Evaluating the role of critical nodes in disrupting diffusion in independent cascade diffusion model

Raj Gaurav Ballabh Kumar

Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

Part of the Computer Sciences Commons

Recommended Citation
Kumar, Raj Gaurav Ballabh, "Evaluating the role of critical nodes in disrupting diffusion in independent cascade diffusion model" (2019). Graduate Theses and Dissertations. 17723. https://lib.dr.iastate.edu/etd/17723

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Evaluating the role of critical nodes in disrupting diffusion in independent cascade diffusion model

by

Raj Gaurav Ballabh Kumar

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Samik Basu, Co-major Professor
Pavan Aduri, Co-major Professor
Gurpur Prabhu

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University
Ames, Iowa
2019

Copyright © Raj Gaurav Ballabh Kumar, 2019. All rights reserved.
DEDICATION

To my parents - Raj Ballabh Kumar and Shashi Kumari. None of this would have been possible without their hard work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
</tbody>
</table>

## 1. INTRODUCTION

1.1 Influence Maximization .......................... 1
1.2 Disrupting Diffusion .......................... 3
1.3 Contribution .......................... 4
1.4 Organization .......................... 5

## 2. LITERATURE REVIEW AND RELATED WORK

2.1 Modelling a Social Network .......................... 7
2.2 Diffusion Models .......................... 8
   2.2.1 IC Model .......................... 9
   2.2.2 LT Model .......................... 11
   2.2.3 Variations on the LT and IC Models .......................... 12
2.3 Influence Maximization Problem .......................... 15
2.4 Why should we study the problem of Disrupting Diffusion .......................... 20
2.5 Managing and Controlling Information Diffusion ........................................ 20

3. DISRUPTING DIFFUSION: FORMULATION BASED ON CRITICAL NODES 23
   3.1 Illustrative Example of Disrupting Diffusion by using Critical Nodes ........... 23
   3.2 Formalizing Criticality ............................................................................ 24
   3.3 Re-characterization of Critical Nodes as “Impact”-ful Nodes ..................... 26
   3.4 Algorithm to compute Critical Nodes .................................................... 28
      3.4.1 Impact Computation using Random Reachability ............................... 29
      3.4.2 Incremental Computation of Marginal Gain in Impact ....................... 30
      3.4.3 Efficient Implementation of Incremental Computation ....................... 32

4. IDENTIFYING CRITICAL NODES: ALGORITHMS, DATA STRUCTURES, AND PROPERTIES 34
   4.1 Using dependencyMatrix to capture dependsOn relationship .................... 34
   4.2 $ICN(G,k,S)$ Problem: Using the Seed Set Context to identify Critical Nodes .... 38
   4.3 Comparison methods for the Crit-Set Algorithm ....................................... 40
      4.3.1 TOP-CRIT: One-Shot approach of selecting Critical Nodes ............... 40
      4.3.2 TOP-INFL: Removing Most Influential Nodes .................................. 41
   4.4 Does Removing Critical Nodes Guarantee Maximal Disruption ................... 41
      4.4.1 Seed context is important: Experimental results depend on the chosen Seed Set ................................................................. 41
      4.4.2 Removing the most influential node can be a better option than removing the most critical node .............................................. 42
      4.4.3 Greedy vs. One-Shot ...................................................................... 43

5. EXPERIMENTAL EVALUATION 46
   5.1 Previous Results by Bharadwaj [2018] ..................................................... 46
      5.1.1 Given an influenced graph, the influence of a random seed is significantly reduced due to removal of critical nodes ............................ 46
5.1.2 Max-seed influence significantly reduced to removal of critical nodes . . . . . 47
5.1.3 Importance of Budget . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
5.2 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
  5.2.1 Objectives of the experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
  5.2.2 Data Setup and System details . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
  5.2.3 Contextualized critical node identification . . . . . . . . . . . . . . . . . . . . . . . 48
  5.2.4 Importance of Greedy (CRIT-SET) vs One-shot (TOP-CRIT) : How to evaluate . . 49
  5.2.5 Applicability in real-data (Twitter/Hong Kong) . . . . . . . . . . . . . . . . . . . 50

6. FUTURE WORK 57

7. GENERAL CONCLUSION 59
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 5.1</td>
<td>Datasets</td>
<td>49</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Criticality-Indicator &amp; Importance with Seed Set as Context</td>
<td>49</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Importance of Submodularity</td>
<td>50</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Time-Step-0</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Time-Step-1</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Time-Step-2</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>All-Activated-Nodes</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Time-Step-0</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Time-Step-1</td>
<td>13</td>
</tr>
<tr>
<td>2.7</td>
<td>Time-Step-2</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Entire Graph</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Node with the maximum influence removed</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>Critical Node removed</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>$G^r_i$</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>$dependencyMatrix$ before removing ${3}$</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>$G^r_i - {3}$</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>$dependencyMatrix$ after removing ${3}$</td>
<td>37</td>
</tr>
<tr>
<td>4.5</td>
<td>$G^r_i\epsilon$</td>
<td>39</td>
</tr>
<tr>
<td>4.6</td>
<td>$isCriticalVector$ before removing ${2}$</td>
<td>39</td>
</tr>
<tr>
<td>4.7</td>
<td>$G^r_i\epsilon - {2}$</td>
<td>39</td>
</tr>
<tr>
<td>4.8</td>
<td>$isCriticalVector$ after removing ${2}$</td>
<td>39</td>
</tr>
<tr>
<td>4.9</td>
<td>Validation of Critical Nodes depends on seed used for calculating influence</td>
<td>42</td>
</tr>
</tbody>
</table>
Figure 4.10  Example for a case when removing the most influential node is a better option than removing the node with the highest impact.  

Figure 4.11  Example for a case when the Modular approximation of the impact function in the Algorithm 2 gives a higher reduction in influence from a fixed seed when compared to the Submodular approximation of the impact function of the same algorithm.

Figure 5.1  Critical nodes identified (Twitter Verified Users) when the seed set is not known.

Figure 5.2  Critical nodes identified (Twitter Un-Verified Users) when the seed set is not known.

Figure 5.3  Critical nodes identified (Twitter Verified Users) when the seed set is known.

Figure 5.4  Critical nodes identified (Twitter Un-Verified Users) when the seed set is known.
ACKNOWLEDGMENTS

This work would not have been possible without the constant support, guidance, and assistance of my major advisors Dr. Samik Basu and Dr. Pavan Aduri. Their levels of patience, knowledge, and ingenuity is something I will always keep aspiring to. I would also like to thank Dr. Gurpur Prabhu for accepting to be a part of my POSC. I would be amiss if I did not mention my lab mates, Madhavan and Xiaoyun, who were always ready to help with any questions that I had.
How can we mitigate the unwanted diffusion of information in a social network? In this work we look at this problem and propose a solution through the identification of critical nodes. If we know which nodes act as the enablers to the spread of diffusion by a varied set of sources, then by removing these enablers from the network we can minimize the spread of diffusion from a large fraction of the sources. We call these enablers the critical nodes in a network. Identifying $k$ critical nodes such that removal of these nodes maximally disrupts the influence from any possible seed is the $ICN(k)$ problem. We use the notion of impact of a set of nodes and use it to characterize the $ICN(k)$ problem in the IC Model. Informally, impact of a set of nodes quantifies the necessity of the nodes in the diffusion process. We develop heuristics that rely on greedy strategy and modular or submodular approximations of impact function. We empirically evaluate our heuristics by comparing the level of disruption achieved by identifying and removing critical nodes as opposed to that achieved by removing the most influential nodes. We also run our algorithm on real-world Twitter data and show that the critical nodes identified by our algorithm can be considered critical to the diffusion of information.
CHAPTER 1. INTRODUCTION

The use of social networks by social scientists to study human interaction can be dated back to at least as early as 1940 (Radcliffe-Brown [1940]). In recent years, the unprecedented growth in the Online Social Networks (OSNs) and the large amounts of data that have become readily available have led to the formulation and study of a number of research questions in this field, both, from academia, and from corporations. A large section of this research is focused on the analysis of influence and information diffusion in social networks. One part of the study in the field of influence propagation focuses on mitigating the spread of influence i.e. if an unwanted diffusion is detected in a social network, what are the different ways in which we can try and minimise the spread of said diffusion. In this work, we look at the method of identifying nodes that are “critical” to the spread of diffusion and through empirical evaluations show that removing these nodes causes higher disruption in the spread of diffusion when compared to other techniques like removing the most influential node. Another way we validate our claim is by running tests on real-world Twitter data and show that the nodes identified as “critical” by our algorithm can in-fact considered to be critical in the general sense of the word.

1.1 Influence Maximization

A lot of research in the recent years has been focused on the computational analysis of social networks. When we talk about social networks, we will always refer to them as a Graph $G = (V, E)$ where the nodes represent the people or entities being studied in the social context and the edges represent the relationship among the entities. This relationship can be in the form of interaction (a user retweeting a tweet from another user), collaboration (researchers working as co-authors on a paper), or influence (viral marketing/“influencers”). A graph thus enables us to model these interacting entities in an intuitive and efficient way. The availability of these large datasets, often
in the form of graphs (Leskovec and Krevl [2014]) and through publicly available APIs (Twitter-API [2019]) has enabled rigorous study of their basic properties and helped researchers identify recurring structural features and patterns that have been exploited to develop efficient algorithms. Our experimental section makes extensive use of these datasets for validation purposes.

Richardson and Domingos [2002] were the first to study influence maximization as an algorithmic problem that was motivated by viral marketing applications. The problem was tackled by means of a probabilistic model of interaction, and heuristics were given for choosing the users to market to that maximised the expected lift in profit. In the seminal work of Kempe et al. [2003], the Influence Maximization (IM) problem was formulated and studied as an optimization problem. Influence maximization is the problem of selecting a small set of seed nodes \( S \) in a social network, such that the expected number of influenced nodes, \( \sigma(S) \), is maximized. (The solution to the \( IM(k) \) problem is a set of cardinality \( k \) that is referred to as the max seed set. We will be using both \( IM(k) \) and max seed set interchangeably.) The main result in their paper was showing the IM problem to be monotone and submodular, which would hence allow the simple greedy algorithm to provide a \( 1 - 1/e \) approximation (Nemhauser et al. [1978]). Two primary models were used in their work - the Independent Cascade (IC) model and the Linear Threshold (LT) model. Both IC and LT models are stochastic models characterizing how influence is propagated through the network - starting from the initial seed nodes - and have been widely studied since their seminal work. In our work, we will be restricting our focus to the IC model.

Computing the influence of the cascade is a \#P-hard problem (Chen et al. [2010]). As a result, the objective function cannot be efficiently computed, and we wish to estimate the function value by sampling techniques. Borgs et al. [2014] proposed the uniform reverse sampling technique which was later improved by Tang et al. [2014a], Tang et al. [2014b], and Nguyen et al. [2016], and it has been widely applied to other problems regarding influence diffusion in social networks. We make use of the framework developed by Borgs et al. [2014] in our algorithm to make computations faster.
1.2 Disrupting Diffusion

Diffusion can be understood as the spread of an entity in a network. Consider a tweet about release of a movie that is posted by a movie studio. That tweet is seen by all of the followers of that movie studio. Those followers then retweet that information which is seen, in turn, by their followers, and the process continues. The information is said to diffuse through the network - sometimes also referred to as an information cascade. In these scenarios, the studio often wants to maximize the number of people that see the tweet. They can therefore pay some highly influential people in the network to tweet about the movie. Assuming that the budget for promotion is limited, which $k$ nodes should the studio target to initially send out the tweet so that the information reaches the maximum number of people? This question is answered by the IM problem explained previously.

But sometimes, the diffusing entity might not be something that is needed in the network. Think of the spread of a disease in a community, or the spread of a rumor in OSNs. We want to mitigate the diffusion such that the people exposed to the rumor/disease is minimized. There are multiple ways to do this. In this work, we approach this problem by trying to identify the nodes that are critical to the spread of diffusion and removing them from the network. Often there is a cost associated with removing nodes. Think of the cost associated with quarantining people in the case of a disease outbreak, or suspending accounts of people on OSNs that could lead to backlash from the users. Hence we want to maximally reduce the spread of diffusion by removing a relatively small number of $k$ nodes.

More formally, the goal is to identify a set of nodes $C$ of cardinality $k$, such that, after removal of $C$ from the network, $\sigma(S)$ is maximally reduced from every possible seed set $S$. We refer to such entities $C$ as critical nodes, and we call the problem of computing such nodes as the identifying critical nodes (ICN) problem.

Consider the ICN problem when $k$ equals 1, i.e, identify a single critical node. A naive approach to critical node identification works as follows: for each node $v$, remove it from the network. Then for every possible seed set $S$ compute how much $\sigma(S)$ is reduced due to removal of $v$. Finally, return the node $v$ that gives the maximum reduction in influence from every possible seed set as
the critical node. This approach has at least two problems. It is immediate that such strategy is not viable even for reasonably small networks as one has to cycle through all possible seed sets \( (O(n!)) \). Secondly, this approach may not even find such a \( v \). For example, consider the following scenario: let \( v_1 \) and \( v_2 \) be two nodes and \( S_1 \) and \( S_2 \) be two seed sets such that removal of \( v_1 \) will maximally reduce \( \sigma(S_1) \), whereas removal of \( v_2 \) maximally reduces \( \sigma(S_2) \). There is no single vertex whose removal will maximally reduce both \( \sigma(S_1) \) and \( \sigma(S_2) \). To get around this problem, Bharadwaj [2018] introduced the concept of Strength of a graph and Impact of a set of nodes. We will be making use of these concepts in this work as well.

