# Approximation of Spread Minimization in Bootstrap Percolation

Andrew Babb Stephanie Martinez Tim Wang

6 July 2022

## 1 Introduction

Let $G = (V, E)$ be an undirected graph, where each vertex may be either active or inactive in state. Choose a subset of vertices $A_0 \subseteq V$, called a *seed set*, to be activated initially. Then, fix an integer $k \geq 1$, called the *threshold*. We would like to explore the activation process on $G$ defined as follows: For the $i^{\text{th}}$ iteration of the process, the set of *activated vertices* $A_i$ is given by

$$A_i = A_{i-1} \cup \{v \in V : k \leq |N(v) \cap A_{i-1}|\}.$$

A vertex $v$ may be called *active* if $v \in A_i$; otherwise it is called *inactive*, and an inactive vertex becomes *activated* or *infected* if it has $k$ or more active adjacent vertices. This activation process is called *$k$-neighbor bootstrap percolation*. For any seed set $A_0$, the set $\langle A_0 \rangle = \bigcup_i A_i$ is the set of vertices that eventually become activated by the seed set $A_0$. If $\langle A_0 \rangle = V$, then $A_0$ is called a *contagious set*.

Bootstrap percolation was first studied on the Bethe lattice in [4] and has since been studied on other types of graphs such as grids [2], trees [1], hypercubes [3], and random graphs [6, 5, 10]. The types of random graphs include the Erdős–Rényi graph as in [6] and the stochastic block model as in [5, 10]. Bootstrap percolation has potential for simulating real-world systems such as social networks. For example, bootstrap percolation may provide a simple simulation for how disease spreads through a population where a person needs enough infected neighbors to become infected themselves. Additionally, bootstrap percolation may simulate the diffusion of trends or ideas throughout social networks. In a graph, users can be represented by vertices and friendships can be represented by edges. Bootstrap percolation has also been studied for a potential application in viral marketing [7, 9] where social groups must be chosen to maximize the influence of an advertisement.

### 1.1 Spread Minimization

The goal of spread minimization is to find a seed set $A_0 \subseteq V$ of size $n$ such that $|\langle A_0 \rangle|$ is minimized for a threshold $k$. Spread minimization is believed to be an NP-hard problem which implicates that there may not be a solution that is all efficient, optimal, and general. Hence, several approaches to spread minimization focus on particular cases or sub-optimal solutions. We explored possible approaches to spread minimization, taking inspiration from past works for the minimum contagious set problem, and observe how these approaches perform.

### 1.2 Minimum Contagious Set

A *contagious set* is a seed set $A_0$ that eventually infects an entire graph such that $|\langle A_0 \rangle| = |V|$. A contagious set $A_0$ is a *minimum contagious set* if there exists no other contagious set with size less than $|A_0|$. The size of the minimum contagious set for a graph $G$ and threshold $k$ is denoted by $m(G, k)$. The minimum contagious set problem is NP-hard as described in [9] and it is believed that spread minimization may also be NP-hard.

## 2 Investigations

To gain a better understanding of the percolation process, we investigated spread in several types of undirected graphs while varying parameters such as the threshold $k$, distribution and size of the seed set $A_0$, and size of the graph. For these graphs, we observed conditions in which $A_0$ is a contagious set or $|\langle A_0 \rangle|$ is maximized or minimized.

### 2.1 Complete Graphs

Given the size of the seed set, $|A_0|$, and the threshold, $k$, it can be determined if further activations can be made in a complete graph $K_n$. Furthermore, there are exactly two cases for $|\langle A_0 \rangle|$ in which $|\langle A_0 \rangle| = |V|$ or $|\langle A_0 \rangle| = |A_0|$ for graph $K_n$. Either the entire vertex set of $K_n$ is activated or there is no spread beyond the seed set.

