Influence Maximization via Vertex Countering

Jiadong Xie
Guangzhou University
Chinese University of Hong Kong
jdxie@se.cuhk.edu.hk

Zehua Chen
Guangzhou University
czeh@e.gzhu.edu.cn

Deming Chu
University of New South Wale
deming.chu@unsw.edu.au

Fan Zhang*
Guangzhou University
zhangf@gzhu.edu.cn

Xuemin Lin
ACEM, Shanghai Jiao Tong University
xuemin.lin@sjtu.edu.cn

Zhihong Tian
Guangzhou University
tianzhihong@gzhu.edu.cn

ABSTRACT

Competitive viral marketing considers the product competition of multiple companies, where each user may adopt one product and propagate the product to other users. Existing studies focus on a traditional seeding strategy where a company only selects seeds from the users with no adopted product to maximize its influence (i.e., the number of users who will adopt its product). However, influential users are often rare, and the gain from traditional seeding will degrade as the number of seeds increases. Therefore, in this paper, we study the promising *countering* strategy which is to counter some users who initially use other products s.t. they will turn to adopting the target product and recommending it to others.

We propose the problem of *influence countering*: given a graph, a budget b, a target company C_t, and a set S of the seeds adopting different companies (where each seed adopts one company), we counter b users in S who do not adopt C_t to turn to adopt C_t s.t. the expected number of users who eventually adopt C_t in the influence diffusion is maximized. Following existing studies, we formalize the diffusion process by the Multi-Campaigner Independent Cascade model. We prove the influence countering problem is #P-complete and its influence computation is #P-hard. Then, we propose two novel algorithms *MIC* and *MIC*$^+$ to address the problem. In general, *MIC* estimates seed influence by its empirical average influence in multiple graph samplings, while *MIC*$^+$ improves *MIC* by reducing the cost of influence estimation and the required number of samples. Given pre-set ε and l, both algorithms return a $(1-\varepsilon)$-approximate solution with at least $1-n^{-l}$ probability. We also design an index for *MIC*$^+$ to efficiently process graphs that are frequently updated. The experiments on 8 real-world datasets show that our algorithms are efficient in practice while offering strong result quality.

PVLDB Reference Format:
Jiadong Xie, Zehua Chen, Deming Chu, Fan Zhang, Xuemin Lin, and Zhihong Tian. Influence Maximization via Vertex Countering. PVLDB, 17(6): 1297 - 1309, 2024.
doi:10.14778/3648160.3648171

* Fan Zhang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 17, No. 6 ISSN 2150-8097.
doi:10.14778/3648160.3648171

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/mao-qiu/countering.

1 INTRODUCTION

The word-of-mouth effect of social networks is important in viral marketing [7, 12, 16, 41, 42], where a company selects some *influential users* as seeds to adopt and propagate the product of the company through user interactions s.t. there will be a large information cascade of the product. Kempe et al. are the first to formulate the above process as a combinatorial optimization problem, the *influence maximization* problem [21]: given a graph G and a budget b, the problem is to find b seed users in G s.t. the number of the users influenced by the information spread from the seeds (i.e., the users adopt the product selected by the seeds) is maximized.

Traditional viral marketing assumes that only a single company is promoting its product. But in reality, the company must compete with other competitors for market shares. Thus, *competitive viral marketing* [5, 9, 10, 18, 32] allows multiple companies to compete for influence spread, where each seed user adopts the product of a company and each user will influence the choice of other users through information cascade. A line of studies [5, 10, 33] aims to maximize the influence spread of a *target company* (i.e., a target product) in competitive viral marketing. They assume that some users have already adopted the competing products, and the company only selects new seeds from the remaining users who have not adopted any product.

However, influential users are often rare, and the gain from traditional seeding will degrade as the number of seeds increases. For instance, when a newcomer company plans to promote its product on a social network, it may find that the majority of influential users have already adopted other products and it is hard to find an influential user in the remaining users. In comparison, if the company successfully counters some users who initially adopt other products, then the company can have influential users to promote its product and also suppress the spread of the competitors. Therefore, in this paper, we focus on the strategy to *counter* some seed users who initially use other products to start a cascade of adopting the target product. The effectiveness of the countering strategy is validated in Section 3.2 through detailed comparisons with other seed selection methods.

In this paper, we propose the problem of *influence countering*: given a graph, a budget b, a target company C_t, and a set S of seeds that adopt different companies (where each seed adopts one company), we aim to counter b users in S not adopting C_t to adopt and recommend C_t s.t. the expected number of users who eventually adopt C_t in the influence diffusion is maximized, under the MCIC model [22]. Besides, due to the dynamic nature of social networks [23, 24] and the dynamic influence strength (activation probability) between two users, we also study the efficient update of the solution for graph dynamics. Apart from the applications of marketing, influence maximization via vertex countering can also be used in other domains, e.g., restraining the spread of misinformation and disseminating positive information.

Challenges. To the best of our knowledge, we are the first to study the problem of influence countering, and no existing works consider the countering strategy in influence maximization. We prove that the problem is #P-complete and the influence computation of the problem is #P-hard. Firstly, we design a baseline (*BIM*) based on the influence maximization method under the IC diffusion model [16]. However, *BIM* does not consider the influence competition of multiple companies, and the selected seeds are less influential in the competition. Then, we propose a greedy baseline (*BGA*) with Monte-Carlo simulations that consider influence competitions in seed selection.

The influence spread of *BGA* is larger than *BIM* by an average of 17%, as shown in Exp. 2 of Section 7.2, but it still suffers from a large time cost, e.g., it cannot finish in one day for a graph with 420K edges in our experiments. In short, conventional approaches cannot solve our studied problem effectively within an acceptable time cost. This motivates us to propose a novel solution from the first principle.

Our Solution. We present *MIC* and *MIC*$^+$, two algorithms for *M̲aximizing I̲nfluence via C̲ountering* using a new framework. We first prove that the expected influence of the target company always increases by a constant $\sigma(s)$ for countering a seed s, regardless of the countering of other seeds (Theorem 3). That is, for a given seed set, the expected influence gain from countering a seed is a constant irrelevant to the countering of any seed combination. Based on this observation, *MIC* counters b seeds with each seed s having a top-b gain $\sigma(s)$, which results in an optimal result if $\sigma(s)$ is exactly computed for each candidate s.

We also analyze the computation of $\sigma(s)$ and the necessary conditions of an approximate guarantee for *MIC*. Then, We propose *MIC*$^+$, an algorithm that estimates $\sigma(s)$ by computing the probability that a set of vertices will reach s by reverse samplings, which reduces the estimation cost. Furthermore, we propose a lower bound estimation method to reduce the number of graph samples required in the computation. The approximation guarantee is retained in *MIC*$^+$ with the well-designed techniques. To efficiently process dynamic graphs, we propose an index for *MIC*$^+$ by storing and carefully updating the reverse samplings of the graph. Then, we can efficiently process various update cases such as edge insertion/removal and seed addition/deletion.

On the theoretical side, *MIC* and *MIC*$^+$ return a $(1-\varepsilon)$-approximate solution with at least $1 - n^{-l}$ probability, and both of them run in $O((lnm \log n)/(b\varepsilon^2))$ time, where b is often set to a proportion of n, e.g., $b = 0.001n$. On the practical side, our experiments demonstrate that *MIC*$^+$ outperforms the baseline *BGA* in runtime by up to 4 orders of magnitude, and outperforms *MIC* by up to 3 orders.