### 1.3 Contribution

The formulation of criticality in terms of strength, that was originally introduced in Bharadwaj [2018], has been developed further. In this thesis, we focus on exploring the strategy for evaluating the effectiveness of such formulation, which, in turn, provides valuable insights toward the viability of disruption strategy using critical node removal. The following outlines the contributions of this thesis.

1. Bharadwaj [2018] implements a heuristic for updating the criticality of nodes; such an update is central to greedy strategy for identifying a set of critical nodes. The heuristic is based on the assumption that the node that is considered critical at a step in greedy strategy and the nodes that contribute to its criticality are sufficient to realize the update of criticality of all nodes in subsequent steps of greedy strategy. We discuss under what scenarios such assumptions may significantly impact quality of the heuristics. In other words, the assumption was that if the reachability from a node \( u \) to \( w \) did not depend on \( v \), then the removal of any node from the graph (other than \( u, v, w \) would cause that to remain unchanged. In reality, removal of some node \( t \), might cause the reachability from \( u \) to \( w \) to start depending on \( v \).

2. We present a detailed discussion on greedy strategy and one-shot strategy for identifying critical set \( S \) (of size \( k \)) of nodes. In the former, the solution set \( S \) is generated by identifying
the $i^{th}$ most critical nodes in the context of $(i - 1)^{th}$ critical nodes (already present in the partial solution set). The latter, on the other hand, computes the top $k$ critical nodes and returns this as the solution set. We describe the scenarios when one can provide better quality results than the other.

3. We present new experimental evaluation strategies to further explain the effectiveness of critical nodes in disrupting diffusion. We consider large publicly available social networks (SNAP Leskovec and Krevl [2014]) and discuss the way we have quantified disruption. We have also developed a mock-up real-life scenario based on publicly available Twitter data on Hong Kong protest (mis)information diffusion. We observe that critical nodes identified as per our strategy are likely conduits for diffusion (even if these nodes are not the most influential ones).

1.4 Organization

In Chapter 2 we look at the Influence Maximization problem introduced by Kempe et al. [2003]. We look at the challenges involved with its efficient computation and the frameworks developed by some future works like Borgs et al. [2014] that overcame those challenges. We discuss the importance of studying the problem of disrupting diffusion and present some of the existing techniques in literature that are used for mitigating the spread of influence.

In Chapter 3 we do a review of the notion of $\mathcal{ST}(G)$ and $\mathcal{IM}(S)$ introduced by Bharadwaj [2018] and how they are used to formulate the $ICN(k)$ problem. We present our algorithm for $ICN(k)$ problem and show the use of Random Reachable ($RR$) sets to enable an efficient implementation for the algorithm.

In Chapter 4 we introduce the version of the $ICN(k)$ problem that requires the Seed Set Context as input as well - the $ICN(G,k,S)$ problem. We also present some examples for cases where removing the critical node does not necessarily lead to the largest reduction in the influence.
In Chapter 5 we present the experimental results of the comparison between removing the most influential nodes and removing the critical nodes. We also validate our algorithm against real world data related to the Hong Kong protests that was collected from Twitter.

In Chapters 6 and 7, we look at some of the future directions of this work and finally end with the general conclusions from this work.
CHAPTER 2. LITERATURE REVIEW AND RELATED WORK

We study the Linear Threshold and the Independent Cascade models and use them to formally define the Influence Maximization problem. We will see why it took days to compute the solution to the Influence Maximization (IM) problem. We will look at some of the advances in algorithmic computation, that were introduced in the later years, that managed to significantly reduce the running time of the IM Problem through approximations. Once we have seen how we can maximize the influence in a given network, we will turn our attention to the problem of mitigating influence (same as disrupting diffusion). First we will look at why we need to study the problem of disrupting diffusion in a network. Then we will look at existing literature and how they go about disrupting diffusion. This work builds upon Bharadwaj [2018] where they disrupted diffusion in a given network through the removal of nodes. The approach that is the closest to their work disrupts diffusion by removing edges instead of nodes. In the context of specifically controlling the spread of misinformation in OSNs, researchers have studied the introduction of competing cascades of true information in the network to mitigate the spread of misinformation. We will look at some techniques introduced by them as well. We also discuss some Machine Learning based approaches that identify and remove the source of the misinformation itself.

2.1 Modelling a Social Network

We model a social network as a directed graph $G = (V, E)$. The set of nodes $V$ depicts the people or the entities being studied in the social context. The edges $E \subseteq V \times V$ is the set of directed edges connecting a pair of nodes.

Each node $v \in V$ is assumed to be in either one of two states - active or inactive. Intuitively, a node is active if it has accepted/infected-by the information/idea/disease spreading through the social network, and is inactive otherwise.
We also assume that the model is progressive. This means that nodes can switch from being inactive to being active, but can not switch in the other direction. In contrast, models in which nodes may switch back and forth between active and inactive states are called non-progressive models. Progressive models are typically used to model the diffusion of the adoption of new technologies or products such as buying a new smart phone, or watching a new movie, since these adoptions are typically associated with a purchase behavior and are not easily reversible. Non-progressive models, on the other hand, can be used to model the diffusion of ideas and opinions, such as the attitude towards a news event or the support of different political proposals, which may switch back and forth based on new information gathered from the network.

At a time step $t$, the set of nodes that are active are known as the activated set of nodes. The activated nodes at time step $t = 0$ are referred to as the seed set $S$. The seed set can be understood as the set of nodes from which the diffusion process starts. Often the seed set is the set of nodes that are manually selected to initiate the influence diffusion process in a social network. Think about a company paying a small number of individuals $p$ to tweet about their product. $p$ becomes the seed set in this case.

Thus, the diffusion process from the perspective of an initially inactive node $v$ looks like this: at each subsequent time step, more and more of $v$’s neighbors become active; at some point, this may cause $v$ to become active, and this may in turn trigger nodes to which $v$ is connected to, to become active. The diffusion process terminates when there is no new node that can be further activated.

### 2.2 Diffusion Models

In their seminal work, Kempe et al. [2003] introduced two basic diffusion models - Independent Cascade (IC) model and the Linear Threshold (LT) model. Our work focuses only on the IC model. Below we give a brief description of the 2 models.
2.2.1 IC Model

Associated with each directed edge \((u, v) \in E\), there is a probability \(p_{uv}\). At time step \(t = 0\), we have only the seed set \(S\) that is activated. Suppose a node \(u\) becomes active at the end of time step \(t - 1\). At the beginning of time step \(t\), \(u\) is given a single attempt to activate each of its inactive outgoing neighbors \(v\) with a probability of \(p_{uv}\). If \(u\) succeeds, \(v\) will be considered as newly activated at the start of time step \(t + 1\). Whether or not \(u\) succeeds, it cannot make any further attempts to activate \(v\) in subsequent rounds.

Below, we give a simple example of diffusion in the IC model. The figure corresponding to this is Fig 2.1-2.4.

Suppose our seed set is \(S = \{A, C\}\).
At the start of time step \(t = 0\), the activated nodes are \(A\) and \(C\). During time step \(t = 0\), \(A\) will try to activate its currently inactive neighbors \(B\) and \(D\). Similarly, in the same time step, \(C\) will try to activate it’s currently inactive neighbors \(B, G,\) and \(H\). Notice that \(B\) has multiple newly activated neighbors trying to activate it. In this case, \(A\) and \(C\)’s attempts will be sequenced in an arbitrary order. Suppose \(A\) manages to activate \(B\) but not \(D\). When it’s \(C\)’s turn, \(C\) won’t try to activate \(B\) since \(B\) has already been activated. Suppose \(C\) could not activate either \(G\) or \(H\).

At the start of time step \(t = 1\), the newly activated nodes are only \(B\). During time step \(t = 1\), \(B\) will try to activate its inactive neighbors \(E, F,\) and \(G\). Suppose \(B\) manages to activate both \(E\) and \(G\).

At the start of time step \(t = 2\), the newly activated nodes are \(E\) and \(G\). During time step \(t = 2\), since they have no neighbors, they aren’t able to activate any new nodes.
At the start of time step \(t = 3\), since there are no newly activated nodes, the diffusion stops. Thus at the end of the diffusion process, the activated nodes are \(\{A, C, B, E, G\}\).

In IC model, diffusion of information between each pair of nodes \((u_1, v_1)\) is mutually independent of every other pair of vertices \((u, v)\). This makes the IC model suitable for modelling behavior like an epidemic spread or spread of some information where exposure to just one source might be enough to activate a particular individual.
Figure 2.1  Time-Step-0

Figure 2.2  Time-Step-1

Figure 2.3  Time-Step-2

Figure 2.4  All-Activated-Nodes
Another way to visualize the IC Model is as follows: given a graph $G = (V, E)$, a seed set $S$, and a probability $p_{uv}$ associated with each edge $(u, v)$. For each edge $(u, v)$, toss a biased coin with probability $p_{uv}$ of it returning heads. If the coin returns head, consider that edge to be a live edge, else consider it to be a blocked edge. Now, in order to find the set of activated nodes, return the set of nodes that are reachable from the seed set $S$ by traversing only those edges that were considered to be live edges.

### 2.2.2 LT Model

Unlike IC model, where a single exposure can be sufficient to activate a node, there are some social networks that require exposure from multiple individual sources for a specific individual to change their behavior. For example, consider the case of people waiting to leave a long boring lecture, sitting impatiently, only because others haven’t yet left. People vary in their thresholds which - among a combination of multiple personality traits - is related to how many people have left the lecture thus far (Granovetter [1978]).

The Linear Threshold (LT) model was developed by Kempe et al. [2003] as a generalization for such behavior. In this model, a node $v$ is influenced by each incoming neighbor $u$ according to a weight $b_{u,v}$. The weights are all normalized such that $\sum_u b_{u,v} \leq 1$ where $u$ has an outgoing edge to $v$. Every node $v$ chooses at random a threshold $\theta_v$ from the interval $[0, 1]$. The random selection of $\theta_v$ from 0 to 1 reflects our lack of knowledge of the individuals’ internal thresholds.

Then the diffusion proceeds as follows: just like the IC model, we start with a seed set $S$ that are active at the start of time step $t = 0$. At any time step $t$, for every inactive vertex $v$, we activate $v$ if $\sum_u b_{u,v} \geq \theta_v$ such that $u$ has an outgoing edge to $v$ and $u$ is active at end of time step $t - 1$. Similar to IC model, the process stops when there are no newly activated nodes.

Below, we give a simple example of diffusion in the LT model. The figure corresponding to this is Fig 2.5-2.7. The numbers on the edges correspond to the weights and the numbers next to the vertices correspond to the thresholds $\theta_v$. 
Suppose our seed set is $S = \{A, C\}$.

At the start of time step $t = 0$, the activated nodes are $A$ and $C$. During time step $t = 0$, $A$ and $C$ combined manage to activate $B$ because their total edge weight of $0.3 + 0.2$ becomes greater than the threshold of $B$, which is 0.4. There can be no other activations possible at this stage and hence we move to the next time step.

At the start of time step $t = 1$, $B$ and $C$ can now jointly activate $G$ because their edge weights of $0.3 + 0.1$ now exceed the threshold of $G$. $B$ also manages to activate $F$, but not $E$.

At the start of time step $t = 2$, the newly activated nodes are $F$ and $G$. No new activations are possible in this time step.

At the start of time step $t = 3$, since there are no newly activated nodes, the diffusion process stops.

Thus at the end of the diffusion process, the activated nodes are $\{A, C, B, F, G\}$.

### 2.2.3 Variations on the LT and IC Models

Both of the models presented above have been widely studied in literature. They have also spawned other variations of these models that are used to model variations of the diffusion process.

We have the **Competitive Linear Threshold (CLT) Model** that tries to model diffusion of information when there are 2 competing cascades in the network. One cascade is considered as the +ve cascade whereas the other one is considered as the -ve cascade. In this case, each node is assigned two thresholds $\theta_v$ - one corresponding to the +ve cascade and another to the -ve cascade. Each edge weight $b_{u,v}$ is also separated into two different edge weights - one for the +ve and another for the -ve cascades to model the fact that the two cascades might have different propagation rates (He et al. [2012]).

Along the same lines of the CLT model, we have the **Competitive Independent Cascade (CIC) Model**. The underlying idea is the same - each edge $(u, v)$ has two different propagation probabilities $p_{uv}$ - one for the +ve cascade and another for the -ve cascade. We don’t need to assign thresholds to the nodes in this case since we are dealing with a variation on the IC model (Budak et al. [2011]).
Figure 2.5  Time-Step-0

Figure 2.6  Time-Step-1

Figure 2.7  Time-Step-2
In the CLT and CIC models, it is assumed that a node can exist in 3 states - activated by the +ve cascade, activated by the -ve cascade, and inactive. Both the models are progressive - meaning once a node becomes activated (either by the +ve or -ve cascade) it does not become inactive again. There is another restriction - a node that has been activated by the +ve cascade cannot “switch” over to being activated by the -ve cascade and vice-versa. It is this rule that gives these kinds of models differentiation from just having two different cascades separately in two different networks. By activating a node \( v \) with +ve cascade, we have, in a way, blocked the -ve cascade from activating \( v \). This is also one of the methods employed by some of the other works that deal with misinformation containment in OSNs. The -ve cascade can be considered to be a rumor that is spreading in a given OSN. To control the number of nodes activated by the -ve cascade, we introduce a +ve cascade, which can be considered to be the true information, into the network so that the number of nodes activated by the -ve rumor is minimized. In this model, there is also the additional complexity of a tie-breaking rule that needs to be considered. The two competing cascades might simultaneously arrive at a node \( v \) and manage to activate it in the same time step. At this point it becomes \( v \)’s prerogative to decide which cascade it wants to be activated by. Giving cascades priority then develops other models.

