2017

Targeted Influence Maximization In Labeled Social Networks with Non-Target Constraints

Naresh Somisetty
Iowa State University

Targeted influence maximization in labeled social networks with non-target constraints

by

Naresh Somisetty

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

Jin Tian

Iowa State University

Ames, Iowa

2017

DEDICATION

I would like to dedicate this thesis to my wife Lakshmi and to my parents Subba Rao and Savithri without whose support I would not have been able to complete this work. I would also like to thank my friends and family for their encouragement and support during the writing of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to Dr Samik Basu and Dr Pavan Aduri for their immense support and guidance through out the research and writing of this thesis.I would also like to thank Dr. Jin Tian for his efforts and contributions to this work. This work is partially supported by NSF Grant CCF 1555780.

ABSTRACT

The objective of influence maximization problem is to find a set of *highly influential* nodes that maximizes the spread of influence in a social network. Such a set of nodes is called *seed set*. Targeted labeled influence maximization problem is an extension that attempts to find a seed set that maximizes influence among certain labeled nodes. However, in certain application areas such as market and political sciences, it is desirable to limit the spread of influence on certain set of nodes while maximizing the influence spread among different set of nodes. Motivated by this, in this work we formulate and study Constrained Targeted Influence Maximization problem where a network has two types of nodes—targets and non-targets. For a given k and θ, the objective is to find a k size seed set which maximizes the influence over the targets and keeps the influence over the non-targets within the threshold θ. We propose two algorithms based on the greedy approach and establish certain approximation guarantees. We extend this greedy algorithm to a Multi-Greedy algorithm. However, the pure greedy methods are not practically viable due to prohibitively high time overhead. To address that, we develop two-phase framework that will enable us to use multiple heuristic choices as subroutines. We experimentally show that several of these heuristic algorithms produce solutions whose quality is close to the quality of solutions produced by the greedy algorithm. We have developed a prototype framework and evaluated all the algorithms using social networks with different types and sizes.

CHAPTER 1. INTRODUCTION

The rapid increase in the number and types of online social networks in the past decade has paved way for a new marketing strategy: viral marketing, where a small number of individuals or communities are targeted to initiate a process of advertising (Larson (2009)) and the dynamics of the information propagation through the network is relied on to spread the advertised information without further external stimuli. The initiators of the process constitute a set referred to as the *seed*. Understanding and identifying the nature and size of the seed that will help in maximizing the spread of desired information about a product, therefore, is one of the main focus of research in social sciences that relate to viral marketing. This is called the diffusion/influence maximization problem. In the recent years, the power of computing and data sciences have been brought to bear upon this problem–the objective has been to design computationally efficient algorithms with provable guarantees on the quality of the seed set generated by such algorithms.

1.1 Background

Central to this problem are two features: the dynamics of the network in which the diffusion is being examined and the diffusion model. The network dynamics capture both the topology and the way it expands and contracts as the new groups or individuals become active and inactive in the network (Erdös and Rényi (1960); Watts and Strogatz (1998); Barabási and Albert (1999)). It has been experimentally validated that social networks typical exhibit behavior of a small-world network (Kautz et al. (1997)), which has high node clustering and same/similar node distances. The diffusion model, on the other hand, captures the conditions (both qualitative and quantitative) under which an individual can influence (or can be influenced by)

another individual in the network (Kempe et al. (2003)). Independent Cascade Model (IC) and Linear Threshold Model (LT), and their generalizations are the diffusion models considered in the past. The IC model considers the probability with which an individual can influence his/her neighbors and the LT model, on the other hand, considers the number of neighbors of node that must become influential to influence node under question. For both these models, finding the seed with a specific size, that will maximize the result of diffusion is an NP-Hard problem (Kempe et al. (2003)). In this context, Kempe et al.(Kempe et al. (2003)) were the first to present a greedy algorithm with an approximation guarantees of $(1 - 1/e)$ on the quality of the seed set. Their algorithm is an iterative process, where at each step the selection of an element to add to a partially computed seed is driven by the highest marginal gain the element is expected to produce after its addition. The proof that this algorithm has a provable approximation guarantee crucially relies on the fact that the influence maximization function is *monotone and sub-modular*. The greedy algorithm, however, is not viable in practice and several heuristics (Chen et al. (2009); Ohsaka et al. (2014); Li et al. (2011)) have been proposed and empirically validated in the recent past. At its core, these heuristics rely on simple and efficient mechanism (such as one-step probabilistic behavior: Degree Discount(Chen et al. (2009)) or non-probabilistic reachability set computation of graphs generated by Monte Carlo Simulation(Ohsaka et al. (2014))) for ordering the possible candidates for seed set in each step.

1.2 Driving Problem

We focus on a variant of the diffusion problem, where, unlike the existing setting, the nodes in the network have labels, say, T and N (properties of individuals in the network). The objective is to find a k size seed set to initiate the diffusion that will maximize the spread among the nodes with a label T, while keeping the diffusion among the nodes with label N within a pre-specified bound θ. We will refer to the individuals with label T as the targets and the individuals with label N as non-targets. This variant directly follows from several real-life scenarios in market science, where advertising among the non-target groups or communities may not only lead to a waste of valuable resources but may also negatively impact the advertising (when

the non-target groups actively spread information that negates the impact of advertisement) (Aaker et al. (2000)). The problem is also relevant in political campaign (in particular, in political fund-raising campaigns), where the objective of the campaign is to reach out to as many sympathetic individuals (with same or similar political viewpoint) without influencing and inadvertently energizing the individuals with opposing political views. We call this problem as Constrained Targeted Influence Maximization Problem (CTIM for short).

1.3 Contributions

From the computational aspect, the problem is as hard as the conventional diffusion maximization problem. Furthermore, we first observe that the natural greedy algorithm may not have an approximation guarantee of $(1 - 1/e)$, as the maximization function in our scenario is not monotone and sub-modular. Nevertheless, we provide two types certain approximation guarantees on the quality of solution produced by the greedy algorithm. In this context, we introduce a new notion of optimality–robust optimality, where the optimal solutions, i.e., the candidate seed sets are constrained such that addition of k target individuals to the candidate will still keep the expected diffusion among the non-target individuals within the bound θ. From practical point of view, robust optimality aims to bring in the importance of non-target threshold in the computation of the optimal solution. We prove that the greedy algorithm when applied to constrained targeted influence maximization problem renders itself the approximation guarantee with respect to robust optimality. We also prove that the greedy algorithm has an approximation guarantee containing both multiplicative and additive errors (whereas for the standard influence maximization, the greedy algorithm's approximation only involves multiplicative factors). Building on this we propose a new greedy algorithm called *Multi-Greedy* Algorithm and theoretically show that this algorithm performs at least as good as the normal greedy algorithm. Experimentally, we observe that Multi-Greedy algorithm performs a lot better than the basic greedy algorithm. However, this algorithm is expensive and is inefficient even for networks with just several thousand nodes.

To address this issue, we further refine the Multi-Greedy algorithm and implement using a two-phase approximation strategy. In the first phase, we estimate for each node $v \epsilon V$, the number of non-target individuals that might get influenced if v is selected as a member of the seed. In the second phase, we iteratively compute the seed by adding at each step, the individuals that will impact the diffusion (to the target individuals) the most if added to the seed while still keeping the number of non-target individuals being influenced below the threshold. We utilize a tree-structure to keep track of the search space over the (partial) candidate solutions for the seed set, which, in turn, allows us to efficiently compute the seed set. We first develop a set of methods, where in the first phase greedy (simulation based) algorithm is used for estimation and in the second phase, heuristics relying on degree discount (Chen et al. (2009)) and reachability analysis of graphs generated by Monte Carlo simulations (Ohsaka et al. (2014)) are used for the seed set computation. We prove that the computed result is as good as the results obtained from the Multi-Greedy algorithm. We then develop pure heuristic method, where the first phase estimation also uses heuristics. While the theoretical guarantees regarding the quality of the result cannot be maintained, these heuristics are very efficient and we have empirically shown that the results obtained using the heuristics is close to the optimal solutions. We have developed a prototype implementation and have evaluated our algorithms on a variety of social networks of different sizes.

1.4 Organization

The rest of the paper is organized as follows. In Chapter 2, we talk about existing methods to maximize the influence in social networks . In Chapter 3, we formally define the problem and complexity of the problem. Also we discuss various algorithms for the CTIM problem. We present the notion of robust optmaility and show that the greedy algorithm for the constrained targeted influence maximization problem renders itself the approximation guarantee with respect to robust optimality . In Chapter 4, we present our system overview, experimental results performed on various real world social networks and comparison of various algorithms with the greedy algorithm . In Chapter 5, we summarize our contributions, and discuss the possible extensions to this work.

CHAPTER 2. REVIEW OF LITERATURE

Domingos and Richardson (2001) introduced the influence maximization problem in a social network represented as $G = (V, E)$, where V is the set of nodes in the graph and $E \subseteq V \times V$ the edge relation between pairs of nodes. The graph also captures propagation probability $p(\langle u, v\rangle)$ for each $\langle u, v\rangle \in E$, which quantifies the amount of influence node u can exert on its neighbor v. The information in the network is diffused as follows: At the beginning certain set of nodes (seed set) are activated/influenced. At every time step, a newly activated node u will activate/influence its neighbor v with probability $p(\langle u, v\rangle)$. This process results in diffusion of information in the network. Given a seed set $S \subseteq V$, $\sigma : \mathcal{P}(S) \to \mathbb{R}$ is a function capturing the expected number of set of influenced nodes at the end of the diffusion process. The objective of the maximization problem is to identify a set of nodes S of size k $(= |S|)$ (defined apriori) such that the spread of influence $(\sigma(S))$ is maximized. This model of information propagation is known as the Independent Cascade (IC) Model. Several variants of this propagation model such as linear threshold model have been studied in literature Kempe et al. (2003). In this work we focus only on IC model.

2.1 Independent Cascade Model

We adopt the *Independent Cascade* (IC) model of diffusion, formalized by Kempe et al. (2003) In the IC model, given a directed graph G = (V,E), a propagation probability $p(\langle u, v\rangle) \in [0, 1]$ for each $\langle u, v\rangle \in E$, and a seed set $S \subseteq V$, we first activate vertices in S. Then the process unfolds in discrete steps based on the following randomized rule: When a vertex u becomes active in step t for the first time, then for every vertex $v \in V_{out}(u)$, u has a single chance to activate vertex v with probability $p(\langle u, v\rangle)$. If u succeeds, then v will become active in step

$t+1$ and remains active in all the subsequent steps. This process runs until no more activation is possible. We denote the set of activated vertices given a seed set S by A_S. The influence spread of a seed set S under the IC model is defined as the total number of active vertices given a seed set S. We denote the influence spread of S by $\sigma(S)$.

Figures 2.1 and 2.2 illustrate the diffusion process under IC Model.Let us consider v_3 and v_4 be the initial seed set. The green colored vertices represent the active nodes and the blue colored vertices represent the nodes that are considered for activation by the active nodes in the current time step. In the first iteration, we activate the nodes v_3 and v_4 and push the nodes to the process queue. Subsequently, we process every node in the process queue. In the next step, v_3 attempts to activate v_2 (as v_2 is the inactive neighbor of v_3) and successfully activates v_2 (as $p(\langle v_3, v_2\rangle)$ is 0.9) and pushes v_2 to the process queue. In the next step, v_4 attempts to activate v_5, v_6 and v_7 and activates only v_5 and v_6 (as $p(\langle v_4, v_7\rangle)$ is very less, hence v_4 fails to activate v_7) and we push the nodes v_5 and v_6 to the process queue. In the next step, v_2 activates v_1 and pushes to the queue. As v_5, v_6 and v_1 do not have any inactive neighbours, this marks the completion of diffusion process under the IC Model.

