COL761 – Homework 2 Report

The Virality Problem: Optimizing Social Contagion

Team:

Arnav Raj [2022CS51652]
Ayush Gupta [2022CS11114]
Arpit Agrawal [2022CS11612]

April 2, 2025

Abstract

This report addresses three main tasks:

1. A rigorous proof of NP-hardness for the influence maximization (or virality) problem under the Independent Cascade (IC) model.
2. The design and analysis of a greedy approximation algorithm to select k seed nodes to maximize the expected spread of infection.
3. A construction of a hypothetical dataset along with a refined example that illustrates why the greedy algorithm can yield a sub-optimal seed selection under the IC model.

We follow the framework outlined in Kempe, Kleinberg, and Tardos (KDD 2003) for the NP-hardness proof, then detail our algorithm and its complexity, and finally present an example with visual aids to demonstrate the limitations of the greedy approach in stochastic settings.

1 Task 1: NP-Hardness of the Virality/Influence Maximization Problem

1.1 Model Setup and Statement of the Problem

We have a directed graph $G = (V, E)$ whose nodes represent people (or entities), and each directed edge $(u, v) \in E$ has an associated probability $p_{u,v} \in (0, 1]$ indicating the likelihood that infection (or influence) spreads from node u to node v once u becomes infected.

> *Given a directed graph $G = (V, E)$ with edge probabilities $\{p_{u,v}\}$, and a budget k, choose an initial "seed set" $S \subseteq V$ of k nodes to infect. The infection then spreads according to a discrete-time diffusion model — in our case, the Independent Cascade (IC) model. The goal is to find a seed set S of size k that maximizes the expected total number of infected nodes.*

Under the **Independent Cascade (IC) Model**, when a node u becomes infected at time t, it is given a single opportunity (in step $t+1$) to infect each of its out-neighbors v. The attempt succeeds

with probability $p_{u,v}$ (independent of past events), and once u has attempted all its neighbors, it never tries again.

Let $\sigma(S)$ denote the expected number of eventually infected nodes when the seed set is S. Our objective is to maximize $\sigma(S)$ over all seed sets of size k.

1.2 NP-Hardness Proof for the IC Model

We reduce from the NP-complete problem Set Cover. Recall:

Definition 1 (Set Cover)**.** Given a ground set $U = \{u_1, \ldots, u_n\}$ and a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ with each $S_i \subseteq U$, and a budget k, the Set Cover problem asks whether there exist k sets in $\mathcal{S}$ whose union covers all of U.

Proof of NP-hardness for the IC Model. Let $(U, \mathcal{S}, k)$ be an instance of Set Cover. We construct an instance of the Influence Maximization problem under the IC model as follows:

- Create a directed bipartite graph $G = (V, E)$ with two parts:

  $V_{\text{sets}} = \{s_1, \ldots, s_m\}$ (one node per subset), $V_{\text{elems}} = \{u_1, \ldots, u_n\}$ (one node per element).

- For each set $S_i \in \mathcal{S}$ and each element $u_j \in S_i$, add a directed edge (s_i, u_j) with probability $p_{s_i,u_j} = 1$.

- There are no other edges. The budget remains k, and we may choose seeds from both V_{sets} and V_{elems}.

Claim. A seed set A of size k infects all element nodes in V_{elems} if and only if it corresponds to a set cover of size k.

- If a set cover of size k, say $\{S_{i_1}, \ldots, S_{i_k}\}$, exists, then seeding the corresponding set nodes $\{s_{i_1}, \ldots, s_{i_k}\}$ will infect every element node (since each u_j is in at least one chosen set).

- Conversely, if a seed set A of size k infects all element nodes, then the chosen set nodes must collectively cover U, as the only way to infect an element node is via an edge from a set node.

Since deciding whether there exists a set cover of size k is NP-complete, the influence maximization problem under the IC model is NP-hard. □

2 Task 2: Approximate Algorithm and Complexity Analysis

2.1 Algorithm Description

Due to NP-hardness, we adopt a greedy approximation strategy based on submodularity. Our goal is to select a seed set $S \subseteq V$, $|S| = k$, that approximately maximizes $\sigma(S)$.

The algorithm works as follows:

1. **Simulation Setup:** Generate R simulation instances of the graph by performing independent coin tosses on each edge (using its probability) to convert the stochastic graph into R deterministic instances.

2. **Seed Selection:** Start with $S = \emptyset$. For each candidate node $v \in V \setminus S$, estimate the marginal gain by computing the expected additional infections when v is added to S, averaging over the R simulation instances.

3. **Greedy Choice:** At each iteration, select the node with the highest average marginal gain and add it to S. Update the baseline spread for each simulation.

2.2 Pseudocode

Algorithm 1 details the greedy seed selection process.

