Approximability of the Firefighter Problem: Computing Cuts over Time

Shaked Levi, Almog David, and Yuval Bubnovsky

Department Of Computer Science, Ariel University

August 25, 2024

Abstract

This paper extends the work on approximation algorithms for the Firefighter problem presented in the original study by **Anshelevich, Chakrabarty, Hate, and Swamy**[1]. We focus on both the non-spreading and spreading models of infection and vaccination propagation in Directed graphs and Directed-Layered-Networks, addressing the Max-Save and Min-Budget problems. First, we provide detailed pseudo-code implementations of the original algorithms, followed by their implementation in Python. This offers a concrete, executable foundation for further research in this domain. Second, we propose two novel heuristic approaches aimed at improving upon the results of the original algorithms. These heuristics are designed to provide better results in terms of how many nodes were saved or how minimal the budget can be, by leveraging local search problem solving with a greedy approach. We implement our heuristics in Python and conduct extensive comparative testing against the original algorithms. Our experimental results demonstrate improvements over the original algorithms, particularly for larger graphs and in scenarios with lower graph density. The heuristic approach shows effectiveness in minimizing the budget required to save all target nodes (MinBudget) and competitive performance in maximizing saved nodes (MaxSave) in denser graphs.

1 Background

1.1 The Model and the Firefighter Problem

We model our network as a graph $G = (V, E)$. This model of infection spread has been studied in the literature as the Firefighter problem. The infection/fire starts at a given node s (or a set of nodes) at time $\tau = 0$. At every subsequent time step, the infection/fire deterministically spreads to all nodes that have an already infected neighbor. To stop the infection, we are allowed to vaccinate/defend at most B nodes per time step, where B is a budget representing how much we are able to affect the network in a single time step.

A vaccinated node can no longer contract the infection, and therefore cannot pass it on to others. Once infected or vaccinated, the vertex remains so for the rest of the time. The process ends when the infection can no longer spread.

It's important to note that in the end, the set of vaccinated nodes form a vertex cut between the set of infected nodes and set of non-infected nodes. However, unlike previous works that examine the static problem of vaccinating a 'cut' before the infection has started spreading, the article aims to find the best "cut over time" (where best depends on the considered objective). This temporal nature of this optimization problem makes it significantly different and more challenging than finding a cut in a "static" network.

1.2 Summary of Anshelevich, Chakrabarty, Hate, and Swamy's Work

Model Description

- In our model, we use a directed graph since an undirected graph is simply a directed graph where each edge in the underictd graph can be represented as two edges in a directed graph. This graph is a graph $G = (V, E)$ with two sets of vertices V and edges E, and a dedicated source vertex s from which the virus spreads.
- We define $n = |V|$ and $m = |E|$.
- The graph is given as a set of vertices u, v and the edge between them is $(u, v) \in E$.
- Each vertex can be in one of the following three states: healthy, infected, and vaccinated.
- At time $\tau = 0$, all vertices in the graph are healthy except for vertex s which is infected.
- At time $\tau = 1$ and later, B additional vertices can become vaccinated. After that, any vertex v that is a neighbor of an infected vertex and is not vaccinated becomes infected.
- Any vertex that is infected or vaccinated remains so for the rest of the algorithm's run.
- A strategy is valid if and only if all vertices that are vaccinated at any given time are not able to be infected by another vertex.

Vaccination Strategy
A vaccination strategy is a set $\Psi \subseteq V \times J$ where V is the set of vertices of graph G and $J = \{1, 2, \ldots, |V|\}$. The vertex v is vaccinated at time $\tau \in J$ by the vaccination strategy Ψ if $(v, \tau) \in \Psi$. A vaccination strategy Ψ is valid with respect to budget B, if the following two conditions are satisfied:

1. if $(v, \tau) \in \Psi$ then v is not infected at time τ,
2. let $\Psi_\tau = \{(v, \tau) \in \Psi\}$; then $|\Psi_\tau| \leq B$ for $\tau = 1 \ldots |V|$.

The first condition implies we can only vaccinate vulnerable nodes, and the second condition requires us to obey the budget constraint.

Anshelevich, Chakrabarty, Hate, and Swamy [1] considered two primary variants of the Firefighter problem:

1. **Non-spreading model**: Vaccinating a node only prevents that node from being infected, while the virus keeps spreading in each timestamp to vulnerable nodes.
2. **Spreading model**: Vaccination itself is an infectious process, spreading to neighboring nodes in subsequent time steps.

In the non-spreading model, vaccinating a vertex simply means that this vertex cannot be infected. The vaccination of a vertex does not affect the vaccination status of its neighboring vertices. In contrast, in the spreading model, the vaccination spreads to all neighboring vertices that are still vulnerable, consequently vaccinating them as well. Thus, at time step $\tau > 0$, if vertex v is vaccinated and has a vulnerable neighbor u, then at time step $\tau + 1$, vertex u will also be vaccinated. In the spreading model, we define priority for the spread of vaccination over the spread of the virus.

In the spreading model, we define two types of vaccination:

- **Direct vaccination:** A vertex is directly vaccinated if it received the vaccine through the vaccination strategy.
- **Indirect vaccination:** A vertex is indirectly vaccinated if it received the vaccine through the network (by being a neighbor of a vaccinated vertex).