Contributions. Our principal contributions are as follows.

- We motivate and formulate the problem of influence countering, proving it is #P-complete and influence estimation is #P-hard. To our best knowledge, we are the first to study this problem.
- We propose *MIC* that returns a $(1 - \varepsilon)$-approximate solution for our problem with a high probability guarantee. Then, we devise *MIC*$^+$ that significantly improves the efficiency of *MIC* and retains the approximation guarantee.
- We extend *MIC*$^+$ to process dynamic graphs, using a well-designed index that can efficiently handle the change of edges, vertices, propagation probabilities, and seeds.
- The experiments on 8 real-world graphs show that our algorithms are effective in result quality and efficient in time cost.

2 RELATED WORKS

The problem of influence maximization is surveyed in [1, 3, 28]. Domingos et al. [13] first study the influence of social network users in marketing. Kempe et al. [21] model this problem as influence maximization, and propose a $(1 - 1/e)$-approximate algorithm for the influence maximization under independent cascade (IC) and linear threshold (LT) diffusion models. Borgs et al. [7] propose a novel method for influence maximization based on reverse sampling. After that, reverse sampling is widely adopted by subsequent works [2, 4, 16, 19, 35–37, 40–42] to improve the scalability of influence maximization. Some works consider influence maximization from specific perspectives, e.g., topic-based [6, 11], learning-based [45], location-aware [15, 26, 47], time-constrained [29–31, 50] and the regret of seed users [51].

Competitive influence maximization considers the competition between companies. Most existing works focus on minimizing the influence of competitors [9, 18, 48, 49] or maximizing the influence of a target company [5, 33]. Lu et al. [32] consider influence maximization from the perspective of the network host, i.e., maximizing the gain of all the companies. Li et al. [27] study the best influence maximization strategy of a target company using Nash Equilibrium. Goyal et al. [14] propose a 2-phase influence model including the switching phase being aware of the product and the selection phase to choose products. He et al. [17] prove that Goyal et al.'s model is an instance of a threshold model. Borodin et al. [8] extend the threshold model to the OR model where the awareness of each product is diffused independently. Tsaras et al. [43] propose the ATI model that considers the similarity between user preferences and product features. Lu et al. [33] study the complementary effect between products in influence propagation. Different from the above studies, this paper aims to counter the seeds from the competitors such that the influence of the target company is maximized. We mainly focus on the multi-campaigner IC model as the diffusion model, and we further extend our algorithms to triggering models.

As social networks are often evolving, some works study influence maximization on dynamic graphs. Ohsaka et al. [38] design a $(1 - 1/e)$-approximate algorithm for dynamic influence maximization based on reverse sampling. Wang et al. [46] propose a streaming algorithm for influence maximization over social streams. Peng et al. [39] design the SOTA algorithm for graphs with only edge insertions. Bevilacqua et al. [4] alleviate the issue of large memory usage in dynamic influence maximization.

Table 1: Summary of notations

Notation	Definition
$G = (V, E)$	a directed graph with vertex set V and edge set E
$V(G)$; $E(G)$	the vertex set of G; the edge set of G
n; m	the number of vertices/edges in G (assume $m > n$)
N_u^-; N_u^+	the set of in-neighbors/out-neighbors of vertex u
d_u^-; d_u^+	the in-degree/out-degree of vertex u
S	a fixed set of seeds in graph G
b	budget, i.e., the number of seeds to counter
C_i; C_t	the i-th/target company ($i \in \{1, \cdots, K\}$)
$S_{\neg t}$	the seeds that do not adopt C_t, i.e., $\{s \mid s \in S \wedge c_s \neq C_t\}$
c_u	the company that vertex u adopts before countering
A	the set of countered seeds, i.e., setting $c_A(s) = C_t$ if $s \in A$
$c_A(s)$	$c_A(s) = C_t$ if $s \in A$; and $c_A(s) = c_s$ otherwise
$I(C_t, A)$	the influence spread of C_t when propagating with c_A
g	a graph sample, it removes $(u, v) \in G$ with $1 - p_{u,v}$ probability
$\delta_g(u, v)$	the shortest distance from a vertex u to a vertex v in g
g^r; $\delta_g^r(u, v)$	the reverse of g, and the shortest distance u to v in g^r
$\sigma(s)$	the spread of a vertex s (see Theorem 3)
$p(s)$	the probability that s activates a random vertex in the diffusion
$\hat{\sigma}_i(s)$; $\hat{p}_i(s)$	an estimation of $\sigma(s)/p(s)$ on the i-th graph sample
$\mathbb{E}[x]$	the expected value of x
OPT	the maximum $\mathbb{E}\big[I(C_t, A)\big] - \mathbb{E}\big[I(C_t)\big]$ for any A with size-b

However, the sampling for our problem requires the activation probability assignment of each vertex according to its distance from the seeds, which is not considered in the sampling of IM problems (as illustrated in Sections 3.2 and 4). So, the methods of influence maximization cannot be used to efficiently solve the influence countering problem.

3 PRELIMINARIES

In this section, we introduce the MCIC diffusion model, different seed selection methods, the influence countering problem, and the baseline algorithms. Table 1 summarizes the notations.

3.1 MCIC Diffusion Model

Consider a social network $G = (V, E)$ in which each directed edge (u, v) is associated with a *propagation probability* $p_{u,v} \in [0, 1]$. The *Multi-Campaigner Independent Cascade (MCIC)* model [22] formalizes a diffusion process where K companies $C_1, \cdots, C_K$ are competing for influence propagation as follows:

(1) At timestamp 1, we *activate* a set S of seed vertices, and set the remaining vertices as *inactive*. Each $s \in S$ is activated with a pre-assigned company $c_s \in \{C_1, \cdots, C_K\}$.
(2) If a vertex u is activated with company c_u at timestamp i, then for each edge that points from u to an inactive vertex v, u has $p_{u,v}$ probability to activate v with company c_u at timestamp $i + 1$. After that, u cannot activate any vertex.
(3) If a vertex u is successfully activated by vertices $w_1, \cdots, w_x$ at the same timestamp, then the company of u is uniformly selected from $c_{w_1}, \cdots, c_{w_x}$ with probability $1/x$.
(4) Once a vertex is activated, it remains active, i.e. it will not be inactivated.

The above process mimics the spread of K competing products in a social network: people may choose a product bought and recommended by friends. The *influence spread* of C_t, denoted by $I(C_t)$, is the number of vertices that are activated with company C_t when the MCIC diffusion converges, i.e., no more vertices can be activated.

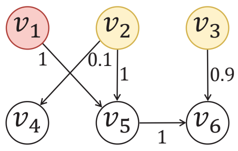


Figure 1: Spread

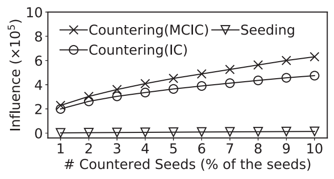


Figure 2: Seed seletion

3.2 Seed Selection Methods

We first illustrate the differences in seed selection methods and then show the effectiveness of countering by a case study. The graph depicted in Figure 1 has three seed vertices, in which seed v_1 adopts the target company C_1 (marked in red), while seeds v_2 and v_3 adopt the competing company C_2 (marked in yellow). After selecting the seeds by each method, we compute the influence of countering the seeds for each company by the MCIC model [22].