2.2 Greedy Algorithm

The influence maximization problem is proved to be NP-hard (Kempe et al. (2003)). Kempe et al. (2003) proposed a greedy algorithm which acheives a constant approximation, 1- $\frac{1}{e}$, to the optimum solution. Kempe et al's greedy approach starts with an empty seed set $S = \emptyset$, and then iteratively add vertex v to the seed set, such that v has the maximum marginal influence spread i.e $v = argmax_{v \in V}(\sigma_{S \cup v} - \sigma_S)$. The influence spread of S, σ_S is estimated by R repeated simulations, Monte Carlo Simulations, of IC Model. The greedy algorithm acheieves constant approximation due to non-negativity, monotonicity and submodularity of influence spread function σ_S for the IC model (Kempe et al. (2003)). However, greedy algorithm has a major limitation, which is its efficiency. The reasons for the limitaion is two fold : (1) The algorithm requires repeated computes of the spread function for various seed sets. The problem of computing the spread under IC is #P-hard . As a result, it is estimated by running Monte Carlo Simulations for R rounds on IC Model, which results in very long computaion time. (2)

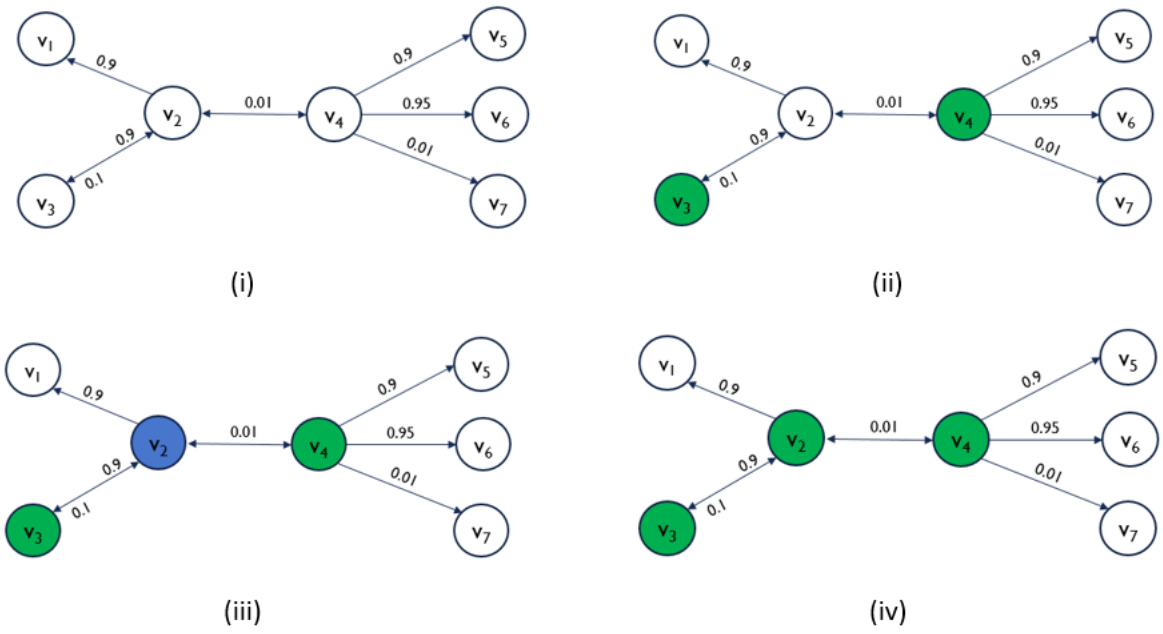


Figure 2.1: Illustration of ICM

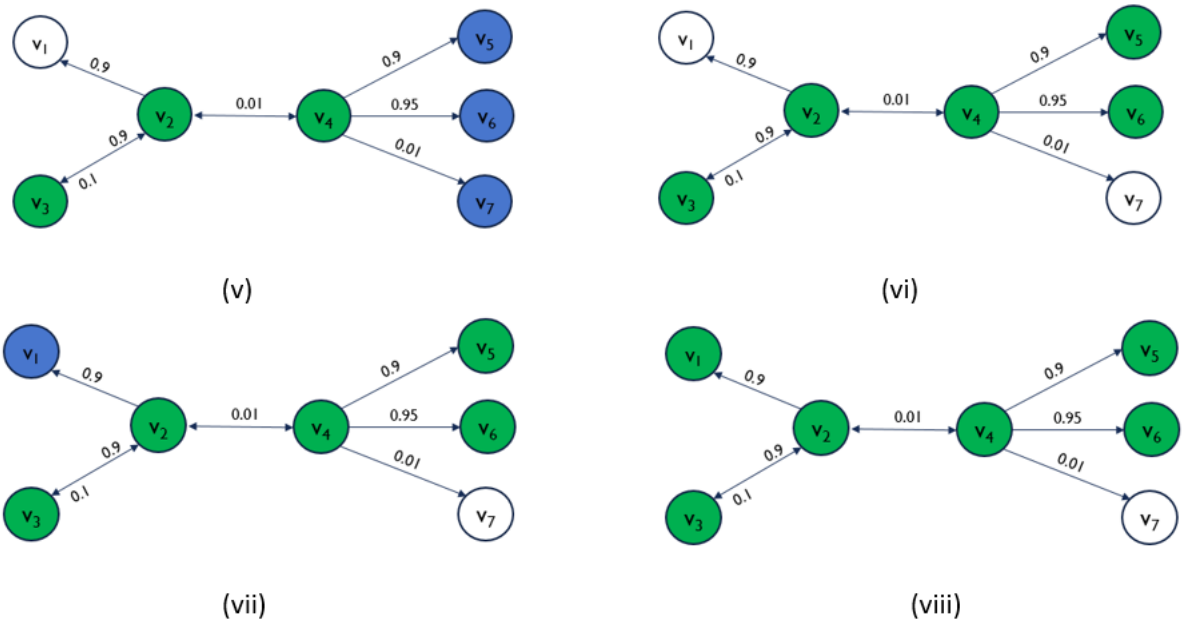


Figure 2.2: Illustration of ICM Contd

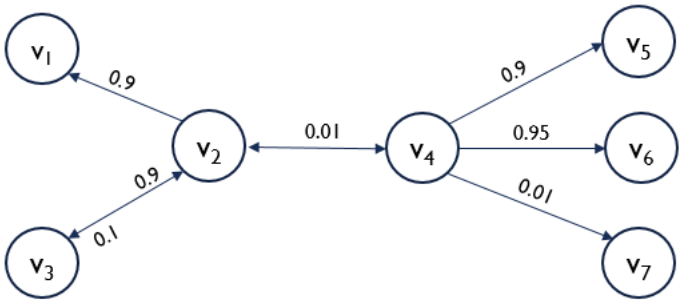


Figure 2.3: Sample Graph to illustrate Influence Maximization

In each iteration, the simple greedy algorithm searches all the nodes in the graph as a potential candidate for next seed node. As a result, this algorithm entails a quadratic number of steps in terms of the number of nodes. The greedy algorithm takes $O(knRm)$ time to complete.

2.3 CELF Optimized Greedy Algorithm

Leskovec et al. (2007) proposed agorithm, CELF (Cost-Effective Lazy Forward) based on "lazy-forward" optimization in selecting seeds. This algorithm tackles the issue (2) quadratic nature of the greedy algorithm by exploiting the submodularity property of σ. The main idea is that marginal influence gain for a node at step i cannot be more than that at step j $(i > j)$. CELF maintains a priority queue $\langle u, \Delta u(S)\rangle$ sorted on $\Delta u(S)$ in decreasing order, where S is the current seed set and $\Delta u(S)$ is the marginal gain of u w.r.t S. In every interation $\Delta u(S)$ is re-evaluated only for the top node at a time and if needed, the queue is resorted. The node at the top is picked as the next seed. Leskovec et al. (2007) empirically shows that CELF dramatically (approx. 700 times speed-up) improves the efficiency of the greedy algorithm with the same performance, in terms of influence spread, as the original greedy algorithm. Subsequently, Goyal et al. (2011b) further optimized and have proposed variants of CELF method for information diffusion problems. However, it still takes a few hours to complete in a graph of tens of thousands of vertices.

Priority Queue

Node	Gain	Iteration
V_4	3	1
V_3	3	1
V_2	2	1
V_5	1	1
V_6	1	1
V_7	1	1

Priority Queue

Node	Gain	Iteration
V_3	3	1
V_2	2	1
V_5	1	1
V_6	1	1
V_7	1	1

Priority Queue

Node	Gain	Iteration
V_3	3	2
V_2	2	1
V_5	1	1
V_6	1	1
V_7	1	1

Priority Queue

Node	Gain	Iteration
V_2	2	1
V_5	1	1
V_6	1	1
V_7	1	1

Priority Queue

Node	Gain	Iteration
V_5	1	1
V_6	1	1
V_7	1	1
V_2	0	3

Priority Queue

Node	Gain	Iteration
V_6	1	1
V_7	1	1
V_2	0	3
V_5	0	3

Priority Queue

Node	Gain	Iteration
V_7	1	1
V_2	0	3
V_5	0	3
V_6	0	3

Priority Queue

Node	Gain	Iteration
V_7	1	3
V_2	0	3
V_5	0	3
V_6	0	3

Figure 2.5: Illustration of CELF Greedy

Figure 2.5 illustrates the CELF Greedy algorithm for the sample graph in Figure 2.3. Let us consider the budget k to be 3. In the first iteration, we calculate the gain for every node and update the priority queue sorted by gain (Figure 2.5). We will remove the node at the top of the queue which is v_4, and add it to the seed set. In the next iteration we pick v_3 as it is at the top of the queue and we re-calculate the gain as the current iteration of v_3 is 1. We will add v_3 to the seed set as it remains at the top even after re-calculating the gain. In the next step for v_2, we will re-calculate the gain which is 0 (as the nodes influenced by v_2 are already activated by the current seed set). We update the queue with re-calculated gain of v_2. Similarly, we re-calculate and update the gain for v_5, v_6 and v_7. We will add v_7 to the seed set as it remains at the top for the current iteration. Hence v_4, v_3 and v_7 will be the seed set produced by CELF algorithm.

2.4 Random Graph Reachability Tests Algorithm

Ohsaka et al. (2014) proposed efficient algorithm, Pruned Monte Carlo Simulations Algorithm, by estimating the influence spread using reachability tests. As reported in (Ohsaka et al. (2014)) simulating the IC model is equivalent to testing reachability on random graphs. The main idea of the algorithm is to (1) maintain and incrementally update the outcome of Monte-Carlo simulations, by which it reduces the number of necessary simulations and the simulation cost (2) BFS pruning for reachability tests, by which it reduces the computation time. Ohsaka et al's algorithm starts with generating R random DAG's based on the following rule: every edge $\langle u, v\rangle \in E$ lives with probability $p(\langle u, v\rangle)$. Then, compute the strongly connected components (SCC's) for each DAG. Consequently, the i-th vertex-weighted DAG $G_i = (V_i, E_i)$ is constructed as follows: $V_i = \{comp_i[v] | v \in V\}$, $E_i = \{(comp_i[u], comp_i[v]) | uv \in E_i'\}$ where $comp_i[v]$ denotes a SCC containing $v \in V$, $weight_i[v]$ denotes the number of vertices in a strongly connected component v. The vertices reachable from set S in DAG G_i is denoted by $\sigma_{G_i}(S)$ and is computed as

$$\sigma_{G_i}(S) = \sum_{v \in V_i : \exists t \in S, comp_i[t] \rightsquigarrow^{G_i} v} weight_i[v]$$

.

The seed set is selected according to the greedy startegy based on maximum marginal gain. An approximate value of marginal gain $\sigma(S \cup v) - \sigma(S)$ is obtained by averaging the gain of v in each DAG.

2.5 Degree Discount Heuristic Algorithm

Chen et al. (2009) propose the Degree Discount heuristic to efficiently find the effective seed nodes. Degree Discount assumes that the propagation of influence has lower potentials to spread globally. And thus it is natural to consider only one-step neighbor nodes and select nodes with high degree values, which tend to have higher expectations of influence, to be the seed ones. The central idea is to compute and update such expectations of influence in each round of selection. If one wants to select k seed nodes, Degree Discount will be performed k times. After selecting node w as a seed in each round, we will re-calculate the expectation of influence of each w's neighbor v, as v's expectation of influence will get discounted as w is in the seed set. The expectation of influence E_v is calculated by

$$E_v = (1-p)^{|t_v|}.(1 + (|d_v - t_v|).p)$$

where $d_v = \{u | u \in V \ and \ (v,u) \in E\}$, $t_v = \{u | u \in V \ and \ (u,v) \in E \ and \ u \in S\}$. In other words, E_v is the expectation value that v is not only never influenced by existing seed nodes, but also able to activate the nodes that are not selected as the seed nodes.