A recent work by Tong et al. [2018] considers a general version of the problem. They assume for one topic there are two groups of cascades - misinformation cascades and positive cascades. Each of the individual members of these 2 groups are different information cascades because they have different sources (different seed sets) and exhibit different levels of reliability (have different propagation probabilities). So now when competing cascades arrive at an inactive node \( v \) and manage to activate it in the same time step, \( v \) has to choose, from all of the cascades, the one cascade it should be influenced by. The authors extend the IC model and introduce the concept of cascade priority which defines a priority among the different cascades with respect to each vertex \( v \).

Note that the models described above do not lend to efficient algorithms. Hence many works, when using these models, consider a simplified version of the model where the propagation probabili-
ties of the +ve cascade and the -ve cascade are considered to be equal along with some model-specific approximations.

The above models assume infinite time of diffusion - i.e. the process continues until there are no new nodes left to activate. But often we might be interested in maximizing the number of activated nodes within a given time limit. For example, a corporation might want to promote its surprise 2-day sale. A different model that takes into account the time delay might be required for this purpose because we are only interested in maximizing the number of activated nodes within the 2-day period. Any nodes activated after the 2-days are inconsequential to us. Liu et al. [2012] and Chen et al. [2012] introduced some models to factor in the time aspect. Liu et al. [2012] introduced the Latency-Aware Independent Cascade Model. In this model when a node \( u \) is first activated at time step \( t \), it activates its currently inactive neighbor \( v \) in time step \( t + \delta_t \) with a probability \( P_{u,v} * P_{u}^{\text{lat}}(\delta_t) \) where \( P_{u,v} \) is the usual activation probability associated with edge \((u,v)\) and \( \delta_t \) is the influencing delay and is randomly drawn from the delay distribution \( P_{u}^{\text{lat}} \).

In this work, we will be dealing with the original IC model proposed by Kempe et al. [2003].

2.3 Influence Maximization Problem

Richardson and Domingos [2002] were the first to study influence maximization as an algorithmic problem, motivated by viral marketing applications. The problem was tackled by means of a probabilistic model of interaction based on Markov Random field. Heuristics were given for choosing the users to market to that maximised the expected lift in profit. Lift in profit, intuitively, was the difference between the expected profit obtained by employing a marketing strategy and the expected profit obtained using no marketing at all.

Kempe et al. [2003] defined influence of a set of nodes \( S \), denoted \( \sigma(S) \), to be the expected number of activated nodes at the end of the diffusion process. The set \( S \) is hence the seed set in this case because it is the set of nodes that is active at time step \( t = 0 \) - diffusion starts from this set \( S \). The Influence Maximization Problem then asks: given a graph \( G = (V,E) \), and a budget \( k \), find a seed set \( S \) of size \( k \) such that the \( \sigma(S) \) is maximised. Kempe et al showed that this problem
is NP-Hard. They also showed that the influence function $\sigma(\cdot)$ is submodular and monotone. A submodular function can be intuitively understood as the diminishing returns property, i.e., the marginal gain from adding an element to a set $S$ is at least as high as the marginal gain from adding the same element to a superset of $S$.

Formally, a function $f$ is \textit{submodular} if it satisfies:

$$f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$$

for all elements $v$ and all pairs of set $S$ and $T$ such that $S \subseteq T$. A function is said to be monotone if adding an element to a set cannot cause the value of the function being calculated on that set to decrease.

Formally, a function $f$ is \textit{monotone} if it satisfies:

$$f(S \cup \{v\}) \geq f(S)$$

for all sets $S$ and all elements $v$.

Owing to the result from Nemhauser et al. [1978] the General Greedy Algorithm allows us to get a $(1 - 1/e)$ approximation to any set function $f$ that is submodular and monotone. Formally:

$$f(S_G) \geq \left(1 - \frac{1}{e}\right) f(S^*)$$

where:

Set $S^* = \arg \max_{|S| \leq k} f(S)$

Set $S_G$ is the greedy solution returned from the General Greedy Algorithm

$e$ is the base of the natural logarithm.
Algorithm 1: General Greedy Algorithm

**Input:** budget $k$, submodular and monotone function $f$

**Output:** Required solution set $S$

1. Initialize $S = \emptyset$
2. for $i = 1$ to $k$
3. $u = \arg\max_{v \in V \setminus S} (f(S \cup \{v\}) - f(S))$
4. $S = S \cup \{u\}$
5. return $S$

From Kempe et al, we know the influence function to be, both, submodular and monotone in the LT and IC models. Hence we should be able to use the Algorithm 1 to get a simple greedy algorithm that gives us a $(1 - 1/e)$ approximation. Difficulty lies in the fact that the result of Nemhauser et al. assumes that the greedy algorithm can evaluate the underlying function $f$ exactly, which is not the case for the influence function $\sigma(S)$. Line 3 of the algorithm requires us to calculate $\sigma(S)$ which has been proved to be a #P-hard problem (Chen et al. [2010]). Kempe et al got around this problem by using Monte Carlo simulations of the diffusion process and then taking the average of the number of activated nodes over all the diffusion processes. Since the Monte Carlo simulations return an approximate value of the number of activated nodes, the approximation guarantee was changed to account for it. Taking all this to account, the authors finally showed that the greedy hill climbing algorithm stated as Algorithm 1 gave a $(1 - \frac{1}{e} - \epsilon)$ approximation. Hence we get the following result, assuming a given budget $k$:

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

where:

- set $S_G$ is the solution to the IM Problem as returned by the General Greedy Algorithm
- $S^*$ is the optimum solution of the IM Problem
- $e$ is the natural logarithm base
- $\epsilon > 0$ depends on the accuracy of the Monte-Carlo estimate of the influence spread given a seed set.
The drawback of the above algorithm using Monte Carlo simulations is that it is very slow, reportedly taking hours on a modern-ish server to select 50 seeds in a moderate sized graph (15K nodes and 31K edges). This makes it infeasible for direct application to modern OSNs which contain millions of nodes and edges.

To solve this problem, techniques were introduced to estimate the function using sampling techniques. The most successful framework was that proposed by Borgs et al. [2014]. We use this framework in our algorithm as well. Below we provide an intuitive explanation of their work. Suppose the problem was to find a single node in the graph $G$ that has the highest influence. In the IM problem, we saw that one way to do this is by using Monte Carlo simulations. We also saw that correct estimations require multiple simulations which take a lot of time. Borgs et al suggested an alternate way to view this. Their process is divided into 2 steps:

a) uniformly at random select a node $v \in V$

b) simulate a single diffusion process from $v$ in a reverse direction using the transpose of the original graph $G$, $G^T$, keeping each incoming edge $(u,v)$ in the original graph $G$ with probability $p_{uv}$, until no node can be further reached, and collect the set $S_v$ of the traversed nodes.

After each sampling, we obtain a sample $S_v$ which is a subset of the nodes. The idea is that all of the nodes present in $S_v$ can influence $v$ in the original graph $G$. Then, given a collection of such samples $S_v$, the most influential node would be the node $u$ that “covers” the maximum number of samples and we return $u$ as the most influential node.

They show that the probability that a node $u$ appears in a set $S_v$ is proportional to $E[\sigma(u)]$. For this, they repeatedly apply the random sampling technique described above to generate a sparse hypergraph representation of the network. A hypergraph edge corresponds to a set of individuals that was influenced by a randomly selected node in the transpose graph. This hypergraph encodes the influence estimates: for a set of nodes $S$, the total degree of $S$ in the hypergraph is approximately proportional to the influence of $S$ in the original graph.

Suppose we were provided with an edge-weighted directed graph $G$ and a seed set $S$. Let $m$ and $n$ denote the number of edges and nodes in the graph $G$. We write $g \sim G$ to mean that $g$ is drawn
from the random graph distribution $G$. Given a set of vertices $S$ and a directed graph $g$, $C_g(S)$ represents the set of nodes that are reachable from $S$ in $g$. Thus $C_g(S)$ is the set of nodes influenced by $S$ in $g$, i.e. $\sigma_g(S)$ will be $|C_g(S)|$. We also write $E_G[\sigma(S)] = E_{g\sim G}[\sigma_g(S)]$ which is the expected value of the influence of a set of nodes $S$. One of their main observations is that the influence of a set of nodes $S$ is precisely $n$ times the probability that a randomly selected node $u$ influences any node from $S$ in the transpose graph $g^t$.

$$E_{g\sim G}[\sigma_g(S)] = \sum_{u \in g} Pr_{g\sim G}[\exists v \in S \text{ such that } u \in C_g(v)]$$

$$= \sum_{u \in g} Pr_{g\sim G}[\exists v \in S \text{ such that } v \in C_{g^r}(u)]$$

$$= n \cdot Pr_{u,g\sim G}[\exists v \in S \text{ such that } v \in C_{g^r}(u)]$$

$$= n \cdot Pr_{u,g\sim G}[S \cap C_{g^r}(u) \neq \emptyset]$$

This implies that we can estimate the $E_G[\sigma(S)]$ by estimating the probability of the event $S \cap C_{g^r}(u) \neq \emptyset$. Note that this sampling method allows us to estimate the influence of a node, but not the marginal gain of adding a node to a partially constructed seed set. Hence, for returning the max seed set of size $k$, we iteratively select the node $v$ that has the highest degree (in the manner that the hypergraph is constructed, this selects node with the highest expected influence as well) and add it to the seed set $S$. During each iteration, once we have selected and added $v$ to the max seed set $S$, we perform an additional step where we remove all of the incident edges from $v$.

Given a Graph $G = (V,E)$ and an integer $R$, we generate $R$ Random Reachable Sets (RR Sets) where each RR set corresponds to a sample $S_v$. The above result then allows us to approximate the influence of a node $v$ as:

$$\hat{\sigma}(v) = \frac{c}{R} \cdot n$$

where:

c is the number of RR Sets that contain the node $v$
$R$ is the number of RR Sets generated

$n$ is the number of nodes in the graph $G = (V,E)$

It follows from Chernoff bounds that $\hat{\sigma}$ approximates $\sigma$ with relative error $\epsilon$ when $N = O(|V|/\epsilon^2)$. We will use the same framework to generate RR Sets, both, to estimate influence in a graph, and to estimate the importance of a particular node to the diffusion process.

2.4 Why should we study the problem of Disrupting Diffusion

So far we have been looking at the propagation of diffusion in a network and the problem of selecting a seed set of fixed size such that the number of activated nodes in the network at the end of the diffusion process is maximized. But the objective function might not always be to maximize influence, it might sometimes also be related to mitigating the spread of influence. Consider a disease spreading through a community, or a malware spreading over the internet, or a rumor in OSNs - all of these scenarios require us to develop techniques that would minimize the number of infected/activated nodes. There can be multiple ways to go about doing this and we look at some of them in the next section.

2.5 Managing and Controlling Information Diffusion

The problem of mitigating the spread of diffusion in a network has garnered major attention in the recent years due to the proliferation of fake news and misinformation on OSNs (Sharma et al. [2019], Kumar and Shah [2018] ). One of the proposed applications of this work can be a way to mitigate the spread of misinformation on OSNs. Hence it is of interest to see what are some other techniques that the existing works, that are specifically designed towards blocking misinformation, apply to deal with this problem.

IM Problem deals with the diffusion of a single entity in the network - release of a single movie or promotion of a single product. While interesting in its own right, the problem does not always refer to what we are normally accustomed to seeing on OSNs. More often than not, there are 2 or more competing products, ideas, or pieces of information that are propagating through a network. True
information (+ve cascade) and misinformation (-ve cascade) can be viewed as 2 such competing cascades. Intuitively, the way that misinformation spread is controlled is this - detection of a -ve cascade in a network would require us to select a seed set of size \( k \) such that when these \( k \) nodes are selected as the seed for the +ve cascade, the number of nodes activated by the -ve cascade is minimized. The idea is that introducing the +ve cascade in the network would cause a certain section of nodes (that, in the absence of the +ve cascade, would have gotten activated by the -ve cascade and would have hence been influenced by the misinformation) to get activated by it. Once a node becomes activated by a particular cascade it does not “switch” over to being activated by the competing cascade. Hence, if a node gets activated by a +ve cascade - it stays activated by the +ve cascade, and thus we have managed to mitigate the spread of misinformation in the network.

The works that use these techniques normally use some variation of the LT and IC models as described in section 2.2.3. This idea forms the basis of works like Budak et al. [2011] and He et al. [2012]. They consider variations on the IC and the LT model respectively. As described on page 14, a similar work done by Tong et al. [2018] looks at a general version of this scenario.

Another approach based on similar lines, but slightly different, was proposed by Nguyen et al. [2012]. Instead of introducing a competing cascade, they focused on decontaminating the nodes that were infected by the misinformation. They designed a greedy algorithm to select the best set of seed users from which to start the diffusion process (once the entire diffusion from the false cascade had been completed) for true news, so that at least a \( \beta \) -fraction of the users infected by the misinformation can be decontaminated.

Another line of work tries to mitigate diffusion by identifying the misinformation at the source itself. These techniques use Content-Based identification by using Machine Learning algorithms that use Convolutional Neural Networks to predict if a particular piece of text might be fake or not. Their argument is that the textual content in fake news differs from that in true news in some quantifiable way. This difference could be in terms of language, writing style, and/or sentiment. So if we have a information about which sources are more likely to try and spread fake news, we
can monitor them, and flag content being presented by these sources as false or true, thus enabling early detection of fake news. These works can be found in Wang [2017], Qian et al. [2018].

In the work presented by Khalil et al. [2014], the authors focus on removing edges for disrupting diffusion in linear threshold model. They prove that the function $f(E) = \sum_{v \in V} \sigma_{G/E}(v)$, where $E$ is a set of edges and $G/E$ corresponds to the network $G$ with edges in $E$ removed, is a supermodular function. The objective of disruption is achieved by minimizing $f$, which involves maximizing the negation of $f$—negation of $f$ being a submodular function. In short, the critical edge identification problem in linear threshold diffusion model reduces to maximizing a submodular function.