(1) If we use the traditional seeding approach under the IC model [21] (the one-campaigner model), i.e., iteratively selecting a user with no adopted company with the largest influence gain (denoted by *Seeding*), as both v_5 and v_6 will certainly be activated by v_1, we will select v_4 as the next seed. According to MCIC, in the influence computation (different from seed selections as we have multi-campaigners), v_6 is first activated by v_3 with 0.9 probability and the influence spread of seeding $\{v_1, v_4\}$ becomes $1(v_1) + 1(v_4) + 0.5(v_5) + 0.1 \times 0.5(v_6$ by v_1 through $v_5) = 2.55$;

(2) If we use the countering approach under the IC model, i.e., iteratively selecting a user who adopted a non-target company with the largest influence gain (denoted by *Countering(IC)*), as v_5 and v_6 are activated by v_1 under IC, we will select v_2 as the next seed. After selecting the seeds, in the influence computation by MCIC, v_6 is first activated by v_3 with 0.9 probability, the influence spread becomes $1(v_1) + 1(v_2) + 0.1(v_4) + 1(v_5) + 0.1(v_6) = 3.2$;

(3) If we use the countering approach under the MCIC model, i.e., iteratively selecting a user who adopted a non-target company with the largest influence gain (denoted by *Countering(MCIC)*), v_4 will be activated by v_2 with 0.1 probability, v_5 will be activated by either v_1 or v_2, while v_6 will first be activated by v_3 with 0.9 probability and then be activated by v_5 with 0.1 probability. So, the next seed is v_3 and the influence spread is $1(v_1) + 1(v_3) + 0.5(v_5) + 0.9(v_6$ by $v_3)+0.1 \times 0.5(v_6$ by v_1 through $v_5)$= 3.45, which is higher than 3.2 by counteringthe IC model v_2 in case (2).

Then, we evaluate the effect of different seed selections by a case study on `Orkut` [25]. We select the influential users in the network (top 1% of all the users) to form the initial seed set S, in which 1/5 users adopt our target company and the others adopt other companies. The influential users are iteratively selected according to the influence gain under the IC model. For each selection method, we continue to select more seeds for the target company from 1% to 10% of the vertices in S and report the influence of the target company by the MCIC model. Figure 2 shows that the influence by *Countering(IC)* is much higher than *Seeding*, because the vertices in seed set S are more influential than others. *Countering(MCIC)* further outperforms *Countering(IC)* because it considers the influence competition of multiple companies and the selected seeds are more influential in the competition.

3.3 Problem Definition

Assume C_t is the *target company* we are interested in. We aim to counter a set A of seeds that initially adopt other companies, to adopt C_t, s.t. the expected influence spread of C_t is maximized, i.e., activating the largest number of vertices to adopt C_t in expectation.

Definition 1 (Influence Countering Problem)**.**

- **Given:** a graph G, a budget b, a target company C_t, and a seed set S with a company assignment $c : S \rightarrow \{C_1, \cdots, C_K\}$;
- **Find:** a size-b countered set A chosen from the set of all seeds that are not with the target company, i.e., from $S_{\neg t} = \{s \mid s \in S \wedge c_s \neq C_t\}$ where c_s is the company that vertex s adopts;
- **to Maximize:** $\mathbb{E}\big[I(C_t, A)\big]$, the expected spread of C_t when propagating the diffusion with a countered company assignment, i.e., if $s \in A$, $c_A(s) = C_t$ and $c_A(s) = c_s$ if $s \in S \setminus A$.

3.4 Hardness of Problems

Lemma 1. *Let $f(s,t)$ be the number of subgraphs of G in which there is a path from s to t. Deciding if $f(s,t) \geq k$ is #P-complete.*

Proof. We prove this by a reduction from the problem of counting $f(s,t)$. Given an instance of the counting problem, we repeatedly binary search on the number of subgraphs $f(s,t)$, and decide if $f(s,t) \geq k$. The binary search ends in $O(|E(G)|)$ rounds, as the number of subgraphs is at most $2^{|E(G)|}$. Since counting $f(s,t)$ is #P-complete [44], deciding if $f(s,t) \geq k$ is also #P-complete. □

Theorem 1. *The problem of influence countering is #P-complete.*

Proof. We prove this by a reduction from Lemma 1. Consider an instance: given a graph G, a constant k, and two vertices s, t, we decide if $f(s,t) \geq k$. This problem is equivalent to computing $p(s,t)$, the probability that s is connected to t, when every edge in G has probability 1/2, because we have $f(s,t) = 2^{|E(G)|} \cdot p(s,t)$.

We build two graphs G', G'' to help decide if $p(s,t) \geq k/2^{|E(G)|}$. We first set the probability of every edge as 1/2 in G. Let I_0 be the spread of s in G, i.e., the expected number of vertices that s can reach in G. Then, we copy G into a new graph G', and insert into G' a vertex t' and an edge (t, t') with probability 1. Let I_1 be the spread of s in G', we have $I_1 = I_0 + p(s,t) \cdot p_{t,t'} = I_0 + p(s,t)$. Note that s activates t iff s is connected to t through sampled edges, i.e., $p(s,t)$ is equivalent to the probability of s activating t. Next, we copy G into another graph G'', and insert into G'' a vertex t'' and an edge (s, t'') with probability $k/2^{|E(G)|}$. Let I_2 be the spread of s in G'', then it is clear that $I_2 = I_0 + k/2^{|E(G)|}$.

We construct a corresponding problem of influence countering. Assume that C_1 (target) and C_2 are competing in $G' + G''$, the seed set contains $\{s \text{ in } G', s \text{ in } G''\}$, and both seeds adopt C_2 initially. Given this influence countering problem with $b = 1$, deciding the better seed is equivalent to deciding $I_1 \geq I_2$, i.e., deciding $p(s,t) \geq k/2^{|E(G)|}$. If there is a polynomial time solution for the influence countering problem, then we can decide if $p(s,t) \geq k/2^{|E(G)|}$ in polynomial time, and then derive whether $f(s,t) \geq k$. □

From Theorem 1, we know the problem of influence countering is at least as hard as NP-complete problems [44].

Theorem 2. *For any $A \subseteq S_{\neg t}$, computing $\mathbb{E}[I(C_t, A)]$ is #P-hard.*

Proof. We prove the theorem by a reduction from the influence spread computation under the independent cascade (IC) model [21], which is #P-hard [12]. The IC model is the single-company version of the MCIC model. Initially, we activate a set $S' \subseteq V$. If a vertex u is activated, then for each edge from u to an inactive vertex v, u has $p_{u,v}$ probability to activate v. The influence spread of S' asks the number of vertices that are activated when this process converges.