The process will stop when there are no vulnerable vertices in the graph that are neighbors of infected vertices, so the virus can no longer spread. This must happen before time n, where n is the number of vertices in the graph.
For each model, they addressed two optimization problems:

1. **MaxSave(G, B, s, T)**

   Instance: A rooted graph $(G(V, E), s)$, integer $B \geq 1$ and $T \subseteq V$

   Objective: Find a valid vaccination strategy Ψ such that if s is the only infected node at time 0, then at the end of the above process the number of non-infected nodes that belong to T is maximized.

   This problem is also referred to as the Firefighter Problem when $T = V$.

2. **MinBudget**(G, s, T)

   Instance: A rooted graph $(G(V,E),s)$, and $T \subseteq V$

   Objective: Find a valid vaccination strategy Ψ with minimum possible budget B, such that if s is the only infected node at time 0, then at the end of the above process all nodes in T are saved.

In other words, in MaxSave the goal is saving as many nodes of T as possible given a fixed budget, and in MinBudget the goal is finding the minimum necessary budget to save all nodes in T.
The results of the article are algorithms for each objective under each spreading model, with approximation and hardness proofs, we will not provide these proofs here

2 Algorithms

We'll address each model (spreading/non-spreading) seperatley while providing some definitions and theorems **verbatim** from the article - we will not provide the proofs of these lemmas and theorems, but simply state them for better understanding of the algorithms which will be presented.

2.1 Spreading Vaccination Model

2.1.1 Characterizing Saved Vertices in the Spreading Model

To allow us to present the psuedo-code and discuss the algorithms presented in the article, we will define several terms, lemmas and theorems.

Let $d(u,v)$ be the shortest distance between the nodes u and v in graph G, and let $N(v,i)$ be the set of all the nodes that are a distance of at most i from v.

Lemma 1 *At time τ, all nodes in the neighborhood $N(s,\tau)$ will either be vaccinated or infected.*

Following this, the article defines a set $\Gamma(v) \subseteq V \times J$ for every node $v \in V$ which will be used to characterize if v is saved by a vaccination strategy Ψ or not. Let

$$\Gamma(v) := \{(u,\tau) | u \in V \text{ and } 0 < \tau \leq d(s,v) - d(u,v)\}.$$

Recall, the tuple (u,τ) represents the direct vaccination of the node u at time τ. The following theorem states that a vertex v is saved by a vaccination strategy Ψ in the spreading model, if and only if the strategy vaccinates some u directly at time τ, such that $(u,\tau) \in \Gamma(v)$.

Theorem 1 *A node $v \in V$ is saved by the vaccination strategy Ψ if, and only if, $\Psi \cap \Gamma(v) \neq \emptyset$.*

2.1.2 The MaxSave Problem

Using Theorem 1, the article proves that the MaxSave problem in the spreading vaccination model can be cast as the problem of maximizing a monotone submodular function over a partition matroid constraint.
Given a ground set U, a function $f : 2^U \to \mathbb{R}_{\geq 0}$ is a monotone submodular function iff

- $f(A) \leq f(B)$ whenever $A \subseteq B \subseteq U$; and
- $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)\ \forall A, B \subseteq U$

A set system $(U, \mathcal{F})$ is a partition matroid if there is a partition of the ground set $U = U_1 \cup \cdots \cup U_k$, and there exists integers $\ell_1, \ldots, \ell_k$ such that $X \subseteq U$ is in $\mathcal{F}$ if and only if $|X \cap U_i| \leq \ell_i$, for all i. The problem of maximizing a monotone submodular function f over a partition matroid $(U, \mathcal{F})$ is to find a subset $X \in \mathcal{F}$ which maximizes $f(X)$.

Theorem 2 *There is a deterministic algorithm which achieves a 2-approximation for the problem of maximizing a monotone submodular function over any matroid [1]*

Theorem 3 *In the spreading vaccination model, MaxSave is a special case of maximizing a monotone submodular function over a partition matroid constraint.*

As part of the proof of Theorem 3, which will not be brought here in full, the article defines and proves the following:
Define $\mathcal{E}$ as the set of all possible direct vaccination tuples (v, τ) where $v \in V$ and $\tau = 1 \ldots n$. Note that from Lemma 1, we can assume $\tau \leq d(s, v)$ for all $(v, \tau) \in \mathcal{E}$. $\mathcal{E}$ can be partitioned as follows:

$$\mathcal{E} = \bigcup_{\tau=1}^{n} \mathcal{E}_\tau \quad \text{where } \mathcal{E}_\tau = \{(v, \tau) | (v, \tau) \in \mathcal{E}\}$$

Proposition 1 *A vaccination strategy Ψ is valid if and only if $|\Psi \cap \mathcal{E}_\tau| \leq B$, for all τ. That is, the collection of vaccination strategies forms a partition matroid over $\mathcal{E}$.*

Using this proposition, the article proves that the optimal Ψ, the vaccination strategy which maximizes the number of saved nodes, also maximizes a certain submodular function over $\mathcal{E}$. This function will be a coverage function. That is, for each element $(u, \tau) \in \mathcal{E}$, define a set $S_{(u,\tau)}$, and the function $f : 2^{\mathcal{E}} \to \mathbb{R}_{\geq 0}$ is given as

$$f(\Psi) := \left| \bigcup_{(u,\tau) \in \Psi} S_{(u,\tau)} \right|.$$

Where $S_{(u,\tau)}$ is defined as:

$$S_{(u,\tau)} := \{v \in V : (u, \tau) \in \Gamma(v)\}.$$

Thus a node v is saved if and only if $v \in S_{(u,\tau)}$ for some (u, τ) in Ψ.

