COL761-HW2

Team - Whocares {2024AIB2286, 2024AIB2287, 2024AIB2295}

March 2025

Task 1: NP-hardness Reduction

Problem Statement: Given a directed graph $G = (V, E, p)$, where each directed edge $e \in E$ has a probability $p(e) \in (0, 1]$ representing the likelihood of infection transmission, the objective is to select a subset of nodes $A_0 \subseteq V$ with $|A_0| = k$ to maximize the expected size of the infected nodes set $E(|A_\infty|)$ after the infection diffusion reaches completion.

0.1 Equivalence of Virality and IC version of Influence Maximization:

We show the equivalence between the given *Virality Problem (VP)* and the well-known *Influence Maximization (IM)* problem under the Independent Cascade (IC) model.

The problem of Influence Maximization under Independent Cascase can be described as -

Given a directed graph $G' = (V', E', p')$, probabilities $p' : E' \to (0, 1]$, and an integer k', select a seed set $S \subseteq V'$, $|S| = k'$, to maximize the expected spread $E(|Spread(S)|)$.

We define an instance of VP as follows:

$$
\begin{aligned}
V &= V' \\
E &= E' \\
p(e) &= p'(e), \quad \forall e \in E \\
k &= k'
\end{aligned}
$$

The diffusion rules in VP match exactly those of the IC model:

1. Each newly infected node has exactly one chance to infect each neighbor independently according to edge probability.
2. Diffusion occurs in discrete time steps.
3. Each node attempts infection only once.

Thus proof showing that IC verision of Influence Maximization is NP Hard would be sufficient to show that the given virality problem is also NP-Hard.

0.2 NP Hardness Proof of Influence Maximization

The Influence Maximization Problem is NP-Hard

Set Cover Problem

Input:

- Universe $U = \{u_1, \ldots, u_n\}$
- Family of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$, $S_i \subseteq U$
- Integer k

Goal: Decide if there exists $\mathcal{C} \subseteq \mathcal{S}$ with $|\mathcal{C}| \leq k$ covering U.

Polynomial-Time Reduction

Construct Influence Maximization instance creating a bipartite graph (G, k) :

1. **Node Creation** (Time: $O(m+n)$):
   - Left partition $L = \{x_1, \dots, x_m\}$ for sets $\mathcal{S}$
   - Right partition $R = \{u_1, \dots, u_n\}$ for universe elements
2. **Edge Construction** (Time: $O(mn)$):

   $$E = \bigcup_{1 \le i \le m u_j \in S_i} (x_i, u_j), \quad p(x_i, u_j) = 1$$

   Non-edges implicitly have $p = 0$.
3. **Budget**: k (copied directly from Set Cover input)

Reduction Complexity:

- Total nodes: $m + n$
- Total edges: At most mn
- Time complexity: $O(mn)$, polynomial in Set Cover input size

Equivalence Proof

Forward Direction (Set Cover IM): If $\exists$ set cover $\mathcal{C}$ with $|\mathcal{C}| = k$:

- Seed set $A_0 = \{x_i \in L \mid S_i \in \mathcal{C}\}$
- Influence spread: $E[|A_\infty|] = k + n$

Reverse Direction (IM Set Cover): If $\exists$ seed set A_0 with $|A_0| = k$ and $E[|A_\infty|] = k + n$:

- A_0 must contain only $x_i \in L$ (since $u_j \in R$ have no outgoing edges)
- $\{S_i \mid x_i \in A_0\}$ forms a valid set cover for U

Key Observations

1. **Optimal Seed Selection**: Any optimal solution uses only L-nodes. Choosing $u_j \in R$ gives $E[|A_\infty|] \le k$, while L-nodes give $\ge k$.
2. **Probability Generalization**: The $p = 1$ edges represent a valid special case of $p \in (0, 1]$.
3. **Verification**: Checking $E[|A_\infty|] \ge k + n$ requires solving Set Cover.

Conclusion

The Influence Maximization problem is NP-hard because it generalizes the NP-hard Set Cover problem through a polynomial-time reduction. We can say that set cover problem is NP-Hard, hence Influence maximization problem is NP-hard. This holds even for edge probabilities $p \in (0, 1]$ as our construction uses $p = 1$ cases which are contained within the general problem formulation and it is known that reduction of NP-Hard problem to any case of given problem is sufficient to show that the given problem is NP-Hard.

