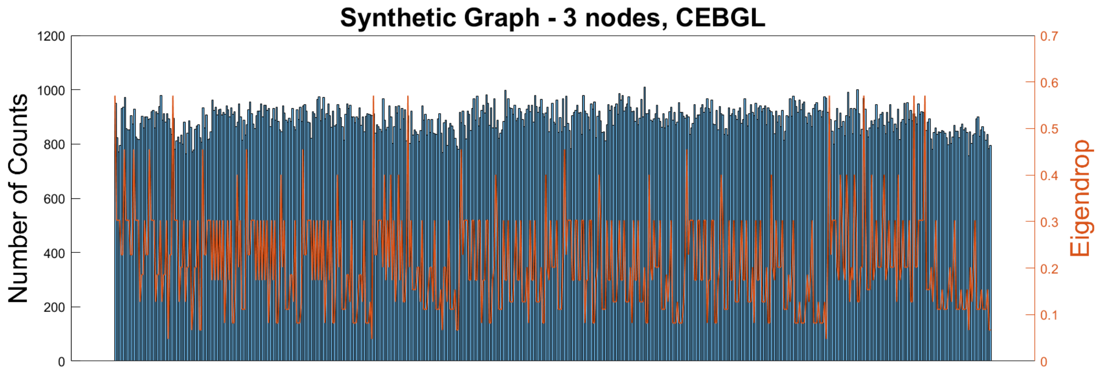
Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 5 9	18214	0.570724830163117	1 10 13	1070	0.302775637731994
2 5 9	6157	0.302775637731994	8 10 13	1060	0.302775637731994
1 8 9	6087	0.302775637731995	3 8 12	1059	0.127447890570919
1 5 11	6061	0.302775637731993	2 8 12	1047	0.127447890570919
1 5 16	6042	0.570724830163117	2 9 15	1037	0.201772648116536
1 5 10	6035	0.302775637731993	6 12 13	1033	0.302775637731994
1 9 14	6030	0.570724830163117	2 3 16	1033	0.18589247985766
1 6 9	6014	0.302775637731995	3 6 13	1031	0.302775637731995
5 9 15	5995	0.570724830163117	12 14 16	1030	0.228462344680052
1 5 12	5989	0.302775637731993	8 10 16	1030	0.173905306725911

TABLE 5.5: 10 most removed sets for DBGL (left) and CDBGL (right)

The case $k = 3$ is very interesting in terms of comparing the different Laplacians, because the samples for the variants significantly differ from each other. The DBGL has the same most likely set as the CSGL, however, in the DBGL it is chosen far more likely than in the CSGL. The other most chosen sets can be grouped together again, however in contrast to the CSGL, they are not the highest cumulative degree sets. These sets consist of two peripheral centers and either one peripheral node from an opposite arm or the opposite bridging node. All these sets are chosen with approximately the same frequency, but a lot less likely than the most likely set.

In the sample of the CDBGL, the most likely sets all have approximately the same frequencies, and from the plot we can conclude that there are generally many sets with approximately the same frequency. The sets can be grouped into three groups, with each group having approximately the same frequency. Looking at the plot, the high eigendrop sets seem to be less likely than other sets. This looks very different from the SGL, but generally, both these variants seem unsuitable for eigendrop maximization as for now.





Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 5 9	1468	0.570724830163117	3 6 9	1009	0.201772648116537
1 5 13	1365	0.302775637731995	1 10 12	1000	0.127447890570919
5 9 13	1328	0.302775637731995	4 8 15	999	0.17390530672591
1 9 14	1314	0.570724830163117	1 11 13	991	0.302775637731994
1 9 13	1268	0.302775637731995	3 7 16	986	0.302775637731994
5 9 15	1249	0.570724830163117	4 10 14	981	0.302775637731994
4 5 9	1237	0.302775637731994	10 12 13	979	0.302775637731995
1 5 16	1218	0.570724830163117	4 6 13	979	0.302775637731995
1 6 9	1214	0.302775637731995	3 11 13	978	0.302775637731994
1 5 15	1209	0.201772648116536	3 7 14	978	0.173905306725911

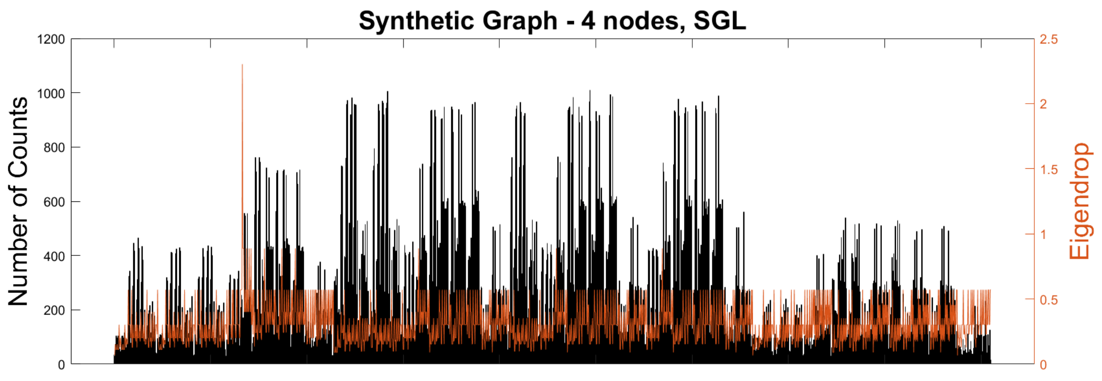
TABLE 5.6: 10 most removed sets for EBGL (left) and CEBGL (right)

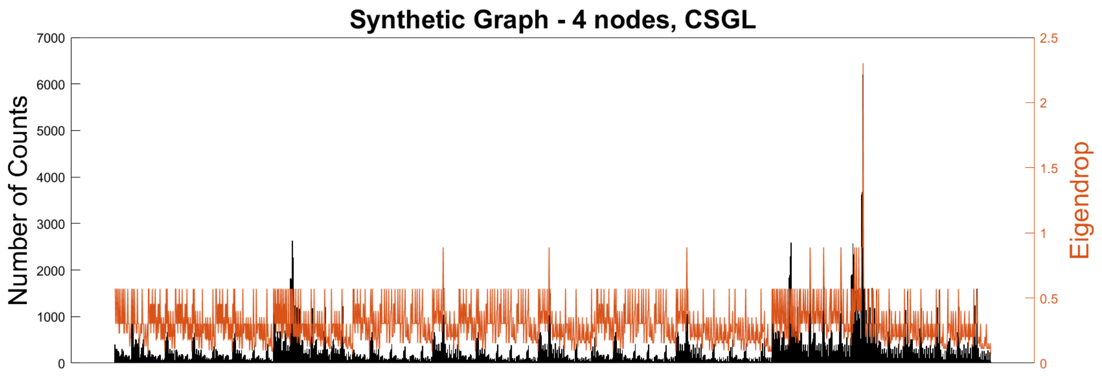
The EBGL has the same highest frequency set as the other two promising variants, but in the case of the EBGL this set is only slightly preferred. The most chosen sets are somewhat similar to the CSGL, but again, their relative frequency

is almost similar to insignificant sets. We see the same results as in the case $k = 1$, where the most chosen sets are 'good', but there is just a lot of noise that makes this variant less preferable than the others.
The CEBGL exhibits an almost uniform frequency distribution with some noise. It does not give a lot of insight about the maximal eigendrop set.

5.1.3 Removing 4 nodes

In the case of $k = 4$ there is one best set, which achieves an eigendrop of **2.3028**, which is also the maximal eigendrop that is possible. The best set is $\{1, 5, 9, 13\}$ and consists of the four peripheral centers and the central node. There are 1820 possible sets in total.

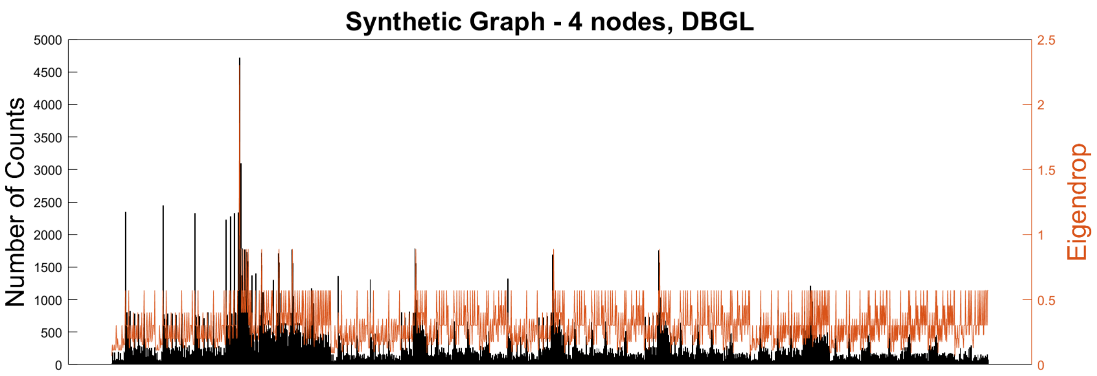


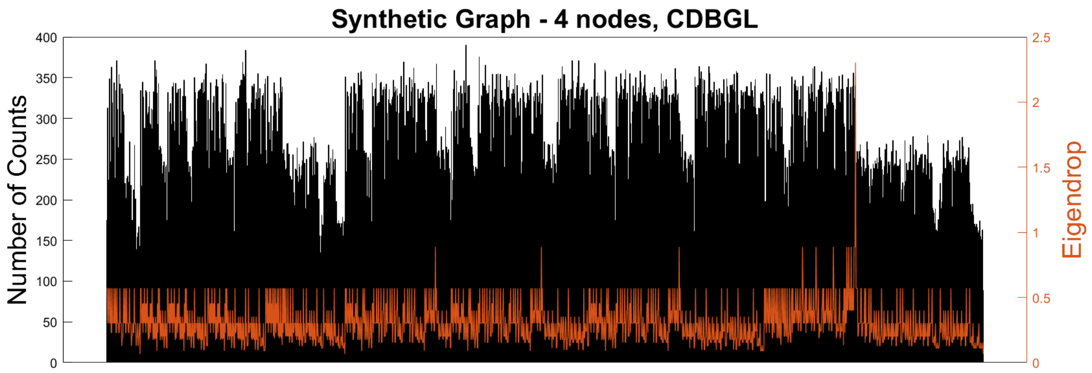


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
3 7 8 11	1010	0.153814495982358	1 5 9 13	6196	2.30277563773199
2 4 8 12	1006	0.153814495982359	1 5 9 14	3679	0.888562075358899
3 8 10 12	993	0.15381449598236	1 5 9 15	3625	0.888562075358899
4 8 11 12	988	0.15381449598236	1 5 9 16	3614	0.888562075358899
3 8 11 12	986	0.15381449598236	5 9 13 15	2622	0.570724830163117
3 6 8 12	984	0.153814495982358	1 9 13 14	2580	0.570724830163117
2 3 7 12	981	0.153814495982359	1 5 13 16	2578	0.570724830163117
4 6 8 11	977	0.153814495982358	1 5 13 15	2333	0.302775637731995
2 3 6 12	972	0.153814495982359	1 9 13 16	2292	0.302775637731995
2 4 7 10	970	0.153814495982359	1 5 13 14	2271	0.302775637731995

TABLE 5.7: 10 most removed sets for SGL (left) and CSGL (right)

The SGL seems to prefer combinations of peripheral nodes with low eigendrop and the plot shows that the highest eigendrop set is a quite unlikely set compared to the others. These results confirm again, that the SGL should not be used to maximize the eigendrop.
For the CSGL the results are the opposite. The best set is also the most chosen set with almost twice the frequency compared to the next most chosen set. The next three most chosen sets can again be grouped together and are the sets with the next highest cumulative degree. This pattern continues for the remaining most chosen sets.