Following this, and by using the 2-approximation algorithm provided in [1], we devised the following psuedo-code for the MaxSave algorithm in the spreading model:

Algorithm 1 Spreading MaxSave

```
Require: Graph $G(V, E)$, source node $s$, target set $T \subseteq V$, budget $B$
for each $v \in V$ do
    Calculate $\Gamma(v) = \{(u, \tau) \mid u \in V$ and $0 < \tau \leq d(s, v) - d(u, v)\}$ ▷$d(x, y)$ is the shortest distance between $x$ and $y$
end
for each possible $(u, \tau)$ pair do
    Calculate $S(u, \tau) = \{v \in T : (u, \tau) \in \Gamma(v)\}$ ▷Nodes in $T$ saved by vaccinating $u$ at time $\tau$
end
$\varepsilon \leftarrow$ all possible $(u, \tau)$ pairs grouped by $\tau$
$\Psi \leftarrow \emptyset$
t $\leftarrow 1$
while the infection can spread and $t < |\varepsilon|$ do
    Spread the vaccination to the vaccinated nodes neighbors
    for $i \in \{1, 2, \dots, /B/\}$ do
        Find $(u^*, \tau^*) \in \varepsilon_t$ which maximize the saved nodes from $T$ that haven't been saved yet by $\Psi$
        Add $(u^*, \tau^*)$ to $\Psi$
        Vaccinate the selected node
        Remove $(u^*, \tau^*)$ from $\varepsilon$
    end
    Spread the virus to the infected nodes neighbors
end
return $\Psi$ and the saved target nodes
```

2.1.3 The MinBudget Problem

Again, we will provide the theorems and propositions verbatim from the article, without proof, following with the psuedo-code we derived from these.

Theorem 4 *In the spreading vaccination model, the MinBudget problem is as hard as set cover.*

Proposition 2 *There is a set cover of size B if and only if there is a vaccination strategy immunizing at most B nodes per time step.*

Theorem 5 *There is a polynomial time* $\ln n$*-factor approximation algorithm for MinBudget in the spreading model.*

Following these, we derived the following algorithm:

Algorithm 2 Spreading MinBudget

```
Require: Graph $G(V, E)$, source node $s$, target set $T \subseteq V$
1: Perform a binary search on the size of the target group $T$
2: set $B$ as the current middle value of the binary search
3: Run the MaxSave algorithm on the parameters of the problem with the budget $B$
   that we found in the binary search
4: Match the budget $B$ so that it is minimal but still able to save all nodes of $T$
5: return $B$ and $\Psi$ for the returned $B$
                                              ▷ The approximated minimal budget
```

2.2 Non-Spreading Vaccination Model

The non-spreading model is considerably more difficult to reason about than the spreading model. For instance, the structural lemma, Lemma 1 (or any simple modification of it), is no longer true in this model. Thus, it is difficult to characterize the set of nodes which are saved by a vaccination strategy. The article shows that the **MaxSave** problem is hard to approximate in this model to any nontrivial factor. Therefore, this section will only contain algorithms for the **MinBudget**, one that achieves a $O(\sqrt{n})$ approximation for directed graphs, and a $O(\log n)$ approximation for directed layered networks.

To simplify notation, the article considers the following equivalent problem: **we add a new node t with edges from all nodes in T to t**, and consider the problem of saving t with minimum budget *under the additional constraint that t itself cannot be vaccinated.* Call s the source and t the sink.

Theorem 6 *There is a $2\sqrt{n}$ factor approximation algorithm for the MinBudget problem.*

Simply based on this theorem, we derived the following algorithm:

Algorithm 3 Non-Spreading MinBudget

```
Require: Graph $G(V, E)$, source node $s$, target node $t$
1: Find Min Cut of $s$-$t$ (source, target) using Edmond Karp's / Ford Fulkerson algorithm
2: set $B$ as the Minimum cut size
3: set $\Psi$ where the Minimum cut nodes are vaccinated at time step 1
4: return $B$ and $\Psi$
```

Before we continue, we must first define what a Directed-Layered Graph is (we call this in short: DirLay graph or DirLay network)

Definition: Directed-Layered Graph
A *directed-layered graph* is a directed graph $G = (V, E)$ where:

- The vertex set V is partitioned into k disjoint subsets $V_1, V_2, \ldots, V_k$ called *layers.*
- Every directed edge $(u, v) \in E$ satisfies $u \in V_i$ and $v \in V_{i+1}$ for some $i \in \{1, 2, \ldots, k-1\}$.
- There are no edges between vertices within the same layer, i.e., $(u, v) \notin E$ if $u, v \in V_i$ for any i.