This work builds up on the techniques introduced by Bharadwaj [2018]. The authors in Bharadwaj [2018] studied another way of disrupting diffusion which was by changing the underlying topology of the graph by the deletion of nodes. The authors in Bharadwaj [2018] proved that if independent cascade diffusion model is considered, the optimization function is neither submodular nor supermodular, unlike the supermodularity obtained in Khalil et al. [2014]. Work done in Bharadwaj [2018] was also different from works that tried to mitigate diffusion by introducing additional cascades of information in the network in the sense that Bharadwaj [2018] mitigated diffusion only through changing the underlying diffusion network structure itself. No additional cascades of information were introduced. Unlike the works discussed that required identifying the source of the misinformation, the authors in Bharadwaj [2018] assumed no knowledge of the source of misinformation, i.e. without knowing where the misinformation might start from, pre-emptively, which are the best nodes to remove from the network in such a way that the expected diffusion from any possible seed is minimized. In the next chapter we present a detailed discussion of the work that was done by the authors in Bharadwaj [2018].
CHAPTER 3. DISRUPTING DIFFUSION: FORMULATION BASED ON CRITICAL NODES

We present a detailed discussion of critical nodes as introduced in Bharadwaj [2018]. We study the notion of criticality and the problem of Identifying the Critical Nodes (ICN Problem) in the network in the terms of strength and impact. We also provide the NP-Hardness proof for the ICN problem. We discuss the algorithm that we will be using for identifying critical nodes in a network. We first list out the simple version of the algorithm and talk about what makes it inefficient. We then use the sampling techniques introduced in the earlier section to come up with a faster version of the same algorithm. We also give the theoretical background about how we could design efficient data-structures for this faster version of algorithm to make computations more efficient.

3.1 Illustrative Example of Disrupting Diffusion by using Critical Nodes

Consider the graph showed in Figure 3.1. The node 4 has edges to nodes numbered from 5 up through to 99. The model that we are using here is the IC Model. Hence each edge in the network has to be associated with a probability that captures the probability of a node $u$ activating another node $v$. For this example, we assume this probability to be 1 for simplicity. The direction of the edge indicates that the diffusion moves from $u$ to $v$. For this example, we assume this probability to be 1 for simplicity. Following the IC model, each node gets one chance to influence its neighbors, which it does with a probability of 1. The objective to identify one node $v$ in the graph $G$ such that when $v$ is removed from $G$, the resulting influence from any possible seed is maximally reduced. Let’s call this node $v$ as a critical node.

So, given this objective to identify one such critical node in the graph $G$ in figure 3.1. Which node should we select? Let’s compare 2 strategies. One strategy would be to assume that the most influential node in the network is also the critical node in the network. So let’s remove the most
influential node and that should cause maximum reduction in the influence from any possible seed. In Fig. 3.1, the most influential node is 0 because it can influence the entire network. So we remove the node 0 and we get the resulting graph shown in 3.2. (Note that when we talk about removing a node, we are in effect removing all of the incident edges on it.) But notice that in this graph, if we select our seed to be 2 or 3, we still manage to activate a large portion of the network i.e. 97 nodes.

Now notice what happens when we select the critical node to be 4 instead. As before, we remove 4 from the original network shown in 3.1 and get the graph in 3.3. Now in this network, the maximum number of nodes that you can influence is only 4 and that is if you select node 0 as your seed. This shows 2 things - first, identifying the critical node in a graph is a different problem from the IM Problem. Second, the fact that removing 4 causes a greater reduction in influence compared to removing 0 shows that 4 is “more” critical to the diffusion process when compared to 0. This means that with the right kind of formulation, we can “measure” the criticality of a vertex. This is what we do in the next section where we formally introduce the notion of critical nodes.

### 3.2 Formalizing Criticality

We have been given a directed graph $G = (V, E)$ where $V$ is a finite set of nodes and $E : V \times V \rightarrow [0, 1]$ is a directed edge relation between nodes annotated with a probability measure. The direction in the edge $u \xrightarrow{p_{u,v}} v$ indicates the direction of diffusion from $u$ to $v$ and the annotation $p_{u,v}$ indicates the probability (propagation probability) of that diffusion, i.e., $u$ can activate $v$ with a probability of $p_{u,v}$. An undirected edge is viewed as a bi-directional edge with the same propagation probability in both directions. Our objective is now to find the critical nodes in this network. As we saw in the example in the section 3.1, identifying a critical node is a different problem from identifying the most influential node.
Figure 3.1  Entire Graph

Figure 3.2  Node with the maximum influence removed

Figure 3.3  Critical Node removed
3.3 Re-characterization of Critical Nodes as “Impact”-ful Nodes

The first formulation of the ICN problem is given as follows. We will change this formulation later because of certain limitations.

**Problem 1** (Identifying Critical Nodes Problem (ICN)).

Given a network $G = (V, E)$ and $k$, the ICN($k$) involves computing a set of $k$ nodes such that removal of these $k$ nodes from $G$ results in a network $G' = (V', E')$ where $\forall S \subseteq V' : \sigma_G(S) - \sigma_{G'}(S)$ is maximized.

Limitations:

a) The notion of criticality, as stated above, is too restrictive. As noted in the introduction on page 3, such a set of critical nodes may not exist in the graph at all.

b) Recall that $\sigma(\cdot)$ was itself an expected value and the way we calculated it was through Monte Carlo simulations. The definition, as stated above, should hold $\forall S \subseteq V'$. If we are using the brute force method outlined on page 3, it makes the function very expensive to compute even for small networks.

To address this the authors of Bharadwaj [2018] introduce the concept of impact of a node(s) and strength of diffusion of a graph $G$.

**Definition 1** (Strength of Diffusion).

Given a network $G = (V, E)$, the strength of diffusion in $G$, denoted by $ST(G)$, is $\sum_{v \in V} \sigma_G(v)$.

For example, for the graph in Figure 3.1, the strength of diffusion is $\sum_{v=0}^{99} \sigma_G(v) = 486$.

Intuitively, the strength of diffusion indicates sum of the expected number of nodes each node in the graph may influence. Thus if the strength of diffusion in a network is high, then it indicates that the network has “many nodes” that can influence a lot of nodes of the network. This can be interpreted as: the network has many good seed sets that can collectively influence a large population of the network. Conversely, if the strength of influence is small, it is an indication that there are no (or very few) seed sets having high influence. Thus if removal of a set of nodes from a
network causes the strength of diffusion to go down, then it indicates the influence of all (or many) seed sets is also reduced. Thus a set of nodes whose removal will cause maximal reduction in the strength of diffusion can be considered as critical nodes. Based on this, they introduced the notion of *impact* as follows.

**Definition 2** (Impact of Node(s)). :

*Given a network* $G = (V,E)$, the impact of $S \subseteq V$, denoted by $IM_G(S)$, is $ST(G) - ST(G/S)$.

The impact, therefore, corresponds to the decrease in the strength of diffusion in the network after removing a set of nodes $S$. Going back to the example in Figure 3.1

- $IM_\{}(\{0\}) = ST(G) - ST(G/\{0\}) = 486 - 386 = 100$
- $IM_\{}(\{4\}) = ST(G) - ST(G/\{4\}) = 486 - 103 = 383$

Using the Definitions 1 and 2, they reformulated Problem 1 as follows:

**Problem 2** (ICN as Identifying Impactful Nodes). :

*Given a network* $G = (V,E)$ and $k$, the ICN($k$) problem involves identifying a set $S \subseteq V$ of size $k$ such that $IM_G(S)$ is maximized.

The authors in Bharadwaj [2018] also showed that the function $IM_G(S)$ is monotonic, neither submodular nor supermodular, but submodular if there is at most one path between any two nodes in $G$.

We will be using the same definitions and problem formulations discussed above in this work as well.

Finally, below we present the NP-Hardness proof of ICN($k$) problem.

**Theorem 1.** ICN($k$) problem (See Problem 2) is NP-Hard.

*Proof.* In Yannakakis [1978], Yannakakis proved that the the problem of removing an optimal number of nodes from a graph resulting in subgraph satisfying some hereditary property is NP-Hard. Hereditary property of a graph is one that is preserved in all induced subgraphs.

In our setting, consider the decision version of ICN($k$): does there exists set $S$ of $k$ nodes, whose removal from the graph $G$ results in $ST(G/S) \leq T$? Note that, the strength of a graph is
an hereditary property, i.e., \( ST(G) \leq A \Rightarrow ST(G/S') \leq A \) for all \( S' \subseteq V \), where \( V \) is the set of nodes in \( G \). Therefore, our decision problem is a member of the NP-Hard class of node removal problems, where the property being considered is hereditary property.

### 3.4 Algorithm to compute Critical Nodes

In the previous section, we defined Impact \( \mathcal{I}_G \) to be as:

\[
\mathcal{I}_G(S) = ST(G) - ST(G/S)
\]

\[
= \sum_{v \in V} \sigma_G(v) - \sum_{v \in V} \sigma_{G/S}(v)
\]

Next we introduce the term marginal gain in the \( \mathcal{I}_G \). Let \( S \) be a set and \( v \not\in S \) be a node, then the marginal gain in terms of \( \mathcal{I}_G \) is defined as follows:

\[
\text{imgain}_G(S, v) = \mathcal{I}_G(S \cup \{v\}) - \mathcal{I}_G(S)
\]

In the context of IM problem, where the influence function is monotonic and submodular, we saw how Kempe et al. [2003] proposed a greedy algorithm with \((1 - 1/e)\) approximation guarantee. The authors in Bharadwaj [2018] established that \( \mathcal{I}_G \) function is neither submodular nor supermodular. So a greedy strategy would not give the approximation guarantee that we need, but the greedy strategy could still serve as a viable heuristic for a general network. We use this to present our first version of the algorithm to solve the \( ICN(k) \) problem.

**Algorithm 1:** Greedy Computation of Critical Nodes

```plaintext
input : Network \( G = (V, E) \) and \( k \)
output: \( S \subseteq V \)
1 GreedyImpact
2 \( S = \emptyset \)
3 while \( |S| < k \) do
4     \( w = \arg \max_{v \in V} \text{imgain}_G(S, v) \)
5     \( S = S \cup \{w\} \)
6 end
7 return(\( S \))
```
The costliest step in the above algorithm is step 4. It requires us to calculate the vertex $v$ that gives us the maximum marginal gain in $\mathcal{I}M_G(S)$. So we will have to cycle through all the vertices in the the graph $G$ to find the one with the maximum value of the function $imgain_G(S, v)$. The calculation of $imgain_G(S, v)$ - for each new vertex being considered - itself requires the calculation of $\mathcal{I}M_G(S \cup \{v\})$ which in turn requires the calculation of $\mathcal{S}T(G/(S \cup \{v\}))$ which in turn requires the computation of $\sum_{u \in V} \sigma_{G/(S \cup \{v\})}(u)$ and we know that computing $\sigma(S)$ is $\#P$-hard problem even when the set $S$ contains a single element as shown by Chen et al. [2010], Chen et al. [2013].

To get around this problem, we now turn back to the Random Reachable Sets framework ($RR$ sets) that was introduced by Borgs et al. [2014] and we looked at back on page 18. To recap, what we observed was that the marginal gain in influence due to a vertex $v$ with respect to some set $S$ can be computed by considering the number of $RR$ elements which contains $v$ but none of the elements of $S$. Then, computing the next vertex $v$ that provided the largest marginal gain in influence was done as follows: at each iteration identify the vertex $v$ that covers the maximum number of existing $RR$ sets and remove all the $RR$ elements that that vertex $v$ covers before proceeding to the next iteration. Repeat this process until you have the required $k$ number of nodes and that becomes your max seed set.

In the following section, we will present the strategy that we use to compute the marginal gain in impact due to a vertex with respect to a given set using random reachable set.

### 3.4.1 Impact Computation using Random Reachability

In our context, we need to compute the impact of a set $S$. As defined earlier:

$$\mathcal{I}M_G(S) = \mathcal{S}T(G) - \mathcal{S}T(G/S)$$

$$= \sum_{v \in V} \sigma_G(v) - \sum_{v \in V} \sigma_{G/S}(v)$$

So, in order to compute Impact of a set of nodes $S$ efficiently, we need a way to compute the influence function $\sigma(\cdot)$.

Let $N$ indicate the total number of $RR$ sets generated.
Let $M^v$ indicate the number of elements in $RR$ sets that contains $v$.

Let $M_S^v$ indicate the number of elements in $RR$ sets that contain $v$ such that there is at least one path to $v$ that does not contain any node from $S$.

Let $M_S^v$ indicate the number of elements in $RR$ sets that contain $v$ such that every path to $v$ involves some node in $S$.

As per the result of Borgs et al. [2014], we know:

$$\hat{\sigma}_{G/S}(v) = |V| \times \frac{M^v}{N}$$

$$\hat{\sigma}_G(v) = |V| \times \frac{M^v}{N}$$

$$\therefore \hat{\sigma}_G(v) - \hat{\sigma}_{G/S}(v) = \frac{|V|}{N} \times (M^v - M_S^v) = \frac{|V|}{N} \times M_S^v$$

Impact $\mathcal{IM}_G$ can then be re-written as -

$$\mathcal{IM}_G(S) = \sum_{v \in V} \sigma_G(v) - \sum_{v \in V} \sigma_{G/S}(v)$$

$$= \sum_{v \in V} (\sigma_G(v) - \sigma_{G/S}(v))$$

$$\sim \sum_{v \in V} (\hat{\sigma}_G(v) - \hat{\sigma}_{G/S}(v))$$

$$= \sum_{v \in V} \frac{|V|}{N} \times M_S^v$$

$$= \frac{|V|}{N} \sum_{v \in V} M_S^v$$

Therefore, $\mathcal{IM}_G(S)$ can be estimated by counting, for every vertex $v$ in the Graph $G$, the number of $RR$ sets in which every path to $v$ contains some node from $S$, i.e., reachability of $v$ depends on $S$.