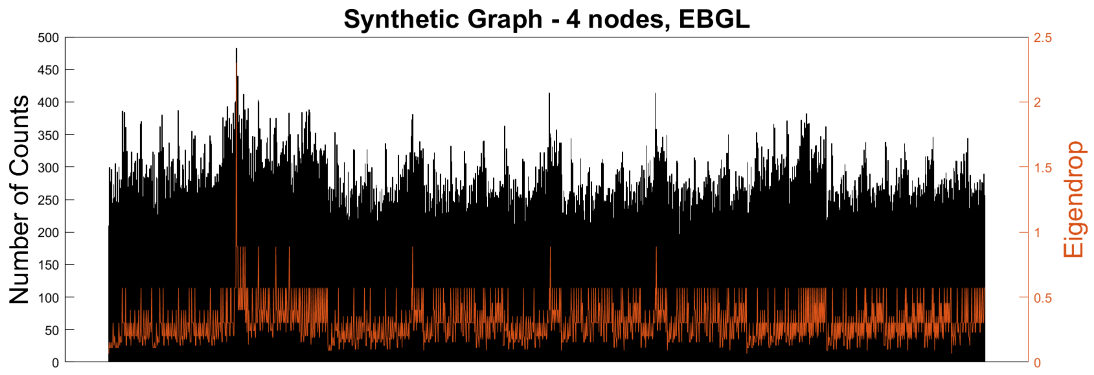


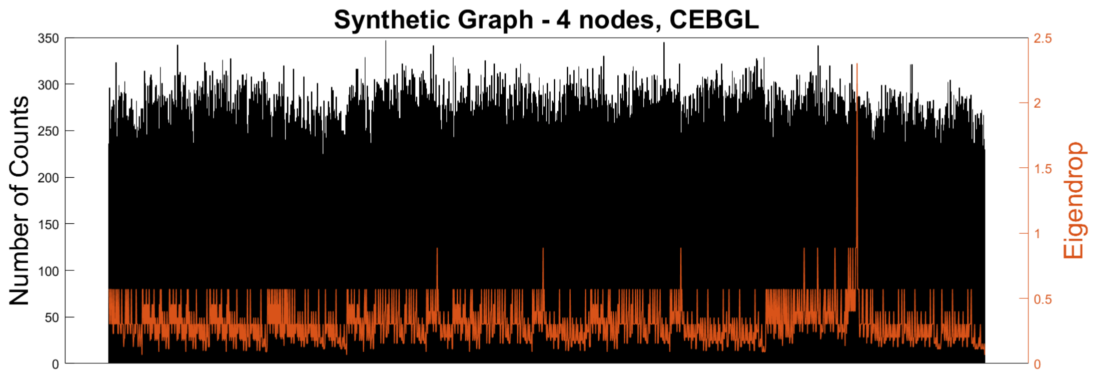


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 5 9 13	4716	2.30277563773199	3 10 11 16	390	0.173905306725909
1 5 9 16	3097	0.888562075358899	6 8 9 15	384	0.570724830163117
1 5 9 15	3068	0.888562075358899	3 8 14 15	376	0.228462344680051
1 5 9 14	2975	0.888562075358899	10 12 14 16	371	0.228462344680053
1 3 5 9	2441	0.570724830163117	8 10 11 16	371	0.17390530672591
1 2 5 9	2344	0.570724830163117	3 4 8 13	371	0.302775637731995
1 5 9 10	2332	0.570724830163117	3 4 7 8	371	0.0980521056056411
1 4 5 9	2327	0.570724830163117	6 8 10 15	369	0.352919813368347
1 5 8 9	2321	0.570724830163117	4 6 14 16	369	0.45501657270942
1 5 7 9	2276	0.570724830163117	2 11 14 15	368	0.45501657270942

TABLE 5.8: 10 most removed sets for DBGL (left) and CDBGL (right)

For the DBGL the results are similar to the CSGL. The most chosen set is the best possible set and the next three sets have the highest cumulative degree. In comparison to the CSGL however, the best set is chosen relatively less frequently. The other difference is in the sets that come after that. The DBGL picked sets that consist of the peripheral centers and one peripheral node in contrast to the highest degree sets that were picked by the CSGL.
The plot of the CDBGL looks a bit similar to the respective $k = 3$ plot, with a majority of sets being chosen with roughly the same frequency and some sets having lower frequencies. The preferred sets have low eigendrops and the highest eigendrop set has way lower frequency than the most likely sets.





Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 5 9 13	483	2.30277563773199	4 8 9 13	347	0.570724830163117
1 5 9 16	440	0.888562075358899	2 6 8 12	345	0.153814495982358
1 5 9 14	426	0.888562075358899	7 11 12 15	342	0.352919813368348
1 5 9 15	424	0.888562075358899	4 5 11 12	341	0.241955508822762
3 5 9 13	414	0.570724830163117	1 7 9 13	341	0.570724830163117
4 5 9 13	414	0.570724830163117	4 5 12 13	332	0.570724830163117
1 5 12 13	412	0.570724830163117	2 10 12 15	330	0.185892479857659
1 6 9 13	403	0.570724830163117	4 9 14 15	329	0.570724830163117
1 5 9 12	401	0.570724830163117	3 14 15 16	329	0.570724830163117
1 6 9 14	400	0.888562075358899	2 3 4 8	329	0.0897206853195351

TABLE 5.9: 10 most removed sets for EBGL (left) and CEBGL (right)

As we seen in the previous samples of the EBGL, the most chosen sets correspond to the most chosen sets of the CSGL to a great part, but with way lower relative frequencies. The distinction between more and less significant sets is barely present, even with a very large sample. This variant does not converge like the others do and it is only slightly more likely to find a good set rather than a bad one.
The CEBGL is not significantly differentiating between sets and looks random. It is not clear on what grounds the sets are chosen and the frequencies do not seem to have any correlation to the eigendrop.

5.1.4 Remarks and assumptions

The first round of experiments validated properties of the forest method, but also provided ground for new assumptions. We saw that the chosen nodes repel each other and some variants of sampling show interesting results.
The most promising methods seem to be the CSGL and the DBGL, since they are differentiating between the sets and manage to find high eigendrop sets more often. The CSGL seems to be strongly connected to the cumulative degree of node sets. In the performed experiments, the most chosen sets were always ordered according to the degree of the sets, even for $k > 1$.
The DBGL had similar results as the CSGL for $k = 1$, which leads to the assumption that this could always be the case $k = 1$. For $k > 1$ however, these variants differed and the DBGL seemed to not be connected to the degree as much. The most chosen set was still the highest degree set, but the other sets did not always follow that pattern, like e.g. in the case $k = 3$. This means we need to further understand the DBGL and raises the question, how does it function?
The EBGL definitely performed differently that expected prior to the experiments. It is good and bad in the context of eigendrop maximization at the same time. In a large sample, it would most likely lead to the same decisions as the CSGL and the DBGL, but its relatively less likely to pick a good set than in the other variants. The EBGL is interesting, because there is something right about its mechanisms, but also the final results are not quiet good.
The SGL, the CDBGL and the CEBGL showed less relevant results. All these variants do not seem suitable for eigendrop maximization. This also makes sense to some degree, since their respective complements perform well.

Even though some of the described variants are significantly worse at identifying relevant sets in a network, we decided not to discard any of them for the next round of experiments. The main reason being, that the Synthetic Graph is very symmetrical and does not resemble a real network. These properties could have been tied to the topology of the Synthetic Graph, and might not be valid for less ordered networks. Yet, there are hypotheses about the performance of the different options that we will test by performing similar experiments on a different graph. This will show if we correctly understand the underlying mechanisms and also provide a base to decide, which option should be used for further experiments.

5.2 The Karate Club Network

After the experiments on the Synthetic Graph gave us an insight in how the forest method performs, we conducted the same experiments on an irregular, more complex graph. We chose the well known network of Zachary's Karate Club, a small social network which models the relationships of karate club members. Two nodes are connected, if the respective club members interacted with each other outside the karate club. Interestingly, the network consists of two groups surrounding the two leaders, node 1 and node 34, that represent the instructor and the president of the club, who are in an argument and thus not directly linked. However, there are also nodes that are part of both groups and bridge the gap between the two leaders and make this network so interesting to analyze. The unweighted graph consists of 34 nodes and 76 undirected edges and its maximal eigenvalue is 6.6728. We will refer to this network as "*Karate Club*".