Theorem 7 *If the network is a layered directed graph with ℓ layers, then there is a poly-time $\lceil H(\ell) \rceil$ approximation algorithm to the MinBudget problem.*

Fact 1 *Given a matrix M' with possibly fractional entries, one can obtain another integral matrix M such that for every row i and column j, we have (i) $M_{ij} \in \{\lfloor M'_{ij} \rfloor, \lceil M'_{ij} \rceil\}$; (ii) the row-sum of row i in M is the floor or ceiling of the row-sum of row i in M'; and (iii) the column-sum of column j in M is the floor or ceiling of the column-sum of column j in M'.*

Combining these theorems and fact, the article presents it's algorithm for DirLay graphs:

Algorithm 4 DirLayNet

```
1: Set the capacity of each vertex $v \in L_i$ at $\frac{1}{iH(\ell)}$.
2: Find the minimum $s - t$ cut in this capacitated network, let it be $(N_1 \cup \cdots \cup N_\ell)$
   with $N_i \subseteq L_i$.
3: The algorithm vaccinates the vertices $N_j$ in $j$ days as follows. Construct the following
   upper-triangular matrix $M'$. Let $M'_{ij} := |N_j|/j$, for all $1 \le i \le j \le \ell$. Note that for
   any column $j$, the column sum of $M'$ is exactly $|N_j|$.
4: Apply **Fact 1** to construct the corresponding integral matrix $M$ from $M'$.
5: The vaccination strategy is as follows: on time step $i$, vaccinate $M_{ij}$ nodes from layer
   $j$, for all $i \le j \le \ell$.
```

The way **Algorithm 4** is presented in the article presented us with a challenge - how can we find the minimum $s - t$ cut in a capacitated network when all of the capacities are defined in the vertices?
We addressed this by implementing a reduction to transfer the capacities from the vertices to the edges, which allows to run several well-known algorithms for finding cuts in a capacitated network [2].

Algorithm 5 Non-Spreading DirLayNet MinBudget

```
Require: Directed layered graph $G(V, E)$, source node $s$, target node $t$, $l$ layers
 1: Set capacity of each vertex $v \in L_i$ at $\frac{1}{i \cdot H(l)}$ ($l$ = number of layers)
 2: Find the min $s$-$t$ cut in this capacitated network:
 3:     Apply reduction to transfer capacitated network from nodes to edges:
 4:         Construct new graph $G'$:
 5:             Replace each vertex $v$ with two vertices $v_{in}$ and $v_{out}$
 6:             Add edge $(v_{in}, v_{out})$ with capacity equal to $v$'s capacity in $G$
 7:             For each edge $(v, u)$ in $G$, add edge $(v_{out}, u_{in})$ with capacity $+\infty$
 8:         Apply Edmond Karp's / Ford Fulkerson algorithm on $G'$, denote results as $H'$
 9:         Find min $s$-$t$ cut in $H'$ to get required nodes
10:     Denote result as $(N_1 \cup \cdots \cup N_l)$ with $N_i \subseteq L_i$
11: The algorithm vaccinates the vertices $N_i$ in $i$ days as follows:
12:     Construct upper triangular matrix $M'$:
13:         $M'_{ij} := \frac{|N_j|}{j}, 1 \leq i \leq j \leq l$ (For any col $j$, col sum is exactly $|N_j|$)
14: Apply **Fact 1** to construct the corresponding integral matrix $M$ from $M'$
15: set $B$ as the maximum sum of the rows of $M$
16: $\Psi$: on time step $i$, vaccinate $M_{ij}$ nodes from layer $j$, for all $i \leq j \leq l$
17: **return** $B$ and $\Psi$
```

3 Proposed Heuristics

The article discusses several algorithms to find an approximate solution for the fire-fighter problem, where a common theme between these algorithms is that for each "viable" option, they look ahead and calculate to check if this choice maximizes the objective function. We propose a different approach using a local search method.

To address the MaxSave and MinBudget problems, we adopted a local search algorithm. In the original algorithms, the authors calculated all possible paths across the entire graph, allowing node selection for the vaccination strategy to consider the entire graph. Consequently, the algorithm might sometimes bypass a current target node in favor of a non-target node that could potentially save more nodes in the future.

Our approach focuses on the immediate state of the graph. Instead of looking ahead to future possibilities, we concentrate on the current infected nodes, selecting a node for direct vaccination from their immediate neighbors. The selection process follows these rules:

1. Select a node if it is a target node.

2. If multiple nodes qualify, choose the one with the most targeted neighbors.

3. If there is still a tie, select the node with the most overall neighbors.

These rules ensure that the algorithm prioritizes targeted nodes that would be infected in the next round if not vaccinated immediately.

For the heuristic MinBudget algorithm, recall that it makes calls to the MaxSave algorithm to determine if the current budget is enough to save all target nodes, therefore, we decided to retain the same algorithm as the original article, but calling our heuristic MaxSave as a part of it.

