Seeding Strategy Comparisons for Graph-Based Simulations of Information Spread

By: Anirud Mohan, Mahima Beltur, Prakhar Tiwari, William C. Loe

Overview

This project investigates the modeling of information diffusion across a large-scale social network using a directed graph data structure. The primary objective is to explore the scalability and effectiveness of graph-based algorithms—specifically Kruskal’s algorithm and centrality-based metrics—for selecting initial seed nodes that maximise information spread in diffusion simulations.

Graphs are a fundamental data structure in computer science, widely used to represent relationships in various real-world systems such as social networks, transportation grids, and biological networks. In the context of information spread, graph algorithms allow for the analysis of how influence propagates through connected entities. This project focuses on implementing Kruskal’s algorithm, a classic method for constructing Minimum Spanning Trees (MSTs) in undirected, weighted graphs, to examine underlying structural properties of the network. Although MSTs do not directly apply to directed graphs, insights derived from MST-based clustering can support the identification of influential regions within the network.

Additionally, the project incorporates centrality measures—including degree, pagerank, and community centrality—to identify key nodes that could serve as effective starting points for information diffusion. These metrics provide heuristic strategies for maximizing spread by targeting nodes with high connectivity or strategic network positions. Implementing these algorithms at scale introduces several challenges. Kruskal’s algorithm requires efficient edge sorting and disjoint set management, which can become computationally intensive with millions of edges. Common approaches to address these issues include the use of Union-Find data structures with path compression and parallelized sorting techniques. Similarly, calculating centrality metrics in large graphs is computationally expensive, often requiring approximation methods or distributed computation frameworks to maintain performance.

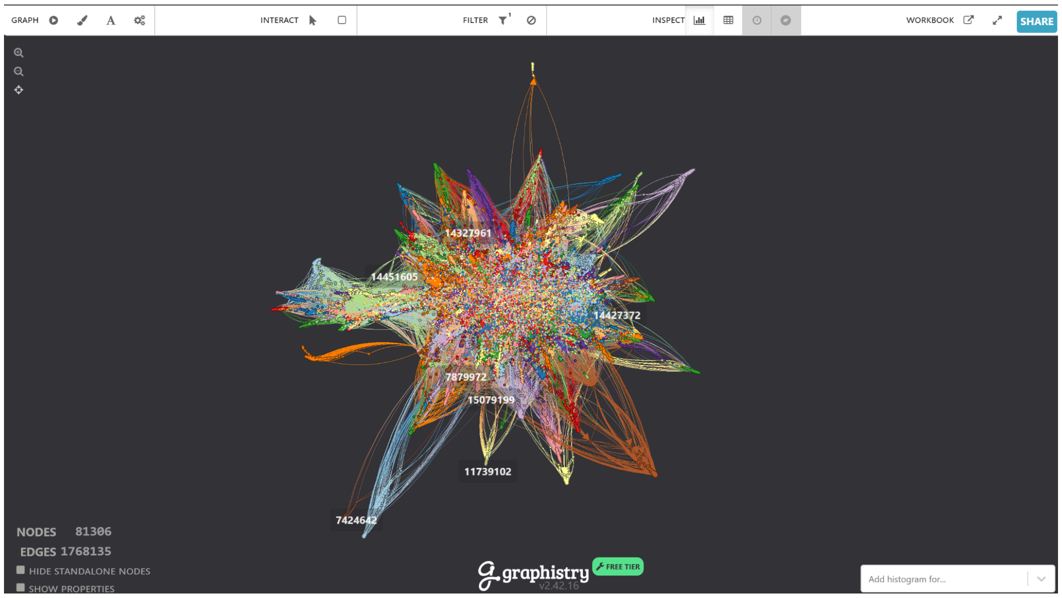
Implementation

This project implements an Information Campaign Planner tailored to simulate and optimize the diffusion of information across a large-scale, real-world social network. The core of the system is based on graph-based data structures and algorithms, particularly leveraging centrality metrics and a custom Minimum Spanning Tree (MST) approach to identify influential nodes for initiating information spread. The target application simulates how a campaign—such as a public health message or marketing initiative—can be seeded in a network like Twitter to maximize its reach.