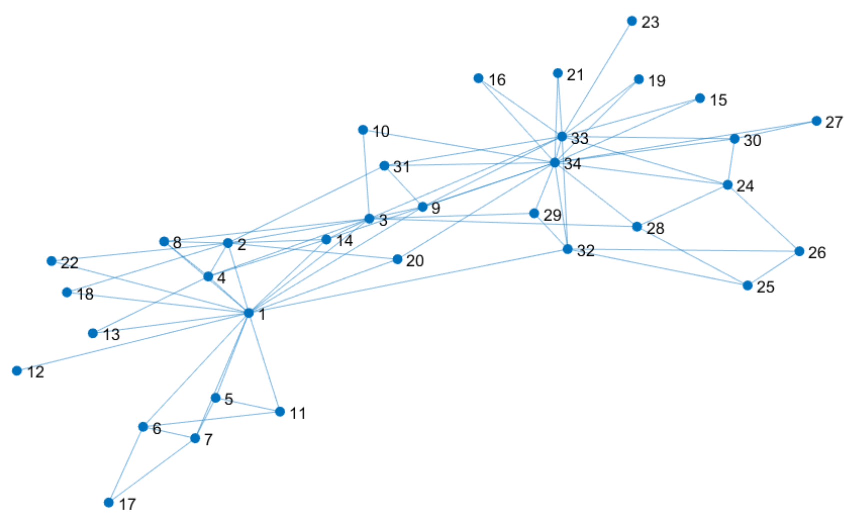


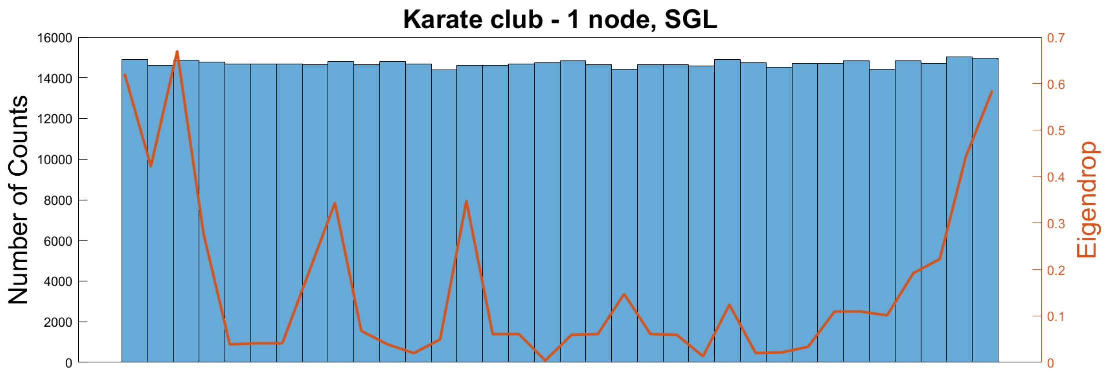
FIGURE 5.2: The Karate Club Network

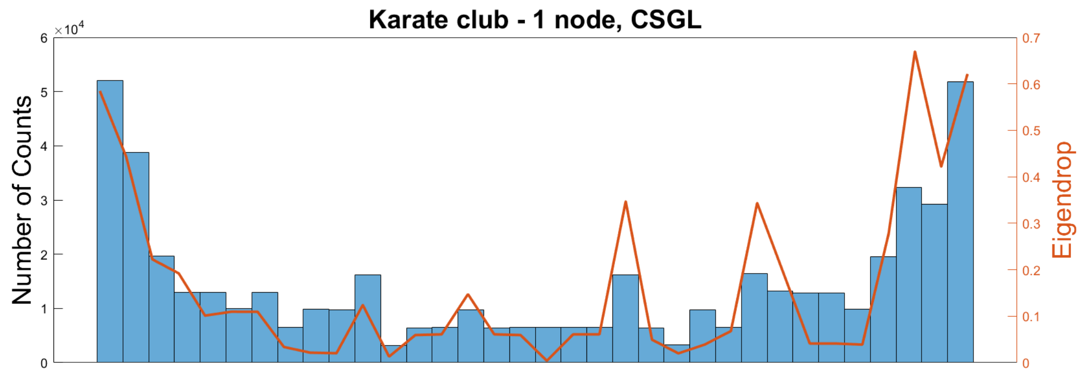
Just like for the Synthetic Graph, we performed 500.000 experiments for all variants of graph Laplacian. Also, similarly to the Synthetic Graph, the sets in the

plots are sorted in increasing order for the regular options and in decreasing order for the reverse options. The bars represent the frequencies of the sets and the orange line is the respective eigendrop.
Further, we only include the experiments for $k \leq 3$ in this thesis, since the other experiments did not offer additional insight, while taking a lot of time to compute the eigendrops of all possible sets.

5.2.1 Removing 1 node

In the case of $k = 1$ the best node to be removed is node 3, with an eigendrop of **0.6696**. Interestingly, this is not the highest degree node and also neither the president nor the instructor, but someone, who is on a path between these two nodes, thus not the trivial choice.
In total, there are 34 different sets that the algorithm can sample. Given the insights from theorem 4.2.2 and the previous experiments, we expect the CSGL and The DBGL to sample the sets according to their relative degree, and thus not to detect the highest eigendrop set as the most likely set.

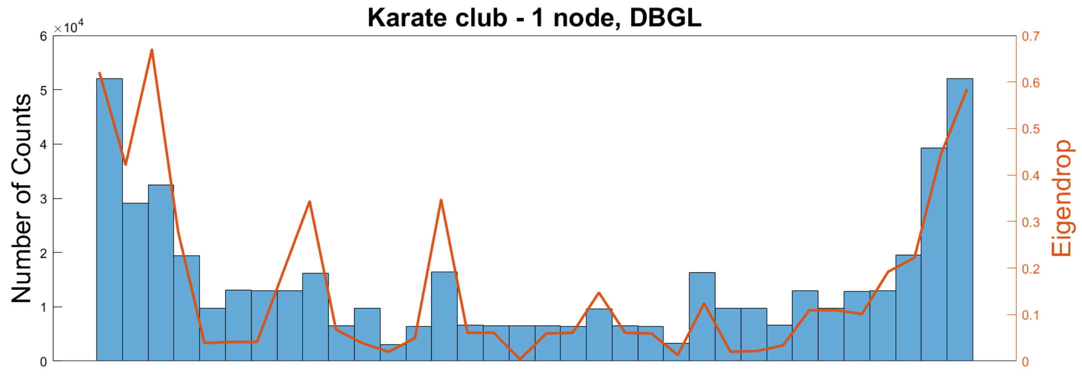


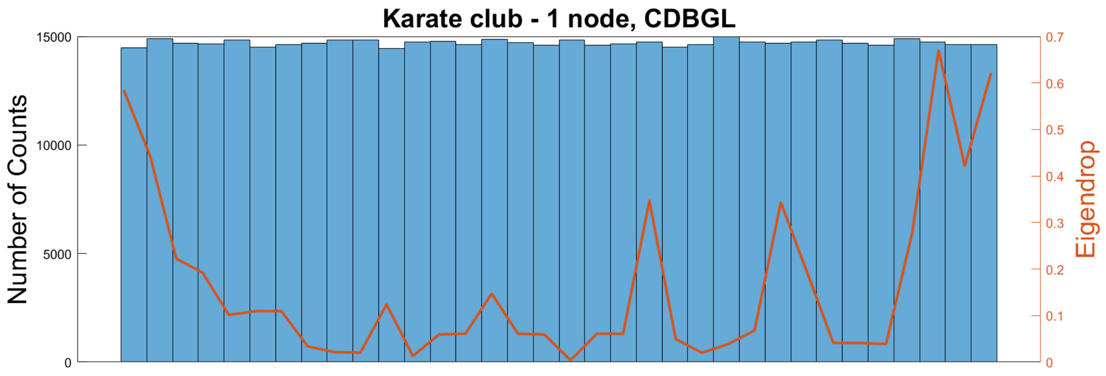


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
33	15021	0.443485957637915	34	52056	0.584749560379755
34	14979	0.584749560379755	1	51795	0.621300659442355
1	14901	0.621300659442355	33	38767	0.443485957637915
24	14886	0.124005157709807	3	32353	0.669618922959452
3	14859	0.669618922959452	2	29174	0.421758110889268
29	14843	0.109579107317447	32	19634	0.222238619473861
31	14839	0.19216420482459	4	19490	0.27765305653167
18	14826	0.059401299239191	9	16397	0.343431646284466
11	14819	0.0388164045542814	14	16211	0.347022628052687
9	14812	0.343431646284466	24	16161	0.124005157709807

TABLE 5.10: 10 most removed sets for SGL (left) and CSGL (right)

For the SGL all sets are chosen with approximately the same frequency and the algorithm does not distinctively discriminate between the sets. The three highest degree sets are also the three most chosen sets, but this could also be a coincidence, since they only lead by a few samples.
For the CSGL we find exactly what was expected, namely that the frequencies are distributed as the relative degrees. Thus, the highest eigendrop set is only the fourth most chosen set.
These results confirm what was expected of this experiment and thus show consistency in the forest method.

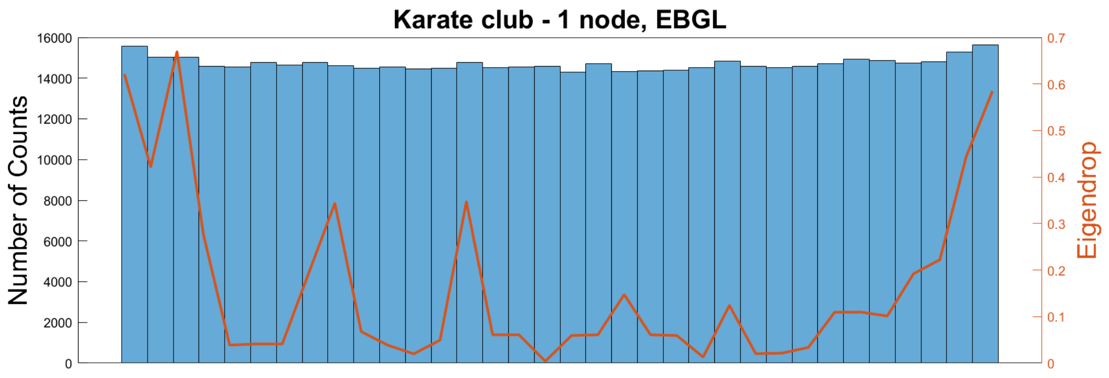


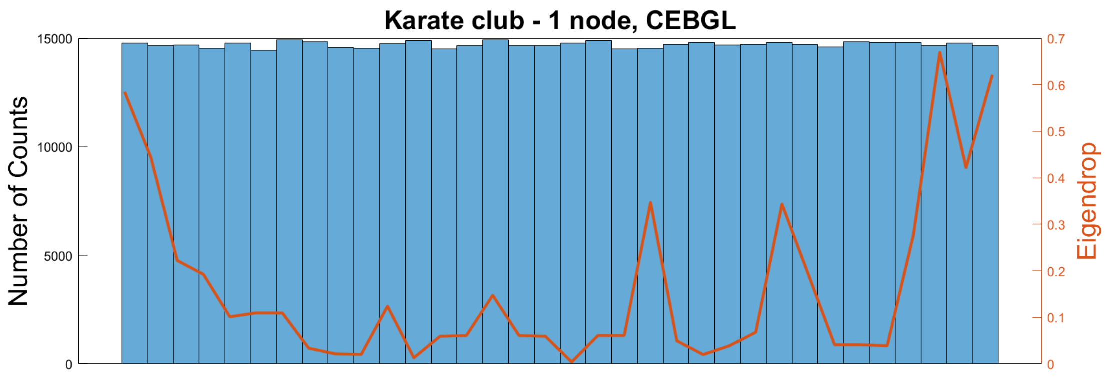


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
34	52064	0.584749560379755	11	14990	0.0388164045542814
1	52008	0.621300659442355	33	14896	0.443485957637915
33	39242	0.443485957637915	4	14893	0.27765305653167
3	32473	0.669618922959452	20	14864	0.14734832918326
2	29121	0.421758110889268	7	14848	0.041148972815833
32	19568	0.222238619473861	17	14839	0.00393807610205332
4	19470	0.27765305653167	25	14838	0.0202755961136312
14	16402	0.347022628052687	30	14833	0.101189161472203
24	16279	0.124005157709807	26	14823	0.0215171587130829
9	16142	0.343431646284466	22	14783	0.0594012992391892

TABLE 5.11: 10 most removed sets for DBGL (left) and CDBGL (right)

As expected, the plot of the DBGL looks just like the mirrored version of the plot of the CSGL. The frequencies are approximately the same as in the CSGL and overall it seems like the DBGL always produces a degree biased sample in the case of $k = 1$. Also, as expected, the CDBGL samples all sets almost uniformly. Thus, both these experiments confirm the assumptions made about the forest measure after the experiments on the Synthetic Graph.