3.1 Heuristic MaxSave

Algorithm 6 Heuristic - MaxSave Algorithm

```
Require: Graph $G(V,E)$, source node $s$, target set $T \subseteq V$, budget $B$, spreading
$\Psi \leftarrow \emptyset$
t $\leftarrow$ 1
while the infection can spread do
    if spreading then
        Spread the vaccination to the vaccinated nodes' neighbors
    end
    for $i \in \{1, 2, \ldots, B\}$ do
        $C \leftarrow$ set of unvaccinated neighbors of infected nodes
        $PQ \leftarrow$ empty max priority queue
        foreach $u \in C$ do
            $inTarget \leftarrow [u \in T]$ ▷1 if true, 0 if false
            $targetNeighbors \leftarrow |\{v \in N(u) : v \in T$ and $v$ is not vaccinated$\}|$
            ▷Priority tuple:  (inTarget, targetNeighbors, |N(u)|)
            ▷1.  Prioritize nodes in target set T
            ▷2.  If tie, prioritize nodes with more unvaccinated target
             neighbors
            ▷3.  If still tied, prioritize nodes with higher degree
            Enqueue u into PQ with priority $(inTarget, targetNeighbors, |N(u)|)$
        end
        if PQ is not empty then
            $u^* \leftarrow$ Dequeue PQ
            Add $(u^*, t)$ to $\Psi$
            Vaccinate $u^*$
        end
    end
    Spread the virus to the infected nodes' neighbors
    Increment t
end
return $\Psi$ and the saved target nodes
```

3.2 Heuristic MinBudget

Algorithm 7 Heuristic MinBudget

```
Require: Graph $G(V, E)$, source node $s$, target set $T \subseteq V$
1: Perform a binary search on the degree of the source node - $N(s)$
2: set $B$ as the current middle value of the binary search
3: Run the Heuristic-MaxSave algorithm on the parameters of the problem with the
   budget $B$ that we found in the binary search and declaring if the vaccination is also
   spreading or not.
4: Match the budget $B$ so that it is minimal but still able to save all nodes of $T$
5: return $B$ and $\Psi$ for the returned $B$
                                                  ▷ The approximated minimal budget
```

These heuristic algorithms are versatile, applicable to both spreading and non-spreading methods. The only difference lies in whether the vaccination spreads; the underlying mechanism and decision-making process remain consistent.

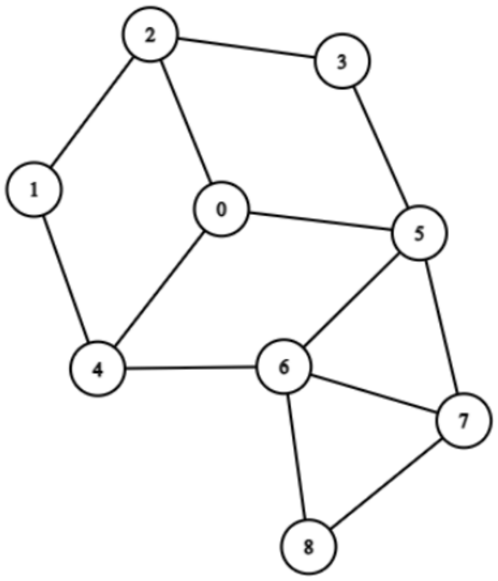


Figure 1: Small example graph

In order to show the differences between the article's algorithms and ours, we will provide a small example of a single run of one of the algorithms.

Consider the graph in **Figure 1**, where the source node is 0, and the target nodes are 1, 2, 6, and 8. We will begin with the **Spreading MinBudget Algorithm**:

1. **Initial Step**: Run the MaxSave algorithm using the same parameters as the MinBudget input. The budget is set to half the size of the target node group, which in this case is 2.

2. **Calculation**: Perform calculations of Γ, $S_{(u,\tau)}$, and ε.

$$\begin{aligned}
\Gamma(1) &= \{(2,1),(4,1),(1,1),(1,2)\} \\
\Gamma(2) &= \{(2,1)\} \\
\Gamma(3) &= \{(2,1),(5,1),(3,1),(3,2)\} \\
\Gamma(4) &= \{(4,1)\} \\
\Gamma(5) &= \{(5,1)\} \\
\Gamma(6) &= \{(4,1),(5,1),(6,1),(6,2)\} \\
\Gamma(7) &= \{(5,1),(7,1),(7,2)\} \\
\Gamma(8) &= \{(4,1),(5,1),(6,1),(6,2),(7,1),(7,2),(8,1),(8,2),(8,3)\} \\
S_{(1,1)} &= \{1\} \\
S_{(1,2)} &= \{1\} \\
S_{(2,1)} &= \{1,2\} \\
S_{(3,1)} &= \{\} \\
S_{(3,2)} &= \{\} \\
S_{(4,1)} &= \{1,6,8\} \\
S_{(5,1)} &= \{6,8\} \\
S_{(6,1)} &= \{6,8\} \\
S_{(6,2)} &= \{6,8\} \\
S_{(7,1)} &= \{8\} \\
S_{(7,2)} &= \{8\} \\
S_{(8,1)} &= \{8\} \\
S_{(8,2)} &= \{8\} \\
S_{(8,3)} &= \{8\} \\
\varepsilon_1 &= \{(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1)\} \\
\varepsilon_2 &= \{(1,2),(3,2),(6,2),(7,2),(8,2)\} \\
\varepsilon_3 &= \{(8,3)\}
\end{aligned}$$

3. **MaxSave Algorithm - First Round**:

- The first node selected for direct vaccination is node 4, as it saves the most nodes within the target list (specifically, nodes 1, 6, and 8).
- With a starting budget of 2, a second node can be vaccinated in the same round. The second node chosen is node 2, as it is the only one capable of saving itself. Thus, node 2 becomes the second choice in the vaccination strategy.