Figure 2.6 illustrates the DegreeDiscount algorithm for the sample graph in Figure 2.3. Let us consider the budget k to be 3. In the first iteration, we calculate the expectation of influence E_v for every node. We will pick the node with the maximum E_v which is v_4 and add it to the seed set. In the next iteration, we will re-evaluate the E_v for every node and pick the node with maximum E_v, which is v_2. As we can see that in Figure 2.6, the E_v changes for the nodes in every iteration. Subsequently, we will pick the node v_7. Hence v_4, v_2 and v_7 will be the seed set produced by DegreeDiscount algorithm.

Node	E_v
V_3	1.9
V_2	2.01
V_4	2.87
Other	1

Node	E_v
V_3	1.9
V_2	1.98
V_5	0.1
V_6	0.05
V_7	0.99
V_1	1

Node	E_v
V_3	0.9
V_5	0.1
V_6	0.05
V_7	0.99
V_1	0.1

Figure 2.6: Illustration of DegreeDiscount Algorithm

2.6 Variations of Influence Maximization

Several other variations of the influence maximization problem have been studied in literature Li et al. (2015); Chen et al. (2015); Goyal et al. (2011a); Li et al. (2011). Chen et al. (2015) proposed the Topic Aware Influence Maximization problem, which, given a topic-aware influence maximization (TIM) query, finds k seeds from a social network ,where each edge is associated with a topic distribution, such that the topic-aware influence spread of the k seeds is maximized. Li et al. (2015) proposed Keyword-Based Targeted Influence Maximization (KB-TIM) problem, to find a seed set that maximizes the expected influence over users who are relevant to a given advertisement. Goyal et al. (2011a) proposed a data based approach for the influence maximization, which leverages the existing propagation traces to learn how influence propagates and uses this to estimate the expected influence spread. Li et al. (2011) proposed the Labeled Influence maximization problem, which aims to find the set of seed nodes which maximize the influence spread and profit given a labeled social network with influence probabilites on edges, set of target labels, profit values for each label and a budget in terms of number of seed nodes to be selected. They first proposed an algorithm based on the greedy and heuristics methods of original influence maximization. Later they proposed the Maximum Coverage algorithm, the central idea of which is to compute off-line the pairwise proximities of nodes in the labeled social network and then find the set of seed nodes online. However,

the labeled influence maximization problem considers no information regarding the non-target customers. In our work, we consider non-target customers and have constrained the influence maximzation problem with non-target threshold.

CHAPTER 3. PROPOSED METHOD

3.1 Problem Definition

We will first formalize the earlier discussed variant of the information diffusion problem. Given a network $G = (V, E)$, assume that each node is either labeled as a *target node* or as a *non-target node*. For any seed S, $\sigma_T(S)$ and $\sigma_N(S)$ capture the expected number of target nodes and non-target nodes, respectively, influenced by S. Given a threshold value θ, we define $\sigma_T(S, \theta)$ as $\sigma_T(S)$ if $\sigma_N(S) \leq \theta$; otherwise $\sigma_T(S)$ equals zero. Thus if a set influences more than θ non-target nodes, then $\sigma_T(S, \theta)$ is defined as zero. We will now define our problem called.

Problem 1. Constrained Targeted Influence Maximization (CTIM) *Given k and θ, find a seed set S of size $\leq k$ such that $\sigma_T(S, \theta)$ is maximized.*

We use $OPT_{k,\theta}$ to denote the number of target nodes influenced by S when S is a solution to the Constrained Targeted Influence Maximization problem. Note that when the number of non-target nodes is 0, then this problem is precisely the Standard Influence Maximization problem.

Figure 3.1 illustrates the problem objective. There are 11 nodes, green colored nodes represent target T nodes and red colored nodes represent non-targets N. The diffusion probabilities are annotated on each neighbor relation. Consider that the objective is to compute a seed of size $k = 2$ and non-target threshold $\theta = 1$. Traditional diffusion maximization objective does not consider the labels of the nodes and will select the nodes v_1 and v_6 as the seed set (as these nodes contribute to highest marginal gain in terms diffusion to any node). On the other hand, for the proposed CTIM problem, the best seed is $\{v_1, v_9\}$—these nodes have the maximum probability to affect nodes labeled T, while restricting the affect on N-labeled nodes to 1 .

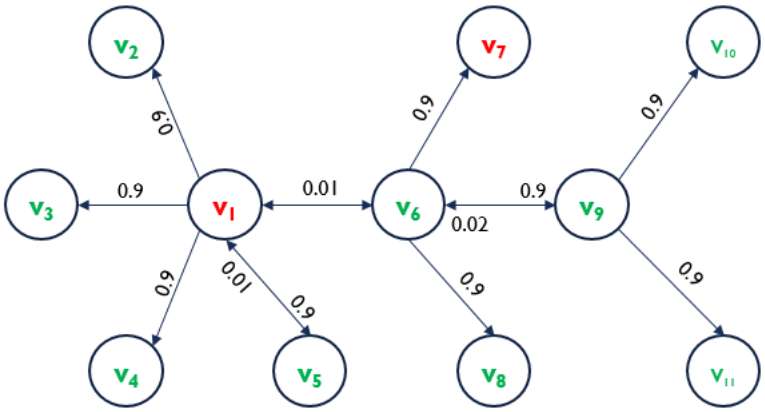


Figure 3.1: A sample graph to illustrate the CTIM problem

A direct extension of the greedy algorithm in our setting will be to select the node at each step such that the inclusion of the node will have maximum impact (maximal marginal gain) under the constraint that its inclusion will not result in violation of the non-target threshold. For instance, in Figure 3.1, in the first iteration v_6 will be the candidate node with influence spread of 5 T-labeled nodes and 1 N-labeled node. In the next iteration as influenced N-labeled nodes reached the threshold $\theta = 1$, the probable nodes will be any of the nodes v_2, v_3, v_4 and v_5 with the total influence spread on the T-labeled nodes being 6. We will show that the guarantees this method provides with respect to a new notion of optimality (robust optimality).

However, as noted above, the greedy extension still does not lead to the seed $\{v_1, v_9\}$, which can achieve influence spread over 7 T-nodes while still conforming to the non-target threshold. In the following section, we will present the details of the BaseLine Greedy and Multi Greedy algorithm, introduce a two-phase framework and discuss their effectiveness (both in terms of quality of computed result and efficiency) in addressing the CTIM problem.

3.1.1 Complexity

It is very easy to see that the CTIM problem is NP-hard by a reduction from the standard (one that does not consider node labels) influence maximization. Given an instance of such standard influence maximization, we can always construct an instance of CTIM with all labels

as target labels and $\theta = 0$, and the solution of our problem embeds a solution of the standard problem.

Theorem 1. *Constrained Targeted Influence Maximization problem under the IC model is NP-hard.*

3.2 Algorithms for Targeted Influence Maximization

In this section, we propose two basic algorithms to solve the $CTIM$ problem. Note that when there are no non-target nodes in the network, this problem is exactly same as the standard influence maximization problem. We know that a greedy algorithm achieves an approximation guarantee of $(1-1/e)$. Thus a natural approach to the CTIM problem is to start with a greedy algorithm and show that the greedy algorithm produces $(1 - 1/e)$ approximation guarantee. However, this approach does not seem to succeed here. Known proof that shows that the greedy algorithm achieves an approximation guarantee of $(1 - 1/e)$, heavily relies on the fact that influence function is monotone and sub modular. However, for our problem even monotonicity does not hold. This is because of the following: Let S be a set such that $\sigma_N(S) = \theta$. Now, if we add a vertex v to S, it could be the case that $\sigma_N(S \cup \{v\}) > \theta$. Thus $\sigma_T(S \cup \{v\}) = 0$ an thus $\sigma_T(S)$ is more than $\Sigma_T(S \cup \{v\})$.

In spite of this fundamental difficulty, we first show that this Baseline Greedy algorithm still has certain approximation guarantees. Next, we will extend this greedy algorithm (called Multi Greedy Algorithm) that attempts hold multiple candidate solutions at any instance. We will show that the quality of the solution produced by this Multi-Greedy algorithm is at least as good as the quality of the solution produced by the Baseline Greedy algorithm. Finally, we introduce speed-up techniques using pruning, reachability tests and heuristics to improve the efficiency of the Tree Based Greedy algorithm. We first start with description and analysis of the baseline greedy algorithm.

3.2.1 Baseline Greedy Algorithm

Algorithm 1 describes the baseline greedy algorithm. In this algorithm, we start with an empty seed set $S = \emptyset$, and then iteratively add vertex v to the seed set, such that v has the maximum influence spread on the target nodes ($\sigma_T(S \cup v)$) and has a influence spread on non-targets ($\sigma_N(S \cup v)$) less than non-target threshold (θ) , i.e

$$v = argmax_{v \in V \setminus S_{i-1}} \{\sigma_T(S_{i-1} \cup \{v\}, \theta)$$

The influence spread is estimated by R repeated simulations, Monte Carlo Simulations, of IC Model. This algorithm takes $O(knRm)$ time as , algorithm runs for k iterations , in each iteration we calculate the influence spread for every vertex (n vertices) for R rounds and calculating the influence spread takes O(m) time. However, this approach has a limitation : local optimal solution at every iteration does not necessarily lead to a global optimal solution. To address this limitation, we propose a Multi-Greedy algorithm which is described in Section 3.2.2

Algorithm 1: *BaselineGreedy*

Data: Labeled Social Network $G = (V, E, L)$, budget k and non-target threshold θ
begin
  $S_0 \longleftarrow \emptyset$
  for $i = 1$ *to* k **do**
    Find v such that $v = argmax_{v \in V \setminus S_{i-1}} \{\sigma_T(S_{i-1} \cup \{v\}, \theta)\}$
    $S_i = S_{i-1} \cup \{v\}$
  return S_k

We establish two types of approximation guarantees on the quality of solution produced by the above greedy algorithm. Recall that $OPT_{k,\theta}$ is the optimal solution when we restrict the number of non-target threshold nodes to θ. Our first theorem relates the quality of the Baseline Greedy algorithm to $OPT_{k,1}$, later we extend the proof and relate the quality of Baseline Greedy to $OPT_{k,\theta}$. The approximation guarantee has an additive error that depends on the graph structure and θ. Let v be a node such that $\sigma_N(v) \leq \theta + 1$ and $\sigma_T(v)$ is maximized. Let $Best_\theta$ denote $\sigma_T(v)$. Given two sets S and R, let $gain_S(R, \theta) = \sigma_T(S \cup R) - \sigma_T(S)$ if $\sigma_N(S \cup R) \leq \theta$. Given a node v and a set S of size at most k, let $gain_S(v)$ denote

$\max\{\sigma_T(S \cup \{v\}, \theta) - \sigma_T(S, \theta), 0\}$. Let $LGain_\theta$ denote the minimum value of $gain_S(v)$ over all S and v. Let $Diff_\theta = Best_\theta - LGain_\theta$.

Theorem 2. $\sigma_T(S_k) \geq (1 - 1/e)[OPT_{k,1} - (k) \times Diff_\theta]$

Proof. The greedy algorithm constructs the seed set in an incremental manner, while ensuring that during each iteration the partial seed set S_i will influence at most θ non-target nodes. During each iteration, we consider an additional set that can influence at most $\theta+1$ non-target nodes. More precisely let u be a node such that

$$u = argmax_{u \in V \setminus S_{i-1} \ \wedge \ \sigma_N(S_{i-1} \cup \{u\}) \leq \theta+1}\{\sigma_T(S_{i-1} \cup \{u\})\}$$

That is u is a node that would achieve best influence when added to S_i while ensuring that total number of non-target nodes influence by $S_i \cup \{u\}$ is at most $\theta + 1$. Let $S'_i = S_i \cup \{u\}$. Consider the following chain of inequalities. Fix an i, $1 \leq i \leq k$. Let $SEED_1$ be a seed set that yields solution whose value is $OPT_{k,1}$.

$$\begin{aligned} OPT_{k,1} &\leq \sigma_T(SEED_1 \cup S_i, \theta + 1) && (3.1) \\ &\leq \sigma_T(S_i, \theta + 1) + \sum_{e \in SEED_1} gain_{S_i}(e, \theta + 1) && (3.2) \\ &\leq \sigma_T(S_i, \theta) + \sum_{e \in SEED_1} [\sigma_T(S'_{i+1}) - \sigma_T(S_i)] && (3.3) \end{aligned}$$

The first inequality follows because, since $SEED_1$ can influence at most 1 non-target node and S_i can influence at most θ non target nodes, $SEED_1 \cup S_i$ can influence at most $\theta + 1$ non-target nodes. The second inequality follows because of the partial sub-modularity—when $S \subseteq R$, $\sigma_N(S \cup \{v\}) \leq \theta$, $\sigma_N(R \cup \{v\}) \leq \theta$, then $\sigma_T(S \cup \{v\}) \geq \sigma_T(R \cup \{v\})$. Note that since S_i can influence at most θ non-target nodes, $\sigma(S_i, \theta+1)$ equals $\sigma(S_i, \theta)$. Finally last inequality follows by the definition of S'_{i+1}.