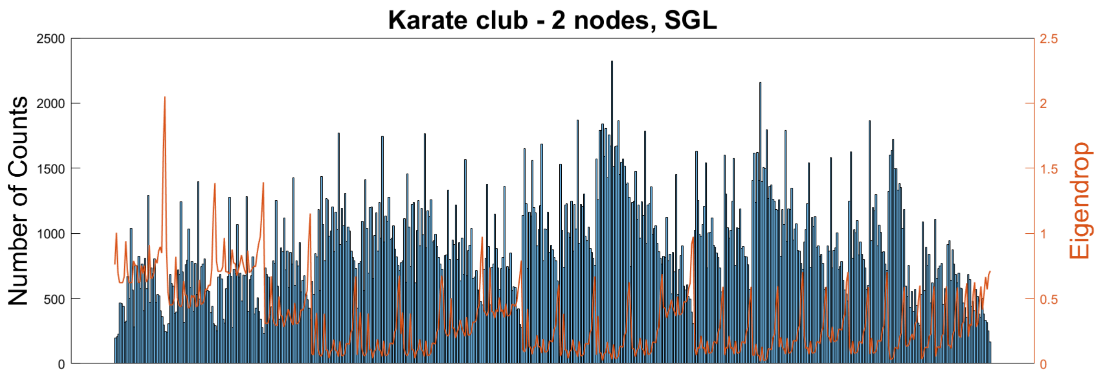
Sets	Counts	Eigendrop	Set	Counts	Eigendrop
34	15624	0.584749560379755	20	14929	0.14734832918326
1	15574	0.621300659442355	28	14922	0.109431800217018
33	15277	0.443485957637915	23	14903	0.0129722953670823
2	15027	0.421758110889268	16	14891	0.0608931474919814
3	15019	0.669618922959452	27	14830	0.0337205548202215
29	14935	0.109579107317447	6	14826	0.0411489728158259
30	14856	0.101189161472203	5	14803	0.0388164045542805
24	14832	0.124005157709807	4	14801	0.27765305653167
32	14804	0.222238619473861	9	14792	0.343431646284466
8	14780	0.192506101537525	12	14791	0.019931184055511

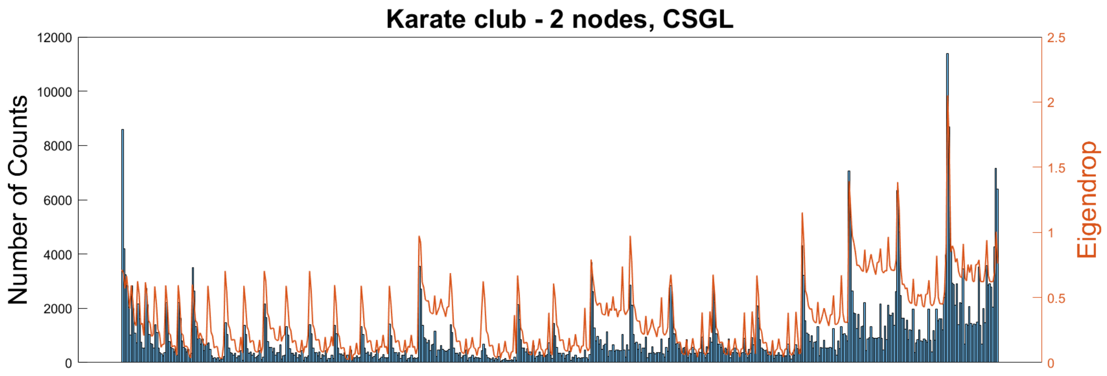
TABLE 5.12: 10 most removed sets for EBGL (left) and CEBGL (right)

The EBGL seems to sample the highest degree sets more often, but only barely, with the 5th highest degree set being sampled more often then the 4th highest degree set. We find the same diluted results as in the experiments of the Synthetic Graph.
The CEBGL is similar to a uniform sample and does not give any insight. Overall, we can clearly see that the forest method is consistent in these variants as well.

5.2.2 Removing 2 nodes

In the case of $k = 2$, out of the 561 possible different sets, there is a single set that achieves the highest eigendrop. That set consists of the instructor and the president, so the two group leaders, which are also the two nodes with the same higehst degree in this network. The eigendrop of the set $\{1, 34\}$ is **2.0508**.

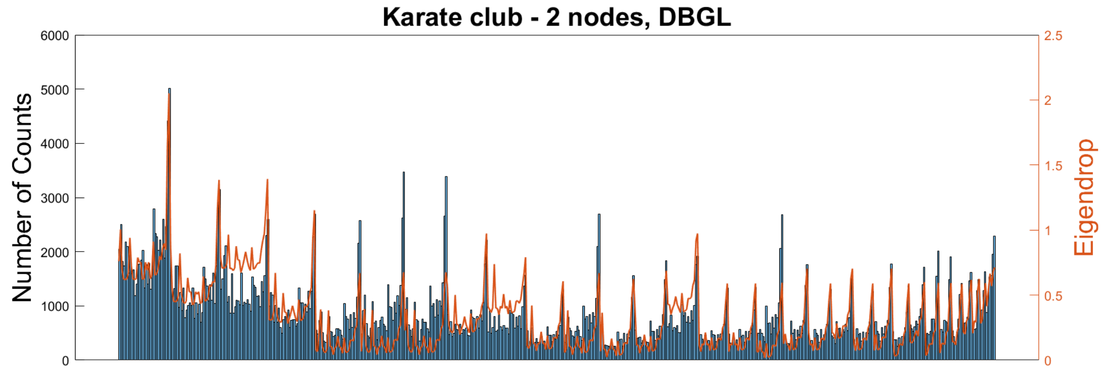


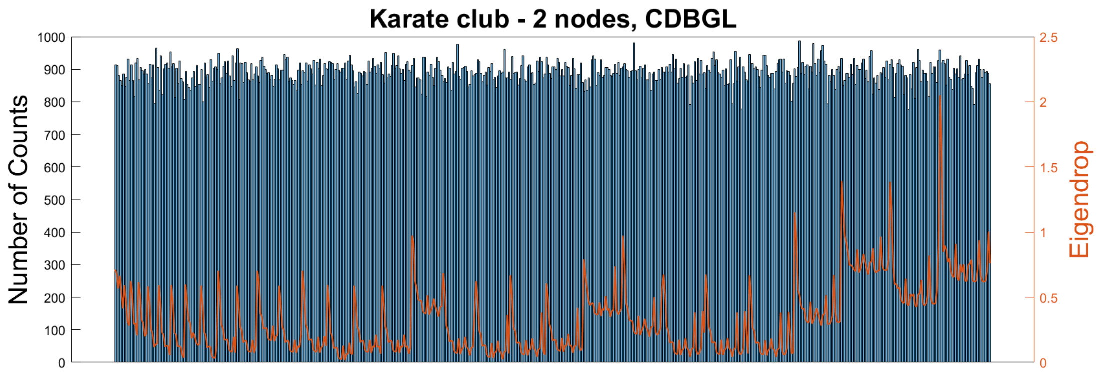


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
12 23	2324	0.0333134365166687	1 34	11394	2.05076034312303
17 23	2160	0.0170140888800354	1 33	8678	1.63653590180451
11 23	1872	0.052662176460994	33 34	8583	0.712440207962737
12 27	1864	0.0547736061527786	1 3	7166	1.00471744952109
22 23	1864	0.0736230900934824	3 34	7075	1.39224673128535
12 17	1838	0.023731632767376	1 2	6396	0.760428343082276
12 19	1805	0.0827793991365873	2 34	6340	1.3853998554762
17 27	1795	0.0379415814875914	3 33	5375	1.16425467061656
12 15	1789	0.0827793991365908	2 33	4811	1.16405551000038
12 16	1789	0.0827793991365908	4 34	4320	1.15242638323867

TABLE 5.13: 10 most removed sets for SGL (left) and CSGL (right)

The SGL produces a random looking plot that offers no insight into important nodes. The plot of the SGL makes it evident, that this variant should not be used for node immunization. The eigendrop line and the height of the bars seem negatively correlated and it is clearly visible that the set with the highest eigendrop has one of the lowest frequencies.
On the other hand, the plot of the CSGL looks very beautiful. It almost looks like the bars follow the eigendrop plot and the frequencies and the eigendrop plot are positively correlated. The by far most chosen set is also the highest eigendrop set. The preferred sets are the sets with the highest cumulative degree, with equal cumulative degree sets having similar frequencies.
These results are consistent with the other experiments and further confirm our understanding of the forest method.