From this point, we can observe that to save all nodes in the target list, a minimum budget of 2 is required. This is because node 2, which must be saved, is the second choice in the Spreading MaxSave Algorithm. If node 2 is not vaccinated in the first round, it will be infected due to its proximity (distance 1) to the source node. Therefore, the minimum budget is 2, and the vaccination strategy will be: $\{(4,1),(2,1),(7,2),(8,2)\}$.

Now, let's compare this with the **Spreading Heuristic MinBudget Algorithm**:

1. **Initial Step**: Begin by running the Heuristic MaxSave algorithm with the same parameters as the MinBudget input. The budget is again set to half the number of the source node's neighbors, which is 2.
2. **Heuristic MaxSave Algorithm - First Round**:
   - The first node selected for direct vaccination is node 2. This is because, in the heuristic algorithm, we prioritize target nodes, followed by those with the most targeted neighbors, and finally, those with the most neighbors overall. Node 2, being a target and a neighbor to the source node, is selected first.
   - With the remaining budget, the second node chosen is node 4, as it saves both nodes 1 and 6, which are targets needing protection.

At this point, the infection will spread to node 5. During the second round, the vaccination will extend to nodes 1, 3, and 6. The only available option for direct vaccination is node 7, as it is the only vulnerable neighbor of node 5. Node 7 is vaccinated, and the algorithm concludes. Since we successfully saved all target nodes, we can adjust the budget to 1 and repeat the process:

1. **First Round**: The process remains the same, but only node 2 is vaccinated.
   - The virus then spreads to nodes 4 and 5.
   - At the start of the second round, the vaccination spreads from node 2 to nodes 1 and 3.
   - We then examine the neighbors of the infected nodes and select the best candidates for direct vaccination. Node 6, being a target and a neighbor to both nodes 4 and 5, is chosen for vaccination. By vaccinating node 6, node 8 is automatically saved as well since it is a neighbor to node 6, and in the next round, the vaccination will spread to it.

This example illustrates that with a budget of 1, all targeted nodes can be saved, with the vaccination strategy being $\{(2,1),(6,2)\}$.

In conclusion, the heuristic algorithm demonstrates better decision-making when the objective is to save all targeted nodes.

4 Methodology

To rigorously assess the effectiveness of our proposed heuristics in comparison to the article algorithms, we precisely structured our comparative analysis into three distinct groups, each representing different categories of algorithms. This approach allowed us to highlight the specific strengths and weaknesses of each algorithm, providing a comprehensive understanding of their performance in various scenarios.

1. **First Group**: This group involved a direct comparison between the *Spreading MaxSave* algorithm and its counterpart, the *Spreading Heuristic MaxSave* algorithm. Both algorithms are designed to maximize the number of nodes saved within a network, but the heuristic approach introduces an additional layer of optimization intended to enhance performance under certain conditions.

2. **Second Group**: In the second group, we focused on the *Spreading MinBudget* algorithm and the *Heuristic Spreading MinBudget* algorithm. These algorithms aim to minimize the budget required to achieve a specific objective within the network, with the heuristic version again offering potential improvements through more efficient resource allocation.

3. **Third Group**: The third group presented a more complex comparison, encompassing the *Nonspreading MinBudget*, *Nonspreading Dirlay MinBudget*, and *Heuristic Nonspreading MinBudget* algorithms. Unlike the first two groups, these algorithms do not involve spreading processes, and their comparison was further complicated by the fact that the *Nonspreading Dirlay MinBudget* algorithm is specifically designed for Directed Acyclic Graphs (DAGs).

For the first two groups, we conducted extensive simulations on randomly generated graphs to evaluate the performance of each algorithm. These graphs varied in terms of node count, with configurations of 20, 50, 100, 200, and 400 nodes being used to ensure a broad spectrum of network sizes. Additionally, the probability of edge creation within these graphs was systematically varied across several levels: 0.1, 0.25, 0.5, and 0.8. By manipulating these parameters, we were able to simulate a wide range of network conditions, allowing for a robust and comprehensive comparison of algorithmic performance across different scenarios.

In the third group, our comparative analysis required a different approach due to the unique characteristics of the *Nonspreading Dirlay MinBudget* algorithm, which is exclusively applicable to DAGs. To provide a fair comparison, we generated random

DAGs with varying numbers of nodes (10, 50, 100, 200, 400) and different layer depths (2, 5, 7, 10, 15). This methodological choice ensured that all algorithms were evaluated within the context of their intended application, thereby providing meaningful insights into their relative performance.

Parameters for Evaluation

The criteria used to determine the superiority of an algorithm were tailored to the specific goals of each algorithm type:

- **MaxSave Algorithms**: For algorithms in the *MaxSave* category, the primary metric of success was the number of nodes that the algorithm successfully preserved from a predefined target list. This parameter is crucial because it directly reflects the algorithm's ability to protect key elements within a network, which is often a critical objective in practical applications. The algorithm that achieved the highest node preservation rate was considered the most effective.