### 3.4.2 Incremental Computation of Marginal Gain in Impact

$$\mathcal{IM}_G(S) = \frac{|V|}{N} \sum_{v \in V} M_S^v$$ allows us to estimate the Impact of a given set $S$. But the line 4 in our Algorithm 1 requires us to calculate the $\text{imgain}_G(S, v) = \mathcal{IM}_G(S \cup \{v\}) - \mathcal{IM}_G(S)$, which is the
incremental gain in the impact by adding a new node $v$ to an existing set of nodes $S$.

$$\text{imgain}_G(S, v) = IM_G(S \cup \{v\}) - IM_G(S)$$

$$= \frac{|V|}{N} \sum_{u \in V} M^u_{S \cup \{v\}} - \frac{|V|}{N} \sum_{u \in V} M^u_S$$

$$= \frac{|V|}{N} \left( \sum_{u \in V} M^u_{S \cup \{v\}} - \sum_{u \in V} M^u_S \right)$$

$$= \frac{|V|}{N} \sum_{u \in V} \left( M^u_{S \cup \{v\}} - M^u_S \right)$$

$M^u_{S \cup \{v\}} - M^u_S$ is equal to the difference between number of graphs in $RR$ where reachability of $u$ involves $v$ or some elements in $S$ and number of graphs in $RR$ where reachability of $u$ involves some elements in $S$. Therefore, $M^u_{S \cup \{v\}} - M^u_S$ is the number of graphs in $RR$ where reachability of $u$ involves $v$ and does not involve any element from $S$.

Incremental computation of $\text{imgain}_G(S, v)$ (and avoid computing $IM_G(S \cup \{v\})$) is realized as follows. Once $IM_G(S)$ is computed using $RR$ set, we remove all elements of $S$ from each $G^i_r \in RR$. After removal, $|V| \times M^u_v / N$ for all $u \in V$ is equal to $M^u_{S \cup \{v\}} - M^u_S$, which, in turn, results in incremental computation of $\text{imgain}_G(S, v)$. Algorithm 2 outlines the method using $RR$ sets.

**Algorithm 2: Greedy using Random Reachability**

```
input : Network $RR = \{G^1_r, G^2_r, \ldots, G^n_r\}$ and $k$

output: $S \subseteq V$

1 $S = \emptyset$

2 while $|S| < k$ do

3     $w = \arg \max_{v \in V} \sum_{u \in V} M^u_v$

4     $S = S \cup \{w\}$

5     Remove $w$ from $RR$ graphs

6 end

7 return($S$)
```
3.4.3 Efficient Implementation of Incremental Computation

Note that, the implementation of incremental computation has two efficiency bottlenecks. For the incremental computation one needs to perform reachability on each graphs in $RR$ set in every iteration. To counter this bottleneck, we develop a data structure that succinctly captures the reachability information in each graphs of $RR$ set and present effective algorithms to construct and maintain the structure, and minimize the re-computation of reachability.

For each node $v \in V$ and for each graph $G_i^r$ in $RR$ set, we maintain a set $\text{dependOn}(v, i) \subseteq V$. The set contains the nodes such that their reachability requires $v$ in $G_i^r$. If $U$ is the set of nodes in $G_i^r$, then $\text{dependOn}(v, i)$ can be computed by subtracting from $U$ the nodes that are reachable in $G_i^r$ after removing $v$. The impact of $v$ proportional to $\sum_{i=1}^{N} \text{dependOn}(v, i)$ (equal to $\sum_{u \in V} M^u_v$).

Updating $\text{dependOn}$ for Incremental Computation. In order to facilitate incremental computation of marginal gain of impact, $\text{imgain}$, the $\text{dependOn}(w, i)$ must be updated for all $w \in V$ and $i \in [1, N]$ once a node $v \neq w$ with the highest impact is selected to be part of the solution. Incrementality requires the removal of $v$ and recomputation of reachability in $G_i^r$. This repeated reachability can be avoided by the following update operation on $\text{dependOn}(w, i)$. If $u \in \text{dependOn}(v, i)$ then remove $u$ from all $\text{dependOn}(w, i)$ ($w \neq v$). This is because $v$ in $G_i^r$ impacts $u$ (removing $v$ will make $u$ unreachable in $G_i^r$); reachability of $u$ cannot be any more falsified (impacted) by further considering $w$. This is illustrated in the following example $G_i^r$.

![Graph](image)

The corresponding $\text{dependOn}$ is represented using as matrix, where the first column represent the input and each cell $(r, c)$ is set to 1, if the $c$-th element is present in the $\text{dependOn}$ of $r$-th element.
If $u_2$ is selected as the one with the highest impact\(^1\), then row corresponding to $u_2$, representing the set $\text{dependOn}(u_2, i)$, will be rendered unreachable in $G_i^r$ by the removal of $u_2$.

Secondly, subsequent computation of impact of nodes $u_0, u_1$ and $u_3$ should not consider the unreachable nodes ($u_2, u_4$ and $u_5$), and hence, their entries (if present) are removed from the $\text{dependsOn}$ of $u_0, u_1$ and $u_3$.

Finally, after removal of $u_2$, the impact of some nodes may improve as well. Such nodes are the ones whose reachability does not depend on $u_2$ and which, if removed in the absence of $u_2$, may render some other nodes un-reachable. For instance, in the absence of $u_2$, removal of $u_1$ will render $u_3$ unreachable. Therefore, $\text{dependOn}$ of $u_1$ includes $u_3$ after removal of $u_2$. Such update is realized by only re-computing the $\text{dependOn}$ relationship of all nodes that do not belong to the $\text{dependOn}$ relation of the node being removed ($u_2$ in our example). The resultant matrix is

\[
\begin{array}{c|ccccc}
 & u_0 & u_1 & u_2 & u_3 & u_5 \\
\hline
u_0 & 1 & 1 & 1 & 1 & 1 \\
u_1 & 1 & 1 & 1 & 1 & 1 \\
u_2 & 1 & 1 & 1 & 1 & 1 \\
u_3 & 1 & 1 & 1 & 1 & 1 \\
u_4 & 1 & 1 & 1 & 1 & 1 \\
u_5 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[^1\text{Note that the above simply illustrates one of the } N \text{ random graphs in } RR. \text{ Impact of a node based on the sum of its impact in all the } N \text{ elements.}\]
CHAPTER 4. IDENTIFYING CRITICAL NODES: ALGORITHMS, DATA STRUCTURES, AND PROPERTIES

We fix one RR Graph, $G_i^e$, out of the entire set of generated RR Graphs, and from the perspective of $G_i^e$ explain the entire process of selection of a critical node and the subsequent update of the same $G_i^e$ as per the algorithm 2 once the critical node has been removed. We also introduce the $ICN(G, k, S)$ formulation of the $ICN(G, k)$ problem. Where $ICN(G, k)$ computes the critical nodes in $G$ independent of any seed set information, the $ICN(G, k, S)$ problem requires the seed set context to be provided. Similar to the $ICN(G, k)$ problem, we fix a RR Graph for the $ICN(G, k, S)$ problem and show the selection of a critical node and subsequent update of a specific RR Graph. We also discuss the reasons behind introducing the $ICN(G, k, S)$ formulation. We highlight the challenges in evaluating the results of our algorithm 2. Primarily, given a graph $G$, experimental evaluation of the $ICN(G, k)$ problem requires us to calculate the influence in $G$ before and after removal of the $k$ nodes, and the influence, in turn, depends on the choice of the seed set $S$. The observation here is that the algorithm for $ICN(G, k)$ problem will always output the same set of $k$ critical nodes, irrespective of the seed set $S$ being used. This means that we can “move around” the seed set $S$, as we please, to make the results of the algorithm look good or bad. The reason behind this phenomenon and its subsequent side-effects are discussed here with examples.

4.1 Using dependencyMatrix to capture dependsOn relationship

For implementing the algorithm 2, step 4 requires us to calculate $\max_{v \in V} \sum_{u \in V} M_{uv}^v$. $M_{uv}^v$ is the number of RR Graphs in which reachability of $u$ depends on $v$. $\sum_{u \in V} M_{uv}^v$ is the total number of vertices among all the generated RR Graphs whose reachability depends on $v$. Hence, $\max_{v \in V} \sum_{u \in V} M_{uv}^v$ will find the $v$ with this largest value. Hence, we need a data structure, which for every vertex $v$, keeps track of the total number of vertices, among all the generated RR
Graphs, whose reachability depends on \( v \). We can have a single array, let’s call it \( CritValue \), of size \( n = |V| \) to store \( \sum_{u \in V} M_{uv}^v \) for every node \( v \). At each iteration, we can find the node \( v \) with the largest value \( CritValue[v] \) and return that \( v \) as the selected vertex \( w \) in the line 4 of the algorithm 2.

The problem with this approach is that during the step 5 of Algorithm 2, we have to remove \( w \) from all RR Graphs. Hence for every RR Graph, \( G_i^r \), that contains the vertex \( w \), we will have to first remove \( w \), then recompute and update the \( CritValue \) of every vertex in \( G_i^r/\{w\} \). This is a very expensive operation. Hence we use an auxiliary datastructure to keep track of the \( dependsOn \) relationship - the \( dependencyMatrix \). Every RR Graph is associated with a \( dependencyMatrix \). For a specific RRGraph, say \( G_i^r \), the \( dependencyMatrix \) associated with it captures the \( dependsOn \) relationship. More specifically, if \( dependencyMatrix[v][u] \) is 1, it means that reachability of \( u \) depends on vertex \( v \). It is 0 otherwise.

Consider figure 4.1 and 4.2 for example. Figure 4.1 is a RR Graph and 4.2 is the \( dependencyMatrix \) associated with this RR Graph. Reachability of vertex 4 depends on 3 because if we removed 3, vertex 4 would no longer be reachable from 0. Hence, in the \( dependencyMatrix \) associated with this RR Graph we would set \( dependencyMatrix[3][4] \) to be 1. On the other hand, reachability of 2 does not depend on vertex 5 because if we removed 5, vertex 0 would still be able to reach 2 through \( 0 \rightarrow 3 \rightarrow 4 \rightarrow 2 \). Hence \( dependencyMatrix[5][2] \) is 0.

How do we update the \( dependencyMatrix \)? Refer figure 4.3 and 4.4. Suppose vertex 3 was chosen overall as the node with the highest \( CritValue \). As per the algorithm, the next step would be to remove 3 from all RR Graphs and at the same time update the \( dependencyMatrix \) as well. Updating the \( dependencyMatrix \) and \( G_i^r \) is done in the following 2 steps:

Step 1 - For every vertex \( u \) whose reachability depends on 3, we remove all outgoing edges from \( u \) in \( G_i^r \) and zero out the row and column containing \( u \) in the \( dependencyMatrix \) associated with \( G_i^r \).

Step 2 - For every vertex \( v \) whose reachability does not depend on 3, recalculate the values in the row \( dependencyMatrix[v] \).
Why do we zero out certain rows and columns in step 1? An intuitive reason behind using RR Sets to calculate the impact of a node can be as follows: recall when we were discussing the RR Sets method to calculate influence on page 18, we mentioned that a RR Set gives a representation of the nodes that can influence the source in the original graph $G$. So, in the figure 4.1, it means that all the nodes $\{0, ..., 7\}$ can influence node 0 in the original graph. Then when we say the reachability of vertex 4 depends on 3, i.e. $\text{dependencyMatrix}[3][4] = 1$ what it means is that in order for 4 to influence 0 in the original graph $G$, 3 is a must. Similarly, since $\text{dependencyMatrix}[5][2] = 0$, it means that even if vertex 5 was not present in the original graph $G$, vertex 2 would still have been able to influence the vertex 0. Hence, in this RR Graph, for each vertex $v$, the row $\text{dependencyMatrix}[v][0..7]$ is keeping track of the vertices who are stopped from influencing 0 if that $v$ is removed. The expectation is that if we sum up the criticality value of all the nodes over all of the RR Graphs, then the node that the algorithm returns as critical would be the node whose removal would block the influence from the maximum number of possible vertices.

So, why do we zero out certain rows and columns in step 1? Once vertex 3 has been globally identified as critical and removed from the RR Graph, all of the vertices whose reachability depends on 3 are now no longer reachable from the source and hence this would imply that in the original graph $G$, they can no longer influence the source. Because they can no longer influence the source, they are no longer critical to the diffusion in this RR Graph, and hence their criticality value should be changed to 0 for this RR Graph $G^r_i$. Hence you see that once we delete 3 from the RR Graph, the vertex 4 has its entire row changed to 0 as well - because reachability of 4 depends on vertex 3 and once 3 has been removed, removal of vertex 4 can no longer block the diffusion from any additional vertex and hence any criticality value it had must now be reduced to 0. We can apply a similar logic for zeroing out the columns as well.

Another point to highlight is that just as the $\text{dependencyMatrix}[u][v]$ for a specific $G^r_i$ can change from 1 to 0, the value can also change from 0 to 1. Consider the example figures 4.1 - 4.4. Before 3 is removed, reachability of vertex 2 does not depend on 5 and hence $\text{dependencyMatrix}[5][2]$ is 0. However, once 3 is removed, the reachability of 2 now starts depending on 5. This is re-
flected in our updated dependencyMatrix where dependencyMatrix[5][2] is now 1. This differs from the work in Bharadwaj [2018] where they assumed that the value of dependencyMatrix[u][v]

can change only from 1 to 0 and not the other way round. Intuitively, you can look at the above process in the same way as the discussion in the previous paragraph - in figure 4.1 if 5 was removed, 2 could still influence 0. But once 3 is removed, there is only one path left for 2 to influence 0 and that is through 5 as has been shown in figure 4.3. Hence the criticality of vertex 5 now increases.