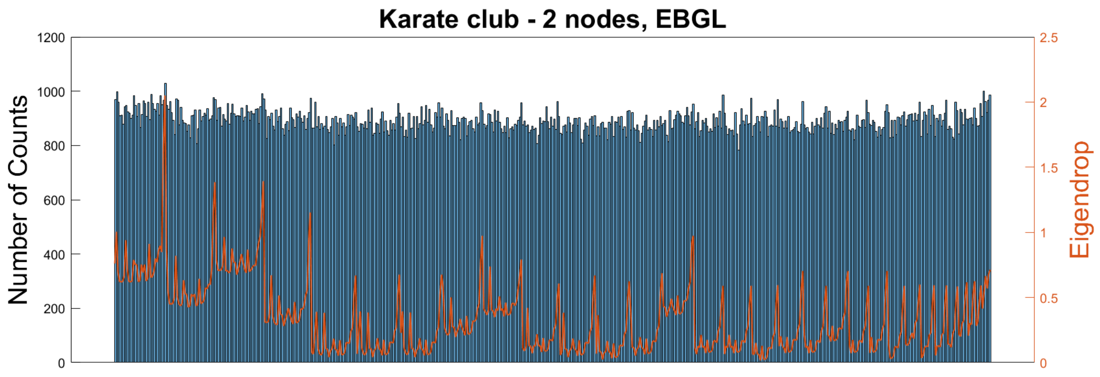


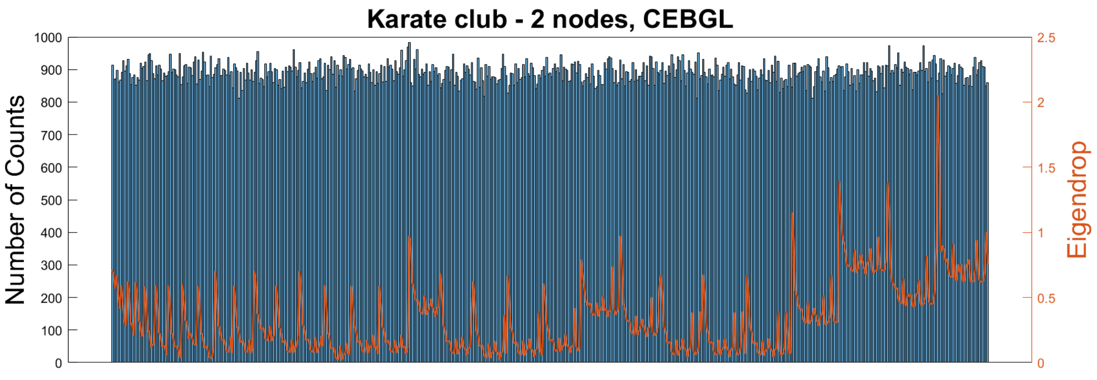


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 34	5009	2.05076034312303	4 31	988	0.514665898734785
1 33	4420	1.63653590180451	8 27	981	0.23771945280041
6 34	3473	0.675383358988293	4 22	979	0.321286046314601
7 34	3385	0.675383358988293	13 25	977	0.0709752312442342
2 34	3145	1.3853998554762	4 16	973	0.367489411524112
1 24	2796	0.911448227910191	27 29	965	0.141487386221694
2 33	2775	1.16405551000038	21 34	963	0.589398807273771
11 34	2699	0.669076654401801	2 16	962	0.525091827883919
4 34	2696	1.15242638323867	12 16	961	0.0827793991365908
17 34	2686	0.59497742963834	1 34	959	2.05076034312303

TABLE 5.14: 10 most removed sets for DBGL (left) and CDBGL (right)

The plot of the DBGL has some similarities with the CSGL, but the lower eigendrop sets are chosen more often relative to the higher eigendrop sets. Both these options would lead to the same decisions if you perform many experiments, but it seems that with the CSGL it is more likely to find a high eigendrop set with few samples.
The CDBGL is close to uniform sampling and does not provide insight into highest eigendrop sets.





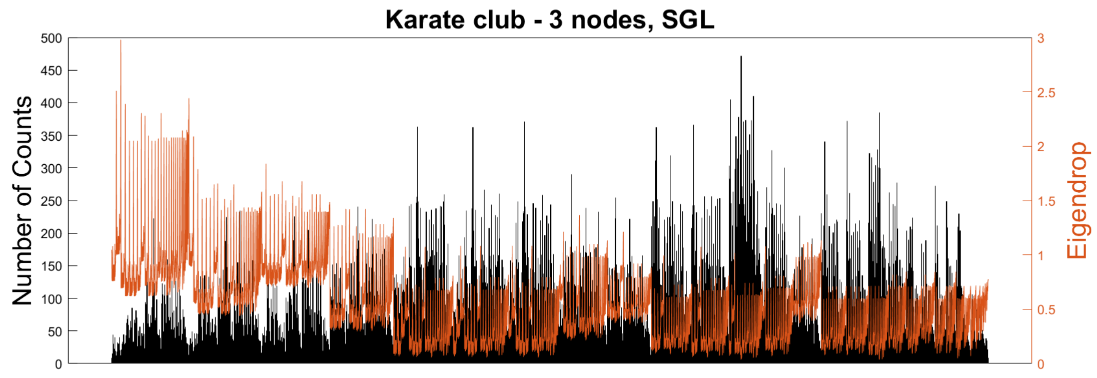
Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 34	1030	2.05076034312303	14 34	984	0.973568996160933
31 33	1001	0.539294559665838	2 33	974	1.16405551000038
1 3	997	1.00471744952109	2 11	973	0.449453738713016
3 33	991	1.16425467061656	15 16	970	0.115519187710932
1 25	988	0.653274385964236	19 23	961	0.0724116110575137
33 34	987	0.712440207962737	14 29	961	0.471448650554144
15 34	985	0.589398807273777	15 20	959	0.212584363892751
1 14	984	0.78891449194975	20 32	956	0.380261781801421
1 31	983	0.897062906126692	23 31	953	0.20327453262076
2 33	977	1.16405551000038	7 10	952	0.112310376884447

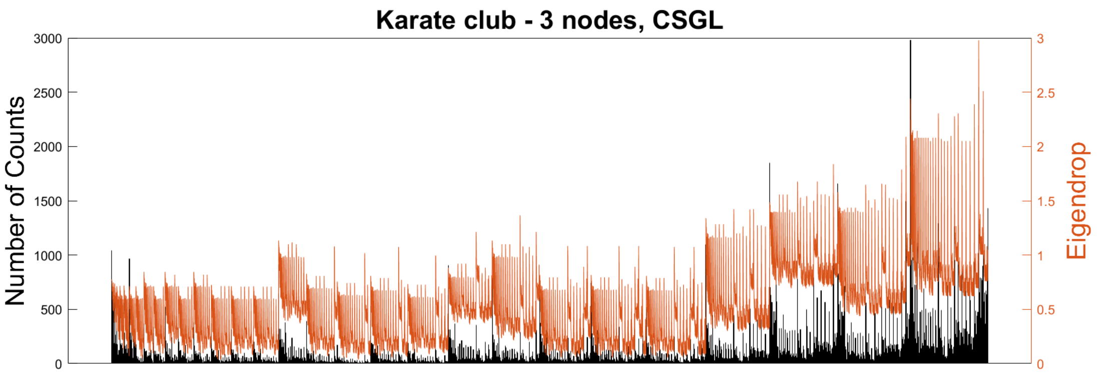
TABLE 5.15: 10 most removed sets for EBGL (left) and CEBGL (right)

For the EBGL and the CEBGL the plots look rather similar. The most likely set in the EBGL it indeed the highest eigendrop set, but it is only barely in the lead. Both these variants seem not to be suitable for eigendrop maximization, even though the EBGL seems to have a slight preference towards high degrees.

5.2.3 Removing 3 nodes

For the case of $k = 3$, the best set in terms of eigendrop is $\{1, 3, 34\}$ with an eigendrop of **2.9786**. This set is a combination of the best sets for $k = 1$ and $k = 2$. There are 5984 possible sets in total.

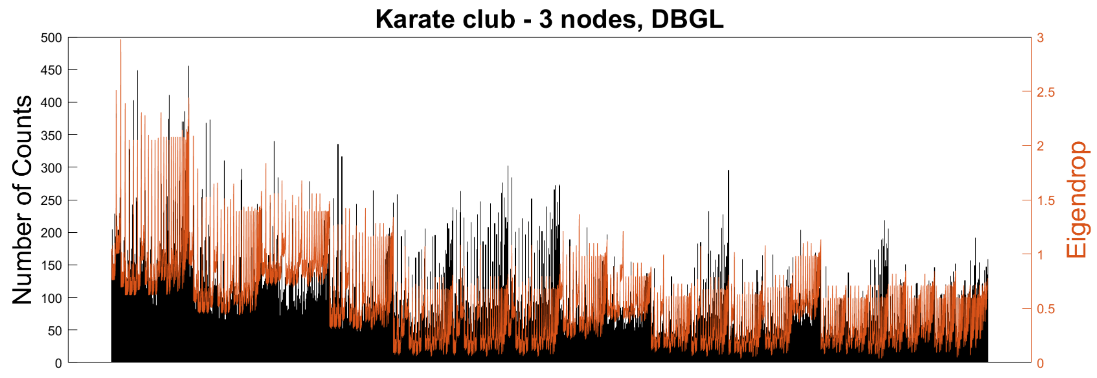


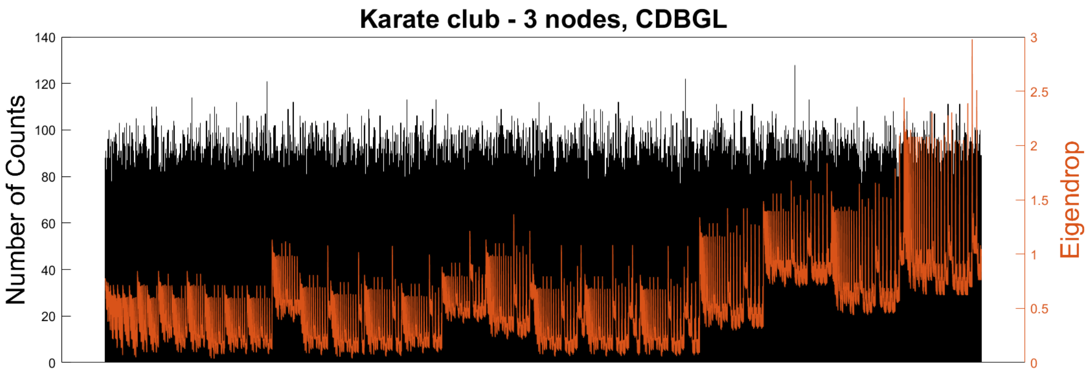


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
12 17 23	472	0.0372167687401967	1 33 34	2980	2.44155072494438
12 23 27	410	0.0678992090946693	1 3 34	2550	2.97857321473091
12 13 23	405	0.0823152156074736	1 2 34	2155	2.50988023709698
17 23 27	385	0.0507534476090914	3 33 34	1848	1.43272717034605
12 16 23	378	0.0946893356660992	1 3 33	1777	2.13350769688457
12 19 23	373	0.0946893356661	2 33 34	1661	1.57995893073488
12 22 23	373	0.0923617950365863	1 2 33	1629	1.9509368691861
16 17 23	372	0.0769549631177497	1 32 34	1545	2.12996656532036
7 12 23	371	0.0736785923288421	2 3 34	1500	2.08909843897758
12 18 23	371	0.0923617950365809	1 4 34	1448	2.38862356466713

TABLE 5.16: 10 most removed sets for the SGL (left) and CSGL (right)

Again, the results of the SGL do not offer a lot of insight and the most chosen sets do not achieve a high eigendrop. For the CSGL, the most chosen set is also the highest cumulative degree set. The best possible set $\{1, 3, 34\}$ is only the second choice of the algorithm and also the set with the second-highest cumulative degree. The list of most chosen sets continues in the same manner and genereally, the frequency of the chosen sets seems to depend on the cumulative degree for this variant.