Given any instance of the above problem, we define a corresponding problem of computing $\mathbb{E}\big[I(C_t, A)\big]$: assume $S = S'$ and all seeds follow the target company, then, $\mathbb{E}\big[I(C_t, \varnothing)\big]$ equals the influence spread of S' under the IC model. □

3.5 Baseline 1: Influence Max Approach (*BIM*)

As discussed in Section 3.2, we can apply the method of influence maximization under the IC model [16] for seed selection, which iteratively selects a user who adopted a non-target company with the largest influence gain, i.e., the *Countering(IC)* algorithm. Then, we compute the influence spread under MCIC for the selected seeds. The time complexity of *BIM* is $O(|\mathcal{R}| \cdot \mathcal{E})$, where $|\mathcal{R}|$ is the number of RR sets in the sampling and $\mathcal{E}$ is the expected running time required to generate an RR set for a random graph sample.

However, *BIM* does not consider the competition of multiple campaigners in seed selection, as the method is designed for one campaigner. In seed selection, the influence spread from the target company is executed before the cascades of all other companies, which is not fair. So, we design a new baseline in the next subsection.

3.6 Baseline 2: Greedy Approach (*BGA*)

We propose another baseline that avoids the limitation of the first baseline. It is inspired by Kempe et al. [21], a greedy approach for influence maximization. In particular, we start from an empty set $A = \varnothing$, and then iteratively inserts into A a seed $u \in S_{\neg t}$ that results in the largest increase of $\mathbb{E}\big[I(C_t, A)\big]$, until $|A| = b$, i.e.,

$$u \quad = \quad \underset{s \in S_{\neg t}}{\arg\max} \left(\mathbb{E}\big[I(C_t, A \cup \{s\})\big] - \mathbb{E}\big[I(C_t, A)\big] \right).$$

The baseline is simple in concept but hard to implement, since the computation of $\mathbb{E}\big[I(C_t, A)\big]$ is #P-hard by Theorem 2. To address this issue, we estimate $\mathbb{E}\big[I(C_t, A)\big]$ by a Monte Carlo method, that is, repeatedly propagating the MCIC diffusion on the graph. Let $\sigma(C_t, A)$ be the number of vertices in G that are activated with C_t after a converged diffusion process of MCIC with the countered assignment $c_A(\cdot)$. We can prove that $\mathbb{E}\big[\sigma(C_t, A)\big] = \mathbb{E}\big[I(C_t, A)\big]$ by the law of big numbers. Therefore, we first simulate the MCIC diffusion for r times, retrieve $\sigma(C_t, A)$ on each simulation, and take the average of $\sigma(C_t, A)$ as an estimator of $\mathbb{E}\big[I(C_t, A)\big]$.

Although the baseline is concise, it suffers from its prohibitive time complexity $O(|S| \cdot brm)$. Specifically, it has b iterations, each requires to estimate the expected spread of $O(|S_{\neg t}|)$ vertex sets. Besides, each estimation simulates the MCIC diffusion for $O(r)$ times, and each diffusion runs in $O(m)$ time. Totally, the baseline needs $O(|S| \cdot brm)$ time as $|S_{\neg t}| \leq |S|$. We adopt the setting $r = 10000$ that is suggested by the previous works [21, 32].

4 BASIC APPROACH: *MIC*

This section presents *MIC*, a method that returns a high-quality solution with a small time cost. *MIC* relies on *graph samples*, where a sample is obtained by removing each edge (u, v) in G with $1 - p_{u,v}$ probability. At a high level, *MIC* consists of three phases as follows:

(1) **Sampling**: This phase iteratively generates r graph samples and puts them into $\mathcal{G}$. The parameter r is *pre-decided*.
(2) **Spread Estimation**: This phase estimates the spread of each seed $s \in S$ by the empirical average spread of s in $\mathcal{G}$.
(3) **Seed Selection**: This phase returns a size-b set A of seeds with the largest estimated spread.

In a graph sample, for influence maximization under the IC diffusion model, the vertices reachable from existing seeds are activated and they are irreverent to the spread of new seeds. However, for the influence countering problem, the vertices that can be reached from existing seeds may still contribute to the marginal gain of a newly countered seed. For instance, if a vertex x is already reached (i.e., activated) by seed s_1 and a newly countered seed s_2 also reaches x with the same distance, the expected probability of activating x by the target company will be increased, which contributes to the spread of s_2. An example is given in Section 3.2.

As analyzed above, the spread computation of influence countering becomes harder, while we prove that the marginal gain of countering a seed is a constant irreverent to the companies of the seeds (Theorem 3). This is because we have an unchanged seed set (only the companies of the seeds may change by the countering), and then the marginal gain of countering a seed is decided by its expected influence ratio of activating every vertex, which is irreverent to the companies of the seeds. Then, the selection of seeds will be easier because we can compute the true combined contributions given the influence gain of countering each seed, i.e., we can find the optimal set of seeds to counter given an accurate sampling, which goes beyond the traditional greedy method.

THEOREM 3. *Let $s \in S_{\neg t}$ be a fixed seed and $\sigma(s)$ be a constant associated with s. Then, for any countered set A that does not contain s, i.e., $A \subseteq S_{\neg t} \setminus \{s\}$, the following equation holds:*

$$\mathbb{E}\big[I(C_t, A \cup \{s\})\big] - \mathbb{E}\big[I(C_t, A)\big] \quad = \quad \sigma(s). \tag{1}$$

That is, whenever we insert s into a valid A, the spread of countering A (i.e., $\mathbb{E}\big[I(C_t, A)\big]$) is always increased by a constant $\sigma(s)$.

PROOF. Given a graph sample g, a seed set S, a seed $s \in S$, and a node $u \notin S$, we let $p_g(s, u)$ be the probability that s can activate u (with its company), when we start a diffusion from S on g. Then, the expected number of vertices that s can activate on g equals $\sigma_g(s) = \sum_{u \in V(g)} p_g(s, u)$. In a graph sample g, a vertex can be activated by one seed. Thus we have the expected number of vertices that S can activate equals $\sum_{s \in S} \sigma_g(s)$. Let $X(G)$ be the set of all possible graph samples from G and $P[g]$ be the probability of sampling g from $X(G)$. By definition of $\mathbb{E}\big[I(C_t, A)\big]$, we can rewrite it with the definitions above:

$$\mathbb{E}[I(C_t, A)] \quad = \sum_{g \in X(G)} P[g] \cdot \left(\sum_{s \in S} \sigma_g(s) \cdot [c_A(s) = C_t]\right),$$

where $[\cdot]$ equals 1 if and only if the condition inside is true, and equals 0 otherwise. That is, $[c_A(s) = C_t]$ equals 1 if and only if the company of s is C_t or s is countered (i.e., $s \in A$). Then, we can rewrite the l.h.s. of Equation 1 as the follows:

$$\begin{aligned}
&\mathbb{E}[I(C_t, A \cup \{s\})] - \mathbb{E}[I(C_t, A)] \\
= &\sum_{g \in X(G)} P[g] \cdot \left(\sum_{s' \in S} \sigma_g(s') \cdot \Big([c_{A \cup \{s\}}(s') = C_t] - [c_A(s') = C_t]\Big)\right) \\
= &\sum_{g \in X(G)} P[g] \cdot \left(\sum_{s' \in S} \sigma_g(s') \cdot [s' = s]\right) \\
= &\sum_{g \in X(G)} P[g] \cdot \sigma_g(s).
\end{aligned} \tag{2}$$