Using this datastructure of dependencyMatrix, we can avoid some recomputations - atleast for those vertices whose reachability depends on the vertex being removed. As mentioned above, for vertices whose reachability does not depend on the vertex being removed, we still have to recompute the dependsOn relationship for every other vertex with it. This gives us certain advantages in making the computation faster - albeit at an overhead of space.
4.2 \textit{ICN}(G,k,S) Problem: Using the Seed Set Context to identify Critical Nodes

The algorithm outlined in 2 does not take a seed set as input. Our comparisons in the experimental section make use of an alternative version of the same algorithm where we are also given the seed set information. This also helps us to get a better validation of our algorithm because of the point stated in Section 4.4.1. We call this problem the \textit{ICN}(G,k,S) problem where S is the given seed set context.

The modified version of our algorithm 2 takes an additional input of a seed set \( S \) of size \( k \). We then limit our search of \( w \) in step 4 of Algorithm 2 to only those RR Sets that contain some vertex from the set \( S \). Let this subset of RRSets that are covered by the seed set be \( E, E \subseteq RR \). Each \( G_i^r \in E \) is represented by \( G_i^r \). A node \( v \) in \( G_i^r \) is termed as critical if removing \( v \) from \( G_i^r \) disconnects all of the seed set nodes in \( G_i^r \) from the source. Here source is the random vertex \( u \) that was chosen as the vertex to begin the probabilistic walk from when the RR sets were being generated.

Let \( CritValue \) be an array of size \( n = |V| \) to keep track of the criticality value of \( v \). For each \( G_i^r \in E \) in which removal of \( v \) causes all of the seed set nodes in \( G_i^r \) to become unreachable from the source, \( CritValue[v] \) is incremented by 1. Thus once we have iterated over the entire set of RR, for each vertex \( v \in V \), we have the count of the number of \( G_i^r \)'s in which \( v \) was critical and hence we can now return the most critical node as the vertex \( v \) with the highest count of \( CritValue[v] \).

We also have to update the data structure \( RR \) once we have computed a critical node \( w \) so that we can compute the next vertex that will provide the largest marginal gain in impact. Let \( w \) be the vertex identified as critical in step 4 of Algorithm 2. Step 6 of Algorithm 2 then proceeds as follows: for every \( G_i^r \in E \) that contains \( w \), remove \( w \) from \( G_i^r \). Check if there is some seed set node that is still reachable from the source. If not reachable, we decrease the criticality value, \( CritValue[v] \), of every other vertex \( v \) in \( G_i^r \) (that were termed to be critical w.r.t \( G_i^r \) at the end of previous iteration) by 1. If the seed is still reachable, for every vertex \( v \) in \( G_i^r/{w} \), we check the criticality of \( v \) and update it's \( CritValue[v] \) score if required.
We refer to this algorithm as the Crit-Set algorithm in the results.

The above process can be illustrated using the example shown in Figures 4.5 - 4.8. This graph corresponds to $G_{ic}'$ - it is one of the RR Graphs from the set $RR$ that contains some seed set nodes. In this case, the seed set nodes it contains are vertices 4 and 5. Just like we mentioned in Section 4.1, we use an auxiliary datastructure associated with each RRGraph called $isCriticalVector$ to store information about whether a vertex $v$ is critical - i.e. removal of $v$ causes all of the seed set nodes in the RRGraph to become disconnected from the source.

For example, removal of vertex 2 would cause seed 4 to become disconnected from the source. But seed 5 would still be reachable. Hence removal of 2 does not cause all of the seed set nodes to become unreachable - and hence its criticality value is 0. The node 1 on the other hand has its criticality value as 1.
Suppose if in step 4 of the Algorithm 2, node 2 was chosen overall as the node \( w \). So now we have to remove 2 from this RR Graph. We remove 2 and find that a seed set node is still reachable. Hence we now have to recompute the criticality value for each vertex. We do that and we get the figures 4.7 and 4.8. As can be observed, the criticality of vertex 3 went up from 0 to 1. This is similar to what we had observed in the previous section 4.1 as well. If instead of vertex 2, 1 had been chosen as the critical node overall, then after removing 1 all of the seed set nodes would have become unreachable from the source. In that case, we would have reduced the criticality value of every vertex in this isCriticalVector to 0.

For performing the experiment and the subsequent comparisons, we need a seed set \( S \) that we can provide as an input to this algorithm. For this, we use Borgs et al. [2014] to find the top \( t \) most influential nodes in the input graph \( G \). The experiments have been performed using \( t = 10 \).

### 4.3 Comparison methods for the Crit-Set Algorithm

The Crit-Set algorithm discussed in the previous section is compared with nodes removed as per two other algorithms - the Top-Crit and the Top-Infl algorithms. Below we explain these two algorithms.

#### 4.3.1 Top-Crit: One-Shot approach of selecting Critical Nodes

Computing impact is NP-Hard and neither submodular nor supermodular. This made it difficult to design efficient algorithms. Hence we assumed the \( \mathcal{IM}(S) \) function to be submodular, and used this submodular approximation as a heuristic to build a solution set of \( k \) critical nodes in a greedy manner. This became the Crit-Set algorithm.

Despite the many approximations and frameworks that we introduced from Borgs et al. [2014], the computation of the vertex \( v \) giving the maximum marginal gain in the step 4 of Algorithm 2 is still an expensive process. Hence we can look at another approximation of the \( \mathcal{IM}(S) \) function - the modular approximation. In order to make computations less expensive we assume the impact function to be modular. A modular assumption means that instead of incrementally building the solution set \( S \), we run the \( ICN(G,k,S) \) version of the Algorithm 2 for one iteration and then return the top \( k \) nodes in one-shot. Hence there is no incremental construction of the solution set \( S \). If we refer back to our array data-structure CritValue that we introduced in Section 4.2, this means running the \( ICN(G,k,S) \) version of the Algorithm 2 for one iteration
and then returning the set of top $k$ nodes with the largest $CritValue[v]$. Hence there are no re-computations involved in calculating the vertex with the maximum marginal gain or updating the RR Graphs. We call this One-Shot algorithm the Top-Crit algorithm. The reduction in running time by the Top-Crit algorithm should be evident, albeit what should intuitively seem, at an expense of reduced quality of the final solution set $S$.

4.3.2 Top-Infl: Removing Most Influential Nodes

For the baseline method for comparison, we chose to compare the Crit-Set against removing the most influential node. The most influential $l$ nodes to remove are identified as follows: for each vertex $v \in V/S$, find the number of sets $G^r_{\mathcal{E}}$ that are covered by $v$. Return the set of top $l$ nodes as the nodes to be removed by this method. Note that this computation is not done incrementally. We run a single iteration and find the number of $G^r_{\mathcal{E}}$ covered by each $v$ and return the top $l$ nodes. We refer to this algorithm as the Top-Infl algorithm in the results.

4.4 Does Removing Critical Nodes Guarantee Maximal Disruption

We will consider 3 strategies to further understand the pitfalls of ICN($k$). First, the context-based strategy, one where the seed node is known apriori, is expected to be more effective than ICN($k$) as the latter does not consider the context in which the disruption needs to be realized. Second, we will consider the strategy that involves removal of most influential set of nodes and give an example in which removing the most influential node is a better choice compared to removing the critical node. Finally, we will consider the Greedy vs. the One-Shot algorithm and show an example where the removal of nodes by the One-Shot algorithm gives a larger reduction in influence when compared to Greedy Algorithm.

4.4.1 Seed context is important: Experimental results depend on the chosen Seed Set

Suppose we were finding one critical node in the Graph $G$ shown in the figure 4.9. Node 6 would be selected as the critical node in $G$. But how do we validate this result? One way is to fix a seed set $S$ at the start of the experiment, find the influence of $S$ in $G$, $\sigma_G(S)$, and then compare it with the influence of the same set $S$ in $G/\{6\}$, $\sigma_{G/\{6\}}(S)$. Now how do we select $S$? In the example of figure 4.9, if $S$ is chosen from $U$, the experiments would correctly show that the influence did go down. However, if $S$ was selected from
Figure 4.9  Validation of Critical Nodes depends on seed used for calculating influence

$V$, it would show no change in the influence values. One way around this problem is to calculate influence from every possible seed set and show that the influence decreases for “most” of them. But this is also not feasible, because of the size of the graphs that we are dealing with (millions of nodes).

4.4.2 Removing the most influential node can be a better option than removing the most critical node

In figure 3.1 - 3.3 and definition of impact in 2, we looked at how removing the node that has the highest impact is better than removing the node that has the highest influence. But we can come up with examples where removing the most influential node is in fact a better choice compared to removing the node with the highest impact.

Consider the graph in Figure 4.10. Assume all the probabilities to be 1. The way we compare removing the most influential node vs. removing the node with the highest impact is - find the most influential node $s$ and fix that to be the seed. In this graph, $s = \{0\}$. Now in the graph $G/\{s\}$, find the most influential node and the node with the highest impact. In this graph, they will be nodes 1 and 5 respectively. Now compute and compare the influence from $s$ in the graph $G_{infl} = G/\{1\}$ and $G_{imp} = G/\{5\}$. We can see that in $G_{infl}$ the node 0 will only be able to influence itself whereas in $G_{imp}$ the node 0 will be able to influence $\{0, 1, 2, 3, 4\}$. Hence in this graph, removing the most influential node is a better option when compared to removing the node with the highest impact.
4.4.3 Greedy vs. One-Shot

The modular approach specified in 4.3.1 can be extended to the simple case where we do not have the seed set context provided, i.e. given the same set of inputs to Algorithm 2, run step 4 just once and return the set of top k nodes with the highest CritValue[v] values. This is the One-Shot algorithm. These two versions of the algorithm 2 give us two ways to arrange the critical nodes to remove. The incremental approach of Algorithm 2 towards building the set to be returned as the critical nodes suggests that it might always give us a better solution to the ICN(G,k) problem when compared to the One-Shot algorithm. Figure 4.11 shows that that is not always the case.

Note that in this scenario, we are working with the general version of the Algorithm 2 where no seed set context has been provided. The graph G shown in figure 4.11 consists of 2 disconnected components having a total of 177 nodes. Nodes 2, 3, and 126 have 21, 101, and 49 outgoing edges respectively. These numbers have been selected so as to achieve the counter-example we are trying to achieve. Suppose we have to return 2 critical nodes, ie. \( k = 2 \). When generating RR sets, each of the 177 nodes have an equal probability of being selected as the starting vertex of the probabilistic walk. Once the RR sets have been generated, the CritValue[3] will be the largest, followed by CritValue[2], and finally CritValue[126]. Hence the One-Shot version of the Algorithm 2 would return the solution \{3, 2\}. Let this be represented by \( S_{mod} \). In the Greedy version of the Algorithm 2, Vertex 3 will be selected as \( w \) in the first iteration. Next step is to update the RR datastructure by removing vertex 3 from all \( G_i \)'s. When 3 is removed, it would cause the CritValue[2] to decrease below the CritValue[126]. Thus, when the next vertex is being selected, the submodular algorithm will choose the vertex 126, and not 2. Let the solution set returned by this algorithm be \( S_{submod} = \{3, 126\} \).
Figure 4.11 Example for a case when the Modular approximation of the impact function in the Algorithm 2 gives a higher reduction in influence from a fixed seed when compared to the Submodular approximation of the impact function of the same algorithm.
So we have $S_{\text{mod}} = \{3, 2\}$ and $S_{\text{submod}} = \{3, 126\}$ returned by the One-Shot and the Greedy versions of the algorithm 2 respectively. Now if we were to select vertex 0 as our seed set, we would find that $\sigma_{G/S_{\text{mod}}} (\{0\}) = 2$, whereas $\sigma_{G/S_{\text{submod}}} (\{0\}) = 24$.

This counter-example shows that Algorithm 2 does not necessarily always give a better answer to the first version of the $ICN(k)$ problem compared to the modular version when no seed set context is provided.

The underlying reason of this is because of what was outlined in Section 4.4.1. The selection of critical nodes is independent of the seed set. Once we find and fix the $k$ critical nodes, we can select our seed set $S$ to calculate $\sigma(S)$ strategically in order to make one method appear better or worse than the other. So, for example, in the above example keeping everything else the same, if node 176 was chosen as the seed set instead of node 0, the results would show a large reduction in influence for the Greedy version whereas no reduction for One-Shot version.
CHAPTER 5. EXPERIMENTAL EVALUATION

We give a brief recap of the results observed by Bharadwaj [2018]. We present the experimental comparisons of our technique of removing critical nodes from a given graph $G$ using a submodular approximation of the impact function (Crit-Set) and compare it with other techniques - reduction in influence obtained in a graph $G$ using a modular approximation of the impact function (Top-Crit) and removing the most influential $k$ nodes (Top-Infl). We also give empirical comparisons between the greedy (Crit-Set) and the One-Shot (Top-Crit) algorithms. We also validate our results on real world data that we crawled from Twitter.

5.1 Previous Results by Bharadwaj [2018]

In this section we will briefly recap the experiments done by Bharadwaj [2018]. The interested reader can refer the document for the complete details.