By adding $(k-1)OPT_{k,1}$ two both sides of the inequality and rearranging terms, and using the fact that the size of $SEED_1 \leq k$, we obtain

$$OPT_{k,1} - \sigma_T(S'_{i+1}) \leq \frac{k-1}{k}(OPT_{k,1} - \sigma_T(S_i))$$

However observe that $\sigma_T(S'_{i+1}) \leq \sigma_T(S_{i+1}) + Diff_\theta$ (due to the definition of $Diff_\theta$). Thus

$$OPT_{k,1} - \sigma_T(S_{i+1}) \leq (1 - 1/k)(OPT_{k,1} - \sigma_T(S_i)) + Diff_\theta$$

Solving the above recurrence we obtain that

$$\sigma_T(S_k) \geq (1 - 1/e)[OPT_{k,1} - kDiff_\theta]$$

This completes the proof. □

In fact, we can extend the ideas from the above proof to relate the quality of the greedy solution to $OPT_{k,\theta}$.

We will next compare the quality of the solution produced by the greedy algorithm with the optimal solution having certain robustness properties. We define a notion of *robust optimal set*. A seed set S is (k,θ)-*robust* if $|S| \leq k$, $\sigma_N(S) \leq \theta$ and for every set $S' \subseteq T$ with cardinality $\leq k$, we have that $\sigma_N(S \cup S') \leq \theta$. That is even after adding all elements of S' to S, the number of non-targets influenced is still at most θ. A seed set S of size is (k,θ)-*robust optimal* if for every S' that is (k,θ)-robust optimal, $\sigma_T(S) \geq \sigma_T(S')$. Let $RobOPT_{k,\theta}$ denote the number of target nodes influence by a (k,θ)-robust optimal seed set.

Theorem 3. $\sigma_T(S_k) \geq (1 - 1/e) \times RobOPT_{k,\theta}$

Proof. Consider a slight modification of the Baseline Greedy algorithm. During each iteration of the loop, the algorithm attempts to find a node v. Consider a modification of the algorithm where we constrain v to be target node. Let $S'_1, S'_2, \cdots S'_k$ be the sets constructed by the greedy algorithm.

$$\begin{aligned} RobOPT &\leq \sigma_T(RobOPT \cup S'_i, \theta) && (3.4)\\ &\leq \sigma_T(S'_i, \theta) + \sum_{e \in RobOPT} \sigma_T^{S'_i}(e) && (3.5)\\ &\leq \sigma_T(s_i, \theta) + \sum_{e \in RobOPT} [\sigma_T(S'_{i+1}) - \sigma_T(S'_i)] && (3.6)\\ & && (3.7) \end{aligned}$$

The first inequality follows because size of $S'_i \leq k$ and due to the definition of $RobOPT$. As before, by solving the above recurrence we obtain that

$$\sigma_T(S'_k, \theta) \geq (1 - 1/e) RobOPT_{k,\theta}$$

Finally note they for every i, $\sigma_T(S'_i, \theta) \leq \sigma_T(S_i, \theta)$. This proves the theorem.

□

3.2.2 Multi-Greedy Algorithm

In this section, we improve the Baseline Greedy algorithm. The Baseline Greedy algorithm keeps track of a single set during every iteration and attempts to improve upon it. Our observation is that if during every iteration, we keep track of multiple sets and attempts to improve upon them, we might arrive at a better solution. To motivate this, consider the following simple scenario where the non-target threshold θ is 1 and k is 2. Suppose that there exist two nodes u and v such that $\sigma_T(u) = 4$ is maximized subject to the constraint $\sigma_N(u) = 0$ and $\sigma_T(v) = 8$ is maximized subject to the constraint $\sigma_N(v) = 1$. Further suppose that there nodes w_1, w_2 and w_3 such that $\sigma_N(u, w_1) = 0$, $\sigma_T(u, w_1) = 7$, $\sigma_N(u, w_2) = 1$, $\sigma_T(u, w_2) = 12$, and let w_3 be the best node such that $\sigma_T(v, w_3) = 10$ is maximized subject to the constraint $\sigma_N(v, w_3) = 1$. Now the greedy algorithm will pick $\{v, w_3\}$ as a seed set. However, in this case the set $\{u, w_2\}$ is a better solution.

The above scenario suggests the following approach. During the ℓth iteration keep track of many subsets $S^0_\ell, S^2_\ell, \cdots$, such that $\sigma_N(S^\ell_i) \leq \theta$ and $\sigma_T(S^\ell_i)$ is maximized subject to a local greedy approach. Then at the end of the kth iteration, we pick the best possible solution among the θ choices $S^0_k, S^1_k, \cdots$. Below we formally describe the algorithm.

Multi Greedy. Initially, the algorithm starts with an empty set $S_0 = \emptyset$. The $\ell+1$th iteration of the algorithm proceeds as follows. Suppose $S^1_\ell, S^2_\ell, S^r_\ell$ are the sets that are present at the end of the ℓth iteration. For each set S^i_ℓ, we extend it in at most θ many possible ways as follows: For every $j \in \{0, \cdots, \theta\}$ find the best vertex v such that $\sigma_T(S^i_\ell \cup \{v\})$ is maximized subject to the constraint $\sigma_N(S^i_\ell \cup \{v\}) \leq j$ (if such a v exists). Set $S^j_{\ell+1}$to $\{v\} \cup S^i_\ell$. Note that this process will generate at most θ new sets that will be considered in the next iteration. Consider all the sets that are generated at the end of the kth iteration. Observe that each of these sets will influence at most θ many non-target nodes. Now, pick the set that will influence maximum number of target nodes.

Set	$\sigma_T(S)$	$\sigma_N(S)$	$\sigma_T(S,\theta)$
$\{V_9\}$	3	0	3
$\{V_6\}$	5	1	5

Set	$\sigma_T(S)$	$\sigma_N(S)$	$\sigma_T(S,\theta)$
$\{V_9,V_2\}$	4	0	4
$\{V_9,V_1\}$	7	1	7
$\{V_6,V_2\}$	6	1	6

Figure 3.2: Illustration of Multi Greedy Algorithm

Figure 3.2 illustrates the Multi Greedy algorithm for the sample graph in Figure 3.1. Let us consider the budget k to be 2 and θ to be 1 for CTIM problem. In the first iteration, the two sets will be v_9 and v_6 as they have maximum marginal gain on target nodes for $\sigma_N = 0, 1$. In the next iteration, we will expand these sets with the nodes resulting in the maximum marginal gain which is as shown in Figure 3.2. As we can see that set $\{v_9, v_1\}$ has the maximum influence on the target nodes, hence the seed set will be $\{v_9, v_1\ \}$.

Though this algorithm is computationally very expensive, it can be shown that the quality of the solution produced is at least as good as the quality of the solution produced by the Baseline Greedy algorithm. Let MG be the solution produced by this and algorithm and G be the solution produced by the Baseline Greedy algorithm.

Theorem 4. $\sigma_T(MG) \geq \sigma_T(G)$

Let us now analyze the time complexity of the above algorithm. If the ℓth iteration of the above algorithm has r many subsets, then the number of sets that will be considered in the $(\ell + 1)$th iteration could be $r\theta$. Thus the total number of candidate sets at the end of the kth iteration could be as big as $O(\theta^k)$ and this makes this computationally infeasible. We will address this by designing a pruning strategy and ensure that during the each iteration, the algorithm considers only θ many sets.

3.2.2.1 Pruning Candidate Sets

We discuss how to reduce the number sets considered during the each iteration of the algorithm. Consider ℓth iteration. Let E and F be two sets that are generated during an iteration. If $\sigma_T(E) > \sigma_T(F)$ and $\sigma_N(E) \leq \sigma_N(F)$, then clearly E is a better solution and E

influences more target nodes than F, and E does not influence more non-target nodes than F. Moreover, any extension of E (while obeying non-target constraint) will be a better solution than any extension of F (while obeying non-target constraint). Thus we can prune F, and remove it. This strategy will drastically reduce the number of sets that are considered during each iteration. Observe that this pruning strategy ensures that for every $j \in \{0, \cdots, \theta\}$ there exists at most one set S such that $\sigma_N(S) = j$. Thus during every iteration, there are at most $(\theta + 1)$ sets to consider. Using this, we analyze the time complexity of the algorithm as follows: Consider an iteration. This iteration starts with $\theta + 1$ many sets. For every set S, we perform the following computation: For every vertex v, compute $\sigma_N(S \cup \{v\})$ and $\sigma_T(S \cup \{v\})$. Thus each iteration of the algorithm takes $O(\theta \times |V| \times Inf)$ time. Where Inf is the time taken to compute the influence spreads $\sigma_N(.)$ and $\sigma_T(.)$. Thus the total time taken by this algorithm is $O(k\theta \times |V| \times Inf)$. Finally, we can show that this pruning strategy does not decrease the quality of the solution. To summarize, we have the following theorem.

Theorem 5. *The* Multi Greedy *Algorithm with Pruning runs in time* $O(k\theta \times |V| \times Inf)$ *and the quality of the solution produced by this algorithm is at least as good as the quality of the solution produced by the Baseline Greedy algorithm.*

3.2.3 Efficient Implementation and a Two-Phase Algorithm

To implement Multi Greedy algorithm, we need to perform $k\theta \times |V|$ number of influence spread computation (for computing $\sigma_N(.)$ and $\sigma_T(.)$); this makes it infeasible for large graphs. We propose various techniques improve the efficiency. There are three computational bottlenecks in the above algorithm. First is that, during every iteration, for every set S, we add *every vertex* to S and estimate influence spread. Do we really need to consider every vertex? The second bottleneck is the estimation of influence spread. This is known to be $SharpP$-complete problem and therefore we can not hope to have a efficient algorithm that will estimate this. Finally, the third bottle neck is that of memory. Holding all the candidate sets during every iteration will make it memory as well as time inefficient. To address these issues two issues, we propose a *two-phase* greedy implementation that build an *influence maximization tree* to reduce memory as well as time.

Consider the first bottleneck. Let S be a set $\sigma_N(S) = x$. If v is a node for which $\sigma_N(v) > \theta$, then definitely $\sigma_N(S \cup \{v\}) > \theta$. Thus, we need not consider such nodes; we build upon this by performing even more aggressive pruning. If $\sigma_N(v) > \theta - x$, then it is likely that $\sigma_N(s \cup \{v\}) > \theta$ (though this may not always hold). Thus when we try to extend a set S, we only consider nodes that will influence up to $\theta - x$ non-target nodes. How do we find such nodes? Instead of searching for such nodes each time, we can perform pre-computation—For each threshold value j maintain a set of nodes that will influence j non-target thresholds. This pre-computation is the first phase of our two-phase algorithm.

Phase 1—Estimating the influence spread on non-targets for each vertex : For each node v in the graph, we estimate $\sigma_N(v)$. We store these estimates in a dictionary $\mathcal{D}$ consisting (key, value) tuples of the for $\langle j, A_j \rangle$, where $0 \leq j \leq \theta$ and A_j is the list of nodes that will influence j non-target nodes. Note that the run time of this phase is $O(|V| \times Inf)$.

Phase 2—Constructing the influence maximization tree: This step is the core of our two phase implementation. We construct a n-ary tree ,influence maximization tree (IMTree), with the following properties

- Each tree node consists of three properties (1) vertex (2) influence spread on target nodes σ_T and (3) influence spread on non-target nodes σ_N
- The height of the tree is equal to budget k
- Each tree node has a branching factor of $O(\theta)$.
- Every path in the tree satisfies the property: $\sigma_N(S) \leq \theta$ where S is the set of nodes in the path.