The Data

The dataset used in this project is the Twitter Ego Network, a large-scale, real-world social network graph. It consists of 81,306 nodes and approximately 1.7 million directed edges, where each edge represents a follower–followee relationship between Twitter users. The graph is inherently directed, as the act of following on Twitter is not necessarily mutual, making it a suitable structure for modeling asymmetric information spread.

The dataset was loaded using cuDF.read_csv to leverage GPU acceleration for scalable processing. No significant data cleaning or preprocessing was required beyond this sampling step, as the dataset was already well-structured and formatted as an edge list with clearly defined source and target columns.



Seeding Strategies

To explore effective seeding strategies for information diffusion, two base methods were implemented and evaluated. The baseline strategy, random seeding, selects k nodes uniformly at random from the network. This approach serves as a strong baseline due to its lack of structural bias, and its stochastic nature can help reduce systematic bias across trials. Despite its simplicity, random seeding often performs competitively in networks with decentralized connectivity.

The primary method in this approach is Kruskal Minimum Spanning Tree to implement Degree-Based Seeding. A custom implementation of Kruskal's algorithm was necessary due to memory limitations with cuGraph.minimum_spanning_tree. The custom MST was constructed by first sorting edges using cuDF, and then applying a GPU-optimized Union-Find structure using CuPy. This strategy offers a scalable, structurally-aware method to find potentially influential nodes based on simplified connectivity paths. Once the MST was constructed, the top-k nodes with the highest degrees in the MST were selected as seeds. For this step, the graph was assumed to be undirected to allow us to find the most central and connected nodes to use as seeds.

Diffusion Simulation

To evaluate the effectiveness of the seeding strategies, a diffusion model was implemented using a stochastic infection process. In each iteration, newly informed nodes had a dynamically adjusted probability (β) of informing their neighbors, mimicking real-world scenarios such as information sharing or rumor spreading where influence propagates probabilistically through social ties. Rather than using a fixed β, the simulation employed an adaptive beta scheduling mechanism for all runs. Specifically, the feedback-based scheduler, feedback_beta(stats, base=β, target_ratio=0.25, k=3.0), updates β as the simulation progresses—starting from the given base value and decreasing it as the proportion of infected nodes increases. This design emulates saturation effects or public fatigue, allowing the model to better mirror the ebb and flow of real-world diffusion dynamics.

Results and Analysis

The heatmaps in Figure 1 offer a side-by-side comparison of Random and Kruskal-MST seeding strategies across key metrics under varying seed set sizes (k) and infection probabilities (β). In this experiment, seeding from the Kruskal-MST, consistently outperforms random seeding in terms of the number of nodes reached. Even with just one seed (k=1) and a low starting β value like 0.05, Kruskal-MST manages to inform over 80,000 nodes, reaching saturation across the network. In contrast, Random seeding struggles to gain traction at low β values (e.g., 0.01) and only begins to perform reasonably well when both k and β are higher.

Trends in the runtimes also highlight key differences. At low β levels, random seeding appears faster because the diffusion stalls early and the simulation ends quickly. As β increases and diffusion becomes more active, runtimes for both strategies converge to around 30–40 seconds. However, Kruskal-MST shows more consistent performance across different conditions, suggesting that the initial overhead of building the MST is balanced out by its efficiency in accelerating spread. Similar trends emerge in the heatmaps measuring the lifetime of the simulation in number of iterations: Kruskal-MST based seeding more consistently maxes out the simulation at 100 iterations across values (especially lower β values) , while random seeding fails to achieve this in the smaller metrics. However, Kruskal-MST typically converges sooner— around 85–90 iterations— at higher β and k values, however this may be attributed to the random chance of propagating the information to each node.

These findings underscore the robustness and scalability of the Kruskal-MST-based strategy across a range of diffusion conditions. Its ability to rapidly reach a critical mass of nodes demonstrates its structural advantage in identifying influential nodes, even under low seeding and propagation parameters. While random seeding offers a baseline, its performance is highly sensitive to parameter tuning and pure chance, and it lacks the consistency observed in MST-driven approaches. The convergence behavior and runtime stability of Kruskal-MST reinforce its practicality for real-world applications where efficiency, reliability, and early saturation are critical despite its initial overhead in seed computation.

