COL 761 HW2

Hasit Nanda: 2024VST9015, Subhojit Ghosal: 2022MT61976, Arnab Goyal: 2022MT61963

1 Task 1: NP-hardness of Virality Problem via Reduction from Set Cover

1.1 Problem Statement

We are given a directed graph $G = (V, E, p)$ where each edge $(u, v) \in E$ is associated with an infection probability $p(u, v) \in (0, 1]$. The goal is to choose a seed set $A_0 \subseteq V$ of size k such that the expected number of eventually infected nodes, $\sigma(A_0)$, is maximized. In this task, we convert the problem to a decision version and show that it is NP-hard by reducing the Set Cover Problem to the Virality Problem.

1.2 Reduction from Set Cover

The *Set Cover* problem is as follows:

- **Input:** A ground set $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ and a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, along with an integer k.
- **Question:** Does there exist a selection of k subsets from $\mathcal{S}$ whose union equals $\mathcal{U}$?

Since Set Cover is NP-complete, a polynomial-time reduction from it will show that Virality Problem is NP-hard.

1.2.1 Graph Construction

Given an instance of Set Cover, construct a directed bipartite graph $G = (V, E)$ as follows:

1. For each subset $S_i \in \mathcal{S}$, create a node $v_i \in V$.
2. For each element $u_j \in \mathcal{U}$, create a node $w_j \in V$.
3. For every pair (S_i, u_j) with $u_j \in S_i$, add a directed edge (v_i, w_j) with infection probability

$$p(v_i, w_j) = 1.$$

Thus, the graph contains $m + n$ nodes, and all infection probabilities are deterministic.

1.2.2 Conversion to a Decision Problem

We now define the decision version of the Virality problem:

> **Decision Problem:** Given the graph $G = (V, E)$ constructed above, an integer k, and a threshold T, does there exist a seed set $A_0 \subseteq V$, with $|A_0| = k$, such that
>
> $$\sigma(A_0) \geq T\,?$$

In our construction:

- Infecting a seed node v_i (representing subset S_i) immediately contributes 1 to the total count.
- If the k chosen subsets cover the entire ground set $\mathcal{U}$, then all n element nodes will be infected.

Thus, if the chosen k subsets form a set cover, the total spread is

$$\sigma(A_0) = n + k.$$

By setting

$$T = n + k,$$

the decision problem asks:

$$\text{Does there exist a seed set } A_0 \text{ of size } k \text{ such that } \sigma(A_0) \geq n + k\,?$$

1.3 Equivalence to Set Cover: Proof of Correctness

We now prove that there exists a seed set $A_0 \subseteq V$ of size k with $\sigma(A_0) \geq n + k$ in the constructed graph if and only if there exists a set cover of $\mathcal{U}$ using k subsets from $\mathcal{S}$.

(1) Forward Direction: Suppose the given Set Cover instance is a *yes* instance; that is, there exists a collection of k subsets

$$\{S_{i_1}, S_{i_2}, \ldots, S_{i_k}\} \subseteq \mathcal{S}$$

such that

$$\bigcup_{j=1}^{k} S_{i_j} = \mathcal{U}.$$

In the graph G constructed as above, choose the seed set

$$A_0 = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\},$$

where each v_{i_j} corresponds to S_{i_j}. By the rules of our deterministic infection process (since every edge has probability 1), the following holds:

1. Each chosen seed node v_{i_j} is infected immediately, contributing exactly k infected nodes.
2. For each element $u \in \mathcal{U}$, there exists at least one subset S_{i_j} containing u, which implies there is an edge from v_{i_j} to the corresponding element node w in G. Since v_{i_j} is infected, every such element node w is infected deterministically.

Thus, all n element nodes are infected, and hence the total number of infected nodes is

$$\sigma(A_0) = n + k.$$

Since $\sigma(A_0) \geq n + k$, the decision problem returns **yes**.

(2) Reverse Direction: Conversely, suppose there exists a seed set $A_0 \subseteq V$ with $|A_0| = k$ such that

$$\sigma(A_0) \geq n + k.$$

Notice that only the subset nodes have outgoing edges that can infect element nodes. If the Virality Problem algorithm selects an element node as part of the seed set, it contributes only by its own infection and does not infect any additional nodes. Therefore, including an element node is suboptimal in maximizing the overall spread. In an optimal seed set achieving the desired spread, all k seeds will be chosen from the subset nodes. Note that:

- Each seed node in A_0 is infected by definition.
- An element node w_j is infected if and only if there is at least one seed node $v_i \in A_0$ with an edge (v_i, w_j), which is equivalent to $u_j \in S_i$.