Algorithm 1 Greedy Seed Selection for Influence Maximization under the IC Model

procedure GREEDYSEEDSELECTION($G = (V, E, p)$, k, R)
  Generate R simulation instances $\{G^{(s)}\}_{s=1}^{R}$ from G
  Initialize seed set $S \leftarrow \emptyset$
  Initialize baseline spread $B[s] \leftarrow 0$ for $s = 1, \ldots, R$
  for $i = 1$ to k **do**
    $bestCandidate \leftarrow$ null, $bestGain \leftarrow -\infty$
    for each candidate node $v \in V \setminus S$ **in parallel do**
      $totalGain \leftarrow 0$
      for $s = 1$ to R **do**
        $gain[s] \leftarrow$ SIMULATESPREAD($G^{(s)}$, $S \cup \{v\}$) $- B[s]$
        $totalGain \leftarrow totalGain + gain[s]$
      end for
      $avgGain \leftarrow totalGain/R$
      if $avgGain > bestGain$ **then**
        $bestGain \leftarrow avgGain$
        $bestCandidate \leftarrow v$
      end if
    end for
    $S \leftarrow S \cup \{bestCandidate\}$
    Output $bestCandidate$
    for $s = 1$ to R **in parallel do**
      $B[s] \leftarrow$ SIMULATESPREAD($G^{(s)}$, S)
    end for
  end for
  return S
end procedure

2.3 Complexity Analysis

Let $n = |V|$, $m = |E|$, and R be the number of simulation instances. Then:

- **Simulation Generation:** $O(R \cdot m)$.

- **Seed Selection:** For each of the k iterations, the algorithm evaluates at most n candidates. For each candidate, it runs R simulations, each taking $O(n+m)$ time. Hence, the cost is $O\big(k \cdot n \cdot R \cdot (n+m)\big)$.

Parallelization may reduce actual runtime, but the worst-case sequential complexity remains as above.

3 Task 3: Example Demonstrating Greedy's Suboptimality with Limited Simulations

In this section, we provide a concrete example to illustrate how the greedy algorithm can select a suboptimal seed set when based on a limited number of simulation runs, and we also comment on its computational expense for large graphs.

3.1 Example Setup: Two Disjoint Clusters

Consider a network that is divided into two disjoint clusters:

- **Cluster A:** Consists of 9 nodes that are highly interconnected with each other. All edges within Cluster A have a high probability (e.g., 0.9) of transmission.
- **Cluster B:** Consists of 4 nodes that are also highly interconnected with each other with the same high probability (0.9).

There are no edges between Cluster A and Cluster B. The optimal strategy for selecting a seed set of size $k = 2$ is to choose one node from each cluster. This would guarantee that Cluster A (9 nodes) and Cluster B (4 nodes) are both activated, potentially resulting in a global spread of up to 13 nodes.

3.2 Simulation with Limited Runs

Assume we run only 5 simulation instances to estimate the marginal gains:

- In Cluster A, even though edges are highly reliable (0.9), they are not perfect. Thus, the spread from a single seed in Cluster A might, in some simulations, cover only 4 or 5 out of 9 nodes.
- In Cluster B, a single seed might cover all 4 nodes in every simulation, because of its smaller size and high internal probabilities.

Scenario:

1. The greedy algorithm selects the first seed from Cluster A because its high internal connectivity shows a high immediate marginal gain.
2. For the second seed, the optimal choice would be a node from Cluster B, which would add 4 new nodes to the spread.

3. However, due to the limited number of simulations (only 5 runs) and the stochastic nature of the diffusion process in Cluster A, there is a chance that the estimated marginal gain from adding a second seed in Cluster A appears competitive. For instance, in some simulations the first seed may not infect all 9 nodes (e.g., infecting only 4 or 5 nodes), so the marginal gain from adding another node in Cluster A might be estimated as 5 or 4 additional nodes on average.

With only 5 simulations, these small differences in estimates may mislead the algorithm. The greedy algorithm might then select the second seed from Cluster A (because its estimated marginal gain appears high due to simulation variance) instead of selecting the seed from Cluster B. As a result, the overall spread might be significantly lower than the optimal $9 + 4 = 13$ nodes, since Cluster B remains unactivated.

3.3 Computational Expense on Large Graphs

It is also important to note that the greedy algorithm is computationally expensive. Its runtime is driven by:

$$O\Big(k \cdot n \cdot R \cdot (n + m)\Big),$$

where:

- n is the number of nodes,
- m is the number of edges,
- R is the number of simulation instances, and
- k is the seed budget.

For large graphs, running a large number of simulations (increasing R to reduce variance) is often computationally infeasible due to resource limitations. As a result, even if one tries to reduce variance by increasing R, practical limitations may force us to use a small R, which in turn leads to higher variance in the marginal gain estimates. This variance can cause the greedy algorithm to select a highly suboptimal seed set.

3.4 Visual Illustration

The following diagram represents the network structure described above:

3.5 Example with Code

For the dataset_3 generated by us present at link:

Image of the graph is also present below.

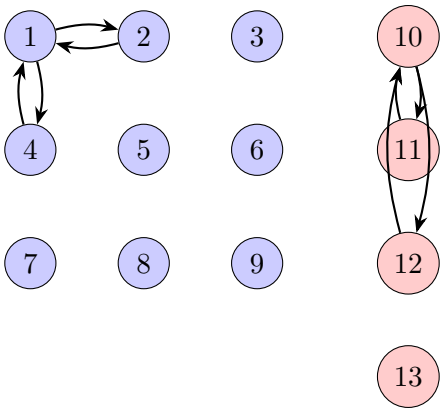

Figure 1: Network with two disjoint clusters. Cluster A (blue) has 9 nodes; Cluster B (red) has 4 nodes. There are no edges between the clusters.