**Theorem 2.1.** *Bootstrap percolation is trivial on $K_n$.*

*Proof.* Consider a complete graph $K_n$ where each of its vertices are $n-1$ regular. For a threshold $k \geq n-1$, no further vertices may be activated if the size of the

seed set $|A_0|$ is below the threshold. Every vertex in $K_n$ will have $|A_0| < k$ infected neighbors and remain inactive. If $|A_0|$ is greater than or equal to $k$, every vertex will have $|A_0| \geq k$ infected neighbors and be activated. □

## 2.2 Petersen Graph

The Petersen graph is an undirected graph with 10 vertices and 15 edges with each vertex connected to three other vertices. We examined outcomes for $|\langle A_0 \rangle|$ given a threshold of $k = 2$ and varied the seed set $A_0$. It aided to divide the Petersen graph's set of vertices into two disjoint sets: the "inner" vertices and the "outer" vertices.

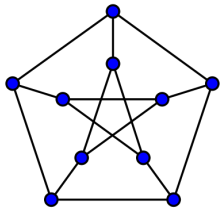


Figure 1: The Petersen graph.

Focusing on the inner set, we experimented with selecting a seed set $A_0$ of size 2. Selecting $A_0$ as two neighboring vertices results in no further activations $|\langle A_0 \rangle| = |A_0|$. Selecting two vertices that do not share an edge results in at most one subsequent activation. The outer set shares the same rules as the inner set.

An automorphism on the Petersen graph can map the inner vertices to the outer vertices. Such an automorphism maps the seed set $A_0$ of two neighboring inner vertices to two outer vertices that do not share an edge. As a result, $|\langle A_0 \rangle|$ after mapping does not create further infections as opposed to before mapping which results in at most one subsequent activation. Therefore, performing the isomorphism on the Petersen graph results in a scenario in which the opposite rules applies.

## 2.3 Bipartite Graphs

A complete bipartite graph, $K_{m,n}$ is similar to the case of the complete graph $K_n$. A complete bipartite graph has a vertex set $V$ which can be split into two disjoint subsets, $A$ and $B$. $A$ has cardinality $m$ and $B$ has cardinality $n$. Every vertex in one subset connects to every vertex in the other subset. For $m, n \geq k$, a large enough and well distributed seed set $A_0$ will infect the entire graph $K_{m,n}$.

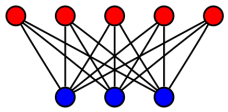


Figure 2: The complete bipartite graph $K_{3,5}$.

A general bipartite graph, $K$ differs from a complete bipartite in that every vertex in the first subset does not have to connect to every vertex in the second subset. We still have vertex subsets, $A$ and $B$. If $\left\lceil \frac{|A_0|}{2} \right\rceil <$ k, we can distribute $A_0$ such that $\min\{|\langle A_0 \rangle| : A_0 \in P_s(V)\} = |A_0|$.

**Theorem 2.2.** *Bootstrap percolation is trivial on the complete bipartite graph $K_{m,n}$.*

*Proof.* If $|A_0| < k$, every vertex in $K_{m,n}$ has less than $k$ infected neighbors and remain inactive. If $|A_0| \geq k$, $A_0$ can be entirely in subset $A$ or $B$. For example, if $A_0$ is chosen to be in subset $A$, subset $B$ will have $|A_0| \geq k$ infected neighbors and become activated during the first iteration of percolation. During the second iteration, all vertices in $A$ will now have $n \geq k$ infected neighbors and be activated. □

**Theorem 2.3.** *If $\left\lceil \frac{|A_0|}{2} \right\rceil < k$, we can distribute $A_0$ such that $\min\{|\langle A_0 \rangle| : A_0 \in P_s(V)\} = |A_0|$.*

*Proof.* Let G be a bipartite graph, composed of disjoint subsets U and V with $|U| \leq |V|$. In the case where $|A_0|$ is even and both $|U|$ and $|V| \geq \frac{|A_0|}{2}$, we choose to make exactly $\frac{|A_0|}{2}$ vertices of both U and V initially active. As $\frac{|A_0|}{2} <$ k, neither the initially active vertices in U or V are able to activate any vertices in the other subset. Therefore, $|A_0| = |\langle A_0 \rangle|$ in this case.

If $|U| < \frac{|A_0|}{2}$, then we choose to make every vertex of U active and $|A_0| - |U|$ vertices of V active. As $|U| <$ k, no vertices of V can be activated. As U is entirely active, there are no vertices left in U to be activated by V. Therefore, $|A_0| = |\langle A_0 \rangle|$ in this case.