The condition $\sigma(A_0) \geq n + k$ implies that at least n element nodes are infected in addition to the k seed nodes. Therefore, every element node w_j must be infected. For each element node w_j, there exists at least one seed node $v_i \in A_0$ such that the corresponding subset S_i satisfies $u_j \in S_i$. Consequently, the collection of subsets corresponding to the nodes in A_0 covers $\mathcal{U}$; that is,

$$\bigcup_{v_i \in A_0} S_i = \mathcal{U}.$$

Thus, a set cover using k subsets exists for the given Set Cover instance, and the decision problem returns **yes**.

1.4 Runtime Analysis of the Reduction

The reduction process involves the following steps:

1. **Node Construction:** Create m subset nodes and n element nodes, which takes $O(m + n)$ time.
2. **Edge Construction:** For each subset S_i, for every element $u_j \in S_i$, add a directed edge (v_i, w_j) with infection probability 1. If the total number of memberships (i.e., the sum of sizes of all subsets) is M, then this step takes $O(M)$ time.

Since M is at most $m \cdot n$, the overall time complexity of the reduction is $O(m + n + M)$, which is polynomial in the size of the input. Therefore, the reduction from Set Cover to the Virality decision problem runs in polynomial time.

1.5 Conclusion

Since Set Cover is NP-complete, and the above reduction is computable in polynomial time, it follows that the decision version of the Virality problem is NP-hard.

2 Task 2: Overview and Time Complexity Analysis of the CELF Algorithm for the Virality Problem

In this section, we describe the CELF (Cost-Effective Lazy Forward) algorithm [1] as applied to the virality problem and provide a detailed analysis of its time complexity. The CELF algorithm leverages the submodularity of the virality function to minimize redundant computations during seed selection.

2.1 Overview of the CELF Algorithm

Let $G = (V, E)$ be a directed social network, where V represents the set of nodes (users) and E represents the edges (connections). Let k denote the number of seed nodes to select. The algorithm proceeds as follows:

Algorithm 1 CELF_Greedy Algorithm

Require: Graph $G = (V, E)$, budget k
Ensure: Seed set S of size k

$S \leftarrow \emptyset$
Create max-heap Q sorted by marginal gain
for each node $u \in V$ **do**
    Compute initial spread $\sigma(\{u\})$
    $u.\text{mg} \leftarrow \sigma(\{u\})$
    Insert u into Q with priority $u.\text{mg}$
    $u.\text{flag} \leftarrow 0$
end for
while $|S| < k$ **do**
    Extract top node u from Q
    if $u.\text{flag} = |S|$ **then**
        Add u to S
    else
        Recompute $u.\text{mg} \leftarrow \sigma(S \cup \{u\}) - \sigma(S)$
        $u.\text{flag} \leftarrow |S|$
        Reinsert u into Q
    end if
end while
return S

The algorithm essentially works by doing the following:

- **Initialization:** Compute the initial influence spread $\sigma(\{u\})$ for each node u, store it as its *marginal gain*, and insert nodes into a max-heap Q sorted by marginal gain. Assign each node a *flag* tracking when its gain was last updated.
- **Seed Selection:** Extract the top node u from Q. If its gain was last computed for the current seed set S, add it to S. Otherwise, recompute its gain, update its flag, and reinsert it into Q.

- **Lazy Evaluation:** Nodes are only recomputed when necessary, relying on the fact that a node's marginal gain cannot increase over iterations. This reduces redundant spread calculations and improving efficiency. The process repeats until k nodes are selected.

This lazy evaluation approach significantly reduces the number of virality spread computations compared to a naive greedy algorithm.

2.2 Time Complexity Analysis

Let:

- n be the number of nodes in the network.
- k be the desired size of the seed set.
- T_{vir} be the time required to compute the virality spread (e.g., via Monte Carlo simulation) for a given seed set.

2.2.1 Initialization

During initialization, the algorithm computes the virality spread for each node $u \in V$:

$$u.\text{mg} = \sigma(\{u\}),$$

which takes $O(n \cdot T_{\text{vir}})$ time. In addition, inserting all n nodes into a max-heap takes $O(n \log n)$ time.