Given g, both $P[g]$ and $\sigma_g(s)$ are constant numbers by definitions, as $P[g]$ is a probability and $\sigma_g(s)$ is the sum of a series of probabilities. Therefore, the l.h.s of Equation 1 (or equivalently, Equation 2) gives a constant number. □

Algorithm 1: *MIC* (G, S, C_t, b)

```
1 $r \leftarrow$ decide the number of graph samples ;            // Sec 4.2
2 Generate $r$ graph samples $\mathcal{G} = \{g_1, \cdots g_r\}$;
3 $\hat{\sigma}(\cdot) \leftarrow$ SpreadEst($\mathcal{G}, r, S$);          // Sec 4.1
4 $S_{\neg t} \leftarrow \{s \mid s \in S \wedge c_s \neq C_t\}$;
5 sort $s \in S_{\neg t}$ in descending order of $\hat{\sigma}(s)$;
6 return $A \leftarrow$ the first $b$ vertices of $S_{\neg t}$;
```

By Theorem 3, $\sigma(s)$ is a constant and thus it is optimal to counter the seeds with the largest $\sigma(s)$. *MIC* selects b seeds with the largest $\sigma(s)$, which leads to the largest increase in $\mathbb{E}\big[I(C_t, A)\big]$. *MIC* estimates $\sigma(s)$ by sampling. Specifically, we first generate some graph samples, then estimate the spread on each sample, and finally take the average spread $\hat{\sigma}(s)$ as an estimation of $\sigma(s)$. By the law of big number, the estimation is unbiased, i.e., $\mathbb{E}\big[\hat{\sigma}(s)\big] = \sigma(s)$.

Algorithm 1 presents the pseudo-code of *MIC*. Lines 1-2 generate a *pre-decided* number (i.e., r) of graph samples. The parameter r must be larger than a certain threshold to guarantee the correctness of *MIC* (Section 4.2). Then, Line 3 estimates the spread $\hat{\sigma}(s)$ for every seed $s \in S$ by the mean spread in $\mathcal{G}$ (Section 4.1). Finally, Lines 4-6 sort any seed $s \in S_{\neg t}$ in descending order of $\hat{\sigma}(s)$ and return the top-b seeds as the final result.

In what follows, we first detail the spread estimation phase, and then analyze the selection of r in the sampling phase. Unless otherwise specified, all logarithms in this paper are to the base e.

4.1 Spread Estimation Phase

This section estimates the spread $\sigma(s)$ of a seed s in Theorem 3 by the empirical average spread of s in r graph samples $\mathcal{G}$. We first discuss how to compute the spread on a graph sample g.

An MCIC diffusion on g is equivalent to a BFS. Specifically, we start a BFS from S, if u is discovered (i.e., activated) for the first time

Algorithm 2: SpreadEst($\mathcal{G}, r, S$)

```
1 for each $g_i \in \mathcal{G}$ do
2     $dag \leftarrow$ the shortest path DAG starting from $S$ on $g_i$;
3     for each $u$ in the reverse topological sorting of $V(dag)$ do
4         $\hat{\sigma}_i(u) \leftarrow 1 + \sum_{v \in N_u^+(dag)} \frac{\hat{\sigma}_i(v)}{|N_v^-(dag)|}$;
5 for any $s \in S$ do $\hat{\sigma}(s) \leftarrow \frac{1}{r}\sum_{i=1}^{r} \hat{\sigma}_i(s)$;
6 return $\hat{\sigma}(\cdot)$;
```

by vertices $w_1, \cdots, w_x$, then u adopts a company from $c_{w_1}, \cdots, c_{w_x}$ uniformly. Each time u activates v in diffusion, it corresponds to the final-hop edge (u, v) on the shortest path from S to v. The sp-dag captures those edges, and we compute the spread on this structure.

Definition 2 (Shortest Path DAG)**.** Given a graph sample g and a vertex set S, the shortest path DAG (aka. sp-dag) rooted at S is a subgraph of g that includes any edge lies on at least one shortest path from sourcing from S.

Algorithm 2 presents the spread estimation of *MIC*. For each $g_i \in \mathcal{G}$, we build the sp-dag of g_i (Line 2), then compute the spread of any vertex u (Line 3-4). Finally, we return the empirical average spread $\hat{\sigma}(\cdot)$ in graph samples (Lines 5-6). Line 2 builds the sp-dag. Let $\delta_g(S, v)$ be the shortest distance from S to a vertex v in g. We first compute $\delta_g(S, v)$ for any vertex $v \in V(g)$ by a BFS, then we put an edge (u, v) into dag if $\delta_g(S, u) + 1 = \delta_g(S, v)$, and return dag as the sp-dag. Lines 3-4 compute the spread on dag, where $N_u^+(dag)$ and $N_u^-(dag)$ are the out-/in- neighbors of u in dag. Recall that each edge (u, v) in the sp-dag corresponds to u activates v. If a vertex u is activated, for any $(u, v) \in dag$, u activates v at the next timestamp and u competes with any vertex w that points to v in dag. Thus, u influences itself and any $v \in N_u^+(dag)$ with probability $1/|N_v^-(dag)|$ (Line 4). The reverse topological sorting (Line 3) guarantees that the $\hat{\sigma}_i(\cdot)$ computation of any out-neighbor of u finishes before u.

Using the law of large numbers, we show that the empirical mean spread $\hat{\sigma}(s)$ in r graph samples accurately approximates $\sigma(s)$.

Lemma 2. *Let $\mathcal{G} = \{g_1, \cdots, g_r\}$ be r graph samples obtained from G, the mean spread of s in r samples, as r approaches ∞, approaches the spread of s in G, i.e., $\mathbb{E}\big[\hat{\sigma}(s)\big] = \lim_{r\to\infty} \frac{1}{r}\sum_{i=1}^{r} \hat{\sigma}_i(s) = \sigma(s)$.*

Algorithm 2 runs in simply $O(rm)$ time, as computing sp-dag and reverse topological sorting both need $O(|E(g)|)$ time for a graph sample g. Together with Lemma 2, we derive the following theorem.

Theorem 4. *Algorithm 2 runs in $O(rm)$ time, and returns $\hat{\sigma}(s)$ that accurately estimates the spread $\sigma(s)$ in G.*

Example 1. Figure 4 shows a graph sample g of Figure 3. The edge $(v_2, v_3) \notin g$ with probability $1 - 0.1$, (v_1, v_4) and (v_2, v_4) exist in g with probability 0.5 and 0.3 respectively, while the rest two edges must be in g. Among all possible graph samples, the probability of sampling g is $0.135 = 0.9 \times 0.5 \times 0.3$.

The thick black edges in Figure 4 form the sp-dag of g, where the edge (v_3, v_4) is not included. This can be verified by $\delta_g(S, v_3) + 1 \neq \delta_g(S, v_4)$ as we have $\delta_g(S, v_3) = \delta_g(S, v_4) = 1$. Consider a MCIC diffusion on g. At timestamp 1, we activate v_1 with C_1 and v_2 with C_1. At timestamp 2, both v_1 and v_2 activate v_4, and v_1 also activates v_3. The sp-dag in Figure 4 accurately describes the activation process in the diffusion.