We initialize the IMTree with a node with dummy vertex, $\sigma_T = 0$ and $\sigma_N = 0$.We construct and process the IMTree level by level until the depth k. For each tree node in the IMTree we find the child nodes based on the following

- Find the partial candidate seed set S at a tree node u by traversing the path from u to root.

Algorithm 2: Constructing Influence Maximization Tree Using Simulation

Data: Labeled Social Network $G = (V, E, L)$, target labels T_L non-target labels N_L, budget k, non-target threshold θ

begin

$\mathcal{D} \longleftarrow$ Dictionary Constructed in Phase 1

$R = 10000$

$root \longleftarrow \emptyset$

$root.vertex = -1$, $root.\sigma_T = 0$ $root.\sigma_N = 0$

$IMTree \longleftarrow root$

$level = 0$

while $level \leq k$ **do**

$tree\_nodes \longleftarrow NodesAtLevel(IMTree, level)$

foreach $tree\_node \in tree\_nodes$ **do**

$S \longleftarrow SeedSetInPath(tree\_node)$

$\sigma_N(S) = NonTargetsInPath(tree\_node)$

for $i = 0$ *to* $\theta - \sigma_N(S)$ **do**

$max\_node = FindMaxInfluentialNode(S, D.get(i))$

$tree\_node.addChild(max\_node)$

return $IMTree$

- Find the total influence spread $\sigma_N(S)$ by adding the influence spread to non-targets by each node present in the path from u to the root.
- If the $\sigma_N(u) = \theta'$, then for each i between 0 till $\theta - \theta'$, find a node v ($\notin S$) with maximal marginal gain (i.e., $\sigma_T(S \cup \{v\} - \sigma_T(S)$ is maximal) and $\sigma_N(v) = i$, and add as a child of u. This step takes $O(|V| \times Inf)$ time

Finding the seed set from the influence maximization tree: In this step, given the IMTree, we find the seed set S such that $\sigma_T(S)$ is maximum. To do so, we need to find a max sum path in IMTree, which is equivalent to standard problem of finding max sum path (leaf to root) in a tree.

Next we describe a pruning strategy to improve the efficiency of constructing the IMTree in Phase 2 as in Section 3.2.2.1.

Pruning IMTree: The IMTree constructed could be very large. We discuss how to reduce the number of tree nodes in IMTree. The following are two scenarios where we can reduce the number of tree nodes

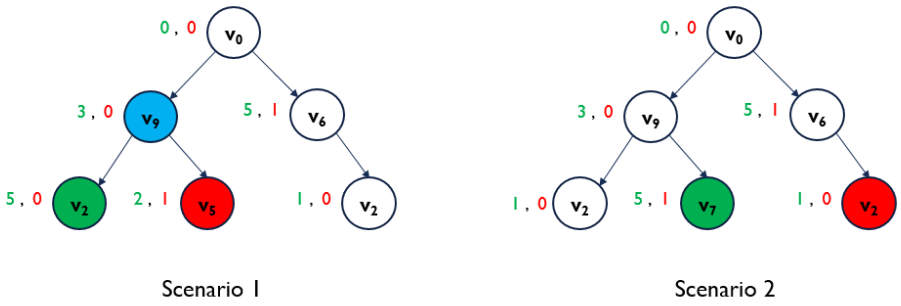


Figure 3.3: Scenarios of IMT Pruning

- For a tree node u, let x and y be two child nodes. If $\sigma_T(x) > \sigma_T(y)$ and $\sigma_N(x) < \sigma_N(y)$, we can ignore y as the child node to u. Clearly with x, with lesser influence spread on non-target nodes we are gaining more influence spread on target nodes. For instance in scenario 1 of the Figure 3.3, we will prune v_5.

- Let x and y be two nodes in the IMTree. If $\sigma_N(S_x) = \sigma_N(S_y)$ and $\sigma_T(S_x) > \sigma_T(S_y)$, where S_x is the set of nodes in the path from x to *root* and S_y is the set of nodes in the path from y to *root*, we can remove tree node y from IMTree due to the same reason as above. For instance in scenario 2 of the Figure 3.3, we will prune v_2.

By pruning the tree nodes in the above scenarios, at each level of tree, we will only have at most of $\theta + 1$ tree nodes, with one tree node for every $\sigma_N = 0, ...\theta$. Hence, the total tree nodes will $O(\theta k)$ compared to that of quadratic number of tree nodes in terms of θ and k. The pseudo code for this algorithm is described as Algorithm 3.

The complexity of the overall strategy is $O(k\theta \times |V| \times \text{inf})$ (see Theorem 5) as the pruned tree contains $O(k\theta)$ nodes and each insertion of node (Step 3 of IMTree construction) takes $O(|V| \times \text{inf})$ time.

This completes the high-level description of the two-phase implementation. To recap, the first phase of the algorithm computes number of non-targets influenced for every node and the second phase constructs a pruned IMTree. Note that in both of these steps, we need to compute influence spread. Thus the time taken for the both of these steps is bounded by Inf.

Algorithm 3: Constructing IMTree With Pruning

Data: Labeled Social Network $G = (V, E, L)$, target labels T_L non-target labels N_L, budget k, non-target threshold θ

begin

$D \longleftarrow EstimateNonTargets(G, N_L)$
$R = 10000$
$root \longleftarrow \emptyset$
$root.vertex = -1$, $root.\sigma_T = 0$ $root.\sigma_N = 0$
$IMTree \longleftarrow root$
$level = 0$
while $level \leq k$ **do**
  $tree\_nodes \longleftarrow NodesAtLevel(IMTree, level)$
  $MaxNodes \longleftarrow \emptyset$
  foreach *tree_node* $t \in$ *tree_nodes* **do**
    $S \longleftarrow SeedSetInPath(tree\_node)$
    $\sigma_N(S) = NonTargetsInPath(tree\_node)$
    $\sigma_T(S) = TargetsInPath(tree\_node)$
    for $i = 0$ *to* $\theta - \sigma_N(S)$ **do**
      $m = MaxInfluentialNode(S, D.get(i))$
      if $\sigma_T(S) + m.\sigma_T) > MaxNodes[\sigma_N(S) + i].\sigma_{T_a}$ **then**
        $\sigma_{T_a} = \sigma_T(S) + m.\sigma_T$
        $MaxNodes[\sigma_N(S) + i) = (m, t, \sigma_{T_a})$
  for $i = 0$ *to* θ **do**
    $(m, t, \sigma_{T_a}) \longleftarrow MaxNodes[i]$
    $t.addChild(m)$
return $IMTree$

Algorithm 4: Estimating Non-Targets Using Simulation

```
Data: Graph G = (V, E), non-target label N
begin
    D ⟵ ∅, R = 10000, s_v = 0 for every v ∈ V
    for node v ∈ V do
        for i = 1 to R do
            A_v = PerformDiffusion(v)
            s_v+ = CountNonTargets(A_v, N)
        s_v = s_v/R /* Estimated number of non-targets influenced by v */
        D.add(s_v, D.get(s_v) ∪ v)
        /*D.get(s_v) returns the current set of nodes associated with s_v */
    return D
```

Next we will discuss several approaches to compute Inf. To compute influence spread in Phase 1, we consider two strategies *simulation* and *random DAGs*. To estimate influence in Phase 2, we consider three strategies *simulation*, *random DAGs* and *Degree Discount*. Note that Degree Discount heuristic can not be used in the phase 1.

3.2.3.1 Phase 1 using Simulation

A pseudocode for this method is shown in Algorithm 4. To estimate the influence spread on non-targets for each vertex $v \in V$, we do the following: We add the vertex v to seed set S and simulate the Independent Cascade (IC) model as described in section 2.1. Let A_v be the set of nodes that are activated at the end of the simulation. We identify node $u \in A_v$ as a non-target if $L(u) \in N_L$ and count the number of non-target nodes in A_v. We repeat the above process for R rounds and take the average of estimate of non-targets.The running time of this algorithm is $O(nRm)$.

3.2.3.2 Phase 2 using Simulation

In phase 2, we need to compute the influence spread while computing the marginal influence gain for each node among the set of probable candidates C (nodes with same σ_N). Again, this algorithm will simulate the diffusion process R many times and take the average. Details are in Algorithm 5.

Algorithm 5: Find Max Influential Vertex Using Simulation

Data: $G = (V, E)$, target label T, Partial Candidate Seed Set S, Probable Candidates C (i.e vertices with same σ_N)

begin

$R = 10000$

for *vertex* $v \in C \setminus S$ **do**

$s_v = 0$

for $i = 1$ *to* R **do**

$A_v = PerformDiffusion(S \cup v)$

$s_v += CountTargets(A_v, T_L)$

$s_v = s_v/R$ - $\sigma_T(S)$ /* *Marginal gain for v* */

$max\_node.vertex = argmax_{v \in C \setminus S}\{s_v\}$

$max\_node.\sigma_T = s_{max\_node.vertex}$ /* marginal gain for the *max_node* */

return *max_node*

Table 3.1: Phase 1 for the graph in Figure 3.1

Non-Targets Estimates (σ_N)	Nodes
0	$\{v_2, v_3, v_4, v_5, v_8, v_9, v_{10}, v_{11}\}$
1	$\{v_1, v_6, v_7\}$

Table 3.1 and Figure 3.5 illustrates the two phases of CTIM problem for graph in the Figure 3.1. Let us consider the budget k to be 3 and θ to be 1. Table 3.1 shows the dictionary constructed in the phase 1 (non-targets estimates). Figure 3.5 illustrates the phase 2 (constructing the IMTree). In the first step, we add the node v_9 to the IMTree as it has the maximum σ_T for $\sigma_N = 0$. In the next step, we add v_6 to the IMTree as it has maximum σ_T for $\sigma_N = 1$. In the next step, we will add node v_2 to v_9. Subsequently, we will add nodes v_1 to v_9 and v_2 to v_6. This marks the end of the construction of IMTree. As we can see that the path $\{v_1, v_9\}$ has the maximum sum of σ_T, hence $\{v_1, v_9\}$ will be the seed set.

3.2.3.3 Phase 1 using Reachability tests on random graphs

Ohasaka et al. Ohsaka et al. (2014) proposed an efficient algorithm to find the marginal influence spread using reachability tests on random graphs, which indeed is used to find the seed set for the influence maximization problem.We borrow this idea to estimate the non-targets. A pseudocode for this algorithm is described in Algorithm 6. Given a graph G, we first generate

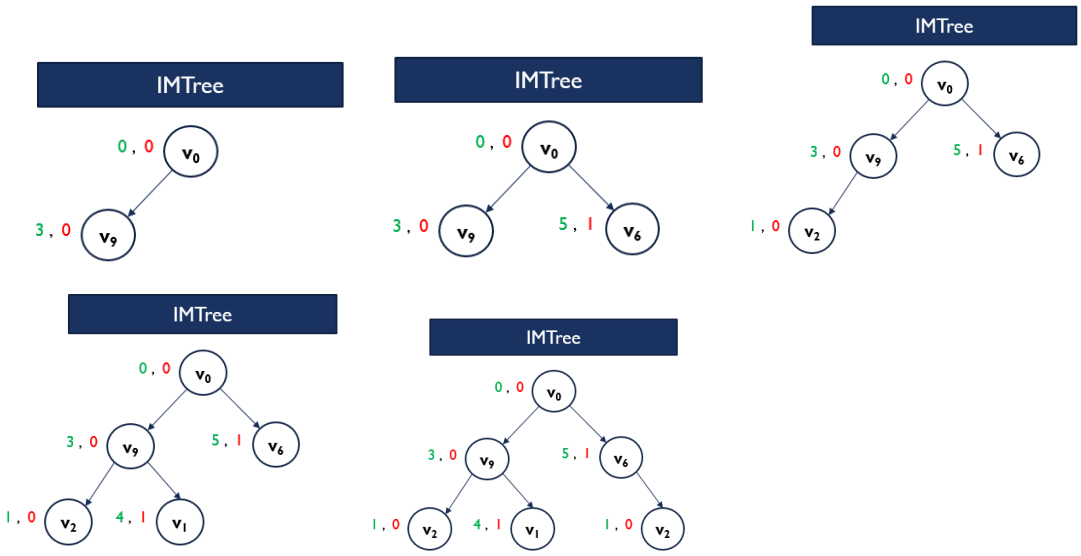


Figure 3.5: Illustration of IMTree construction

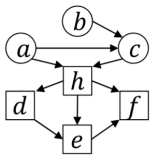
R random graphs using the rule : Each edge $\langle u, v\rangle \in E$ lives with probability $p(\langle u, v\rangle)$. Then, we do BFS on each vertex $v \in V$, let r_v be the set of reachable nodes from v, we store the result in Dictionary. We can estimate the marginal influence spread on non-target nodes for a vertex v by

$$\sigma_N(v) = \frac{1}{R}\sum_{i=1}^{R} CountNonTargets(R_i[v], N_L)$$

However, this method does not improve the time complexity because BFSs still need to be conducted $|V| \times R$ times and it takes $O(|E|)$ for the BFS . In order to make algorithm faster we adopt *cached* BFS (pruned BFS) which reduces the number of nodes visited during the BFSs. The central idea of cached BFS is based on the following: If a vertex v can reach to vertex u, vertices reachable from u are also reachable from v, hence we can prune these vertices while doing BFS from v. Figure 3.6 illustrates BFS pruning on a sample graph.A pseudocode for cached BFS algorithm is described in Algorithm 7.