5.1.1 Given an influenced graph, the influence of a random seed is significantly reduced due to removal of critical nodes

The first set of experiments compared removing the most influential nodes against removing the most critical nodes. The experiment was performed on 3 graphs. For each graph $G$, first they identified the set of most influential nodes of cardinality Seed-Size - call this maxSeedSet. They then performed one diffusion from the maxSeedSet to get the influenced graph $G_{infl}$. The graph $G_{infl}$ was provided as input to the ICN($G, k$) problem. In this $G_{infl}$ graph they then found the budget number of most critical nodes - let this set of nodes be denoted by crit. In this $G_{infl}$ graph they also found the budget number of most influential nodes - let this set of nodes be denoted by infl. For comparing the results, they considered $x\%$ ($x \in [20, 90]$) nodes of the $G_{infl}$ network and created a subgraph of the $G_{infl}$ network consisting of only the $x\%$ of the nodes - call it $G_{X-infl}$. Now in $G_{X-infl}$ they found the Seed-Size number of most influential nodes - let this set be denoted by usedSeedSet (usedSeedSet did not have any overlap with the set crit or infl). Then, finally, they compared the influence of the set usedSeedSet in $G_{infl}/crit$ and $G_{infl}/infl$ for different values
of $x \in [20, 90]$ and averaged out the number of influenced nodes obtained in each of the graphs $G_{infl/crit}$ and $G_{infl/infl}$ to get the results in the columns CRIT-SET and TOP-INFL respectively.

5.1.2 Max-seed influence significantly reduced to removal of critical nodes

Experiments were run to validate that the maximum influence achievable after removal of nodes following CRIT-SET is considerably lesser than that achievable after removal of nodes following TOP-INFL on one graph. In graph $G$ they found the budget number of most critical nodes - let this set of nodes be denoted by $crit$. In the same graph $G$ they also found the budget number of most influential nodes - let this set of nodes be denoted by $infl$. They calculated the max seed set in the graphs $G/crit$ and $G/infl$ and from that calculated the maximum influence in each of the graphs $G/crit$ and $G/infl$. A graph was plotted showing the relation between different values of budget and $\sigma_{G/infl}(maxSeedSet) - \sigma_{G/crit}(maxSeedSet)$.

5.1.3 Importance of Budget

They looked at the role of budget on node removal where they empirically mapped the relation between removing budget number of nodes and the difference between the number of influenced nodes after removing the nodes as per the CRIT-SET and the TOP-INFL algorithm.

5.2 Experimental Results

We now present the experimental results and observations from our techniques of removing the critical nodes.

5.2.1 Objectives of the experiments

The primary objective of our experiments is to compare the reduction in the influence in a graph $G$, from a fixed seed set $S$, when $k$ number of nodes are removed from the graph as per the strategies outlined in Sections 4.2, 4.3.1, and 4.3.2. The expectation is that removing nodes as per the CRIT-SET algorithm gives a greater reduction in influence than the TOP-CRIT algorithm which gives a greater reduction than the TOP-INFL algorithm. We test this on 3 graphs. All the graphs used in this section can be found on http://snap.stanford.edu/data/. Details of the graphs used can be looked up in table 5.1.

Another objective is to stress the importance of submodularity in picking critical nodes. In Section 4.4.3 we presented a counter-example for when the modular approximation of impact function seems a better
choice for node removal than the submodular approximation. The expectation is to empirically show that
submodularity does have its advantages when selecting critical nodes.

The final objective is to run these algorithms on real-world data collected from Twitter and try and
correlate the critical nodes identified by our algorithm to the “perceived” importance of those users in
propagating information on Twitter.

### 5.2.2 Data Setup and System details

All experiments were run on a Linux VM running CentOS7 with 2 Virtual CPUs (2 cores) and 128 GB
of RAM. The code was written in C++. The code for experiment mentioned in Section 5.2.3 can be found
on the github page [https://github.com/rgbk21/Disrupting_Diffusion_By_Finding_Critical_Nodes/tree/CodeContaining_modTopKInf1_modTopKInf1GivenSeed_topCritGivenSeed](https://github.com/rgbk21/Disrupting_Diffusion_By_Finding_Critical_Nodes/tree/CodeContaining_modTopKInf1_modTopKInf1GivenSeed_topCritGivenSeed). The code for experiments in Section 5.2.4 can be found in [https://github.com/rgbk21/Disrupting_Diffusion_By_Finding_Critical_Nodes/tree/CodeContaining_topCrit_WithoutAnySeedSetContextGiven_Exp2](https://github.com/rgbk21/Disrupting_Diffusion_By_Finding_Critical_Nodes/tree/CodeContaining_topCrit_WithoutAnySeedSetContextGiven_Exp2). The code for experiment mentioned in Section 5.2.5 can be found in [https://github.com/rgbk21/GraphManipulations](https://github.com/rgbk21/GraphManipulations).

In all the experiments, following the prior works, we chose $p_{uv} = 1/d_{in}(v)$, where $d_{in}(v)$ is the indegree
of $v$. The size of RR is computed based on the chosen $\epsilon = 2$.

### 5.2.3 Contextualized critical node identification

Table 5.1 presents the basic information about the networks used in the experiments. Three algorithms
are compared - Top-Infl which is considered as the baseline, Top-Crit, and Crit-Set. All 3 have been
discussed in the previous Chapter. The results in this section correspond to the table 5.2.

The basic methodology for experiments was as follows: for each of the 3 graphs in the table 5.1, identify
the max seed set - maxSeedSet. Borgs et al. [2014] was used to find the max seed set. The size of maxSeedSet
was fixed to be 10, denoted as in the column Seed-Size in the table 5.2. Infl-Size shows the influence obtained
in the graph $G$ when the maxSeedSet was used as the seed set. Budget is the number of nodes that were to be
removed from the Graph $G$. This corresponds to the parameter $k$ in our ICN($G,k,S$) Algorithm. Hence the
inputs provided to the ICN($G,k,S$) problem were ICN($G,Budget,maxSeedSet$). Let the Budget number
of critical nodes returned by each of the 3 algorithms Top-Infl, Top-Crit, and Crit-Set be $\inf$, $\mod$, and
$submod$ respectively. Then the values in New Infl column for each of the 3 algorithms Top-Infl, Top-Crit,
and Crit-Set were calculated as $\sigma_{G/\inf}(maxSeedSet)$, $\sigma_{G/\mod}(maxSeedSet)$, and $\sigma_{G/submod}(maxSeedSet)$
respectively. Time is measured in seconds.
The goal of this table is to show that CRIT-SET algorithm gives a larger reduction in the influence when compared to our baseline TOP-INFL. It also shows the comparison between the TOP-CRIT and the CRIT-SET methods and shows the greedy submodular approximation performs better but takes more time because of the recomputations at each iteration as was described in Section 4.2. This is inline with our expectations about the behavior of these 3 algorithms as well. The percentage values show the reduction obtained by TOP-CRIT and CRIT-SET when compared to our baseline TOP-INFL.

Table 5.1 Datasets

<table>
<thead>
<tr>
<th>Network-name</th>
<th># Nodes</th>
<th># Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>com-DBLP</td>
<td>317,080</td>
<td>1,049,866</td>
</tr>
<tr>
<td>DBLP-Tang</td>
<td>613,586</td>
<td>1,990,159</td>
</tr>
<tr>
<td>Web-Google</td>
<td>875,713</td>
<td>4,332,051</td>
</tr>
</tbody>
</table>

Table 5.2 Criticality-Indicator & Importance with Seed Set as Context

<table>
<thead>
<tr>
<th>Infl-Size</th>
<th>Seed-Size</th>
<th>Budget</th>
<th>TOP-INFL</th>
<th>TOP-CRIT</th>
<th>CRIT-SET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>New Infl</td>
<td>Time</td>
<td>New Infl</td>
</tr>
<tr>
<td>Network com-DBLP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21265</td>
<td>10</td>
<td>5</td>
<td>15328</td>
<td>67.80</td>
<td>14164</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>13726</td>
<td>76.60</td>
<td>11882</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>11035</td>
<td>76.80</td>
<td>9694</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>10486</td>
<td>74.40</td>
<td>9440</td>
</tr>
<tr>
<td>Network DBLP-Tang</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23702</td>
<td>10</td>
<td>5</td>
<td>18201</td>
<td>185.98</td>
<td>17560</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>17893</td>
<td>174.97</td>
<td>16755</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>17889</td>
<td>185.59</td>
<td>15313</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>17264</td>
<td>185.16</td>
<td>14529</td>
</tr>
<tr>
<td>Network Web-Google</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4750</td>
<td>10</td>
<td>5</td>
<td>4016</td>
<td>421.20</td>
<td>3791</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>3840</td>
<td>466.39</td>
<td>3586</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>3671</td>
<td>454.75</td>
<td>3278</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>3494</td>
<td>450.06</td>
<td>2982</td>
</tr>
</tbody>
</table>

5.2.4 Importance of Greedy (Crit-Set) vs One-shot (Top-Crit) : How to evaluate

In section 4.4.1 we looked at why it is difficult to validate the results of our Algorithm 2. The dependency of the results on the chosen seed set makes it difficult to argue about the quality of the results returned by the algorithm. In sections 4.4.3 we saw an example where the One-Shot method gives a better result when
compared to the *Greedy* method. In this section, we present an empirical comparison of the *Greedy* and *One-Shot* methods, methods that were outlined in Section 4.4.3, by designing an experiment that does not depend upon the selection of a seed and hence gives us a better picture of the importance of submodularity in the *Greedy* method.

We compute the solution (say, A) using *Greedy* and the solution (say, B) using *One-Shot*. Then we compute the set of nodes $R^{A-B}$ that are likely (probabilistic reachability) to reach the set of nodes $A - B$ (nodes that are present in A and absent in B). Similarly, we compute the set $R^{B-A}$. Intuitively, $R^{A-B}$ (resp. $R^{B-A}$) indicates the set of nodes whose influence in the network is likely to be disrupted due to the removal of nodes in $A - B$ (resp. $B - A$). Proceeding further, if the expected influence of $R^{A-B}$ is larger compared to that of $R^{B-A}$, then we claim that removing nodes in A (*Greedy*) is likely to be more disruptive than removing nodes in B (*One-Shot*). Table 5.3 presents the results of our experiments and affirms the importance of submodularity in *Greedy* method.

### Table 5.3 Importance of Submodularity

<table>
<thead>
<tr>
<th>Budget</th>
<th>$\sigma(R^{A-B})$</th>
<th>$\sigma(R^{B-A})$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network com-DBLP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2781</td>
<td>918</td>
</tr>
<tr>
<td>10</td>
<td>24052</td>
<td>5960</td>
</tr>
<tr>
<td>15</td>
<td>25333</td>
<td>7086</td>
</tr>
<tr>
<td>20</td>
<td>26644</td>
<td>8862</td>
</tr>
<tr>
<td><strong>Network DBLP-Tang</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>74191</td>
<td>3974</td>
</tr>
<tr>
<td>10</td>
<td>33453</td>
<td>1635</td>
</tr>
<tr>
<td>15</td>
<td>32668</td>
<td>10245</td>
</tr>
<tr>
<td>20</td>
<td>91020</td>
<td>15430</td>
</tr>
</tbody>
</table>

#### 5.2.5 Applicability in real-data (Twitter/Hong Kong)

As we briefly mentioned in the Section 2.5, one of the proposed applications of this work can be as a way to mitigate the spread of misinformation on OSNs. In this experiment we try to apply the concept of critical nodes to the Twitter network and find the nodes (user accounts) that Algorithm 2 returns as the critical nodes.
5.2.5.1 Background

As of on the 10th of August 2019, there was a demonstration ongoing in Hong Kong against an extradition bill that had been proposed by the government of Hong Kong (BBC [2019]). During the demonstrations, Twitter found that there were certain accounts that were linked to the government of China (PRC) that were trying to spreading misinformation about the protests in the network (Times [2019]). Twitter later released a list of 936 accounts that it had detected (and later blocked) to be state-backed accounts created by PRC whose sole purpose was to undermine the legitimacy and political positions of the protest movement on the ground. Twitter also reported that they shut down a network of approximately 200,000 other spammy/fake twitter accounts that were created as a result of them blocking the above 936 accounts (Twitter [2019]).

So, here we have been given a network $G$ of people that are specifically talking about Hong Kong (we use key words to filter out the users so that the graph contains only those accounts that have tweeted about Hong Kong). We realise that there might be unknown entities that might try and spread misinformation in the network - i.e. the seed set is not known. The goal is then to identify the nodes/user accounts in this network that are critical to the spread of misinformation. Intuitively, the idea is that if we can preemptively isolate/inform some key users (so that they do not retweet any misinformation) about a possible misinformation campaign then we can minimize the number of people that are exposed to the said misinformation irrespective of where the misinformation might start from. The way we do this is by running the $ICN(G, k)$ algorithm specified in 2 and finding the $k$ most critical nodes. Intuitively, our expectation is that the nodes identified as critical by our algorithm must be critical in the general sense of the word - i.e. the Twitter user accounts identified by the $ICN(G, k)$ algorithm as critical should be strongly correlated and must bear high relevance with the ongoing political discussions regarding Hong Kong in the network.

Thus, the goal is to identify the critical nodes (user accounts) in this network of people talking about Hong Kong.

5.2.5.2 Data Collection and Experimental Setup

For the purpose of this experiment, tweets mentioning Hong Kong were filtered using the Tweepy library. Tweets were collected from Tue Aug 13 21:30:14 +0000 2019 to Thu Aug 15 13:08:30 +0000 2019. In all, around 1.2 million tweets (in the English language) mentioning keywords related to Hong Kong were collected. A graph was created from the collected tweets. The graph $G$ was created in the following manner: there are 3 actions that any user can take on a tweet: retweet, retweet with comment, and reply. All 3 actions were
perceived as engaging with the tweet. An edge was created from a user \(v\) to a user \(w\) if \(w\) took any of the above stated 3 actions on \(v\)'s tweet.