In the case where $|A_0|$ is odd and both subsets are sufficiently large, we choose to make $\left\lfloor \frac{|A_0|}{2} \right\rfloor$ elements of U active, and $\left\lceil \frac{|A_0|}{2} \right\rceil$ elements of V active. Since both $\left\lfloor \frac{|A_0|}{2} \right\rfloor$ and $\left\lceil \frac{|A_0|}{2} \right\rceil <$k, neither the initially active vertices in U or V are able to activate any vertices in the other subset. Therefore, $|A_0| = |\langle A_0 \rangle|$ in this case.

If $|U| < \left\lfloor \frac{|A_0|}{2} \right\rfloor$, then we choose to activate every vertex in U and $|A_0| - |U|$ vertices of V. As $|U| <$ k, no vertices of V can become activated. As every vertex in

U is already active, no new vertices can become activated. Therefore, $|A_0| = |\langle A_0 \rangle|$ in this case. □

# 3 Empirical Findings

In [8], Reichman presents a greedy algorithm for deriving a contagious set $A_0$ for the minimum contagious set problem. An upper bound for the size of the contagious set is also given. The algorithm continuously deletes the vertex of highest degree at least equal to $k$ as well as its neighbors in each iteration. Vertices that have less than $k$ neighbors are ignored. When no more deletions can be made, the remaining vertices are chosen as the contagious set.

**Algorithm 1** A greedy algorithm for finding contagious sets

**Input:** a graph $G$ and threshold $k$
**Output:** a contagious set of $G$

```
while ∃v ∈ V(G) : d(v) ≥ k do
    Choose v ∈ V(G) : d(v) = min {d(u) : u ∈ V(G), d(u) ≥ k}
    G ← G − {v}
end while
return V(G)
```

For the spread minimization problem, the goal is to reduce spread as much as possible rather than infect the entire graph. Therefore, we took a contrary approach to the algorithm for finding a contagious set in [8] and select vertices of minimum degree that are greater than or equal to $k$ as part of the seed set. This is one of the algorithms we tested for the spread minimization problem.

Additionally, we investigated two types of random graphs to generate for testing in order to observe the performance of our algorithms for spread minimization. We looked into the Erdős–Rényi graphs and the stochastic block model. As mentioned in the introduction, bootstrap percolation has potential application for simulating the spread of ideas in social communities. The Erdős–Rényi graph is too simplistic for this application. Therefore, we further investigated the stochastic block model as an alternative to better represent communities.

## 3.1 Stochastic Block Model

A stochastic block model (SBM) is the union of two disjoint Erdős–Rényi graphs $G_1$ and $G_2$. The SBM is represented by $G = (n_1, n_2, p_1, p_2, q)$ where $n_i$ is the number

of vertices per graph, $p_i$ is the probability of connecting vertices within the same community per graph, and $q$ is the the probability for adding inter-connected edges between different communities $G_1$ and $G_2$. We test percolation on the SBM graph as an alternative to the Erdős–Rényi graph which does not accurately represent social communities. While the SBM is still simplistic, it is a step towards the right direction in representing social communities.

In [10], assumptions by [6] are given for the asymptotic bounds for $p_i$ and a formula for the critical number of seeds per community $g_i$. [6] also assumes that $n_1$ and $n_2$ are scalar multiples of each other as well as $p_1$ and $p_2$. The asymptotic bounds for $p_i$ are given in terms of $n_i$ and $k$,

$$1/n_i \ll p_i \ll 1/({n_i}^{1/k})$$

Additionally, a critical number of seeds per community $g_i$ is defined in terms of $p_i$, $n_i$, and $k$,

$$g_i = \left(1 - \frac{1}{k}\right)\left(\frac{(k-1)!}{n_i p_i^k}\right)^{\frac{1}{k-1}}$$

[10] states that when $O(g_1)$ vertices are initially infected, there are extreme cases where percolation either stops such that $|\langle A_0 \rangle| = |A_0|$ or nearly the entire graph is activated. Theorems 3.1 and 3.2 below are given by [10] to arrive at this conclusion.

