Differentiable Influence Minimization via Continuous Relaxation

Final Project Report: CMSC838B

Mihil Sreenilayam

University of Maryland, College Park

December 18, 2025

Motivation: The Cost of Connection

The Problem:

- Networks amplify diffusion: viruses, misinformation, and malware.
- **Influence Minimization (IMIN):** Given a budget k, which k edges should we cut to maximally stifle the spread?

Motivation: The Cost of Connection

The Problem:

- Networks amplify diffusion: viruses, misinformation, and malware.
- **Influence Minimization (IMIN):** Given a budget k, which k edges should we cut to maximally stifle the spread?

The Challenge:

- It is a *combinatorial* problem (discrete choices).
- It is a *stochastic* problem (random spread).
- **Result:** Standard gradients don't work. We usually rely on slow heuristics or greedy algorithms.

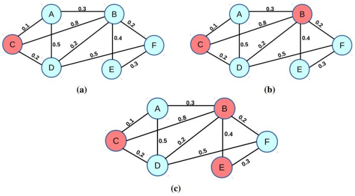

Figure: Network diffusion often relies on critical "bridge" edges.

Theoretical Formulation

We model diffusion via the **Independent Cascade (IC)** model.

The Objective

Minimize expected spread $\sigma(G)$ subject to a budget constraint:

$$\min_{E' \subset E} \sigma(G(V, E')) \quad \text{s.t.} \quad |E| - |E'| \leq k$$

Theoretical Formulation

We model diffusion via the **Independent Cascade (IC)** model.

The Objective

Minimize expected spread $\sigma(G)$ subject to a budget constraint:

$$\min_{E' \subset E} \sigma(G(V, E')) \quad \text{s.t.} \quad |E| - |E'| \leq k$$

Why is this hard?

1. **#P-Hardness:** Calculating $\sigma(G)$ exactly requires summing over $2^{|E|}$ possible live-edge realizations.
2. **Non-Differentiable:** The spread function is discrete.

$$\frac{\partial \sigma}{\partial \text{edge}_{uv}} \text{ is undefined.}$$

The Solution: Differentiable Relaxation

We propose a two-stage differentiable pipeline to bypass these hurdles.

1. **Surrogate Modeling**
   - Replace the expensive Monte Carlo simulator with a **Graph Neural Network (GNN)**.
   - The GNN approximates the influence function differentiably.

The Solution: Differentiable Relaxation

We propose a two-stage differentiable pipeline to bypass these hurdles.

1. **Surrogate Modeling**
   - Replace the expensive Monte Carlo simulator with a **Graph Neural Network (GNN)**.
   - The GNN approximates the influence function differentiably.

2. **Continuous Relaxation**
   - Replace binary edges $\{0, 1\}$ with continuous weights $w_{uv} \in [0, 1]$.
   - Turn the discrete "switch" into a smooth "dimmer knob."

Goal: Allow gradients to flow from the predicted spread back to the graph structure.

Methodology: GNN Surrogate Architecture

We utilize a Graph Convolutional Network (GCN) to map graph topology to a scalar influence score $\hat{y}$.

Input Features: Strictly structural ($x_v = [\text{deg}, \text{clust}, \text{neigh\_deg}]$) to ensure inductive generalization.

Optimization: The Sparsity Penalty

To remove edges, we minimize a compound loss function:

$$\mathcal{L}(\mathcal{W}) = \underbrace{f_\theta(\mathcal{W})}_{\text{Influence Min.}} + \underbrace{\lambda \cdot \sum_{(u,v)\in E} |1 - w_{uv}|}_{\text{Sparsity Penalty}}$$

- **First Term:** Pushes weights to minimize spread (flatten the curve).
- **Second Term:** Penalizes deviation from the original graph (cost of removal).
- λ **(Lambda):** The "aggressiveness" knob.
  - High $\lambda \rightarrow$ Keep edges (Conservative).
  - Low $\lambda \rightarrow$ Remove edges (Aggressive).