2.2.2 Iterative Seed Selection

In each iteration, the algorithm extracts the top node from the priority queue and checks if its marginal gain is current. In the worst-case scenario, each of the n nodes might have their marginal gain recomputed in each of the k iterations. Hence, the worst-case cost for the iterative phase is:

$$O(nk \cdot T_{\text{vir}}).$$

Furthermore, each recomputation requires updating the priority queue, which incurs an additional overhead of $O(\log n)$ per update, contributing:

$$O(nk \log n).$$

NUMBER OF MONTE CARLO SIMULATIONS:

The number of Mote Carlo Simulations play a very important role in time taken and spread achieved.
Theoretically if monte carlo simulation gives exact $E(A_\infty)$, then with the insertion of each seed, the marginal gain got decreases monotonically.
If we do 1 simulation then variance in our expected $E(A_\infty)$ is $N/4$, where N is number of nodes (assuming number of nodes infected by monte carlo simulation follows binomial distribution). If we do M monte carlo simulations and take the average spread then variance is $\frac{N}{4M}$, as each monte carlo simulation is independent of each other and follows the same distribution (iid).
We have N 15000.

No. of Mone carlo Sim	Variance in 1 sim	Time taken on dataset 1
100	40	1 mins
1000	4	5 mins
10000	0.5	40 mins

Table 1: Analysis for different M

Dataset 2 is much more dense than dataset 1, which takes much more time. To deal with this we employ dynamic value of number of monte carlo simulations. Ideally marginal spread should decrease, so we start with a $M = InitialItr$ value, and whenever we find marginal spread non-decreasing we increase M by 500 to maintain correctness with an upper threshold of 10k. We analyse that this is both time and accuracy efficent.

2.2.3 Overall Complexity

Combining the initialization and iterative phases, the worst-case overall time complexity of the CELF algorithm is:

$$O(n \cdot T_{\text{vir}}) + O(n \log n) + O\left(nk \cdot (T_{\text{vir}} + \log n)\right).$$

For large n and k, the dominant term is:

$$O(nk \cdot T_{\text{vir}} + nk \log n).$$

Based on the papers we have read, we note that in practice the algorithm performs significantly better than the asymptotic bound by reducing redundant computations, yielding an average-case performance that is much better than the worst-case bound.

2.2.4 Approximation Bound of the Algorithm

Theorem 1 (Kempe, Kleinberg, and Tardos, 2003)**.** *The problem of selecting k nodes to maximize influence under the Independent Cascade and Linear Threshold models is NP-hard. Moreover, the function $\sigma(S)$, which represents the expected number of influenced nodes given a seed set S, is monotone and submodular. Therefore, a simple greedy algorithm that iteratively adds the node with the highest marginal gain achieves an approximation factor of at least $(1 - 1/e) \approx 0.632$ of the optimal solution.*

The **proof** of this theorem relies on the submodularity of the influence function and follows from the standard result in submodular maximization under a cardinality constraint. We take the proof of this theorem from Kempe, Kleinberg, and Tardos, 2003. [1]

3 Task 3: Counterexample for Greedy Algorithm

In this section, we present a deterministic counterexample that demonstrates how the CELF_Greedy algorithm can select a sub-optimal seed set for the virality problem. In our example, the virality function $f(S)$ is defined as the total number of activated nodes (including the seed nodes).

3.1 Graph Description

Consider a simple bipartite graph with nodes A, B, C, 1, 2, 3, 4, 5, 6 and the edges are as following (all with weight probability of 1):

- **Node** A**:** Activates nodes 1, 2, 3
- **Node** B**:** Activates nodes 2, 3, 4, 5
- **Node** C**:** Activates nodes 4, 5, 6

