Assignment 2

Team Name: Jim Beams
Shreyash Padeer(2024AIB2800)
Piyush Miglani(2024AIB2282)
Somesh Agrawal(2024AIB2292)

April 1, 2025

1 Task 1

Problem. Given a graph $G = (V, E, p)$, where $p : E \to (0, 1]$, and a budget k, determine an initial *seed set* A_0 of infected nodes such that

- $|A_0| = k$, and
- $\mathbb{E}(|A_\infty|)$ is maximized

where A_∞ is the set of infected nodes at time $t = \infty$, and $\mathbb{E}(|A_\infty|)$ is the *expected* size of such a set.

Reduce this problem to a well-known NP hard problem.
The expected size of the final influenced set is given by:

$$\mathbb{E}(|A_\infty|) = \frac{1}{n} \sum_{j=1}^{n} |A_\infty^{(j)}|.$$

At time $t = \infty$, the stochastic edges determine the final state of the graph. If the activation probability exceeds the edge probability, the edge is considered active. This allows us to represent the final graph as a directed unweighted graph.

The problem can be reduced to the Maximum Coverage problem, which is closely related to: **Set Cover Problem** :

- Given a universe U and a collection of subsets $C = \{S_1, S_2, \ldots, S_m\}$,
- The goal is to select k subsets from C such that their union has the maximum size.

The given influence maximization problem can be reduced to the Set Cover problem as follows:

- Let U represent the set of all nodes in the given graph.

- For each subset S_j, introduce a corresponding node v_j.
- Define an edge between v_j and u_i with probability 1 if there exists a directed edge or path between the corresponding nodes in the original problem.
- Set a budget k for seed selection.

If the nodes $\{v_{j_1}, v_{j_2}, \ldots, v_{j_k}\}$ are selected as seeds, then the corresponding subsets $\{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$ form a solution to the Maximum Coverage problem.

Consider a bipartite graph where:

- The left partition consists of seed nodes(of size m).
- The right partition consists of elements from U.
- An edge exists between a seed node and an element if the element is contained within the corresponding subset.

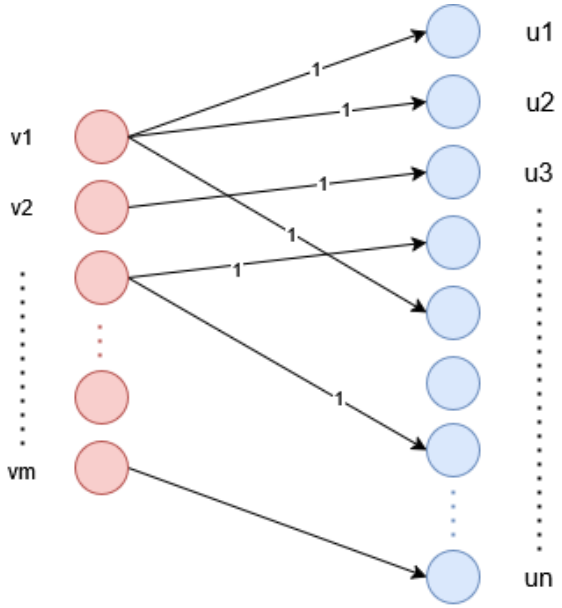

Figure 1: Bipartite Graph between k nodes on left and all nodes in a graph.

A size-k set cover in the Set Cover problem corresponds to selecting k seed nodes that influence at least $k + n$ nodes in the influence maximization problem. The optimal choice of k nodes always involves selecting from the left partition since selecting a right-side node can be replaced by its left-side neighbor to achieve greater influence.

The influence of selecting k seed nodes is given by:

$$\left| \bigcup_i S_{j_i} \right| + k.$$

Thus, maximizing influence is equivalent to maximizing coverage.

Since Maximum Coverage is NP-hard, and we have a polynomial-time reduction to the Independent Cascade (IC) Influence Maximization problem, it follows that the latter is also NP-hard.

2 Task 2

The naive solution would be to use a greedy approximation.