Task 2: Approximate Algorithm and Complexity

Approximate Algorithm: Greedy Hill-Climbing

As established in Task 1, the Influence Maximization problem under the Independent Cascade (IC) model is NP-hard. This motivates the use of approximation algorithms. A well-known result by **Kempe, Kleinberg, and Tardos (2003)** shows that the influence function $\sigma(S)$, which denotes the expected number of infected nodes starting from a seed set S, is:

- **Monotonic**: $\sigma(S) \leq \sigma(T)$ for all $S \subseteq T$
- **Submodular**: $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$ for all $S \subseteq T$

This allows us to use a greedy algorithm that provides a $(1 - 1/e)$-approximation to the optimal solution.

Greedy Algorithm Description

Let $G = (V, E, p)$ be the given graph and k be the budget (number of seeds). The greedy algorithm works as follows:

1. Initialize seed set $S = \emptyset$
2. For $i = 1$ to k:
    - For each $v \in V \setminus S$, estimate marginal gain: $\Delta_v = \sigma(S \cup \{v\}) - \sigma(S)$
    - Select $v^* = \arg\max_v \Delta_v$
    - Update $S = S \cup \{v^*\}$

The influence $\sigma(S)$ is estimated using Monte Carlo simulation, typically repeated R times to obtain accurate results.

Pseudocode

```
Input: Graph G = (V, E, p), budget k, R = # simulations
Output: Seed set S of size k

S =
for i = 1 to k do
    max_gain = -1
    best_node = None
    for each v in V \ S do
        gain = EstimateSpread(S  {v}, R) - EstimateSpread(S, R)
        if gain > max_gain then
            max_gain = gain
            best_node = v
    S = S  {best_node}
return S
```

Approximation Guarantee

The algorithm gives a $(1 - \frac{1}{e})$-approximation to the optimal solution. That is,

$$\sigma(S_{greedy}) \geq \left(1 - \frac{1}{e}\right) \sigma(S^*)$$

where S_{greedy} is the output of the greedy algorithm and S^* is the optimal seed set.

Time Complexity

Let:

- $n = |V|$: number of nodes
- k: number of seeds to select
- R: number of Monte Carlo simulations
- T_{sim}: time to simulate spread for one set of nodes

In each of the k iterations, the algorithm evaluates up to n candidates. For each candidate, it runs R simulations. Hence, the total time complexity is:

$$\mathcal{O}(k \cdot n \cdot R \cdot T_{sim})$$

This can be accelerated using optimizations such as lazy evaluation or CELF (Cost-Effective Lazy Forward selection).

Conclusion

The greedy algorithm is a practical and theoretically justified method for influence maximization in social networks. Its approximation guarantee of $(1 - 1/e)$ and the submodularity of the influence function make it a robust choice, despite the problem's NP-hardness.

Task 3: Example Where Greedy Algorithm is Suboptimal

Graph Definition

We define a directed graph $G = (V, E, p)$, where:

- $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- E is the set of directed edges with associated propagation probabilities $p : E \to (0, 1]$

The edge set is:

$$E = \{(1, 2, 0.6), (1, 3, 0.6), (2, 5, 1.0), (2, 6, 1.0), (3, 7, 1.0), (3, 8, 1.0), (4, 3, 1.0)\}$$

The function $p(u, v)$ gives the probability that node u infects node v. For example, $p(1, 2) = 0.6$, $p(2, 5) = 1.0$, etc.

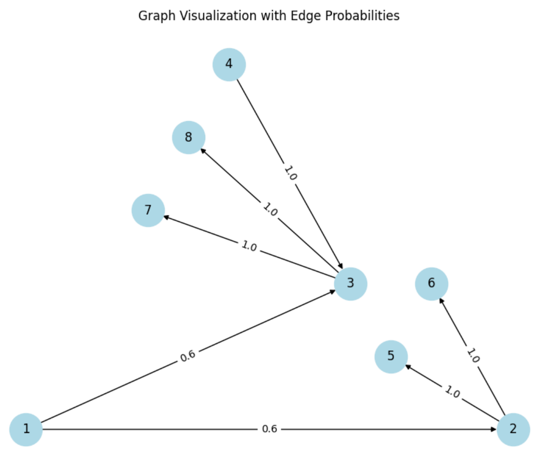

Figure 1: Graph

Greedy Algorithm Output

Using the standard greedy algorithm for influence maximization under the Independent Cascade model with $k = 2$, the greedy algorithm selects the seed set:

$$A_0 = \{1, 4\}$$

The expected spread $E[|A_\infty|]$, estimated using Monte Carlo simulations (100 iterations), is:

$$E[|A_\infty|] = 6.65$$

Optimal Seed Set

However, a better (and likely optimal) seed set is:

$$A_0^* = \{2, 4\}$$

This set achieves a higher expected spread:

$$E[|A_\infty|] = 7.0$$

Analysis of the Standard Greedy Algorithm

The standard greedy algorithm iteratively selects the node that provides the maximum marginal gain in expected spread until k nodes are chosen. Let $\sigma(S)$ denote the expected spread $E[|A_\infty|]$ for an initial seed set S.

Iteration 1: Selecting the first seed ($k = 1$)
We calculate the expected spread $\sigma(\{v\})$ for each node $v \in V$:

- $\sigma(\{1\}) = 1 + E[spreadfrom2] + E[spreadfrom3] = 1 + 0.6 \times (1 + 1 + 1) + 0.6 \times (1 + 1 + 1) = 1 + 1.8 + 1.8 = 4.6$
- $\sigma(\{2\}) = 1 + p(2 \to 5)(1) + p(2 \to 6)(1) = 1 + 1.0(1) + 1.0(1) = 3.0$
- $\sigma(\{3\}) = 1 + p(3 \to 7)(1) + p(3 \to 8)(1) = 1 + 1.0(1) + 1.0(1) = 3.0$
- $\sigma(\{4\}) = 1 + p(4 \to 3)(1 + \sigma(\{3\})startingfrom3) = 1 + 1.0 \times (1 + 1.0(1) + 1.0(1)) = 1 + 3 = 4.0$
- $\sigma(\{5\}) = 1.0$, $\sigma(\{6\}) = 1.0$, $\sigma(\{7\}) = 1.0$, $\sigma(\{8\}) = 1.0$

Node **1** provides the maximum initial spread (4.6).
Current Seed Set $S_1 = \{1\}$. Current Spread $\sigma(S_1) = 4.6$.

Iteration 2: Selecting the second seed ($k = 2$)
We calculate the marginal gain $\Delta(v) = \sigma(S_1 \cup \{v\}) - \sigma(S_1)$ for adding each remaining node $v \in V \setminus S_1$:

- $\Delta(4) = \sigma(\{1, 4\}) - \sigma(\{1\})$. To calculate $\sigma(\{1, 4\})$: Nodes 1 and 4 are initially active. Node 4 activates Node 3 with probability 1.0. Node 3 activates Nodes 7 and 8 with probability 1.0. Node 1 attempts to activate Node 2 with probability 0.6. If successful, Node 2 activates Nodes 5 and 6 (prob 1.0). Node 1 also attempts to activate Node 3 (prob 0.6), but Node 3 is already activated by Node 4. The expected number of active nodes is: $1(node1) + 1(node4) + 1(node3) + 1(node7) + 1(node8) + 0.6 \times (1(node2) + 1(node5) + 1(node6))$. $\sigma(\{1, 4\}) = 5 + 0.6 \times 3 = 5 + 1.8 = 6.8$. $\Delta(4) = 6.8 - 4.6 = 2.2$.
- $\Delta(2) = \sigma(\{1, 2\}) - \sigma(\{1\})$. $\sigma(\{1, 2\}) = 1(1) + 1(2) + 1(5) + 1(6) + 0.6 \times (1(3) + 1(7) + 1(8)) = 4 + 0.6 \times 3 = 4 + 1.8 = 5.8$. $\Delta(2) = 5.8 - 4.6 = 1.2$.
- $\Delta(3) = \sigma(\{1, 3\}) - \sigma(\{1\})$. $\sigma(\{1, 3\}) = 1(1) + 1(3) + 1(7) + 1(8) + 0.6 \times (1(2) + 1(5) + 1(6)) = 4 + 0.6 \times 3 = 4 + 1.8 = 5.8$. $\Delta(3) = 5.8 - 4.6 = 1.2$.
- Marginal gains for adding nodes 5, 6, 7, or 8 will be smaller (e.g., $\Delta(5) < 1$).