**Theorem 3.1.** *For any $\varepsilon > 0$, there exists $c(\varepsilon) \in \mathbb{R}_+$ such that*

$$P\left(|\frac{|G|}{g_1} - x_*| > \varepsilon\right) = O\left(e^{-c(\varepsilon)g_1}\right)$$

*where $x_* > 0$.*

**Theorem 3.2.** *For any $\varepsilon > 0$, there exists $c(\varepsilon) \in \mathbb{R}_+$ such that*

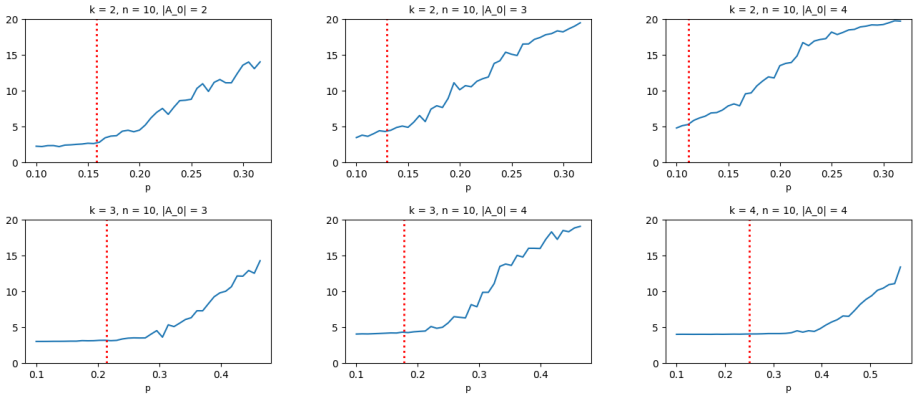
$$P\left(|\frac{|G|}{n} - 1| > \varepsilon\right) = O\left(e^{-c(\varepsilon)g_1}\right).$$

## 3.2 Testing percolation on the SBM

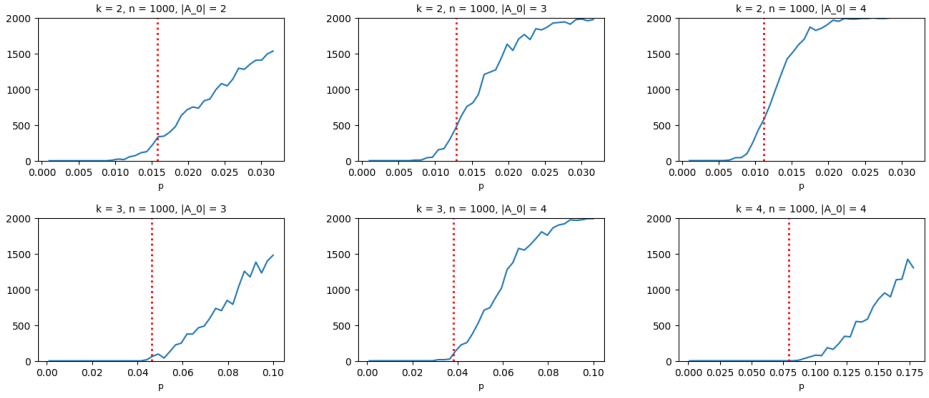
We generated several computations to observe the behavior of percolation in the SBM model by treating the bounds for $p_i$ as non-asymptotic. Generating these computations would allow us to observe how random selection of a seed set performs for percolation in the SBM. We performed iterations of bootstrap percolation for community sizes of 10, 50, 100, 500, and 1000 vertices and generated 40 $p_i$-values bounded between $1/n_i$ and $1/({n_i}^{1/k})$. We set the rest of the parameters as follows,

- $n_1 = n_2$ and $p_1 = p_2$
- $q = 0.75p$
- $2 \leq |A_0| \leq 4$ and $2 \leq k \leq 4$
- $k$ is bounded below by $|A_0|$

According to [10], a phase transition, in which the number of infected vertices $|\langle A_0 \rangle|$ begins to rapidly increase, is likely to be observed when $g_i = \left(1 - \frac{1}{k}\right)\left(\frac{(k-1)!}{n_i p_i^k}\right)^{\frac{1}{k-1}}$. Therefore, when $|A_0|$ and $g_i$ are equal for some $p_i$, we hope to observe an increase in average number of infected vertices $|\langle A_0 \rangle|$. The computations fix $k$, $|A_0|$, and $n_i$ as constants while varying $p_i$.