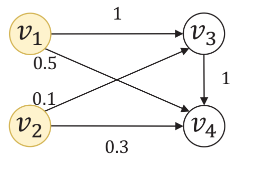


Figure 3: Graph sample g

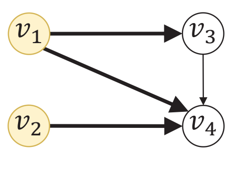


Figure 4: Shortest path DAG

4.2 Select r in Sampling Phase

This section analyzes how to select a parameter r such that *MIC* returns a high-quality solution with high probability. The subsequent analysis frequently uses the Chernoff bounds [34].

Lemma 3. *Let X be the sum of c i.i.d. random variables sampled from a distribution on $[0, 1]$ with a mean μ. For any $\delta > 0$, we have*

$$Pr\Big[X - c\mu \geq \delta \cdot c\mu\Big] \leq \exp\Big(-\frac{\delta^2}{2+\delta}c\mu\Big).$$

Let A^* be the optimal solution to our problem, i.e., the countered set that leads to the largest $\mathbb{E}\big[I(C_t, A^*)\big]$, and $OPT = \mathbb{E}\big[I(C_t, A^*)\big] - \mathbb{E}\big[I(C_t)\big]$ be the maximum increase in the influence. We define $\sigma(A)$ as $\sum_{s\in A} \sigma(s)$. By the Chernoff bounds, we confirm that $\hat{\sigma}(A)$ is an accurate estimator of $\sigma(A)$ when r is sufficiently large:

Lemma 4. *Assume that r satisfies*

$$r \geq 2n \cdot (\varepsilon + 4) \cdot \frac{l \log n}{OPT \cdot \varepsilon^2}. \tag{3}$$

Then, for any countered set A of at most b seeds, the following inequality holds with at least $1 - n^{-l}$ probability:

$$|\hat{\sigma}(A) - \sigma(A)| < \frac{\varepsilon}{2} \cdot OPT.$$

Proof. We regard $r \cdot \hat{\sigma}(A)/n$ regard as the sum of r i.i.d. Bernoulli variables with a mean $\mu = \sigma(A)/n$. By Lemma 2, $\mu = \mathbb{E}\big[\hat{\sigma}(A)\big]/n = \sigma(A)/n$. Then, we have

$$\begin{aligned} &Pr\Big[|\hat{\sigma}(A) - \sigma(A)| \geq \frac{\varepsilon}{2} \cdot OPT\Big] \\ &= Pr\Big[|r \cdot \hat{\sigma}(A)/n - r\mu| \geq \frac{\varepsilon r}{2n} \cdot OPT\Big] \\ &= Pr\Big[|r \cdot \hat{\sigma}(A)/n - r\mu| \geq \frac{\varepsilon \cdot OPT}{2n\mu} \cdot r\mu\Big]. \end{aligned} \tag{4}$$

Let $\delta = \varepsilon \cdot OPT/(2n\mu)$. By the Chernoff bounds, Equation 3, and the fact that $\mu = \sigma(A)/n \leq OPT/n$, we have

$$\begin{aligned} \text{r.h.s. of Eqn. 4} &\leq \exp\Big(-\frac{\delta^2}{2+\delta} \cdot r\mu\Big) \\ &= \exp\Big(-\frac{\varepsilon^2 \cdot OPT^2}{2n(4n\mu + \varepsilon OPT)} \cdot r\Big) \\ &\leq \exp\Big(-\frac{\varepsilon^2 \cdot OPT}{2n(\varepsilon + 4)} \cdot r\Big) \leq \frac{1}{n^l}. \end{aligned}$$

□

Algorithm 3: *MIC*$^+$ (G, S, C_t, b)

```
1 $\theta \leftarrow$ ThetaEst$(G, S, C_t, b)$;                    // Sec 5.2
2 Generate $\theta$ graph samples $\mathcal{G} = \{g_1, \cdots g_\theta\}$;
3 $\hat{\sigma}(\cdot) \leftarrow$ SpreadEst$^+(\mathcal{G}, \theta, S)$ ;    // Sec 5.1
4 $S_{\neg t} \leftarrow \{s \mid s \in S \wedge c_s \neq C_t\}$;
5 sort $S_{\neg t}$ in descending order of $\hat{\sigma}(s)$;
6 return $A \leftarrow$ the first $b$ vertices of $S_{\neg t}$;
```

Let A be the result of Algorithm 1, then $\hat{\sigma}(A) \geq \hat{\sigma}(A^*)$ as *MIC* returns the seeds with the top-b $\hat{\sigma}(s)$. By this fact and Lemma 4,

$$\sigma(A) \;\geq\; \hat{\sigma}(A) - \frac{\varepsilon}{2} \cdot OPT \;\geq\; \hat{\sigma}(A^*) - \frac{\varepsilon}{2} \cdot OPT \;\geq\; (1-\varepsilon) \cdot OPT.$$

Therefore, Algorithm 1 returns a $(1-\varepsilon)$-approximate solution with high probability when Equation 3 holds. By taking the bound $OPT \geq b$ into Equation 3, we set $r \geq 2n \cdot (\varepsilon + 4) \cdot (l \log n)/(b\varepsilon^2)$, then, *MIC* runs in time

$$O(rm) = O\left((\varepsilon + 4) \cdot \frac{2lnm \log n}{b \cdot \varepsilon^2}\right) = O\left(\frac{lnm \log n}{b\varepsilon^2}\right).$$

Thus, we have the following theorem:

Theorem 5. *Given a r that satisfies $r \geq 2n \cdot (\varepsilon+4) \cdot (l \log n)/(b\varepsilon^2)$, MIC runs in $O((lnm \log n)/(b\varepsilon^2))$ time, and returns a $(1-\varepsilon)$-approximate solution with at least $1 - n^{-l}$ probability.*

4.3 Put It Together

Given G, S, c, C_t, b, and two parameters ε and l, *MIC* first decides the number of graph samples by Equation 3 and puts it into r. After that, *MIC* generates r graph samples and estimates the spread $\hat{\sigma}(s)$ for any $s \in S_{\neg t}$ by Algorithm 2. Finally, *MIC* sorts $S_{\neg t}$ by decreasing order of $\hat{\sigma}(s)$ and returns the first b seeds as the final result.

By Theorem 3 and 5, *MIC* eliminates two loops in the baseline and reduces the time from $O(brm \cdot |S|)$ to $O(rm)$. Lemma 4 establishes the minimum r required for the approximate guarantee of *MIC*. By Theorem 5, *MIC* runs in $O((lnm \log n)/(b\varepsilon^2))$ time, and returns a $(1-\varepsilon)$-approximate solution with at least $1 - n^{-l}$ probability.

5 IMPROVED APPROACH: *MIC*$^+$

This section proposes *MIC*$^+$, a method that improves *MIC* by reducing the cost of spread estimation and the required number of graph samples. *MIC*$^+$ consists of three phases:

1. **Sampling**: This phase generates θ graph samples and put them into $\mathcal{G}$. The parameter θ is far less than r (of *MIC*) while offering the same guarantee.
2. **Spread Estimation**: This phase estimates the spread of a seed s by the probability that s activates a random vertex in $\mathcal{G}$. The cost is largely reduced compared with *MIC*.
3. **Seed Selection**: This phase is the same as *MIC*.