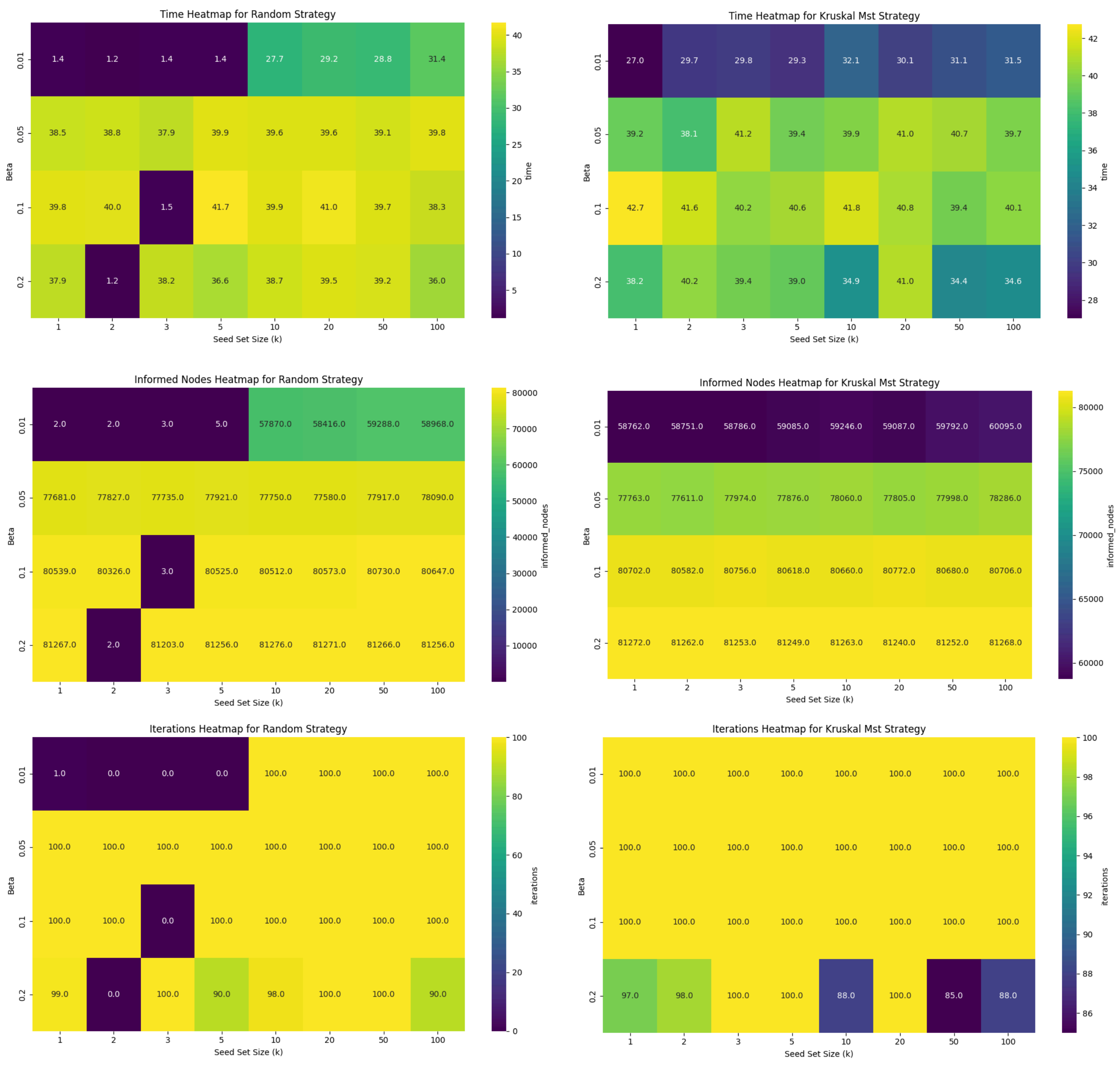


Figure 1. Left side, runtime (s), information reach (nodes), and number of iterations to completion for random seeding. Right side, runtime (s), information reach (nodes), and number of iterations to completion for Kruskal's MST-based seeding.