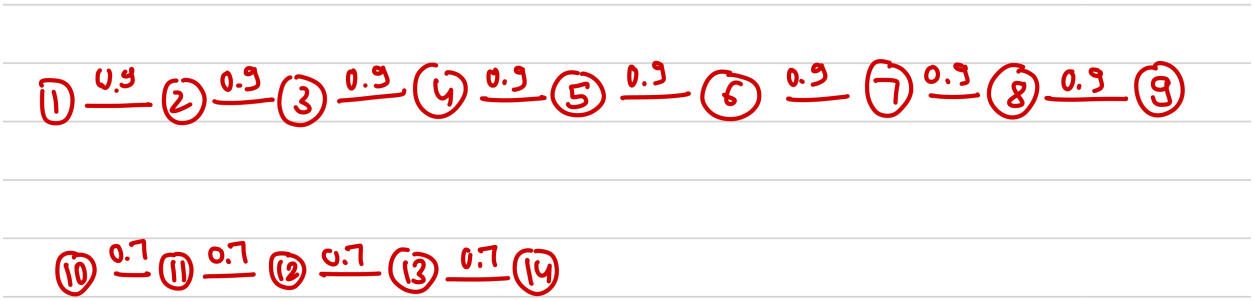

Figure 2: Graph_dataset_3

It can be seen in the screenshot that our algorithm might sometimes select both nodes from the same cluster.

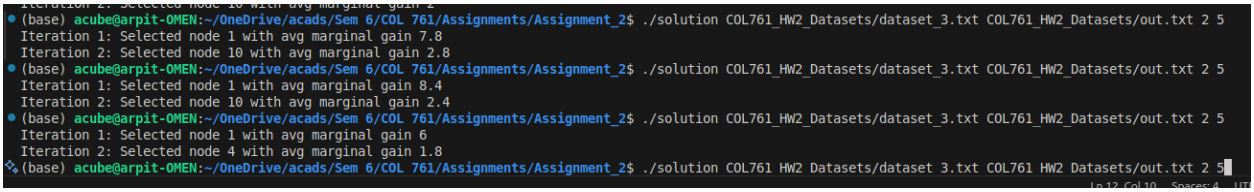

Figure 3: Example from the dataset_3.txt

3.6 Summary

- **Optimal Seed Selection:** With $k = 2$, selecting one seed from Cluster A and one from Cluster B would optimally activate 9 and 4 nodes, respectively.
- **Greedy Pitfall:** With only 5 simulations, the estimated marginal gain for a second seed in Cluster A may appear comparable to that for a node in Cluster B, leading the greedy algorithm to choose both seeds from Cluster A. This results in a lower overall spread.

- **Computational Considerations:** The high computational cost of running a large number of simulations on large graphs often forces practitioners to use a limited R, increasing the risk of suboptimal seed selection.

This example shows how the inherent randomness and limited simulation budget can cause the greedy algorithm to select a highly suboptimal set of seeds, especially in large-scale networks where computational resources are constrained.

4 Contributions

All three members of the team contributed equally to each part of the assignment. Our workflow for this assignment followed these steps:

1. We initially conducted research by exploring existing literature on the influence maximization problem. We reviewed various research papers to understand the heuristics and methodologies used in prior work.
2. Each team member attempted to implement different approaches based on the research papers, analyzing the spread results and discussing the unique aspects and focal points of each method.
3. Initially, we implemented our solutions in Python. However, due to the given time constraints, Arpit and Ayush later translated the Python code into C++ while also performing parallel optimizations to improve efficiency.
4. Arnav Raj, having prior experience with High-Performance Computing (HPC) from previous coursework, guided the team in setting up and utilizing the HPC environment for testing and optimizing our implementations.

Overall, our team maintained a well-balanced contribution, ensuring maximum learning and collaboration among all three members.

5 Conclusion

In this report, we have:

- Shown that the influence maximization problem is NP-hard under the Independent Cascade model via a reduction from Set Cover (Task 1).
- Proposed a greedy algorithm to approximate the optimal seed set, including detailed pseudocode and complexity analysis (Task 2).
- Provided a concrete example—with simulation-based numerical estimates and a visual diagram—to illustrate how limited simulation runs and computational constraints can lead the greedy algorithm to select a suboptimal seed set.

This analysis highlights both the computational challenges in optimizing social contagion and the limitations of simulation-based greedy methods in stochastic settings.

References

[1] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2003, pp. 137–146.

[2] G. Schoenebeck, B. Tao, and F.-Y. Yu, "Limitations of Greed: Influence Maximization in Undirected Networks Revisited," in *Proc. 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, Auckland, New Zealand, May 2020, pp. 1224–1233.

[3] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "CELF++: Optimizing the greedy algorithm for influence maximization in social networks," in *Proc. 20th International Conference Companion on World Wide Web (WWW 2011)*, 2011, pp. 47–48.