3.2.3.4 Phase 2 using Reachability tests

We apply similar technique described in Section 3.2.3.3 to find the max influential node. Given C, S, for every node $v \in C$, we calculate the Gain(v, S) for each DAG. Gain(v, S) is nothing but the number of target nodes reachable from v (excluding the ones which are



Vertex	# of vertices visited during normal BFS	pruned BFS
a	6	2
b	6	2
c	5	1

Figure 3.6: A sample graph to illustrate BFS Pruning

Algorithm 6: Estimating Non-Targets Using Reachability Tests

Data: Labeled Social Network $G = (V, E, L)$, non-target labels N_L
begin
  $D \longleftarrow \emptyset$
  $R = 10000$
  for $i = 1$ *to* R **do**
    $E_i^{'} \longleftarrow$ edges by keeping with probability $p(e)$
    $G_i^{'} \longleftarrow (V, E_i^{'})$
    $R_i \longleftarrow ReachableMap(G_i)$
  for $v \in V$ **do**
    $s_v = \frac{1}{R} \sum_{i=1}^{R} CountNonTargets(R_i[v], N_L)$
    $D.add(s_v, D.get(s_v) \cup v)$
  return D

reachable from S).

$$Gain_i(v, S) = CountTarget(R_i[v] \setminus \bigcup_{u \in S} R_i[u], T_L)$$

An approximate of marginal influence spread is obtained by averaging the gain of v in each DAG.

$$\sigma_T(v) = \frac{1}{R} \sum_{i=1}^{R} Gain_i(v, S)$$

We then select the node with the maximum marginal influence spread.

3.2.3.5 Phase 2 using DegreeDiscount

Even with improved greedy algorithms, the running time is still large and may not be suitable for large graphs. Chen et al. proposed Chen et al. (2009) the degree discount heuristic

Algorithm 7: ReachableMap using BFS Caching

```
Data: Labeled Social Network G = (V, E, L)
begin
    R ⟵ ∅
    for v ∈ V do
        Q ⟵ a queue with only one element v
        r_v ⟵ ∅
        while Q ≠ ∅ do
            u ⟵ Dequeue Q
            if R[u] ≠ ∅ then
                r_v.addAll(R[u])
            else
                r_v.add(u)
                for x ∈ {x : (u, x) ∈ E} do
                    if x ∉ r_v then
                        Enqueue Q with x
        R[v] = r_v
    return R
```

to efficiently find the seed set. We modify the DegreeDiscount heuristic to consider the target label information to find the max influential node among the set of nodes C. The pseudocode for this algorithm is described in Algorithm 8. For every vertex $v \in C$, we calculate the degree discount heuristic value

$$dd_v = (1-p)^{|t_v|}(1 + (|d_v - t_v|).p)$$

where $d_v = \{u : (u, v) \in E \wedge u \in T\}$ and $t_v = \{u : (u, v) \in E \wedge u \in S\}$. Then we choose the node with the maximum dd_v.

Algorithm 8: Find Max Influential Vertex Using DegreeDiscount

```
Data: Labeled Social Network G = (V, E, L), target labels T_L, Seed Set S, Probable
      Candidates C (i.e vertices with same σ_N)
begin
    for vertex v ∈ C \ S do
        dd_v = (1 − p)^{|t_v|}(1 + (|d_v − t_v|).p)
    max_node.vertex = argmax_{v∈C\S}{dd_v}
    max_node.σ_T = dd_{max_node.vertex}
    return max_node
```

CHAPTER 4. EXPERIMENT RESULTS

The experiments are designed to validate and evaluate the following:

1. How does the simulation-based estimation of marginal influence spread on non-targets compare to that computed using DAG method?
2. How does the quality of results (estimated number of influenced nodes for a seed of size k) compare between the baseline greedy algorithm and the different realizations of multi-greedy algorithm?
3. How do the proposed methods compared with respect to efficiency?

4.1 System Overview

In this section, we describe our framework of the two phase implementation of Multi-Greedy for CTIM problem. Figure 4.1 illustrates high level modules in our framework. Our framework consists of four major modules

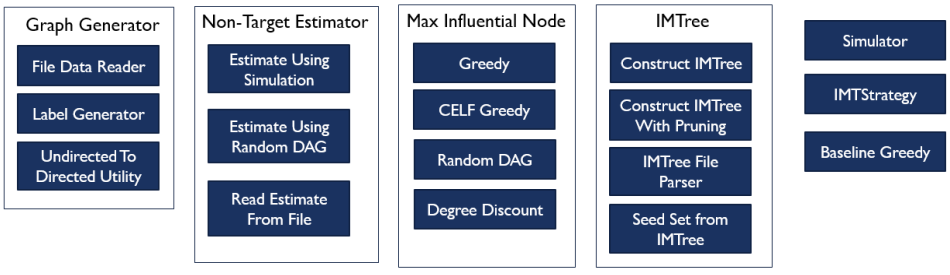


Figure 4.1: System Overview

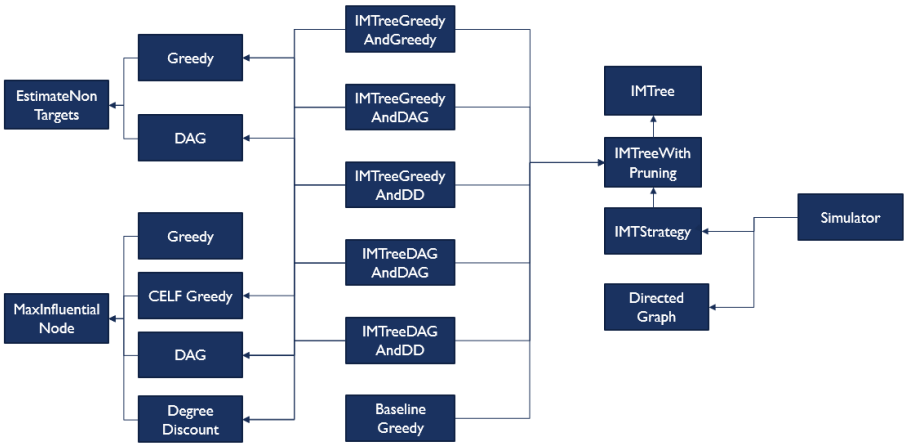


Figure 4.2: Interaction/Class Diagram

1. **Graph Generator Module:** This module is responsible for generating a graph from a file, generating labels for the nodes and converting undirected graph to a directed graph.

2. **Non-Target Estimator Module:** This module is responsible for estimating the non-targets for every node in the graph (Phase 1). This module consists of two strategies (implementations) to estimate the non-targets: 1) Using Simulation 2) Reachability Tests on random DAG's.

3. **Max Influential Node Module:** This module is responsible for finding the max influential node among the given set of nodes (crucial sub step in Phase 2). This module has three strategies : 1) Using Simulation 2) Reachability Tests on random DAG's 3) Degree Discount Heuristic.

4. **IMTree Module:** This module is responsible for constructing the IMTree based on the strategy input and finding the seed set from constructed IMTree.This module has two variations for constructing the IMTree - one with pruning and other without pruning. This module also consists of a utility to read/write IMTree from/to a file.

Figure 4.2 shows the high level class/interaction diagram of the system. Based on the input of the strategy, corresponding implementation of phase 1 and phase 2 will be chosen.

Table 4.1: DataSets

DataSet	Nodes	Edges	Type
ca-GrQC	5,242	14,496	Undirected
ca-HepTh	9,877	25,998	Undirected
wiki-Vote	7,115	103,689	Directed
ca-HepPh	12,008	118,521	Undirected
ca-AstroPh	18,772	198,110	Undirected
soc-Epinions1	75,879	508,837	Directed

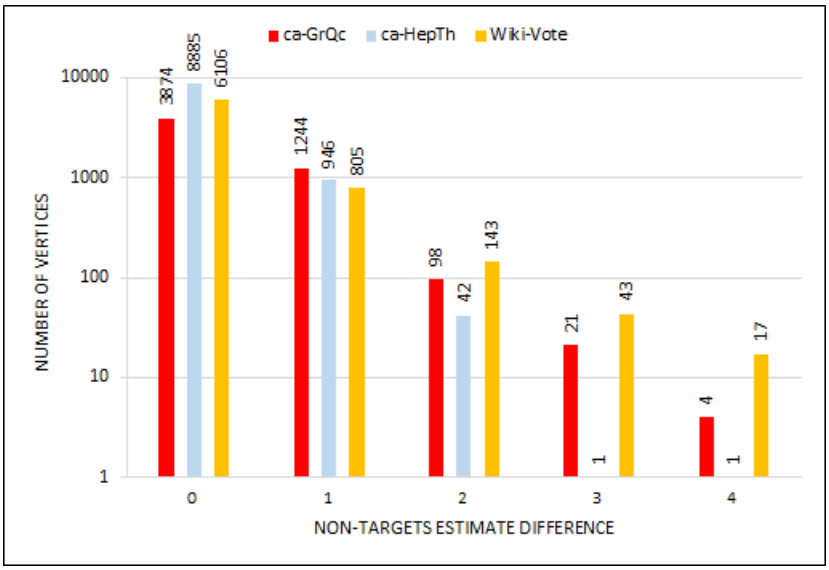


Figure 4.3: Non-Targets estimates difference between Simulation-based and DAG-based algorithms

Framework Benefits: The following are the benefits of our framework

- Every phase is modularized, which enable us to plug in different algorithms (implementations) for different phases very easily.
- Every phase result is cached, hence we can overcome the overhead of running everything every time.
- IMTree is cached, so we can query with varying budget.
- The propagation/diffusion models can be interchanged easily.

4.2 Experimental Setup

Dataset. Our dataset contains 6 social networks (Table 4.1): two directed and four undirected graphs. For the undirected graphs, we consider the influence can propagation in both directions. We have considered networks of different sizes with respect to the number of nodes and edges. There are three types of networks

- Collaboration network: These networks are from the e-print arXiv and covers scientific collaborations between authors papers submitted to a specific category. Each node in the network represents an author, and each edge represents the two authors collaborated on a paper. The categories of scientific discipline considered are **ca-GrQc**: "General Relativity and Quantum Cosmology" category, **ca-HepTh :** "High Energy Physics - Theory" category, **ca-AstroPh :** "Astro Physics" category, and **ca-HepPh :** "High Energy Physics - Phenomenology" category.

- Trust networks: **soc-Epinions1** is a who-trust-whom online social network of site `Epinions.com` where each vertex represents a user and each edge represents a trust relationship.

- Voting network: **wiki-Vote** is a Wikipedia vote network, where each vertex represents Wikipedia user and each edge represents vote relationship (who-voted-whom).

Datasets are available at `http://snap.standford.edu/data/`.

Environment. All experiments are conducted on Linux server (Virtual machine) with AMD Opteron 6320 CPU (8 cores and 2.8 GHz) and 32GB main memory. All the algorithms were implemented in Java.

Notations. We run the proposed algorithms under IC model. In the rest of the section, we will refer to these algorithms as follows.