Extended Components

Alternatives

The second method is degree centrality seeding, which selects the top k nodes with the highest out-degrees, under the assumption that highly connected nodes are well-positioned to spread information quickly. This is particularly effective in scale-free networks, where a few high-degree hubs dominate connectivity. Degree values were computed efficiently using cuGraph.degree() on the directed graph, and sorted to extract the top-ranking nodes.

A more nuanced third method is PageRank-based seeding, which considers both the quantity and quality of incoming links. Nodes are ranked by their PageRank scores, with those linked by other influential nodes receiving higher scores. This method is well-suited for directed graphs and helps target globally influential users, potentially leading to broader and faster reach. Implementation used the GPU-accelerated cuGraph.pagerank() method.

To diversify the reach of the seeding set, a fourth method—community-based PageRank seeding was introduced. This approach first segments the graph into weakly connected components using cuGraph.weakly_connected_components(), then computes PageRank scores within each of the largest components. The top-ranking node from each component is selected, ensuring that seeds are distributed across structurally distinct regions of the network. If additional seeds are required, remaining slots are filled by selecting top global PageRank nodes. This hybrid method balances local influence with global coverage, especially useful in modular or clustered networks.

RESULTS FOR ALTERNATIVE:

Figure 2 compares the performance of all five seeding methods. Across all metrics, Kruskal-MST, Degree Centrality, PageRank, and Community-based PageRank consistently outperform Random Seeding. Notably, all four identified the same top-3 influential nodes as seeds: 813286, 115485051, and 40981798, which highlights the extent to which these nodes hold influence as the central users of this network. The number of informed nodes reached saturated quickly, after which point increasing the seed size (k) made little difference. Kruskal-MST and Degree strategies achieved high levels of spread early. Higher β values naturally accelerated diffusion, but Kruskal-MST and PageRank remained effective even at lower β, thanks to better seed targeting.

The runtime remained efficient across all seeding strategies. The runtime heatmap and line plots confirm that all structured strategies (Kruskal-MST, Degree, and PageRank) achieved near-maximum spread in under 40 seconds across most configurations, while Random often terminated prematurely in sparse conditions due to limited activation. Iteration trends further supported this pattern: as β increased, adaptive feedback scheduling reduced β over time, forcing faster convergence. Structured strategies reached saturation in fewer steps and maintained consistent performance across k values, whereas Random showed erratic iteration counts, especially at low β.