- **MinBudget Algorithms**: In the case of *MinBudget* algorithms, both spreading and nonspreading variants, the key performance indicator was the budget required to reach the algorithm's goal. The budget, in this context, refers to the resources—such as time, computational power, or other relevant metrics needed to execute the algorithm's strategy. The algorithm that accomplished its objective with the minimal budget was deemed superior, as this reflects greater efficiency and practicality, particularly in environments where resources are limited.

Comparative Analysis and Implications

Our structured comparison across these algorithm groups, with clearly defined evaluation parameters, not only highlights the relative strengths and limitations of each algorithm but also underscores the potential improvements introduced by our proposed method. By varying the structure of the graphs and the probabilities of edge creation, we were able to simulate a diverse array of real-world scenarios, providing a thorough and nuanced assessment of algorithm performance. This detailed examination of both node preservation in *MaxSave* algorithms and budget efficiency in *MinBudget* algorithms offers valuable insights into the practical applications of these algorithms.

Furthermore, the findings from this comparative analysis can guide future research and development in the field, offering a foundation upon which more advanced algorithms can be built. By demonstrating the specific conditions under which certain algorithms excel, we provide a roadmap for optimizing algorithmic strategies in various contexts, ultimately contributing to the advancement of the field.

5 Results and Analysis

5.1 Comparison with Original Algorithms

We compare our heuristic approaches against the original algorithms for both the MinBudget and MaxSave problems in non-spreading and spreading vaccination models. Figures 2-7 present these comparisons across different graph sizes and graph density.

5.1.1 MinBudget Problem (Non-Spreading Model)

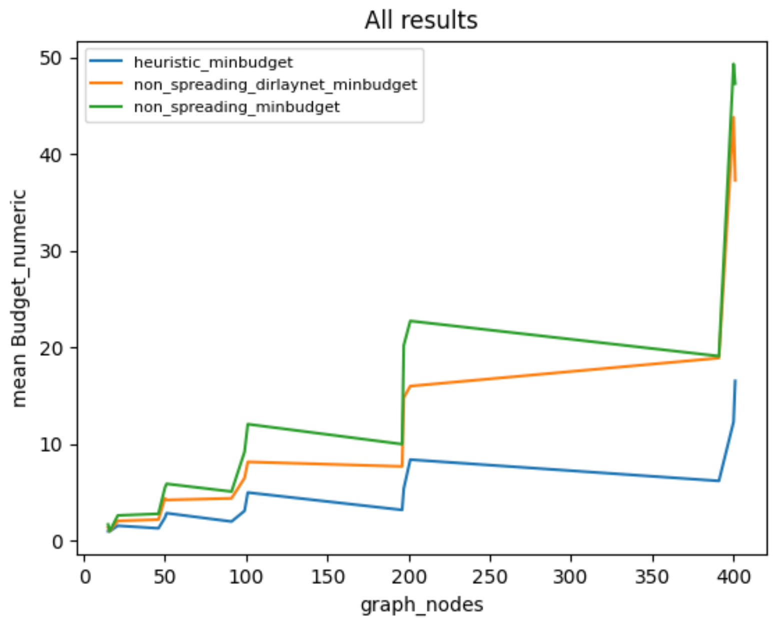


Figure 2: Comparison of MinBudget algorithms for non-spreading vaccination models

For the MinBudget problem in the non-spreading model (Figure 2):

- Our heuristic approach consistently required a lower budget compared to the original algorithm across all graph sizes and edge probabilities.
- The performance gap widened as the number of nodes increased, particularly for higher edge probabilities.