Given a collection of sets $\mathcal{C}$ and a universal set U, the marginal gain in coverage from adding a set $S \in \mathcal{C}$ to the current selection $\mathcal{S}$ is defined as:

$$\sigma(S, \mathcal{S}) = |S \cup \mathcal{S}| - |\mathcal{S}|$$

The greedy algorithm iterates over the sets, computing $\sigma(S, \mathcal{S})$ for each set S in every iteration, and selecting the set that provides the maximum gain. However, some constraints must be considered:

- The marginal gain $\sigma(S, \mathcal{S})$ should be strictly positive.
- The total number of selected sets should not exceed a given budget k.
- The algorithm should terminate once no further gain can be achieved.

Based on these conditions, we define the following greedy algorithm.

Algorithm 1 Greedy Algorithm

1: Set $\mathcal{S} \leftarrow \emptyset$, $S^0 \leftarrow \emptyset$, $L \leftarrow A$. L stands for leftover.
2: **Iterate** starting with $t = 0$ **while** $t < k$.
   1. Compute $\sigma^t(x) = \sigma(S^t, x)$ for all $x \in L$.
   2. Let $x^* = \arg\max_{x \in L} \sigma^t(x)$.
   3. **If** $\sigma^t(x^*) \leq 0$: **stop**.
   4. **Else**: $S^{t+1} \leftarrow S^t \cup \{x\}$ and $L \leftarrow L \setminus \{x\}$.
   5. $t \leftarrow t + 1$.
   6. $S \leftarrow S^{t+1}$.
3: $\mathcal{S}$ contains the greedy solution.

Even though there do not exist any polynomial-time solutions that provide the optimum results, we may still use a greedy approach to find a close approximation. How close is the approximation, As mentioned in the paper by Kempe 2003(mentioned in reference). The approximate bound on the value if greedy approximation is given as follows.

$$\frac{ValueofGreedyApproximation}{ValueofOptimalApproximation} \geq 1 - (\frac{k-1}{k}) \geq \frac{e-1}{e}$$

But the above algorithm was taking a lot of time due to the large number of nodes, and we decided to extend the algorithm using heuristics. Here we are using the PageRank Algorithm, where large weights are given to the nodes with higher outgoing links. For this case, we had to reverse the algorithm, since page rank works for incoming links.

PageRank Algorithm

We use the PageRank algorithm to identify the top N nodes that maximize the spread in a network. The PageRank algorithm constructs a normalized transition matrix based on edge weights to model the influence of nodes. In this case, we will be using outgoing links instead of incoming links as the part of the model.

2.1 Damping Factor and Jumping

To prevent the algorithm from getting trapped in loops and ensure convergence, PageRank introduces:

- A **damping factor** β, which controls the probability of following outgoing edges.
- A **random jump probability** $(1-\beta)$, allowing occasional jumps to any node, ensuring aperiodicity.

2.2 Mathematical Formulation

The PageRank score for any node u in the graph is given by:

$$PR(u) = (1-\beta) + \beta \sum_{v \in \mathcal{N}(u)} \frac{PR(v)}{L(v)}$$

where:

- β is the damping factor (typically set to 0.85).
- $PR(v)$ represents the PageRank score of node v.
- $L(v)$ denotes the number of outgoing edges from node v.
- The summation runs over all nodes v that have an edge pointing to u.

The PageRank vector is the solution to the equation:

$$PR = \beta(M)(PR) + (1-\beta)e$$

where M is the transition matrix, and e is a uniform teleportation vector.