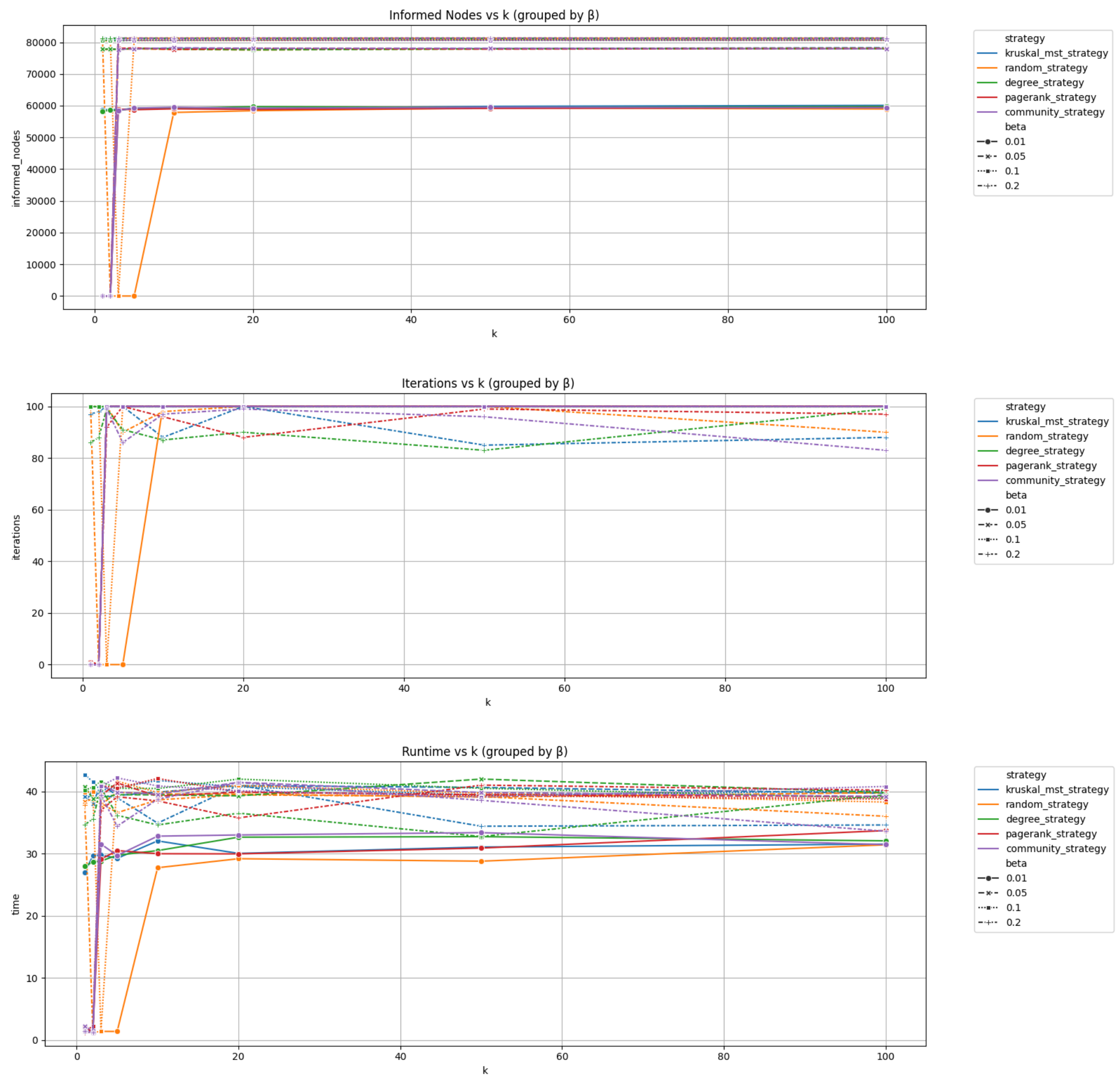


Figure 2: Comparisons across all five seeding methods of runtime (s), information reach (nodes), and number of iterations to completion for random seeding

Proposed enhancements*

One limitation with the current methodology is the computational intensity of running Kruskal's algorithm on large-scale graphs. Built-in functions for this process are currently not supported in GPU libraries as they are in CPU-based ones such as networkX. A potential improvement to remedy this is to implement a more memory-efficient, chunked Kruskal's algorithm that processes edge batches sequentially on the GPU. While we used Union-Find with path compression, performance could be further enhanced with union by rank or incorporating RAPIDS' native graph primitives more strategically. For the diffusion simulation, refining the adaptive β scheduler to be topology-aware—e.g., reducing β more aggressively in highly clustered regions—could lead to more realistic saturation dynamics, aligning better with observed phenomena in social graphs.

Proposed Features*

A valuable extension would be to support dynamic or temporal graphs, enabling us to simulate how influence spreads over time as users enter or leave the network. This would introduce concepts like edge deletions and node insertions, deepening the data structure application via dynamic connectivity handling. Additionally, enabling edge weighting based on activity frequency or trust scores would create opportunities to explore advanced priority queue usage within MST computation and influence models. These features not only enhance realism but also offer a stronger grounding in algorithmic design and complexity management—key pillars of this course.

Interactive Visualization*

To support structural insights, an interactive Graphistry visualization of the Twitter Ego Network was generated. The plot allows users to study the static topology of the social graph interactively. Users can zoom, filter, and hover over nodes to examine local connectivity patterns, and visualize the reach of high-degree nodes, and observe community clusters.

Ethical considerations*

This project investigates how influence and information can spread across networks, with a focus on identifying optimal seeding strategies using graph-based models. While the technical objective is to evaluate the efficacy of diffusion algorithms within social media-like structures, it is important to acknowledge the broader ethical implications of this work. These models possess a clear dual-use potential: while they can be employed for beneficial purposes such as amplifying public awareness campaigns or countering misinformation, they may also be misused for manipulative practices, including the targeted dissemination of propaganda or polarizing content.