Project Contributions

Relationship to Prior Work (DiffIM)

This project builds on the theoretical framework of Lee et al. (2025).

My Specific Contributions:

1. **Reproducible Implementation:** Built the full pipeline in PyTorch Geometric (Surrogate + Relaxation Loop).
2. **Feature Engineering:** Designed the structural feature set ($d = 3$) to enable size-invariant generalization.
3. **The "Lambda Sweep":** Developed an evaluation protocol to map the full Pareto frontier of sparsity vs. influence.
4. **Baseline Comparison:** Conducted rigorous testing against random baselines on synthetic (ER, BA, WS) and real (SNAP) graphs.

Experimental Setup

- **Dataset:** $\approx 1,400$ graphs.
  - *Training:* Mix of Erdős–Rényi, Barabási–Albert, Watts–Strogatz.
  - *Testing:* Held-out synthetic graphs + Real-world SNAP subgraphs (Email-Eu-core).
- **Ground Truth:** Monte Carlo Simulations (500 runs per graph).
- **Baseline: Random Edge Removal**.
  - Crucial: We match the *exact number of edges removed* to ensure fair comparison.

Results: Surrogate Fidelity

Can a neural network actually predict stochastic influence? **Yes.**

Graph Type	**Pearson** r	**Spearman** ρ
Erdős–Rényi	0.982	0.975
Watts–Strogatz	0.971	0.966
Barabási–Albert	0.954	0.941
Overall	**0.969**	**0.960**

Table: Correlation between GNN prediction and Monte Carlo truth on test set.

The high rank correlation (ρ) confirms the surrogate correctly identifies which graph states are "worse" than others.

Results: Sparsity-Influence Trade-off

Analysis:

- **Orange (Random):** Linear/Convex decay. Requires massive deletion to stop spread.
- **Blue (Ours):** Concave decay. Significant influence drop with minimal deletion.
- **Interpretation:** The gradient method identifies *structural bottlenecks* (bridges) that random deletion misses.

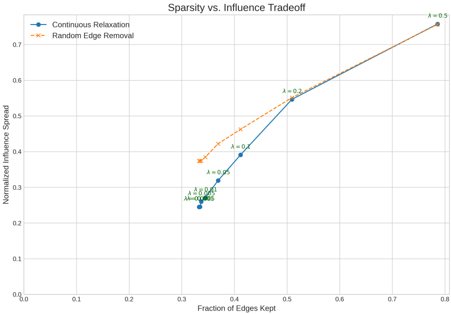

Figure: Influence vs. Edge Retention Rate.

Limitations and Future Work

Current Limitations:

- **Generalization Gap:** Performance drops on real-world graphs ($r \approx 0.65$) due to complex community modularity not present in synthetic training data.
- **Scalability:** Full-batch GCN is memory intensive for massive graphs.

Future Directions:

- **Domain Adaptation:** Few-shot fine-tuning of the surrogate on real-world subgraphs.
- **One-Shot Pruning:** Train a policy network to predict $\mathcal{W}^*$ directly, avoiding the per-instance optimization loop.

Summary

1. We formulated Influence Minimization as a **differentiable learning problem**.
2. We implemented a pipeline using a **GCN Surrogate** and **Continuous Edge Relaxation**.
3. We demonstrated that gradient-based pruning significantly outperforms random baselines, identifying critical structural weaknesses in networks.
4. The framework provides a bridge between combinatorial graph theory and modern deep learning optimization.

Thank You! Questions?

References I

Lee, J., et al. (2025). *DiffIM: Differentiable Influence Minimization with Surrogate Modeling*. arXiv:2502.01031.

Kipf, T. N., & Welling, M. (2017). *Semi-Supervised Classification with Graph Convolutional Networks*. ICLR.

Kempe, D., Kleinberg, J., & Tardos, É. (2003). *Maximizing the spread of influence through a social network*. KDD.

Manchanda, S., et al. (2020). *Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs*. NeurIPS.