Algorithm 3 presents the pseudo-code of *MIC*$^+$. *MIC*$^+$ aims to select (and counter) b seeds with the largest spread $\sigma(s)$, like *MIC*. By Theorem 3, this leads to the largest increase in the resulting influence $\mathbb{E}\big[I(C_t, A)\big]$.

In the following, we detail the differences to *MIC*. We first detail the spread estimation phase of *MIC*$^+$, and then analyze the selection of θ in the sampling phase.

Algorithm 4: SpreadEst$^+(\mathcal{G}, \theta, S)$

```
1  for each $g_i \in \mathcal{G}$ do
2      $x_i \leftarrow$ select a vertex from $V$ uniform at random;
3      $g_i^r \leftarrow$ the reverse of $g_i$;
4      $rdag \leftarrow$ reverse shortest path DAG rooted at $x$ in $g_i^r$;
5      $\hat{p}_i(x_i) \leftarrow 1$;
6      for each $u$ in the topological sorting of $V(rdag)$ do
7          $\hat{p}_i(u) \leftarrow \sum_{N_u^-(rdag)} \frac{\hat{p}_i(v)}{|N_v^+(rdag)|}$;
8  for any $s \in S$ do $\hat{p}(s) \leftarrow \frac{1}{\theta}\sum_{i=1}^{\theta} \hat{p}_i(s)$;
9  for any $s \in S$ do $\hat{\sigma}(s) \leftarrow n \cdot \hat{p}(s)$;
10 return $\hat{\sigma}(\cdot)$;
```

5.1 Improved Spread Estimation

Let *spread probability* $p(s)$ be the probability that a seed s activates a random vertex in G. Recall that $\sigma(s)$ is the spread of s, i.e., the expected number of vertices that s can activate. Then, based on Theorem 3, we can prove that $n \cdot p(s)$ equals $\sigma(s)$.

Lemma 5. $n \cdot p(s) = \sigma(s)$.

Proof. Following the definitions in the proof of Theorem 3,

$$\sigma(s) = \sum_{g \in X(G)} P[g] \cdot \big(\sigma_g(C_t, A \cup \{s\}) - \sigma_g(C_t, A)\big).$$

Observe that the r.h.s equals the expected number of vertices that are activated by s in the diffusion. By the definition of $p(s)$, it follows that $n \cdot p(s) = \sigma(s)$. □

Recall in Section 4.1, we define sp-dag as the activation of a vertex x in a diffusion corresponds to a shortest path from S to x in a BFS. The activation also corresponds to a shortest path from x to S in the *reverse* graph, i.e., edge direction is flipped. We refer to such shortest paths in the reverse graph as a *reverse shortest path*. Then, we can define a counterpart of sp-dag on the reverse of a graph sample (aka. *reverse graph sample*).

Definition 3 (Reverse Shortest Path DAG)**.** Given vertex x and vertex set S, let g^r be a reverse graph sample. The reverse shortest path DAG (aka. rsp-dag) rooted at x is a subgraph of g^r that contains edges lying on a reverse shortest path from x to $S' \subseteq S$, with the distance from x to any $s' \in S'$ being the lowest from x to S.

For any vertex $s' \in S'$, the distance from x to s' is the lowest in g^r, so the distance from s' to x is also the lowest in the original g. On the other hand, for any vertex $s \in S \setminus S'$, the distance from s to x is not the shortest. Therefore, any seed $s' \in S'$ will activate x at the same timestamp, while other seeds will not, i.e., only the vertices in rsp-dag activate x. By this intuition, we compute $\hat{p}(u)$ the probability that u activates x over the rsp-dag. We can accurately estimate $p(s)$ by the average of $\hat{p}(s)$ in multiple graph samples.

Algorithm 4 presents the spread estimation of *MIC*$^+$. For each $g_i \in \mathcal{G}$, we first select a random vertex x_i (Line 2), then build the rsp-dag rooted at x_i (Lines 3-4), and next compute the spread probability in g_i (Lines 5-7). After that, we obtain the average spread probability $\hat{p}(\cdot)$ (Line 8). By Lemma 5, we multiply it by n and return it as the empirical average spread $\hat{\sigma}(\cdot)$ (Lines 9-10).

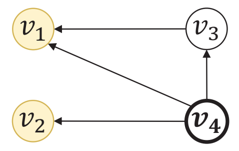


Figure 5: Reverse graph sample g' (rooted at v_4)

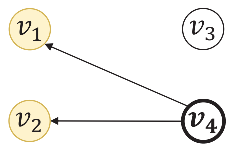


Figure 6: Reverse shortest path DAG (rooted at v_4)

Lines 3-4 build the rsp-dag using a BFS with stop. Let $\delta_g^r(x,s)$ be the distance from x to s in g^r. We start a BFS from x in g^r. Once reaching a seed s in S, we continue to search on any other seed s' on the current BFS level, i.e., any seed $s' \in S$ with $\delta_g^r(x,s) = \delta_g^r(x,s')$. We put s and all s' into a set S' and then terminate the BFS. Finally, we insert into *rdag* any reverse edge lying on a reverse shortest path from x to a seed $s' \in S'$, and return *rdag* as the rsp-dag.

Lines 5-7 compute the probability of activating x. Recall that each edge (v,u) in the rsp-dag corresponds to u activates v. Eventually x is activated and the probability is 1 (Line 5). Before that, for any $(v,u) \in rdag$, u activates v at some timestamp, and u competes with any vertex w satisfying $(v,w) \in rdag$. Thus, u has $1/|N_v^+(rdag)|$ probability to influence any $v \in N_u^-(rdag)$ (Line 4). The topological sorting (Line 3) guarantees that the $\hat{p}_i(u)$ computation of any in-neighbor of u finishes before u.

By the law of large numbers, we show that $\hat{p}(s)$ in r graph samples accurately approximates $p(s)$.

Lemma 6. *Let $\mathcal{G} = \{g_1, \cdots g_\theta\}$ be θ graph samples obtained from G, the mean probability of s activates a random vertex in θ graph samples, as r approaches ∞, approaches $p(s)$ the probability that s activates a random vertex in G, i.e.,*

$$\mathbb{E}\left[\hat{p}(s)\right] = \lim_{\theta\to\infty} \frac{1}{\theta}\sum_{i=1}^{\theta} \hat{\sigma}_i(s) = p(s).$$

By Lemma 6 and the fact that $\mathbb{E}\left[n \cdot \hat{p}(s)\right] = n \cdot p(s) = \sigma(s)$, we prove the following theorem:

Theorem 6. *Algorithm 4 runs in $O(rm)$ time, and returns $\hat{\sigma}(s)$ that accurately estimates the spread $\sigma(s)$ in G.*

Example 2. Figure 5 presents a reverse graph sample g^r that is the reverse of g in Figure 4. Figure 6 shows the rsp-dag rooted at v_4 in g^r. Clearly, $S' = \{v_1, v_2\}$ and (v_4,v_1), (v_4,v_2) form the reverse shortest path from v_4 to S'. The edges (v_3,v_1), (v_4,v_3) are excluded. The sp-dag of Figure 4 contains 3 edges while rsp-dag here contains 2. The size of rsp-dag is smaller. We compute the probability of activating v_4 on rsp-dag as follows. Initially, $\hat{p}_g(v_4) = 1$. As v_4 has two out-neighbors, $\hat{p}_g(v_1) = \hat{p}_g(v_2) = 0.5$ and $\hat{p}_g(v_3) = 0$. This matches the diffusion on g (Figure 4), where v_1, v_2 activate v_4 at timestamp 2 and v_3 never activates v_4.