5.1.2 MinBudget Problem (Spreading Model)

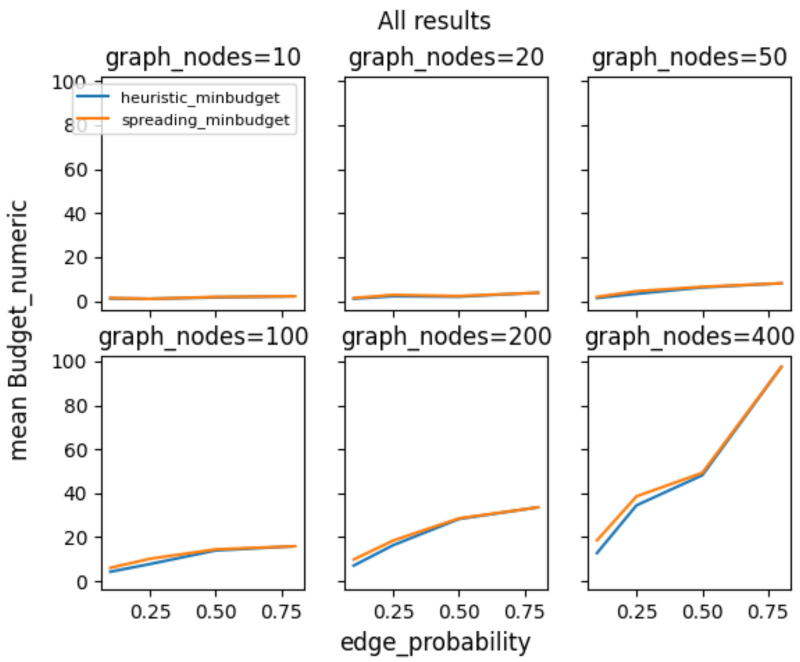


Figure 3: Comparison of MinBudget algorithms for spreading vaccination models

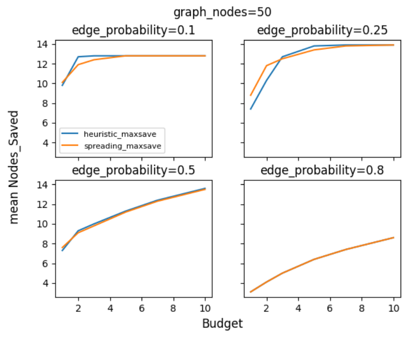
For the MinBudget problem in the spreading vaccination model (Figure 3):

- Our heuristic showed lower or comparable budget requirements to the original algorithm, with improved performance as graph size increased.
- For larger graphs (200-400 nodes), our heuristic outperformed the original algorithm, especially at lower edge probabilities.
- At higher edge probabilities (above 0.5), both algorithms showed similar budget requirements.

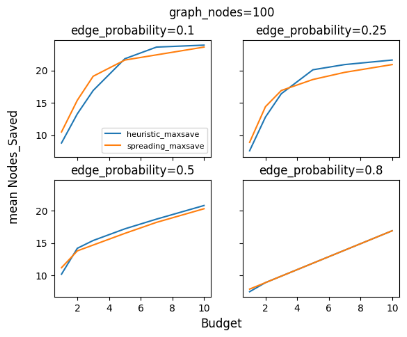
5.1.3 MaxSave Problem

For the MaxSave problem (Figure4):

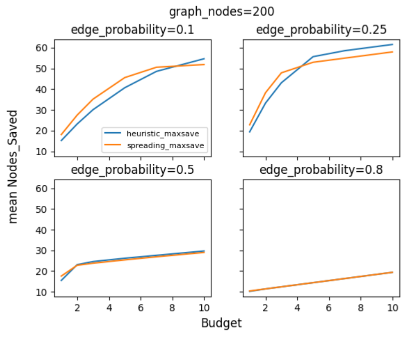
- At lower edge probabilities (0.1, 0.25), our heuristic generally under-performed the original algorithm, saving fewer nodes across all budget levels.
- At higher edge probabilities (0.5, 0.8), the performance difference narrowed, with both algorithms showing similar results in denser graphs.
- In some instances, our heuristic slightly outperformed the original algorithm in denser graphs.



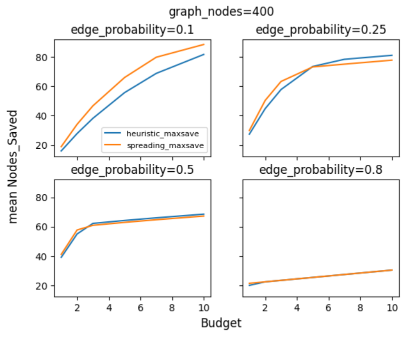
[a]



[b]



[c]



[d]

Figure 4: (a) 50 nodes (b) 100 nodes (c) 200 nodes (d) 400 nodes

5.2 Explaining the Heuristic's Performance

The success of our heuristic approaches can be attributed to their focus on local, adaptive decision-making. By prioritizing the vaccination of immediate neighbors of infected nodes, our heuristic addresses the most pressing threats first, potentially reducing the overall budget needed for saving a particular group of nodes.

Our findings suggest that the heuristic algorithms are more suitable for the MinBudget method, as they make decisions based on the current state of the network at each time step. For the MaxSave method, the heuristic algorithm may miss nodes that aren't immediate neighbors but could save more nodes in the future, explaining its lower performance in sparse graphs.

6 Conclusion

This work extends the approximation algorithms for the Firefighter problem presented by Anshelevich et al.[1], focusing on both spreading and non-spreading models. Our primary contributions include:

- Detailed implementations and pseudo-code for the original algorithms.
- Development of novel heuristic approaches for both MaxSave and MinBudget problems in spreading and non-spreading models.
- Extensive comparative testing on randomly generated graphs, varying in size from 20 to 400 nodes, and with edge probabilities of 0.1, 0.25, 0.5, and 0.8.

In conclusion, our heuristic approach demonstrates significant improvements over the original algorithms, particularly for larger graphs and in scenarios with lower edge probabilities. Its effectiveness in minimizing the budget required to save all target nodes (MinBudget) and its competitive performance in maximizing saved nodes (MaxSave) in denser graphs highlight its potential for practical applications.

Future work could explore further refinements to these heuristic strategies, potentially incorporating more sophisticated predictive elements while maintaining the advantages of local, adaptive decision-making.

References

[1] Ameya Hate Chaitanya Swamy Elliot Anshelevich, Deeparnab Chakrabarty. Approximability of the firefighter problem, computing cuts over time. *Algorithmica*, 2012.

[2] John Kemeny. Reduction from node capacity to edge capacity, 2023. Stack Exchange discussion.