Baseline_Greedy. This is the baseline greedy algorithm (Algorithm 1) with CELF optimization Leskovec et al. (2007).

Multi-Greedy. We have proposed a two-phase efficient realizations of multi-greedy algorithm and realized different variants of the two-phase method. The variants considered are as follows.

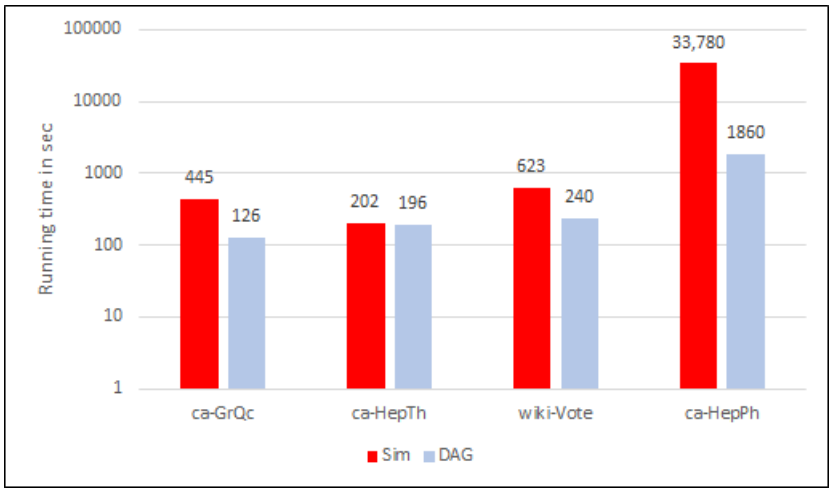


Figure 4.4: Running time for Non-Targets estimates difference using Simulation-based and DAG-based algorithms

Sim_Sim : In this algorithm, both phase 1 and phase 2 are estimated using Simulation with CELF optimization Leskovec et al. (2007).The results on original greedy are not reported since the influence is the same as CELF optimization while its running time is very slow.

Sim_DAG: In this algorithm, Phase 1 is by using Simulation (Algorithm 4) and Phase 2 is by using reachblity tests on random graphs.

Sim_DD : In this algorithm, Phase 1 is by using Simulation (Algorithm 4) and we Phase 2 is by using degree discount heuristic (Algorithm 8)

DAG_DAG : In this algorithm, Phase 1 is estimated using reachbaility tests (Algorithm 6) and Phase 2 is by using reachability tests.

DAG_DD : In this algorithm, Phase 1 is estimated using reachbaility tests (Algorithm 6) and Phase 2 is by using degree discount heuristic (Algorithm 8)

4.3 Experimental Results

4.3.1 Estimation of Influencing Non-targets

Our first objective is to experimentally evaluate the differences between the estimates of influencing non-targets as computed by the simulation-based (Algorithm 4) and the DAG-based (Algorithm 6) algorithms. Figure 4.3 presents the number of nodes (y-axis) with estimated

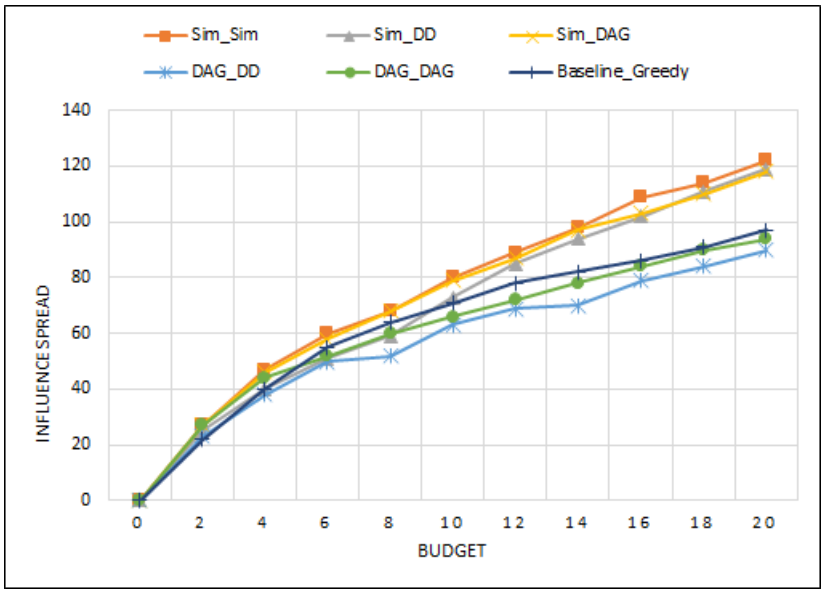


Figure 4.5: Influence spreads of different algorithms on the graph CA-GrQc ($\theta = 10$ and $p = 0.05$)

differences (x-axis) between simulation-based and DAG-based algorithms when applied to ca-GrQc, ca-HepTh and wiki-vote networks. It shows that 97.0% vertices has the difference between the estimates ≤ 1.

Figure 4.4 shows the running time for estimating the non-targets using these algorithms. Note that y-axis is in log scale. As expected, the DAG-based algorithm takes far lesser time compared to simulation-based one. The running time of the simulation-based approach increases significantly with the increase of number of vertices, edges and the propagation probability.

4.3.2 Estimation of Influencing Targets

Our second objective is to evaluate the quality of the results obtained by applying the proposed methods. We considered the threshold $\theta = 10$ and experimented with different values for the size of the seed set ranging from $k = 2, 4, 6 \ldots, 20$ (budget). Figures 4.5, 4.6 and 4.7 show the influence spread of various algorithms on the graphs ca-GrQc, ca-HepTh and wiki-Vote, respectively, under the IC model.

As expected, the Sim_Sim algorithm computes the seed set with highest influence spread among the targets while keeping the non-target influence within the threshold. With respect to

the influence spread achieved using the resulting seed set, we computed the average percentage difference of each algorithm from the Sim_Sim.

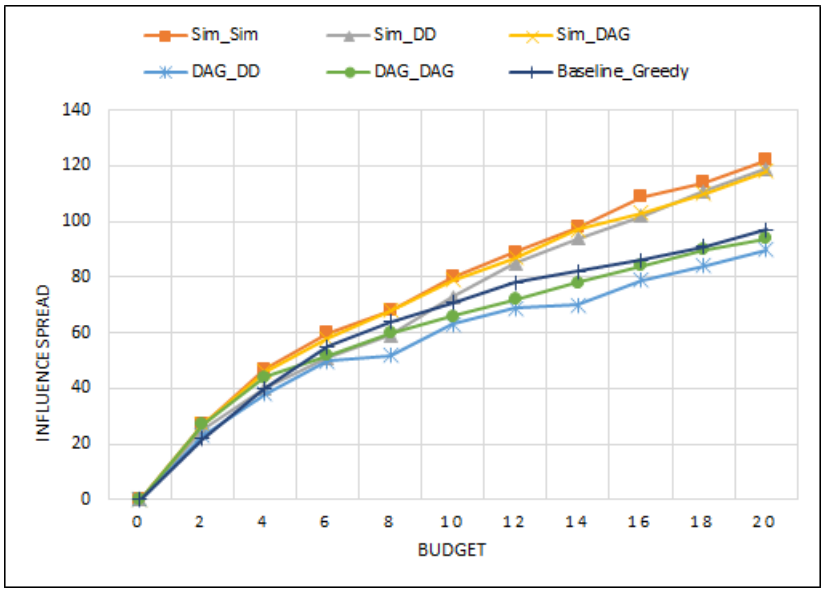


Figure 4.6: Influence spreads of different algorithms on the graph CA-HepTh ($\theta = 10$ and $p = 0.05$)

	% Lower than Sim_Sim by		
	ca-GrQc	ca-HepTh	wiki-Vote
Sim_DAG	6.6	3.2	9.9
Sim_DD	10.0	2.45	11.6
Baseline_Greedy	9.45	14.93	18.18
DAG_DAG	16.46	22.9	38.2
DAG_DD	24.5	26.2	36.8

Note that, the percentage difference is higher for the wiki-vote network. This can be attributed to the structure of the network, where the number edges in the largest SCC is 38.1% compared to that 92.6% in ca-GrQc and 95.5% in ca-HepTh.

Figure 4.8 shows the running time for various algorithms on networks ca-GrQc, ca-HepTh and wiki-Vote for $k = 20$. The y-axis represents time in log scale. Sim_DAG, DAG_DAG takes are almost four orders of magnitude faster than that of Sim_Sim. Sim_DD and DAG_DD are even faster. DAG_DD is the fastest of all algorithms. For even a medium dataset Sim_Sim

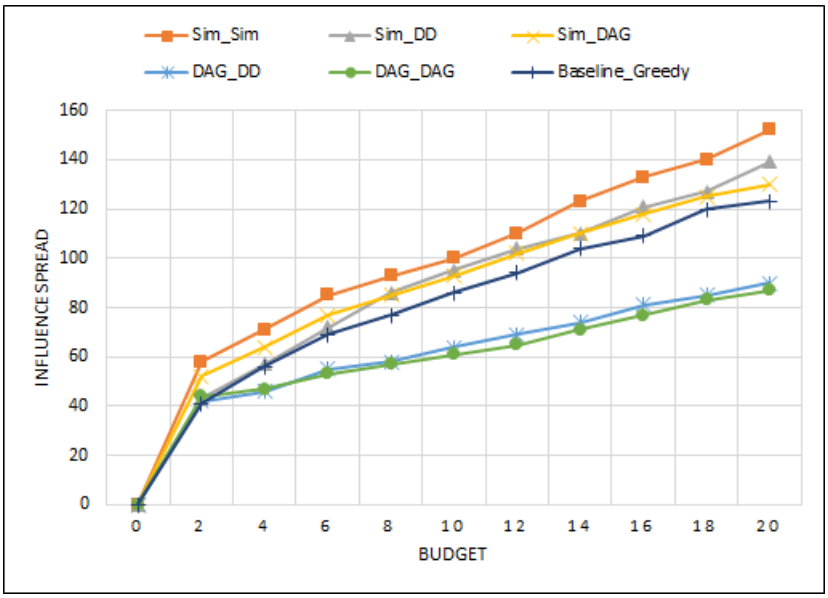


Figure 4.7: Influence spreads of different algorithms on the graph Wiki-Vote ($\theta = 10$ and $p = 0.02$)

algorithm takes several hours and will not be practical to apply for large social networks. After estimating the non-targets, finding seed set takes significantly less time when DD-based and DAG-based algorithms are applied (as opposed to simulation-based ones). DAG_DAG and DAG_DD are the most scalable for the large social networks; however, it is important to recall that the diffusion spread using these algorithms can be 15–40% less than that obtained using Sim_Sim algorithm. We believe that for very large networks, the benefits of finding a result within reasonable time can outplay the cost in terms of the degree of quality.

4.3.3 Application to Large Social Networks

As the Baseline_Greedy, Sim_DAG and Sim_DD algorithm takes several hours on medium size networks, we conduct the experiments on larger social networks only using DAG_DAG and DAG_DD. Figure 4.9, 4.10 and 4.11 shows the influence spread on ca-HepPh, ca-AstroPh and soc-Epinions respectively. The results are very close, for ca-HepPh network DAG_DAG the influence spread is 6.02% more than DAG_DD, for ca-AstroPh network DAG_DAG the influence spread is 5.37% less than DAG_DD and for soc-epinions network DAG_DAG the influence spread is 6.15% less than DAG_DD.