Additionally, many commonly used strategies—such as those based on degree centrality or PageRank—tend to amplify the influence of already prominent users, potentially marginalizing less connected or minority voices within the network. This raises concerns about fairness and the reinforcement of existing power imbalances in online spaces. It is therefore important to consider more inclusive or equitable approaches in future developments.

Social Impacts*

The results of this project feed into discussion about both the potential benefits and risks that arise as a result of information virality. By identifying key central users whose influence disproportionately shapes the network, the simulation sheds light on the dynamics of influence campaigns and the broader implications of digital persuasion. These strategies can serve beneficial purposes, such as disseminating public health messages, but also raise concerns about their misuse in spreading misinformation or propaganda. The ethical tension lies in leveraging these insights for the public good without enabling manipulation or control.

The exploration of community detection and Kruskal-MST also sheds light on how echo chambers form. Echo chambers are a key consideration in understanding how social media shapes peoples political and social views. Insights into cross-community interactions can be used to reduce informational silos, but the same studies can highlight closed-off communities which carry the risk of reinforcing divisions and deepening polarization. Striking a balance between fostering constructive dialogue and avoiding social fragmentation is essential. Moreover, the methodologies applied in this project have strong potential for civic and humanitarian applications. Improved understanding of information diffusion can lead to more effective interventions in public health, disaster response, and civic engagement, reinforcing the role of these strategies as tools for societal benefit.

Conclusion

This project explored scalable strategies for identifying central nodes and simulating information diffusion in large-scale social networks through the use of graph-based algorithms. By leveraging GPU-accelerated tools on the Twitter Ego Network dataset, several seeding strategies were implemented and compared, including a custom GPU-optimized Kruskal-MST, degree centrality, PageRank, and community-based methods. As expected, the findings showed that structurally-aware approaches like Kruskal-MST and PageRank outperform random seeding. Random seeding has very variable results, and occasionally matches the performance of the calculated metrics. The MST, degree-centrality and page-rank based approaches are more consistent, and match each others results.

A key challenge with this implementation involved transitioning the codebase to a GPU-accelerated environment using CUDA, CuPy, and cuGraph, which proved difficult due to various issues with setting up the RAPIDS ecosystem. However, the downloaded graph was too large to simulate and experiment on in the standard CPU.

* Items that are part of our bonus work

Contributions and Reflections

Anirud

Contributions: CPU- based graph implementation, networkx based visualisation for testing, contributed to slides
Reflection: Working on this project reminded me how important it is to get the fundamentals right before scaling up. Prototyping with NetworkX gave me an intuitive understanding of the diffusion process and allowed me to test concepts quickly. Even though the final implementation shifted to GPU, I gained valuable experience bridging small-scale testing with large-scale deployment, and I feel more confident with both ends of that spectrum.

Mahima

Contributions: GPU conversation, visualization with GPU, experiment design, report writing, project scoping
Reflection: Adapting our simulations to run efficiently on large graphs was challenging, especially with memory constraints and long runtimes. I learned a lot about optimizing data pipelines, and how to make GPU workflows more interpretable through visual tools. More broadly, helping guide the direction of helped me think critically about each state of the project, and identifying what additions need to be made for a robust work.

Prakhar

Contributions: ethics, enhancements, additional features-> made implementations that we ultimately had to not use due to their scope
Reflection: Coming up with ways to enhance our process with more dynamic and adaptive methods really helped me think critically about the functionality of our application.. Although some implementations had to be cut for feasibility, they allowed us to critically assess where complexity may not always yield practical benefit. Writing about the ethical dimensions of influence algorithms was a meaningful part of this process—I came away with a stronger awareness of the dual-use nature of network diffusion tools and the responsibility we carry in their design and deployment.

William

Contributions: Benchmarking, experiment setup in code, code maintenance, slides, project scoping and direction
Reflection: Coordinating experiments at this scale taught me a lot about code modularity, reproducibility, and performance monitoring. Writing seeding strategy benchmarks and watching them scale under different parameters gave me insight into the importance of evaluation structure. This experience also reinforced how powerful GPU-based computation can be—but only if the design is sound and well-maintained.