Average size of final infected vertices $|\langle A_0 \rangle|$ as a function of probability $p_i$, $n_i = 10$




Average size of final infected vertices $|\langle A_0 \rangle|$ as a function of probability $p_i$, $n_i = 1000$

The red dotted line for each plot indicates when the critical number of seeds $g_i$ is reached and equal to $|A_0|$ for some value of $p_i$. For the plots where $n = 10$, it can generally be observed that after the red line, there is an increase in the number of infected vertices from $|A_0|$. For some graphs, before this red line or when $|A_0| < g_i$, there is a horizontal line indicating that $|\langle A_0 \rangle| = |A_0|$ or the final size of infected vertices stayed at $|A_0|$. However, after the red line or when $|A_0| \geq g_i$, it can be observed that the number of infected vertices increases from $|A_0|$. The same trend be observed for the rest of the sizes for $n = 50, 100, 500$, and 1000.

## 3.3 Testing algorithms on Erdős–Rényi graphs

An Erdős–Rényi graph $G_1$ is defined in terms of a fixed number of vertices $n$ and an edge probability $p$ as $G_1 = (n, p)$. The graph is created by generating the vertices, and then adding each possible edge in the graph with probability $p$. We investigated various ways of selecting the seed set of an Erdős–Rényi graph, and how they impacted spread on that graph.

Our selection choices were based on the degree sequence of the graph. As a baseline for comparison, the first algorithm chose vertices for the seed set entirely randomly.

In order to remain efficient in terms of computations, we chose to only consider how choice of seed set impacted the vertices directly connected to that seed set, and not how further spread would be impacted. In order to accomplish this, we attempted to minimize the known number of edges between active and inactive vertices.

Our 5 different potential algorithms considered two factors: the degree of vertices chosen, and whether their neighbors should be activated as well or not. We tested choosing the smallest degree vertices, with and without activating the neighbors of a vertex as it was chosen, as well as the largest degree vertices with both neighbor options. Our 5th and final algorithm was one of our own design, which we call GLT - "Greatest Less Than".

The smallest and largest degree vertex algorithms work as follows: Order the degree sequence of $G$ in ascending order for smallest degree, and descending order for largest degree. If we are not considering neighbors, then for a given graph $G$ and seed set $A_0$, with $|A_0| = A$, we select and activate the first A vertices in the degree sequence of $G$.

If we are considering neighbors, then we denote the current number of vertices needing to be activated before the seed set is fully chosen by $A_G$. As we select vertices from the beginning of the degree sequence, we also activate every vertex in $N(v)$, the set of neighbors of $v$. Each time a vertex is chosen, and its neighbors activated

alongside it, we decrement $A_G$ by $d(v) + 1$). If at any point $d(v) \geq |A_G|$, then we activate the chosen vertex and a random $A_G - 1$ sized subset of $N(v)$.

For the GLT algorithm, we define $V_G$ to be the set of vertices eligible for activation, initially equal to the vertex set of the graph $V$. We choose vertices to activate sequentially, with the remaining number of vertices needing to be activated denoted by $A_G$. We search for

$$\max\{d(v) : v \in V_G, d(v) < |A_G|\}. \tag{1}$$

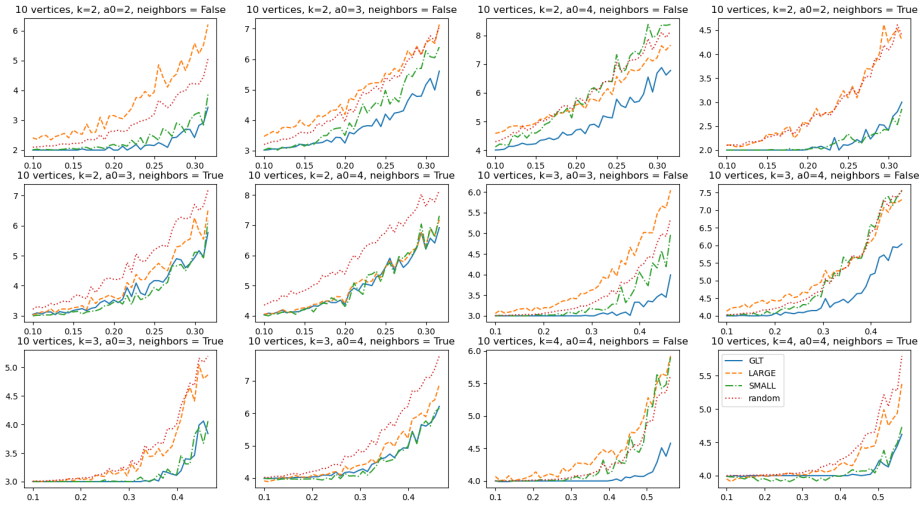
The vertex that satisfies these conditions is activated, as well as $N(v)$. These activated vertices are removed from $V_G$, and $A_G$ is decremented by $(d(v)+1)$, the number of activated vertices. This process repeats until either $A_G = 0$ or there is no $v \in V_G$ that satisfies (1). If $A_G = 0$, the algorithm has chosen the seed set. Otherwise, $V \in V_G$ is selected such that $d(v) = \min\{d(v) : v \in V_G\}$. In this case, we activate v and a random $A_G - 1$ sized subset of $N(v)$.