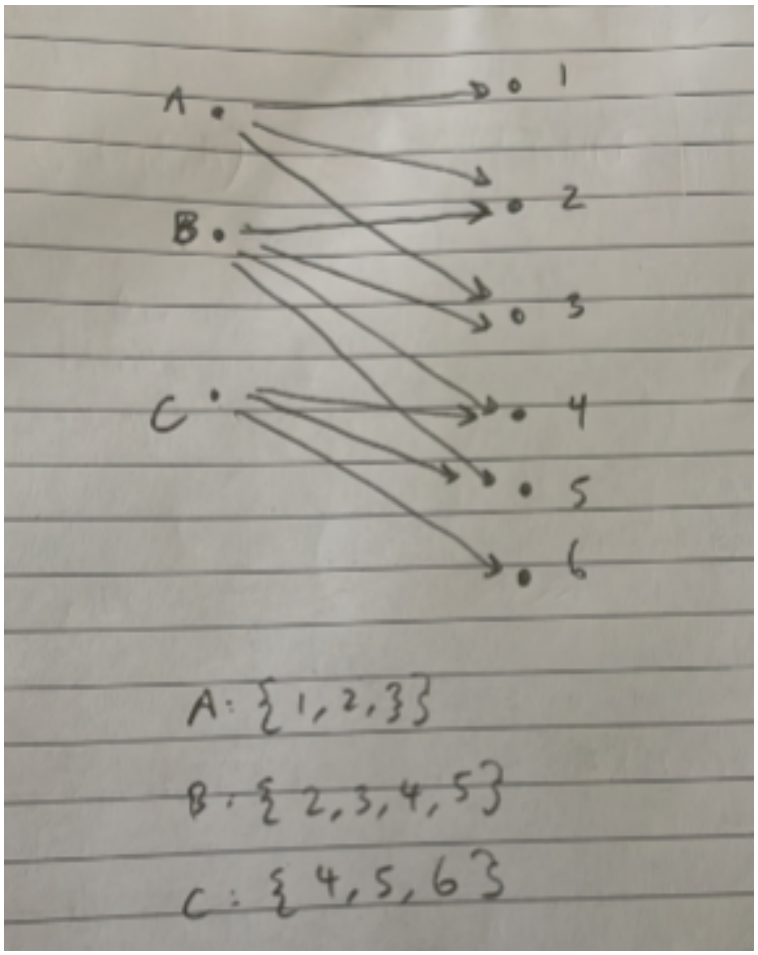

Figure 1: Deterministic counterexample illustrating that the greedy selection of node A leads to a sub-optimal seed set.

3.2 Seed Set Evaluation

When choosing a seed set of size $k = 2$, we evaluate the total virality as follows:

- $f(\{A, B\}) = 7$ since nodes 1-5 are deterministically infected in addition to A and B.
- $f(\{B, C\}) = 7$ since nodes 1-5 are deterministically infected in addition to B and C.
- $f(\{A, C\}) = 8$, since nodes 1-6 are infected in addition to A and C.

Thus, the optimal seed set is $\{A, C\}$ with a total virality of 8, while the CELF_Greedy algorithm, will select B first due to its higher individual virality (5), leading to a sub-optimal overall result.

4 Task 4: Running Algorithms on Provided Dataset

Here we provide a brief overview and analysis of the given datasets.

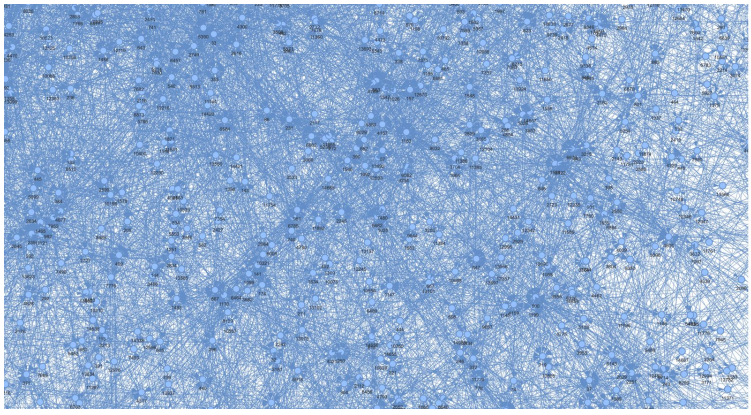

Figure 2: Dataset Zoomed

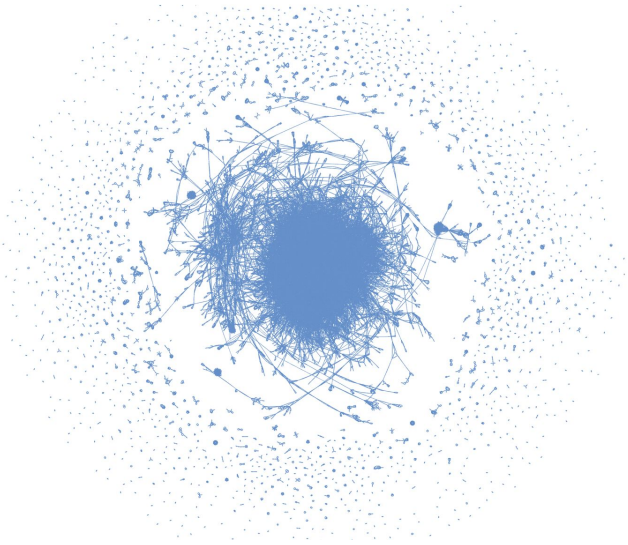

Figure 3: Dataset Unzoomed

From the graph plots, we observe that the graph exhibits disconnected components, most of which have very low probabilities of connectivity. As a result, even with a seed set of size 50, the influence propagation remains limited, infecting only approximately 900 individuals.

5 References

References

[1] David Kempe, Jon Kleinberg, and Eva 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*, pages 137–146. ACM, 2003.