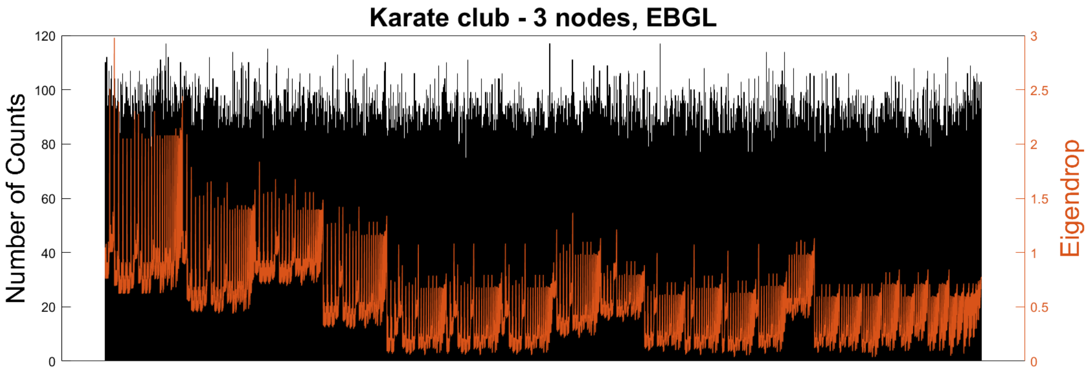


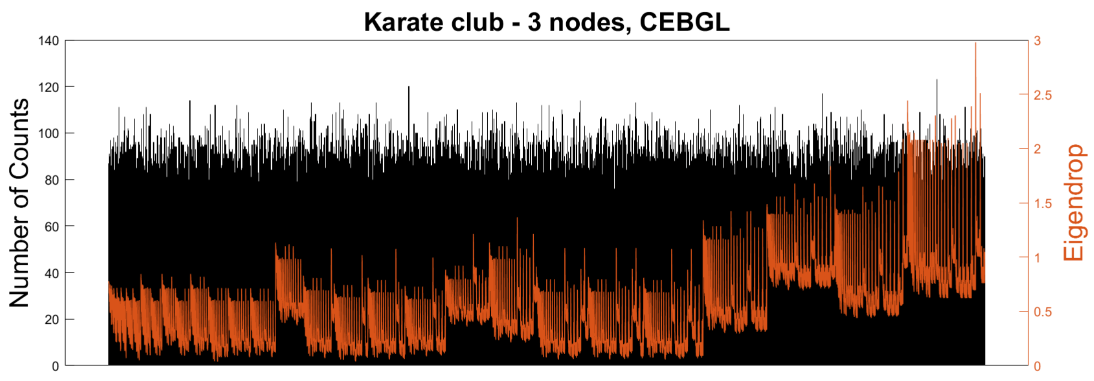


Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 33 34	456	2.44155072494438	3 13 30	128	0.900729823448794
1 7 34	449	2.05076034312302	5 9 27	122	0.426021482391695
1 17 34	411	2.05076034312303	15 17 34	121	0.599712509617064
1 6 33	403	1.63653590180452	18 26 32	114	0.296156363411571
1 6 34	397	2.05076034312302	10 26 27	113	0.12174688948514
1 26 34	386	2.0777060837942	10 12 30	113	0.191602648786304
1 2 34	383	2.50988023709698	3 9 23	113	0.98289531602127
1 3 34	380	2.97857321473091	16 21 31	112	0.291880875204018
1 7 33	375	1.63653590180452	14 17 26	112	0.381986801837182
1 17 33	374	1.63653590180451	7 25 31	112	0.266812989837361

TABLE 5.17: 10 most removed sets for DBGL (left) and CDBGL (right)

The DBGL also chose the highest degree set with the highest frequency. However, the other sets are not ordered as they were for the CSGL, and the best set is barely among the most chosen sets. The likelihood of a set sampled with the DBGL seems to be less dependent on the degree of the set than the CSGL, however this raises the question on what basis sets are chosen instead. It seems like the sampling is still connected to the degree to some extent, since all of the most chosen sets contain at least one, most of them even both, maximal degree nodes. Also, the DBGL is less distinct, meaning that low degree and eigendrop sets are more likely compared to the CSGL.
The CDBGL does not offer any insight in terms of maximal eigendrop.





Sets	Counts	Eigendrop	Set	Counts	Eigendrop
1 19 27	117	0.829510395116735	1 14 22	123	0.791713858390211
7 27 33	117	0.537384486567071	10 27 30	120	0.181285627937247
10 16 19	117	0.179999296429967	3 6 33	117	1.28140618758244
3 6 33	115	1.28140618758244	19 20 26	114	0.236064581467928
2 17 18	114	0.438521295143921	7 10 15	114	0.176446627611258
13 17 34	114	0.697785471872392	13 24 27	113	0.212971546520688
13 28 34	114	0.722395476445894	12 25 32	113	0.247857200181854
4 8 24	113	0.596618468323527	11 24 25	113	0.188038749481635
1 2 13	112	0.761342258313495	8 15 20	113	0.422593309371783
1 20 28	112	0.872894333637231	6 9 10	113	0.4647010008704

TABLE 5.18: 10 most removed sets for EBGL (left) and CEBGL (right)

As in the case $k = 2$, the plots for both the EBGL and the CEBGL look very similar. The most sampled sets seem random and do not achieve a high eigendrop in both cases. For eigendrop maximization these variants do not seem suitable.

5.3 Conclusion

Both experiment series presented in this chapter were meant to explore the different variants of sampling that were previously introduced. Ultimately, the experiments helped in understanding the forest method and showed that some sampling variants were more suitable for eigendrop maximization than others.
The experiments on the Synthetic Graph already showed that the SGL, the CDBGL and the CEBGL were completely invaluable in the context of the problem, and the Karate Club experiments only confirmed this even further.
The EBGL was very interesting to look into, since it was a completely different approach in terms of Laplacian and in theory should have been linked directly to the eigenvalues. However, the experiments showed, that despite finding some high eigendrop sets, the EBGL was inferior to other variants.
Only the CSGL and the DBGL created results that were relevant in the context of eigendrop maximization. From the experiments it is clear that these variants have different sampling mechanisms and can lead to different decisions in a small sample. With the exception of the Synthetic Graph $k = 3$ case, the CSGL always had a bigger difference in frequency for high and low degree sets and was thus

more distinct. That means that in most cases the likelihood to sample a set with high degree was higher for the CSGL than for the DBGL.
Another advantage of the CSGL is the computation time. Sampling many nodes is less computationally costly than sampling only few, because usually fewer experiments need to be performed to get the desired number of roots. For the CSGL $n - k$ roots need to be sampled, which should usually be a bigger number than k, and thus the sampling is faster than for the DBGL.
Finally, although both the CSGL and the DBGL showed promising results, we concluded to continue with experiments on bigger graphs only using the CSGL. The main reasons for that were the computation time, the more distinct results in comparison to the DBGL and the fact that we understood the mechanisms of the CSGL better than the ones of the DBGL. Even though the CSGL seems to pick node sets based on their cumulative degree only, this does not have to be a bad thing. It is still a random algorithm and explores the whole graph and eigendrop maximizing sets are very likely to be high degree sets, especially in real world networks with irregular structures.

Chapter 6

Experiments on real networks

In this chapter we will have a look at how the forest method performs on real networks as well as compare it to both Netshield algorithms. For this we used four networks that model the connections between airports and a social networks that model contact between attendees of a conference. Besides illustrating how the forest method is performing on real networks with Netshield as a sort of benchmark, we will also look into what happens when we introduce the concept of batches, as seen in Netshield+, to the forest method.

We will start with describing the airport networks and why they are relevant for this experiment. After that, a selection of the experiments performed on these networks will be described in detail, as well as conclusions drawn. After that, we will move on to the conference networks. First, the selected network will also be presented. After that, we will introduce the batch method and analyze how it performed in comparison to no batches and in comparison to the Netshields. Finally, we will sum up the results from the chapter and formulate a conclusion.

Throughout this chapter, when we talk about using the forest method, we will use the complement of the roots sampled using the simple graph Laplacian (CSGL). For the forest method variants, only the higehst eigendrop set will be reported here. Additionally, we will apply the rule of a good sample size, but also do an additional check by taking tenfold that number and compare the best result for each with the Netshields.
Comparing the results with Netshield is done for two reasons: The main reason is that we cannot compute the eigendrop for all possible sets to find the "true" best eigendrop set. The second reason is that comparing to Netshield is like comparing to a golden standard, since it is a fast and quite reliable algorithm. This way we can immediately see the deficiencies of the forest method in terms of time and performance.
We will report a measure of time, just to be able to have another parameter to compare the two approaches. However, these times should not be given too large of a weight, as they are based on a single run of the programme. They could probably be improved through better programming and could be made more reliable through averaging out multiple runs. We paid attention that the times we reported were not outliers, but finally they are merely a rough measures and are only meant for a relative comparison of the methods.
We expect the forest method to have a disadvantage in running time when comparing it to Netshield, as for every sampled set the eigenvalues of the resulting

network need to be calculated. This is the costliest part of the forest method and can only be minimized when reducing the sample size of the forest method.
Similar to the previous chapter, the analysis of the experiment was performed using MatLab.