We found that for graphs of sufficiently large size, the smallest tested size being $n = 50$, choosing seed set by the vertices with smallest degree was most effective for minimizing spread.

For graphs smaller than this size, the tested size being $n = 10$, we found that the GLT algorithm performed best at minimizing spread.

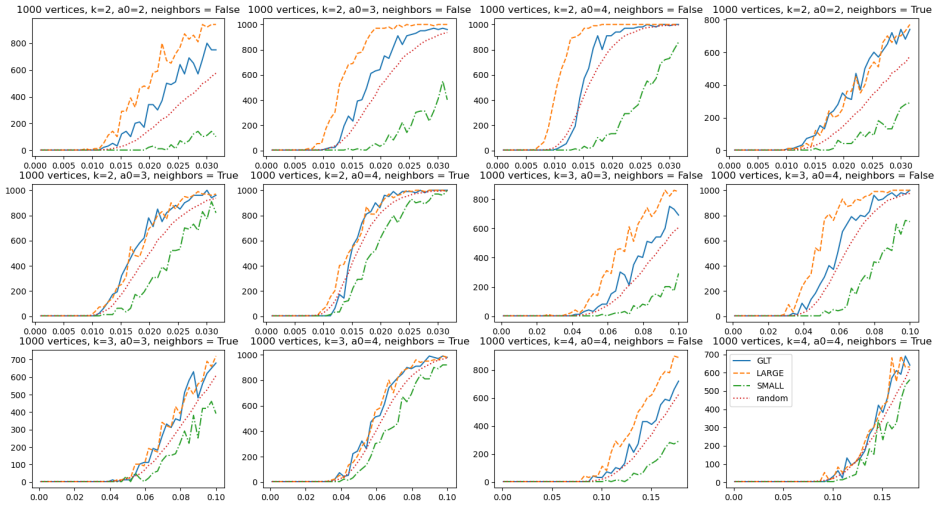
To get these results, we used the same parameters for $n$, $p$, $A_0$, and $k$ as in the SBM computations. With each trial consisting of 100 graphs, we chose seed sets based on each of the various algorithms and plotted their average $|\langle A_0 \rangle|$.

We found that when applied to graphs with only 10 vertices, the algorithms performed in a manner unique to the results on larger graphs.




Effectiveness of algorithms on various 10 vertex graphs

Our GLT algorithm performs better than or similarly to the smallest degree vertex algorithm for any set of parameters on these 10 vertex graphs. With all other parameters being equal, a choice between activating neighbors or not impacts the smallest and largest degree vertex algorithms in an expected fashion. The smallest degree vertex algorithm, when considering neighbors, is activating vertices that we know only one thing about: they have degree equal to or larger than the vertices that would have been chosen without neighbors. This makes it very likely for the algorithm to perform worse when considering neighbors, as those neighbors will likely increase the initial number of active-inactive edges. The largest degree vertex algorithm has an opposite change. As the neighbors being activated necessarily have degree equal to or less than the vertices that would be chosen without neighbors, the algorithm is very likely to perform better when considering neighbors. The trends seen for these 10 vertex graphs appear uniquely in our results.