Node **4** provides the maximum marginal gain (2.2).
Final Greedy Seed Set $S_{greedy} = \{1, 4\}$.
Calculated Greedy Spread $\sigma(S_{greedy}) = 6.8$.
(Note: The calculated exact expected spread is 6.8. Your simulation result of 6.65 is within reasonable bounds for Monte Carlo estimation but we use the exact value for comparison here.)

Optimal Solution Analysis

Consider the alternative seed set $S_{opt} = \{2, 4\}$. We calculate its expected spread:

- Node 2 is active. It activates nodes 5 and 6 with probability 1.0. The contribution from this branch is $1(node2) + 1(node5) + 1(node6) = 3$.
- Node 4 is active. It activates node 3 with probability 1.0. Node 3 then activates nodes 7 and 8 with probability 1.0. The contribution from this branch is $1(node4)+1(node3)+1(node7)+1(node8) = 4$.
- The sets of nodes potentially influenced by {2} (i.e., {2, 5, 6}) and by {4} (i.e., {4, 3, 7, 8}) are disjoint in this graph instance.

Therefore, the total expected spread for S_{opt} is the sum of the contributions from the two disjoint components initiated by the seeds: $\sigma(S_{opt}) = \sigma(\{2, 4\}) = 3 + 4 = 7.0$.

Why Greedy is Suboptimal Here

This example highlights a key limitation of the greedy algorithm. It selects node 1, which has decent local influence via probabilistic edges to nodes 2 and 3. However, node 2 can deterministically influence nodes 5 and 6. Hence, selecting node 2 directly results in higher overall spread.

The greedy algorithm first picks node 1 (due to its immediate marginal gain), and then selects node 4. But if it had picked node 2 instead, the guaranteed influence of nodes 5 and 6 would yield a better global outcome.

Thus, this discrepancy clearly shows how the greedy algorithm can yield a suboptimal solution.

Colab Notebook Demonstration

The code and simulation to support this example are available in the following `Google Colab Notebook`.

Task 4: Team details

- Gaurav Meena - 2024AIB2286
- Utkarsh Giri - 2024AIB2287
- Taki Hasan - 2024AIB2295

Contribution

Equal Contribution by all the team members. The team member divided the work by trying different algorithms and writing different parts of the report.

Reference:

- Yuxin Ye, Yunliang Chen, Wei Han. *Influence maximization in social networks: Theories, methods and challenges. Array*, Volume 16, 2022. `https://doi.org/10.1016/j.array.2022.100264`
- Zainab Naseem Attuah, Firas Sabar Miften, Evan Abdulkareem Huzan. *Extracting Influential Nodes for Maximization Influence in Social Networks. Journal of Physics: Conference Series*, 1818 (2021) 012177. `https://doi.org/10.1088/1742-6596/1818/1/012177`
- Sonia, Kapil Sharma, Monika Bajaj. *DeepWalk Based Influence Maximization (DWIM): Influence Maximization Using Deep Learning. Intelligent Automation Soft Computing*, 2023. `https://doi.org/10.32604/iasc.2023.026134`
- Rahul Kumar Gautam, Anjeneya Swami Kare, Durga Bhavani S. *Heuristics for Influence Maximization with Tiered Influence and Activation thresholds.* arXiv preprint, 2024. `https://arxiv.org/abs/2406.08876`
- Min Gao, Li Xu, Limei Lin, Yanze Huang, Xinxin Zhang. *Influence maximization based on activity degree in mobile social networks. Concurrency Computat Pract Exper.*, 2020. `https://doi.org/10.1002/cpe.5677`

- Tarun Kumer Biswas, Alireza Abbasi, Ripon Kumar Chakrabortty. *A cost-effective seed selection model for multi-constraint influence maximization in social networks. Decision Analytics Journal*, Volume 11, 2024, 100474. `https://doi.org/10.1016/j.dajour.2024.100474`
- Jia-Lin He, Yan Fu, Duan-Bing Chen. *A Novel Top-k Strategy for Influence Maximization in Complex Networks with Community Structure. PLOS ONE*, 10(12): e0145283, 2015. `https://doi.org/10.1371/journal.pone.0145283`
- David Kempe, Jon Kleinberg, Éva Tardos. *Maximizing the Spread of Influence through a Social Network. Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003. `https://doi.org/10.1145/956750.956769`
- ChatGPT for hints and cleaning the code
- Google AI studio - Gemini 2.5 model for latex code generation