6.1 Airport Networks

We performed experiments on four different real networks that we will call Airport 1 through Airport 4, sorted by their size from smallest to largest. We decided to perform experiments on networks modelling airport connections, as we expected them to have a structure similar to social networks. In addition to that, we could also observe how the forest method handles networks increasing in size with roughly similar structure.
In the following we will shortly describe each network and include a plot for better understanding. For the bigger networks we will not provide a plot, since there are too many nodes to get an insightful picture. We will also sum up the results of our experiments in a table and analyze the outcome. In the end of this section we will also formulate a first conclusion.
The airport data sets can be found in [1]

6.1.1 Airports 1

The Airports 1 network is a subset of the the Airports 2 network, containing only the 50 highest degree nodes. It is highly connected with 1756 out of 2450 possible edges and the maximal eigenvalue is 37.301. We included this network into the experiments to see how the forest method will perform on a small, densely connected real network.
We conducted experiments for the case of removing one, five and ten nodes, for which we used Netshield, Netshield+ with batch size $b = 1$, the forest method with $\lceil m/n \rceil = 36$ runs and $\lceil m/n \rceil \cdot 10 = 360$ runs respectively.

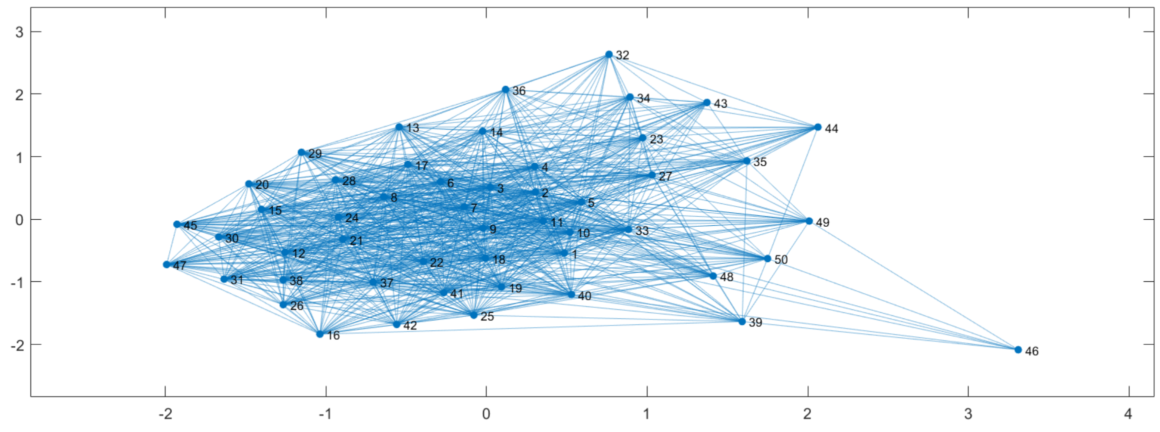


FIGURE 6.1: Airports 1 Network

In table 6.1 the results of the experiments are summed up. In the header row, the abbreviations "ED" stand for eigendrop and the abbreviations "T" stand for time in seconds.

TABLE 6.1: Airports 1 Experiments

	1 ED	1 T	5 ED	5 T	10 ED	10 T
Forest Method	1.2288	0.24	5.4533	0.24	10.8889	0.25
Forest Method x10	1.2288	0.29	5.8831	1.41	10.8109	1.7
Netshield	1.2288	0.13	6.0935	0.11	12.1241	0.13
Netshield+	1.2288	0.25	6.0935	0.31	12.1717	0.43

We can see the the forest method always takes approximate the same amount of time to find a good eigendrop, however finds an at least as good eigendrop in a faster time. The extended forest method with a bigger sample finds a better eigendrop in the case of 5 nodes, but also takes significantly longer doing so and still does not get as high as Netshield. Nevertheless, the sets that the forest method finds have a relatively high eigendrop.

6.1.2 Airports 2

The Airports 2 network consists of 500 nodes that are connected by 5960 edges. The maximal eigenvalue of this network is 48.074. This network was the smallest of the airport data sets. The structure of Airports 2 is comparable to one big clique of highly connected nodes and most other nodes have relatively low degree.
For the Airports 2 network we performed experiments for removing one node, five nodes and 10 percent of the networks nodes. Similar to the previous example we used Netshield, Netshield+ with batch size $b = 1$, the forest method with $\lceil m/n \rceil = 12$ runs and $\lceil m/n \rceil \cdot 10 = 120$ runs respectively.

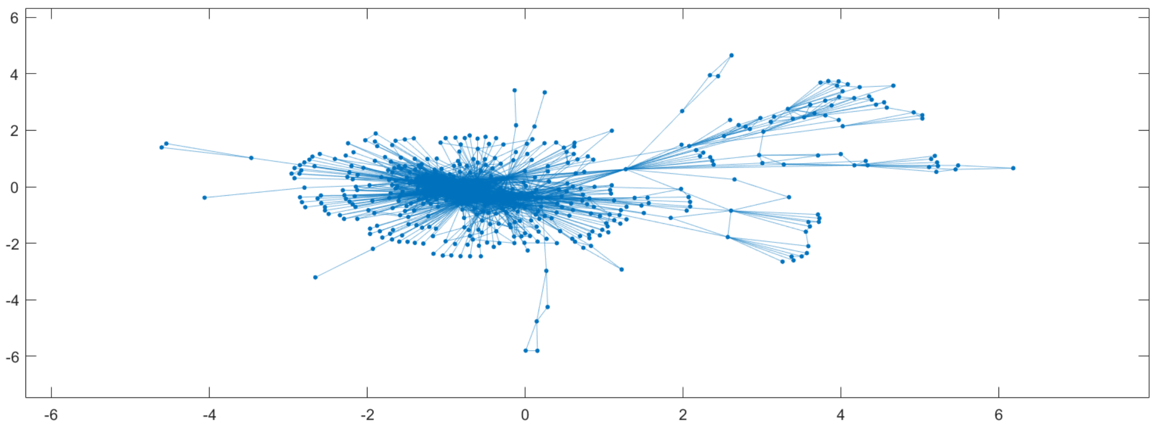


FIGURE 6.2: Airports 2 Network

TABLE 6.2: Airports 2 Experiments

	1 ED	1 T	5 ED	5 T	10 % ED	10 % T
Forest Method	1.3689	0.23	4.9291	0.21	26.1324	0.23
Forest Method x10	1.7706	0.62	6.4912	0.79	27.8537	0.82
Netshield	1.7706	0.11	8.2353	0.17	41.7404	0.27
Netshield+	1.7706	0.16	8.2353	0.19	43.3176	0.44

In table 6.2 we see that in general the forest method achieves worse eigendrops than Netshield and also takes longer to compute. Still, we see that on a much less dense graph, the forest method still finds relatively high eigendrop sets with very few samples required.

6.1.3 Airports 3

The Airports 3 network consists of 1858 nodes and 28236 edges. The highest eigenvalue of the adjacency matrix is 99.118.
For this network we performed exactly the same experiments as for Airports 2, the only difference being that the sample size for the forest method is $\lceil m/n \rceil = 16$.

TABLE 6.3: Airports 3 Experiments

	1 ED	1 T	5 ED	5 T	10 % ED	10 % T
Forest Method	1.0559	0.46	4.5699	0.29	64.2104	0.45
Forest Method x10	1.5837	2.32	5.7140	3.32	69.1802	3.05
Netshield	1.5837	0.25	7.7191	0.28	75.8286	0.31
Netshield+	1.5837	0.23	7.7191	0.32	92.7816	2.13

Similar to the previous two data sets, we can see in table 6.3 that the eigendrop that the forest method achieves stays below the one that can be achieved using Netshield. Using the forest method with tenfold sampling starts taking significantly longer, since we are dealing with quite a large graph, but we also see that Netshield+ gets very slow for the 10 percent removal, as eigenvectors need to be recalculated for each node removed. And we see that the gap between forest method and Netshield for the 10 percent of nodes remove is smaller than in the previous cases which might be related to the structure of the network. There are hub-nodes with high degree and this network has some resemblance with a social network.

6.1.4 Airports 4

The Airports 4 network is the biggest network in the set and consists of 7976 nodes and 30501 edges. It is also the sparsest network out of the set. The maximal eigenvalue here is 62.280. For this network we again performed the same experiments as for the last two. The sample size here was $\lceil m/n \rceil = 4$. Also, for the removal of 10 percent of the nodes we used Netshield+ with a batch size of one, but also took the batch size recommended in [8] of $b \approx k/10 = 80$.

TABLE 6.4: Airports 4 Experiments

	1 ED	1 T	5 ED	5 T	10 % ED	10 % T
Forest Method	0.1473	2.11	2.6209	1.84	51.1380	0.55
Forest Method x10	0.7448	5.48	2.0312	6.14	53.1875	4.04
Netshield	2.0793	0.57	8.2629	1.99	57.1967	0.60
Netshield+	2.0793	0.32	8.2944	0.38	61.27 / 58.05	38.68 / 0.73

In the above table, in the line of Netshield+, the two results in the 10 percent columns stand for the 2 results we got using Netshield+ with different batches. The first result is for batch size $b = 1$ and the second result is for $b = 80$.

In this experiment we see that for removing a small amount of nodes, Netshield performs significantly better. However, when we removed 10 percent of the nodes, the relative gab between the forest method and Netshield got smaller and the running times of both algorithms were approximately the same for the simple versions of the algorithms. This also shows how Netshield+ with batches of one can explode when it comes to running time, when dealing with a large network.

6.1.5 Conclusive remarks

After the first round of experiments, which we performed on airport data sets, we can already draw some empirical conclusions. We saw that with the simple setup of the forest method that we used here, the forest method will in general reach worse results than Netshield or Netshield+. Both in terms of running time and eigendrop, the forest method performed worse than the Netshield algorithms.
Nevertheless, these experiments still showed us that the forest method is effective. Even if it does not find the highest eigendrop sets, it still consistently finds sets that contribute to a good eigendrop. Given that the forest method is a random algorithm and given how many possible sets there are for e.g. removing 5 out of almost 8000 nodes, the algorithm definitely has a tendency towards better sets.
Another observation that we made, is that in sparser networks like Airports 2 to Airports 4, when we removed 10 percent of the nodes, the results of Nethsield and forest method did not differ too much. The bigger the graph became, the better the forest method performed in comparison to Netshield.

6.2 Conference Networks

The networks and their description can be found in [9] In light of the global pandemic that has changed many lives since the year 2019, experimenting with the forest method on a social network seem like a very natural choice. We picked a set of three networks called the conference networks, that recorded the interactions between conference attendee in the course of two and a half days. Every time the participants were facing each other for 20 seconds, a connection was recorded by devices worn by them. Out of this data, we constructed three weighted networks, each network representing a day of the conference. We performed similar experiments on the unweighted conference networks to the ones that we performed on the airport networks, with comparable results, thus we will not duplicate these efforts in this chapter.
In addition to the aforementioned experiments, we wanted to explore two other things, namely the forest method on a weighted graph and the forest method including batches. For this we restricted ourselves to the first of the conference networks, as they were quite similar in all aspects, with the only significant difference being the weight structure of the third network. But since the first two networks were more representative of meeting new people in a closed setting, we decided to pick the network Conference Day 1.