The time complexity of the PageRank algorithm depends on the nature of the transition matrix T. Each iteration involves matrix-vector multiplication, leading to the following complexities:

2.3 Pseuodocode

Algorithm 2 PageRank Algorithm

$d \leftarrow 0.15$ ▷ Damping factor
$N \leftarrow$ Number of nodes in G
for each node v in G **do**
    Initialize $Rank(v) \leftarrow \frac{1}{N}$
end for
while $n > 0$ **do**
    for each node v in G with links from u_i and u_j with no inlinks **do**
        $Rank(v) \leftarrow \frac{d}{N} + (1-d) \cdot \left(\sum \frac{Rank(u)}{\text{InDegree}(u)} + \sum \frac{Rank(u)}{N} \right)$
    end for
    $n \leftarrow n - 1$
end while
return Rank

Algorithm 3 Base Algorithm

$k \leftarrow 50$ ▷ Seed Nodes
$d \leftarrow 0.15$ ▷ Damping factor
$G \leftarrow$ Given Graph
$Nodes \leftarrow$ PageRank(G,d)
Sort($Nodes$) in descending order
return topKnodes
Here **topKnodes** holds the top nodes sorted on the order of decreasing ranks.

2.4 Time Complexity of the Algorithm

- In the **worst case**, when the matrix is dense, the time complexity is:

$$O(I \cdot N^2)$$

  where:

  - I is the number of iterations,
  - N is the number of nodes in the graph.

- Since T is typically sparse, the **average case** time complexity is:

$$O(I \cdot k)$$

  where:

  - k is the number of non-zero entries in T.

- Time Complexity for Sorting and Selecting Top K is

$$O(N \cdot logN)$$

Approximate Bound of the algorithm will be similar to Greedy Algorithm as we are only working with the nodes providing the best nodes at each time which is similar to mentioned in the paper by Kempe(2003) for the Greedy Approximation.