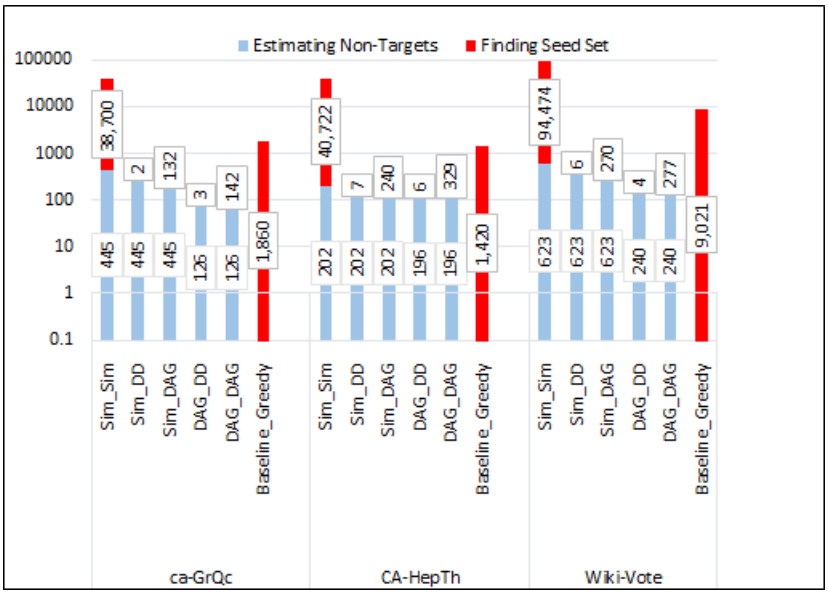


Figure 4.8: Running times of different algorithms on the graph CA-GrQc, Wiki-Vote, CA-HepTh

Figure 4.12 shows the running time of DAG_DAG and DAG_DD algorithms on networks ca-HepPh, ca-AstroPh and soc-Epinions for $k = 51, \theta = 50, p = 0.01$. DAG_DD takes far lesser time compared to DAG_DAG on all the above networks.

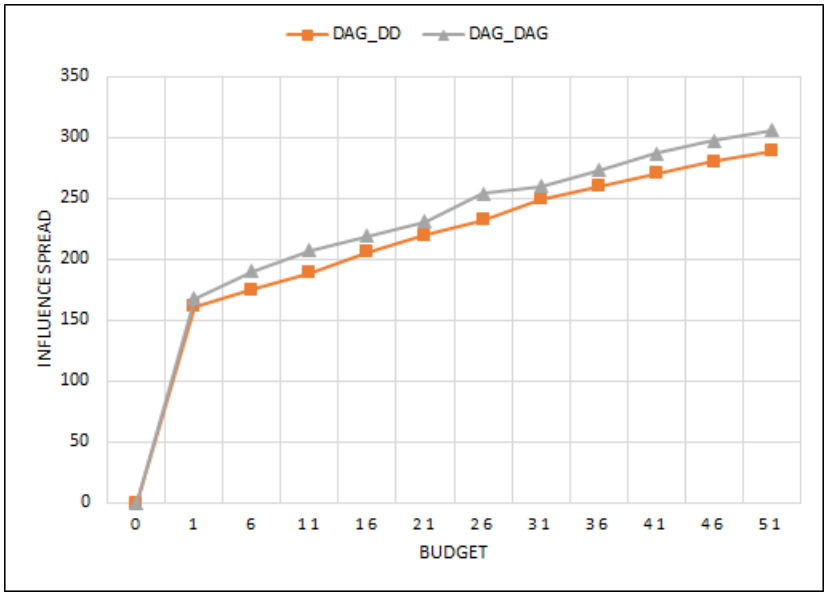


Figure 4.9: Influence spreads of DAG_DAG and DAG_DD algorithms on the graph CA-HepPh ($\theta = 50$ and $p = 0.01$)

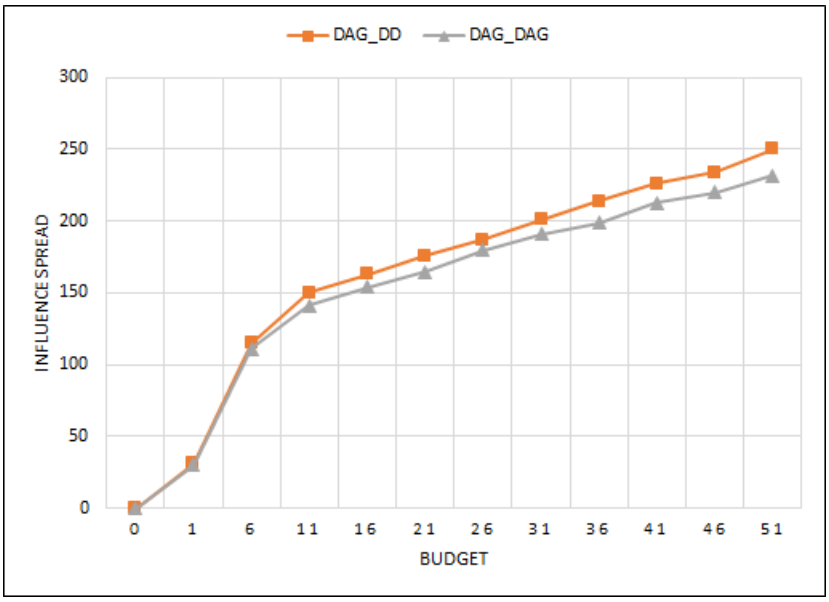


Figure 4.10: Influence spreads of DAG_DAG and DAG_DD algorithms on the graph CA-AstroPh ($\theta = 50$ and $p = 0.01$)

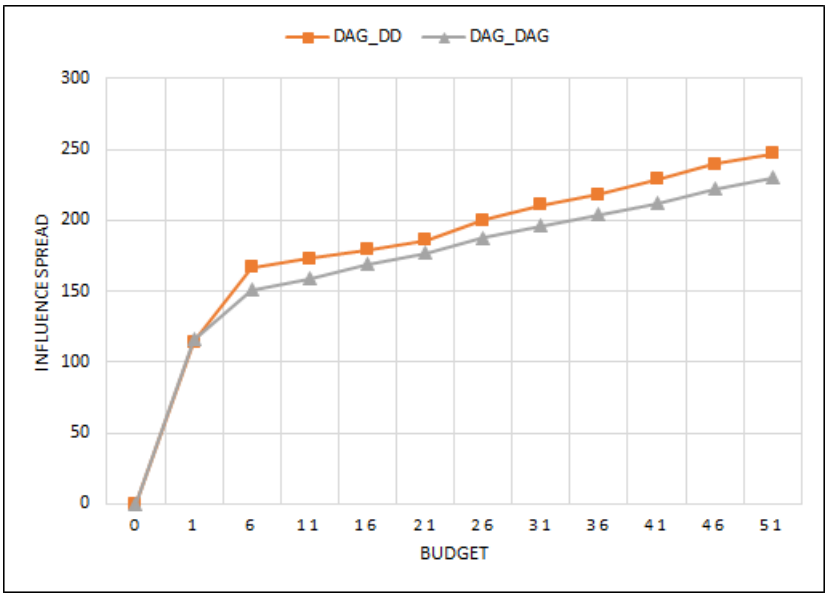


Figure 4.11: Influence spreads of DAG_DAG and DAG_DD algorithms on the graph soc-Epinions ($\theta = 50$ and $p = 0.01$)

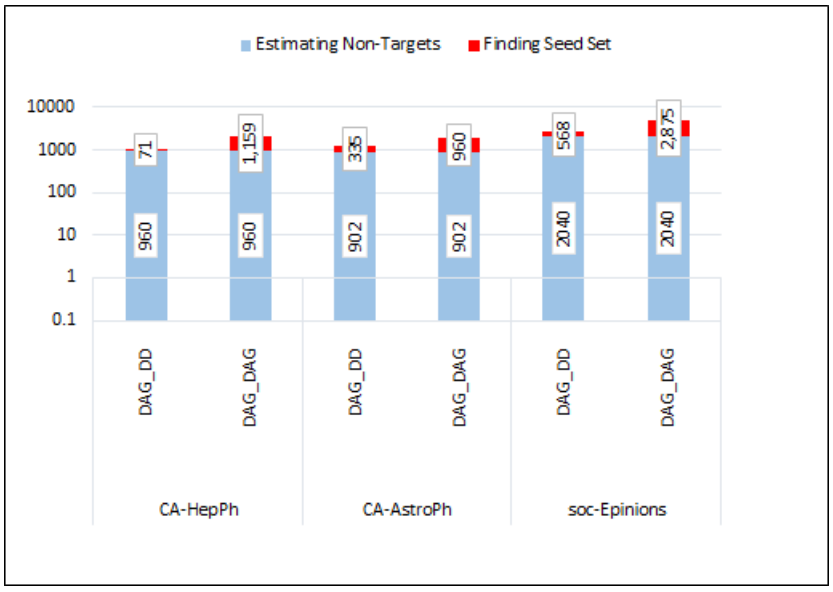


Figure 4.12: Running times of DAG_DAG and DAG_DD algorithms on the collaboration graph CA-HepPh, CA-AstroPh and soc-Epinions

CHAPTER 5. SUMMARY AND DISCUSSION

We introduce the Constrained Targeted Influence Maximization problem in social networks for target marketing which focus on maximizing the influence spread on target nodes while restricting the influence spread on non-target nodes. We propose a baseline greedy algorithm for this problem with guarantee on the influence spread on target nodes. We introduced the concept of robust optimality to characterize the quality of solutions relevant in the context of non-target nodes. We further propose a two phase tree based framework with combination of simulation-based algorithms to further improve the influence spread with a guarantee of solution which is at least as good as the solution produced by the baseline greedy. We have improved the efficiency of the two phase algorithm using combination of heuristics methods. The experiment results suggest that the proposed methods perform faster while quality of the influence spread is not considerably compromised.

5.1 Future Work

1. **Neutral Customers:** In CTIM, we consider only target and non-target customers. We can extend our problem by considering the neutral customers (neither target nor non-target). To motivate this, let us consider the following real world scenario : While marketing you would like to spread the influence to the customers, who has more influence on the target customers, although the customer might not be interested (neutral customer) in the product.

2. **Dynamic Social Networks :** CTIM deals with the influence maximization in static networks. However, in real world, many of the networks exhibit dynamic behaviour, specifically, the network changes over time and the changes can be observed by periodi-

cally probing some nodes for the update of their connections. We can extend the CTIM to maximize influence diffusion in a dynamic social network. .

3. **Topic Distributions:** Evidently, in real-world social networks, users have their own interests, which can be represented by topics, and are more likely to be influenced by users with similar interests. We can extend the CTIM problem with topic distributions by representing the social network as topic aware graph, where each edge represents the distribution of influence probablities on various topics.

4. **Rumor Source Detection:** Identifying rumor sources in social networks plays a critical role in limiting the damage caused by them through the timely quarantine of the sources.We can extend the framework of CTIM to solve this problem by labelling the influenced nodes as target nodes and others as non-target nodes and estimating the seed set using our framework.

5. **Different Heuristics:** We plan to use our framework to evaluate different heuristics and investigate how the structure of the network influences the quality of the result

BIBLIOGRAPHY

Aaker, J., Brumbaugh, A., and Grier, S. A. (2000). Non-target market effects and viewer distinctiveness: The impact of target marketing on attitudes. *Journal of Consumer Psychology*, 9(3):127–140.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439):509–512.

Chen, S., Fan, J., Li, G., Feng, J., Tan, K.-l., and Tang, J. (2015). Online topic-aware influence maximization. *Proc. VLDB Endow.*, 8(6):666–677.

Chen, W., Wang, Y., and Yang, S. (2009). Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 199–208.

Domingos, P. and Richardson, M. (2001). Mining the network valueof customers. In *Proceedings of the 7th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 57–56.

Erdös, P. and Rényi, A. (1960). On the evolution of random graphs. In *The Mathematical Institute of the Hungarian Academic of Sciences*, pages 17–61.

Goyal, A., Bonchi, F., and Lakshmanan, L. V. S. (2011a). A data-based approach to social influence maximization. *Proc. VLDB Endow.*, 5(1):73–84.

Goyal, A., Lu, W., and Lakshmanan, L. V. (2011b). Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 47–48. ACM.

Kautz, H., Selman, B., and Shah, M. (1997). Referralweb: Combining social networks and collaborative filtering. *Communications of the ACM*, 30(3).

Kempe, D., Kleinberg, J., and Tardos, E. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 137–146.

Larson, R. J. (2009). *The rise of viral marketing through the new media of social medial.*

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 420–429.

Li, F., . Li, C., and Shan, M. (2011). Labeled influence maximization in social networks for target marketing. In *2011 IEEE Third Intl Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Intl Conference on Social Computing*, pages 560–563.

Li, Y., Zhang, D., and Tan, K.-L. (2015). Real-time targeted influence maximization for online advertisements. *Proc. VLDB Endow.*, 8(10):1070–1081.

Ohsaka, N., Akiba, T., Yoshida, Y., and Kawarabayashi, K. I. (2014). Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *Proceedings of the AAAI*, pages 138–144.

Watts, D. and Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393:440–442.