The Conference Day 1 network consists of exactly 100 nodes and 946 undirected edges. In the following, we will sum up the experiments that we did for the unweighted graph, including batches and later do the same for the weighted graph.

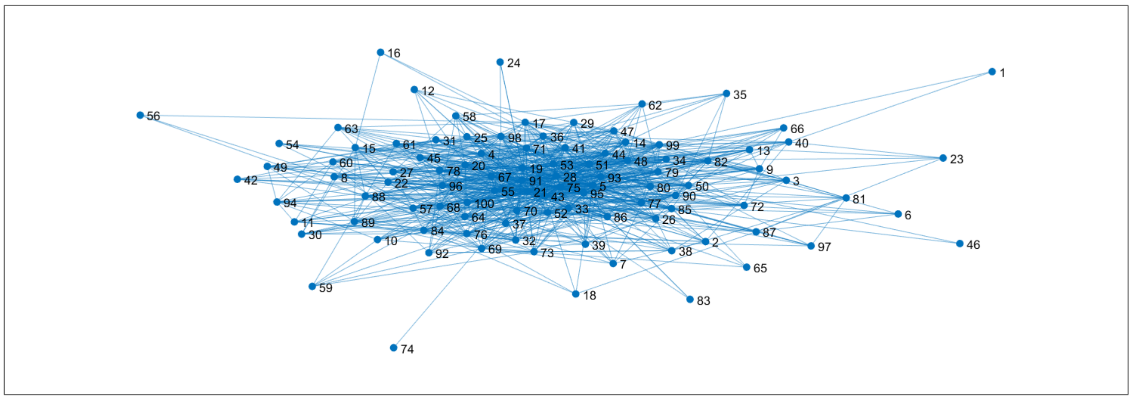


FIGURE 6.3: Conference Day 1 Network

6.2.1 Conference Day 1 unweighted

If we look at the Conference Day 1 network in an unweighted setting, its highest eigenvalue is 26.9197. We performed experiments to remove 5, 10 and 25 nodes, for which we used the forest method with a sample size of $\lceil m/n \rceil = 10$, as well as the tenfold sample size and a batch size of $b_{FM1} = 1$ for 5 and 10 nodes and a batch size of $b_{FM2} = 2$ and $b_{FM3} = 5$ for 25 nodes. As a benchmark, we again used Netshield and Netshield+ with batch size $b_{NS1} = 1$ for all, and additionally $b_{NS2} = 3$ for the case of removing 25 nodes. Where there are multiple eigendrops / times recorded in a cell, the smaller batch size is the first and the larger the second.

TABLE 6.5: Conference Day 1 unweighted Experiments

	5 ED	5 T	10 ED	10 T	25 ED	25 T
FM	3.3245	0.125	7.4922	0.119	12.9096	0.146
FM Batch	4.2801	0.163	7.1824	0.158	12.59 / 13.75	0.178 / 0.157
FM x10	5.5567	0.440	8.5811	0.489	14.3515	0.489
FMx10 Batch	5.0856	0.536	9.9643	0.650	15.24 / 15.35	0.575 / 0.617
NS	7.6531	0.163	12.7244	0.153	18.8100	0.080
NS+	7.6531	0.106	12.7244	0.128	19.02 / 19.02	0.157 / 0.164

In the experiments that we recorded, using batches lead to better results more often than not, however it was also a bit more costly that not using batches in terms or running time, which it to be expected. Even with batches, the eigendrop and running time were not reaching the levels of Netshield, but especially for the bigger sample, the gap became closer and batches seemed to offer a bigger improvement.

6.2.2 Conference Day 1 weighted

The weighted Conference Day 1 network was the first network that we included in the experiments that was not unweighted. In the weighted case, the highest eigenvalue of the network is 367.028.
We also have not used Netshield for a weighted network before, as the algorithm,

as per [8] is meant for unweighted graphs. However, weights don't seem to pose a restriction given the proofs, so it is valid to compare the two methods. We will also compare the below data to table 6.5, to see if the methods perform differently on weighted vs. unweighted networks.

TABLE 6.6: Conference Day 1 weighted Experiments

	5 ED	5 T	10 ED	10 T	25 ED	25 T
FM	138.69	0.162	154.73	0.223	231.98	0.203
FM Batch	112.48	0.173	158.06	0.170	264.42 / 239.99	0.167 / 0.144
FM x10	114.97	0.617	207.41	0.610	265.80	0.596
FMx10 Batch	157.32	0.659	212.25	0.670	283.89 / 278.85	0.718 / 0.657
NS	78.77	0.178	78.80	0.149	231.76	0.108
NS+	196.34	0.109	244.85	0.175	323.56 / 321.23	0.191 / 0.153

The first thing we notice is that the regular Netshield algorithm seems to perform worse than all other methods. Based on this single network, we cannot say if this is generally true, still this is one of our observations that should be further investigated in the future.
Netshield+ is still achieving better eigendrops than the different variants of the forest method, but the gaps are generally smaller for the weighted graph when comparing to the unweighted case. We can also see that except for the case of using the forest method to remove 5 nodes (the first result seems like an outlier), the eigendrop is improving with each additional layer of complexity, be it more sample and / or batch sampling, similar to the unweighted case.
When using batches we (almost) always see an improvement, with only a small increase in cost, as the costliest part of the computation is the evaluation. Thus, batches need to be investigated in more detail in the future, to see if this is generally true.

6.2.3 Conclusive remarks

We introduced two new factors into the experiments: weight structure and batches. We saw that relatively, the forest method performed better on the weighted graph, and also that the regular Netshield was performing worse on the weighted network than the unweighted in our experiment. These findings open up new tracks to research the forest method and require further investigation to see if this is true in general.
We also discovered, that batches offered an improvement in eigendrop, especially for the weighted graph, with only a slight increase in running time. If these results can be confirmed in more generality, sampling in batches can be used as a "cheap" way to improve the outcomes of the sampling.

6.3 Conclusion

After looking at the forest method from different angles and comparing it to Netshield and Netshield+, we can say that none of the variations and methods used by us could keep up with Netshield(+). Does it mean, we should discard this research and move on to something different? - No!
We saw many very promising qualities that the forest methods has to offer. Even

if the random set that the forest method picks is usually not the highest eigendrop set, it still shows a clear preference towards "good" sets and there might be opportunities for improvement in that regard. There are many ways to fine tune the forest method to the *k*-node immunization problem, some of which we merely touched upon in the second part of this chapter.
Another advantage of the forest method is that e.g. in comparison to Nethshield, the forest method explores the space of possible solutions. This could be used to a great advantage, as the forest method offers many good solutions in contrast to one (almost) perfect one. In cases, where immunizing a network would come at a cost per node, the forest method allows to compare different solutions and chose the one with the best trade-off, where Netshield is missing this flexibility.
But the research does not end here. After the finalization of the author's research project, Luca Avena and Alexandre Gaudillière continued the researching and improving the forest method. They could mathematically confirm e.g. that indeed the complementary root set of the simple graph Laplacian was the optimal option when it comes to sampling the highest eigendrop set, as well as other findings that we empirically concluded during the experiments. They also continued developing the batch sampling and the forest method in general, such that now the forest method can achieve at least the same eigendrops as Netshield+ with a comparable complexity. These findings will soon be published in a separate paper.

Bibliography

[1] *Airport Networks*. https://toreopsahl.com/datasets/#usairports.

[2] Kimia Ameri and Kathryn M. Cooper. "A Network-Based Compartmental Model For The Spread Of Whooping Cough In Nebraska." In: *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science* 2019 (2019), pp. 388–397.

[3] Luca Avena, Fabienne Castell, Alexandre Gaudillière, and Clothilde Mélot. "Random Forests and Networks Analysis". In: *Journal of Statistical Physics* 173.3-4 (2018), 985–1027.

[4] Luca Avena and Alexandre Gaudillière. "Two Applications of Random Spanning Forests". In: *Journal of Theoretical Probability* 31.4 (Dec. 2018), pp. 1975–2004. URL: https://hal.archives-ouvertes.fr/hal-02076291.

[5] Simon Barthelme, Nicolas Tremblay, Konstantin Usevich, and Pierre-Olivier Amblard. "Determinantal Point Processes in the Flat Limit: Extended L-ensembles, Partial-Projection DPPs and Universality Classes". In: (Nov. 2020). preprint. URL: https://hal.archives-ouvertes.fr/hal-03012027.

[6] Fred Brauer. "Mathematical epidemiology: Past, present, and future". In: *Infectious Disease Modelling* 2.2 (2017), pp. 113–127.

[7] Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. "Epidemic Thresholds in Real Networks". In: *ACM Trans. Inf. Syst. Secur.* 10.4 (Jan. 2008).

[8] Chen Chen, Hanghang Tong, B. Prakash, Charalampos Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. "Node Immunization on Large Graphs: Theory and Algorithms". In: *IEEE Transactions on Knowledge and Data Engineering* 28 (Jan. 2015), pp. 1–1.

[9] *Conference Networks*. http://www.sociopatterns.org/datasets/hypertext-2009-dynamic-contact-network/.

[10] Gustav R. Kirchhoff. "Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird". In: *Annalen der Physik* 148 (), pp. 497–508.

[11] Istvan Z Kiss, Joel C Miller, and Peter Simon. *(Book) Mathematics of epidemics on networks: from exact to approximate models*. Springer, 2017.

[12] Thomas M. Liggett. *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes*. Springer-Verlag, 1999.

[13] Piet Van Mieghem, Karel Devriendt, and Hale Cetinay. "Pseudoinverse of the Laplacian and best spreader node in a network." In: *Physical review. E* 96 3-1 (2017), pp. 1–22.

[14] P. Van Mieghem R. Pastor-Satorras C. Castellano and A. Vespignani. "Epidemic processes in complex networks". In: *Review of Modern Physics* Vol. 87 (2015), pp. 925–979.

[15] Leo Torres, Kevin S. Chan, Hanghang Tong, and Tina Eliassi-Rad. "Nonbacktracking Eigenvalues under Node Removal: X-Centrality and Targeted

Immunization". In: *SIAM Journal on Mathematics of Data Science* Vol. 3.2 (2021), pp. 656–675.

[16] David Bruce Wilson. "Generating Random Spanning Trees More Quickly than the Cover Time". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 1996, 296–303.