3 Task 3

3.1 Hypothetical Dataset Example

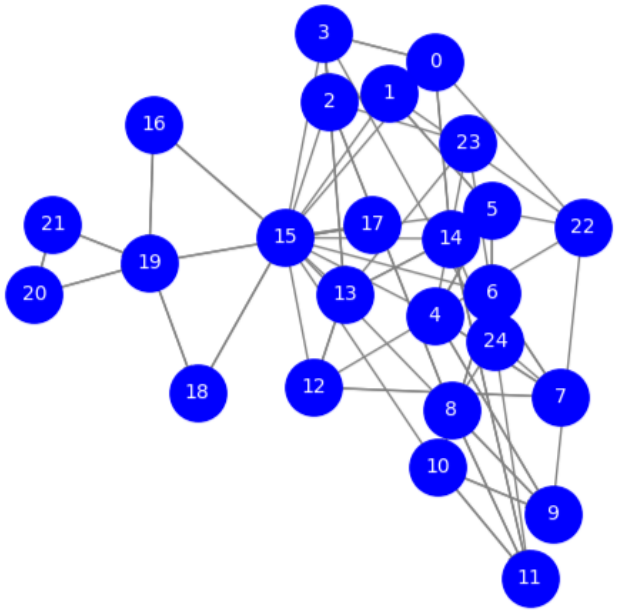

Figure 2: Original Hypothetical Dataset

3.2 Sampling of Edges using Coin Toss

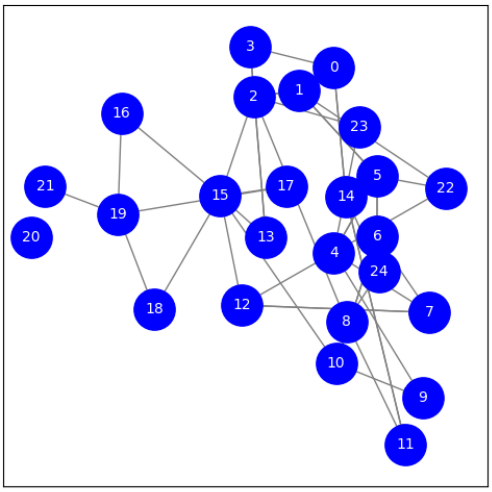

Figure 3: Sampled Graph

3.3 Spread with Optimal Nodes

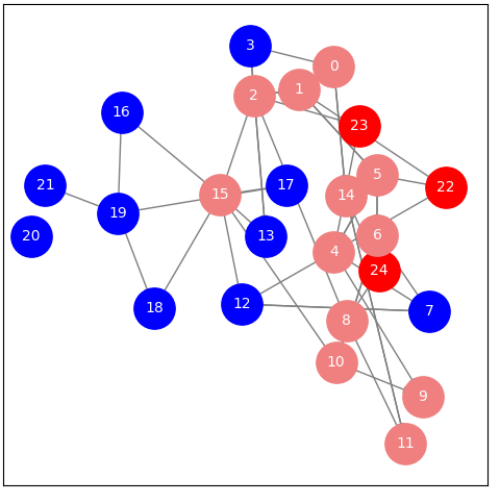

Figure 4: Spread with Optimal Seed Set

3.4 Spread with PageRank Selected Nodes

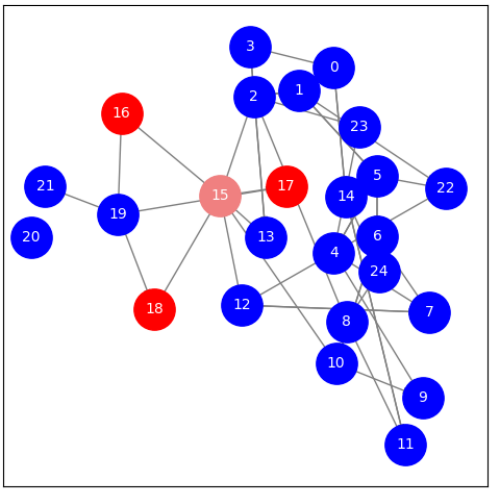

Figure 5: Spread with Sub Optimal Seed Set

3.5 Explaination

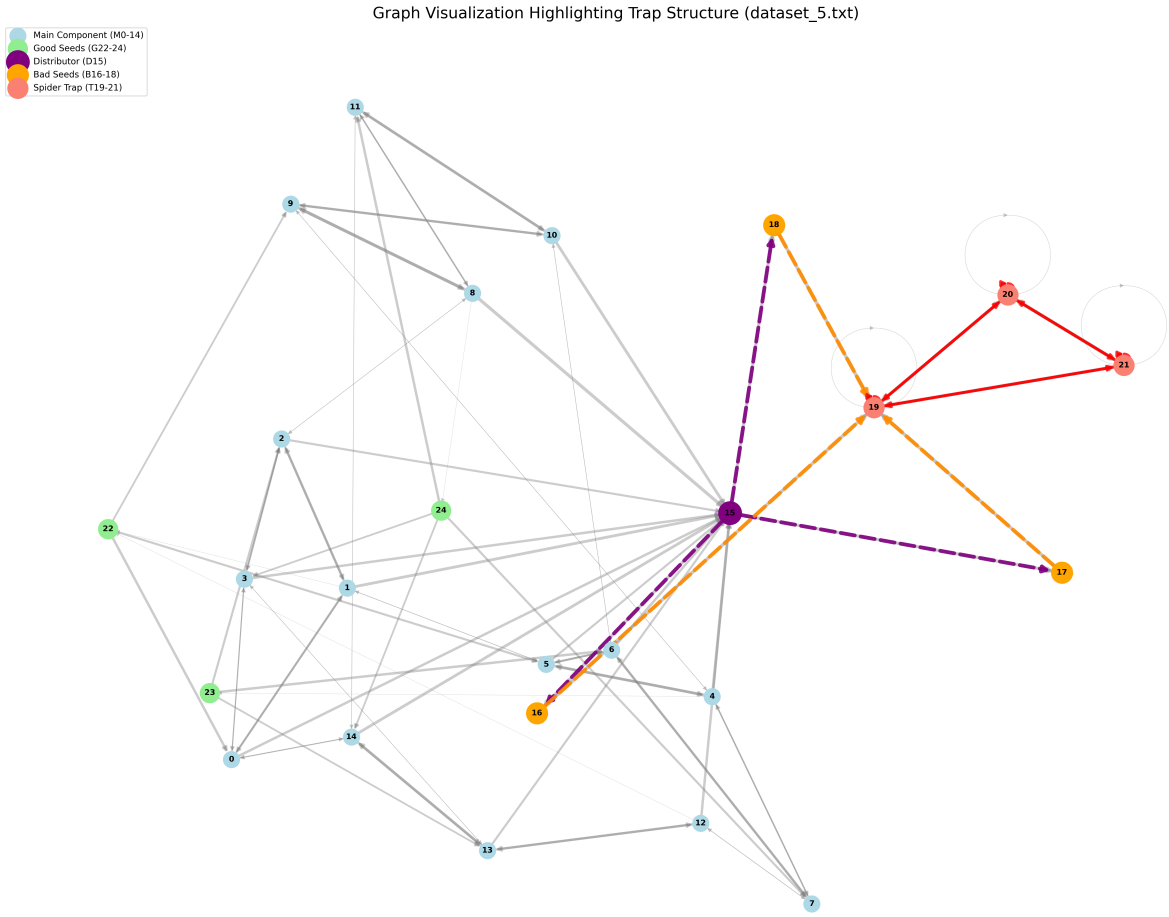

Figure 6: Commponents in the Original graph

Here are the components in the above graph:

- **Main Component (M)**: Nodes {0-14}, forming the core of the network.
- **Distributor (D15)**: A node with a high original in-degree, receiving multiple links from the Main Component.
- **Bad Seeds (B)**: Nodes {16, 17, 18}, strongly linked to D15 and forming a potential misclassification in PageRank.
- **Spider Trap (Trap)**: Nodes {19, 20, 21} connected to the Bad Seeds, forming a closed loop that prevents effective influence spread.
- **Good Seeds (G)**: Nodes {22, 23, 24}, which bypass the trap structure and link into the Main Component with reasonable probability.

3.5.1 PageRank on Reversed Graph

- **D15** accumulates high PageRank due to its strong out-degree in the reversed graph.
- **B16, B17, B18** receive a significant PageRank from D15, causing them to rank among the top nodes.
- The trap nodes **T19, T20, T21** have no outgoing links, making them terminal nodes with zero contribution to spreading influence.

3.5.2 PageRank's Seed Selection and Spread Analysis

The PageRank algorithm is expected to select seeds based on ranking, often favoring {16, 17, 18}. However, this selection leads to inefficient spread due to the following reasons:

- Nodes {16, 17, 18} mostly direct influence into the trap {19, 20, 21}, where spread halts.
- The total spread remains moderate due to the wasted influence on the Bad Seeds and the trap.

3.5.3 Optimal Seed Selection and Spread

An improved selection would include {22, 23, 24}, leading to a more efficient influence distribution:

- **Good Seeds (22, 23, 24)** directly inject influence into well-connected nodes in {0-14}.
- This selection ensures influence propagates effectively across the largest portion of the network.

3.5.4 Comparison of Spread Scores

The difference in spread efficiency between the algorithm's default selection and the optimal set is substantial:

- The algorithm-selected seeds result in limited influence due to the trap mechanism.
- The optimal seeds ensure a broader spread within the Main Component, significantly improving total reach.
- This highlights the limitation of using PageRank alone for seed selection in influence maximization problems.

We can also observe from the final experiments done in Section 3.1 to Section 3.4, Section 3.2 in the simulation done on the original Graph. The optimal nodes given in Figure 4 for this case have given us a wider spread, while in Figure 5, the PageRank has given us these suboptimal nodes due to Spider traps and deadnodes, leading to a very spread in comparison to the optimal seeds.

4 References

- https://web.iitd.ac.in/ aiz218322/blog/celf/
- David Kempe, Jon Kleinberg, and Éva Tardos. 2003. 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). Association for Computing Machinery, New York, NY, USA, 137–146. https://doi.org/10.1145/956750.956769
- https://churchill-aloha.medium.com/pagerank-algorithm-explanation-code-2fb6c0389bed
- https://swang21.medium.com/what-is-googles-pagerank-algorithm-d0ac17cbc167
- Social Network Analysis Course (ELL880) Slides