Effectiveness of algorithms on various 1000 vertex graphs

Above is the results from Erdős–Rényi graphs with 1000 vertices. Similar trends to those discussed below were present in graphs of size 50, 100, and 500. On graphs of these sizes, we found that the most effective algorithm was choosing the smallest degree vertex, while not considering neighbors. There is a distinct contrast from the 10 vertex graphs when considering the GLT algorithm. As the size of the graph grows, the GLT algorithm performs consistently worse, even rarely doing worse than the largest degree vertex algorithm. Regardless of size, all graphs show the same trend regarding neighbor sets. When neighbors are considered, the smallest degree vertex algorithm performs consistently worse, and the greatest degree vertex algorithm performs consistently better. (see top left and top right graphs of each set for a simple comparison).

# 4 Future Work

We hope to continue our work on the algorithmic approach to spread minimization on Erdős–Rényi graphs in two particular areas. The first is probabilistically bounding the eventually infected set of vertices for graphs based on algorithmic choice of seed set. This would likely take the form of a bound with some confidence interval, as we are working with graphs of finite size. The other area is an exploration into which graphs, or classes of graphs, cause certain algorithmic choices of seed set to perform

optimally.

# 5 Acknowledgements

We are incredibly grateful to our mentor, Connor Anderson, for his guidance, insight, encouragement, and sense of humor throughout our investigations. Furthermore, we would like to thank Iva Halacheva, Matej Penciak, and Joshua Wen for so graciously organizing and supporting this program.

We also acknowledge the generous NSF-RTG grant Algebraic Geometry and Representation Theory at Northeastern University, DMS-1645877, without which this research experience would not have been possible, as well as additional support from the Northeastern University Department of Mathematics and Northeastern University College of Science.

# References

[1] J. Balogh, Y. Peres, and G. Pete. "Bootstrap percolation on infinite trees and non-amenable groups". In: (2003). DOI: 10.48550/ARXIV.MATH/0311125. URL: https://arxiv.org/abs/math/0311125.

[2] J. Balogh and G. Pete. "Random disease on the square grid". In: *Random Structures & Algorithms* 13 (1998).

[3] J. Balogh, B. Bollobás, and R. Morris. "Majority bootstrap percolation on the hypercube". In: (2007). DOI: 10.48550/ARXIV.MATH/0702373. URL: https://arxiv.org/abs/math/0702373.

[4] J. Chalupa, P. L. Leath, and G. R. Reich. "Bootstrap percolation on a Bethe lattice". In: *Journal of Physics C: Solid State Physics* 12.1 (1979), pp. L31–L35. DOI: 10.1088/0022-3719/12/1/008. URL: https://doi.org/10.1088/0022-3719/12/1/008.

[5] U. Feige, M. Krivelevich, and D. Reichman. "Contagious sets in random graphs". In: *Ann. Appl. Probab.* 27.5 (Oct. 2017), pp. 2675–2697. DOI: 10.1214/16-aap1254. URL: https://doi.org/10.1214/16-aap1254.

[6] S. Janson et al. "BOOTSTRAP PERCOLATION ON THE RANDOM GRAPH G n,p". In: *The Annals of Applied Probability* 22.5 (2012), pp. 1989–2047. ISSN: 10505164. URL: http://www.jstor.org/stable/41713404 (visited on 07/01/2022).

[7] D. Kempe, J. Kleinberg, and E. Tardos. "Maximizing the Spread of Influence through a Social Network". In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. Washington, D.C.: Association for Computing Machinery, 2003, 137–146. ISBN: 1581137370. DOI: 10.1145/956750.956769. URL: https://doi.org/10.1145/956750.956769.

[8] D. Reichman. "New bounds for contagious sets". In: *Discrete Mathematics* 312.10 (2012), pp. 1812–1814. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2012.01.016. URL: https://www.sciencedirect.com/science/article/pii/S0012365X12000301.

[9] P. Shakarian and D. Paulo. *Large Social Networks can be Targeted for Viral Marketing with Small Seed Sets*. 2013. arXiv: 1205.4431 [cs.SI].

[10] G. L. Torrisi, M. Garetto, and E. Leonardi. *Bootstrap percolation on the stochastic block model*. 2022. DOI: 10.48550/ARXIV.2201.13263. URL: https://arxiv.org/abs/2201.13263.