There were some simplifications made in order to make dealing with the data easier:

a) The sequence of retweet history is not preserved by the Tweet object. Consider the following sequence of events - Alice writes an original tweet \(t_1\). Bob, Alice’s follower, retweets \(t_1\) thus creating tweet object \(t_2\). Charlie, Bob’s follower sees \(t_2\) and retweets \(t_2\) thus creating tweet object \(t_3\). Tweet \(t_3\) will then refer to \(t_1\) as its original tweet, and not \(t_2\). Hence the information that Charlie saw Alice’s tweet through Bob, and not directly from Alice, is lost. There are some works that deal with the problem of uncovering the underlying network of diffusion (Gomez-Rodriguez et al. [2012]). For this experiment, we will draw a directed edge from Alice to Charlie.

b) It is difficult to calculate the probability that a user \(v\) will engage with the tweet of another user \(u\). Even between the same pair of users, the probability might be different for different topics. For simplicity, we consider \(p_{uv} = 1/d_{in}(v)\), where \(d_{in}(v)\) is the indegree of \(v\). The size of \(RR\) is computed based on the chosen \(\epsilon = 2\).

The graph \(G = (V, E)\) in this way contained a total of 607,525 nodes and 994,220 edges. This graph \(G\) and \(k = 20\) were provided as input to the Algorithm 2.

The column headings of the figures from 5.1 - 5.4 should be interpreted as follows:

a) The first column “Rank” ranks the critical nodes in the order that they were returned by the Algorithm 2. Recall that in each iteration, the node \(v\) returned by the algorithm is the \(v\) that provides the maximum marginal gain. Hence, this column can be considered as a ranking of the nodes/user-accounts in terms of how critical they are to the diffusion process, with rank 1 being the highest and 20 the lowest.

b) The second column is “rankedByFollowerCount”. This rank was obtained by looking at the number of the Twitter followers that each of the 20 critical nodes/user-accounts returned by the algorithm had and ranking them in the order. Rank 1 meaning that user had the maximum number of followers among all the 20 critical nodes returned and rank 20 specifying that that user had the minimum number of followers among all the 20.

c) The third column is “rankedByFriendsCount”. It is the same as point b but instead of Twitter followers, we looked at the count of Twitter friends of each of the 20 users.

One point that we would like to mention here - which is also true for all of the figures discussed - is that the Follower and Friends count for each of the 20 users that we use for obtaining the above ranks gives the number of followers of a specific node in the entire Twitter graph. However, as we mentioned earlier, in the
graph that we have constructed and used as input to our algorithms, an edge was drawn from \( u \) to \( v \) only if \( v \) engaged with the tweet sent out by \( u \) - given that the tweet contained at least one of the keywords associated with Hong Kong protests. Hence the second and third column may not give an accurate representation of the outdegree of a particular node. This was done because if we had drawn an edge from a user to every one of their followers, our resultant graph \( G \) would have had hundreds of millions of nodes.

### 5.2.5.3 Observations and Results

**Observation 1)** *The nodes identified as critical by the algorithm do intuitively make sense*

The top 20 nodes that were identified as critical by our Algorithm 2 are shown in the figures 5.1 and 5.2. Most of the nodes among the top 20 nodes returned by the CRIT-SET algorithm were accounts belonging to political scientists and analysts, grassroot activists, and political commentators from Japan, Europe, and the United States. They also included accounts belonging to activists, correspondents, and locals - all of these people either living or reporting from Hong Kong. The nodes also included accounts belonging to news aggregator sites and newspapers. The point of this example is to demonstrate that the algorithm picks nodes whose selection intuitively makes sense - these nodes identified can in fact be considered critical to the spread of information in the network.

**Observation 2)** *The algorithm also identified, as critical, some other nodes that might not intuitively seem important to the information diffusion*

These are the user accounts that had relatively small number of followers (200 - 20,000). Some rudimentary analysis revealed that some of these users had tweets that went viral during the duration of the 3 days that the data was collected.

For the same Graph \( G \) and \( k = 20 \), we also ran the \( ICN(G, k, S) \) algorithm with the seed set context provided. This is the algorithm that we discussed previously in Section 4.2. In addition to \( G \) and \( k \), the \( ICN(G, k, S) \) problem also requires a seed set \( S \) as input. Since this version assumes that we already know the source of the (mis)information spread, we need to select a seed that we could use as the source. The problem of identifying the source of misinformation is not trivial and is outside the scope of the current discussion. For now, the selection of seed was done as follows (using an extremely conservative approach): we looked at the user descriptions of all the 607,525 accounts. If description contained the keywords “China” and “News” in it, we made the assumption that the account belonged to a news organization from China. Media in China comes under the censorship of the government (Xu and Albert [2017]) and hence could be a potential source for misinformation. Some accounts were added and removed from the list based on some
basic background checks done for the accounts. This finally gave us a list of 169 accounts that served as our seed set nodes $S$ for this experiment.

The results for the experiment are shown in the figures 5.3 and 5.4.

We again limited our search to identify the top 20 critical nodes present in the network.

Observation 3) *Most of the critical nodes identified are Twitter accounts belonging to news aggregators or journalists working on topics related to Hong Kong*

An intuitive explanation of the results can be done as follows: we selected the seed set as the Twitter accounts belonging to news stations in China. The general public on Twitter talking about topics related to Hong Kong will probably not be directly following the Twitter accounts of the news stations in China. However, the journalists and news agencies covering stories related to the Hong Kong protests will be following these Twitter accounts belonging to Chinese media.

In this case, the flow of information can be pictured as follows - our source (seed set of 169 nodes that we chose earlier) are the Twitter accounts belonging to Chinese news stations (under state control) that tweet out information - these are picked up by their followers i.e. journalists and news stations working on
Figure 5.3 Critical nodes identified (Twitter Verified Users) when the seed set is known

<table>
<thead>
<tr>
<th>Rank</th>
<th>rankedByFollowerCount</th>
<th>rankedByFriendsCount</th>
<th>user_verified</th>
<th>user_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>17</td>
<td>TRUE</td>
<td>Journalist in Beijing for CNN.</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>6</td>
<td>TRUE</td>
<td>Hong Kong correspondent</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>TRUE</td>
<td>Material distributed by CGTN America on behalf of CCTV.</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>14</td>
<td>TRUE</td>
<td>China Correspondent, British Broadcasting Corporation.</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>13</td>
<td>TRUE</td>
<td>Activist from Hong Kong</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>9</td>
<td>TRUE</td>
<td>Journalist</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>1</td>
<td>TRUE</td>
<td>New York Times reporter in Hong Kong</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>7</td>
<td>TRUE</td>
<td>China Daily EU Bureau, columnist.</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>2</td>
<td>TRUE</td>
<td>Hong KongFP Correspondent</td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>3</td>
<td>TRUE</td>
<td>Journalist writing on Asia</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>15</td>
<td>TRUE</td>
<td>Editor-in-Chief of Chinese Newspaper</td>
</tr>
</tbody>
</table>

Figure 5.4 Critical nodes identified (Twitter Un-Verified Users) when the seed set is known

<table>
<thead>
<tr>
<th>Rank</th>
<th>rankedByFollowerCount</th>
<th>rankedByFriendsCount</th>
<th>user_verified</th>
<th>user_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>12</td>
<td>FALSE</td>
<td>No Description Available</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>19</td>
<td>FALSE</td>
<td>Thai Activist/Writer</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>4</td>
<td>FALSE</td>
<td>Barcelona (Republic of Catalonia)</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>16</td>
<td>FALSE</td>
<td>No Description Available</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>11</td>
<td>FALSE</td>
<td>Journalist</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>18</td>
<td>FALSE</td>
<td>Hong Kong, Reuters Chief Correspondent</td>
</tr>
<tr>
<td>17</td>
<td>20</td>
<td>20</td>
<td>FALSE</td>
<td>No Description Available</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>10</td>
<td>FALSE</td>
<td>No Description Available</td>
</tr>
</tbody>
</table>

Stories related to protests in Hong Kong who then engage with these tweets either by replying, retweeting, or quoting and send the new tweet out to their followers - the followers of these journalists then progressively retweet, reply, or comment on these tweets - and the process cascades to include more people. Hence, the first line of people who are being exposed to PRC propaganda are mostly the Twitter accounts belonging to journalists and news stations reporting on news about Hong Kong. So it makes sense when our algorithm chooses these journalists as critical nodes, because if they did not retweet information from Chinese news stations, majority of the general users on Twitter would not be exposed to that information.

5.2.5.4 Summary of results

Despite certain limitations, the experiments do serve as a proof-of-concept. The nodes that were identified as critical by the Algorithm 2 were mostly related to the Twitter accounts of journalists, news aggregators, and politicians. This result is inline with our real-world expectations. In a politically motivated environment like a public protest, where news is rapidly developing, we would expect media outlets and politicians to be critical in spreading information. Hence, if we pre-emptively informed them about a misinforma-
tion campaign, it would significantly reduce the number of nodes that could be activated by any potential misinformation.

One criticism of the results can be that the politicians and media outlets would already have a significant number of followers. Hence, if we just selected the 20 most influential nodes as the most critical nodes, it would still give us the same kind of results. To verify this claim, we found the max seed set as per the Borgs et al. [2014] algorithm - the same algorithm that we have used to solve the Influence Maximization problem thus far. When we looked at the top 20 nodes returned by the algorithm, the overlap between the critical nodes returned in Figures 5.1 - 5.2 and the max seed set consisted of 11 nodes. Among the nodes missed by the max seed set, but present in the critical nodes set, two were of key interest. One was a Twitter account belonging to Chinese news organization and another was a popular activist from Hong Kong working at the grass-root level. This showed that trying to control misinformation by just controlling the most influential node is not necessarily always a correct strategy. Even in this real-world mock-up we could identify nodes that are not returned by the max seed set algorithm but can actually be considered critical to the (mis)information diffusion in the network.

Some ways to improve upon these experiments is to crawl the Twitter network in order to get a true representation of the underlying follower-followee relationship. The challenge of figuring out the retweet sequence is also something that could be addressed. The probability of information diffusion on each edge is another challenge. Solving one or more of these will give a more complete and better representation of results returned by the algorithm which will in-turn enable better comparison. Of course, this comparison of actual users can lead to a debate about which user someone actually considers to be more critical than another - which is not possible to solve.
CHAPTER 6. FUTURE WORK

We plan to consider different heuristics and implementation strategies to realize the computation of impact; the goal being application to very large networks efficiently without compromising the quality. We only considered heuristics for deleting the nodes - other heuristics could be designed that deal with removing a combination of nodes and edges.

In this work, we also assumed that the removal of each node had the same cost. A more general problem can be developed that deals with associating costs with the removal of nodes and constraining the deletion of nodes based on these costs. If there was such a cost associated with the removal of each node, is there a cost-to-benefit ratio after which the removal of critical nodes would get expensive without providing the necessary reduction in influence. How would this depend on the topology of the graph or the number of seed set elements chosen for diffusion in the graph.

Since the final solution of critical nodes is dependent on the structure of the graph, we can run topology-aware algorithms that find out the best algorithms from a slew of algorithms depending on the underlying graph structure. Variations of the $ICN(k)$ problem can be considered where it may not be possible to remove some nodes.

We compared the removal of most influential node against the most critical node. Along the same lines, it might be interesting to look at influence of most critical nodes and compare it with the influence of the most influential nodes. One version of competing cascades looks at mitigating influence by identifying the most influential node in the influenced graph of misinformation to serve as the seed for spreading true information. How would this compare to a method where we selected the most critical node in the influenced graph of misinformation and used that as the seed for diffusing the true information.

It might be interesting to see the selection of critical nodes when the seed set is not a small set of most influential nodes but instead a large group of low-influence nodes that can simulate behavior of bots that we see frequently in OSNs. Would the critical nodes in this case be just the most influential nodes? Bots also do not always work in synchronization - they try to get around automatic detection by not exhibiting coordinated behavior. In that case, can we model the critical nodes problem as a time dependent function.

Even in general - misinformation diffusion is a time-critical process. Can we incorporate a factor for time? When dealing with seed set context - once we know that the misinformation has affected a certain
section of the nodes, it would not suit well to select a critical node as one of the already infected nodes. A
time based model would help in finding critical nodes that are still inactive.

On the implementation front, one of the bottlenecks is the computation of reachability for every RR
Graph. Since the generation of RR Graphs and the subsequent computation for that RR Grpah are in-
dependent processes to other RR Graphs, concurrency/multi-thread can be explored to try and make the
executions faster. On the experimental validation front, we need better ways to compare two different al-
gorithms. Instead of trying to compare results by finding a global seed to perform diffusion, it might be a
better option to look at the probabilistic close neighbors of nodes being removed and find the reduction in
influence locally and compare those values.
CHAPTER 7. GENERAL CONCLUSION

In this work, we studied the problem of disrupting influence in social network under the IC diffusion model through the identification of critical nodes. Our objective was to identify a set of $k$ nodes, the critical nodes, which when removed from the network can maximally reduce diffusion from any possible seed set. Using the formalization of the problem in terms of impact and strength introduced previously by Bharadwaj [2018], we identified some key properties of the impact function and showed some cases in which removal of critical nodes may not be the best option of nodes to remove. We added modifications to the existing $ICN(G, k)$ algorithm. We expanded upon the existing work to introduce the $ICN(G, k, S)$ algorithm for finding critical nodes when the seed set context has been provided. We performed experimental validations using multiple graphs and showed that the algorithm using a greedy heuristic does in fact give us a better result when compared to the baseline of removing the most influential node. We empirically validated the importance of a submodular approximation of the impact function over the modular approximation. We also validated our algorithms against a real-world data set collected from Twitter that showed that the nodes that were returned as critical by the algorithm can be interpreted as being critical to the spread of information in the Twitter network.
REFERENCES