5.2 Select θ in Sampling Phase

By Theorem 6 and Lemma 4, *MIC*$^+$ accurately estimates $\sigma(A)$ when θ is sufficiently large. Thus, the theoretical guarantees of *MIC* (Theorem 6) also holds for *MIC*$^+$, i.e., given $\theta \geq 2n(\varepsilon+4)\cdot(l\log n)/(b\varepsilon^2)$,

Algorithm 5: ThetaEst(G, S, C_t, b)

```
1 $r \leftarrow (\varepsilon+2)n \cdot \frac{l\log n}{|S_{\neg t}|\cdot\varepsilon^2}$;
2 Generate $r$ graph samples $\mathcal{G} = \{g_1, \cdots g_r\}$;
3 $\hat{\sigma}(\cdot) \leftarrow$ SpreadEst$^+(\mathcal{G}, r, S)$ for any $g_i \in \mathcal{G}$;
4 $\widehat{BPT} \leftarrow \frac{b}{|S_{\neg t}|}\cdot\sum_{s\in S_{\neg t}}\hat{\sigma}(s)$;
5 return $\theta \leftarrow 2n\cdot(4+\varepsilon)\cdot\frac{(1+\varepsilon)\cdot l\log n}{\widehat{BPT}\cdot\varepsilon^2}$;
```

MIC$^+$ runs in $O((lnm\log n)/(b\varepsilon^2))$ time, and returns a $(1-\varepsilon)$-approximate solution with at least $1-n^{-l}$ probability.

Recall that A^* is the optimal countered set of our problem and *OPT* equals $\sigma(A^*)$. The guarantees above are obtained by a lower bound $OPT \geq b$. But $b \ll OPT$ in practice. To tighten the bound, we devise a new lower bound *BPT* for *OPT*:

Definition 4. $BPT = \frac{b}{|S_{\neg t}|}\sigma(S_{\neg t})$ and $\widehat{BPT} = \frac{b}{|S_{\neg t}|}\hat{\sigma}(S_{\neg t})$.

BPT is the expected spread of a size-b countered set. We have $BPT \leq OPT$, as *OPT* is the size-b countered set with the largest spread. By the definition of $\widehat{BPT}$, *MIC*$^+$ can accurately estimate $\sigma(S_{\neg t})$ and $\widehat{BPT}$. This motivates the θ selection algorithm below.

Algorithm 5 presents the θ selection of *MIC*$^+$ in the sampling phase. We first estimate the spread $\sigma(S_{\neg t})$ by Algorithm 4 (Lines 1-3) then estimate $\widehat{BPT}$ (Line 4) and finally return a parameter θ. In what follows, we step by step show that the θ returned by Algorithm 5 is sufficiently good.

We derive the following lemma by taking $2\cdot\varepsilon$ into ε in Lemma 4:

Lemma 7. *Assume that r satisfies $r \geq (\varepsilon+2)\cdot n\cdot\frac{l\log n}{OPT\cdot\varepsilon^2}$. Then, for any countered set A of at most b seeds, the inequality holds with at least $1-n^{-l}$ probability: $|\hat{\sigma}(A)-\sigma(A)| < \varepsilon\cdot OPT$.*

By Lemma 7 and Definiton 4, we prove that $\widehat{BPT}$ accurately estimates *BPT* when r is sufficiently large:

Lemma 8. *Assume r satisfies $r \geq (\varepsilon+2)\cdot n\cdot\frac{l\log n}{\sigma(S_{\neg t})\cdot\varepsilon^2}$, the inequality holds with at least $1-n^{-l}$ probability: $|\widehat{BPT}-BPT| < \varepsilon\cdot BPT$.*

Proof. Suppose we set $b = |S_{\neg t}|$ in our problem, then it is optimal to counter all seeds, and we have *OPT* equals $\sigma(S_{\neg t})$ and $A = S_{\neg t}$. By this fact and Lemma 7, if $r \geq (\varepsilon+2)\cdot n\cdot\frac{l\log n}{\sigma(S_{\neg t})\cdot\varepsilon^2}$, then the inequality $|\hat{\sigma}(S_{\neg t})-\sigma(S_{\neg t})| < \varepsilon\cdot\sigma(S_{\neg t})$ holds with at least $1-n^{-l}$ probability. By Definition 4,

$$\left|\widehat{BPT}-BPT\right| \leq \varepsilon\cdot BPT.$$
$$\Leftrightarrow \left|\frac{|S_{\neg t}|}{b}\widehat{BPT} - \frac{|S_{\neg t}|}{b}BPT\right| \leq \varepsilon\cdot\frac{|S_{\neg t}|}{b}BPT$$
$$\Leftrightarrow |\hat{\sigma}(S_{\neg t})-\sigma(S_{\neg t})| \leq \varepsilon\cdot\sigma(S_{\neg t})$$

then we can derive the lemma by the equivalence above. □

By taking the bound $|S_{\neg t}| \leq \sigma(S_{\neg t})$ into Lemma 7, we set $r \geq (\varepsilon+2)n\cdot\frac{l\log n}{|S_{\neg t}|\cdot\varepsilon^2}$ (Line 1), and then the following inequality holds with $1-n^{-l}$ probability (Lines 2-4):

$$(1-\varepsilon)\cdot BPT \leq \widehat{BPT} \leq (1+\varepsilon)\cdot BPT \leq (1+\varepsilon)\cdot OPT \quad (5)$$

After that, by taking the bound $\widehat{BPT}/(1+\varepsilon) \leq OPT$ into Lemma 4, we return $\theta = 2n \cdot (4+\varepsilon) \cdot \frac{(1+\varepsilon)\cdot l \log n}{\widehat{BPT}\cdot\varepsilon^2}$ (Line 5), and *MIC*$^+$ return a $(1-\varepsilon)$-approximate solution with at least $1 - 2 \cdot n^{-l}$ probability. We can increase the probability to $1 - n^{-l}$ by setting $l = 1 + \log_n 2$.

MIC$^+$ reuses the graph samples and it requires $\max\{r, \theta\}$ graph samples. By $(1-\varepsilon) \cdot b \leq (1-\varepsilon) \cdot BPT \leq \widehat{BPT}$ in Equation 5 and the number r, θ selected above, the runtime of *MIC*$^+$ is

$$O(\max\{r,\theta\} \cdot m) = O\left((\varepsilon+4) \cdot \frac{2lnm \log n}{b \cdot (1-\varepsilon) \cdot \varepsilon^2}\right) = O\left(\frac{lnm \log n}{b\varepsilon^2}\right).$$

By the above, the following theorem holds:

Theorem 7. *MIC*$^+$ *runs in* $O((lnm \log n)/(b\varepsilon^2))$ *time, and returns a* $(1-\varepsilon)$*-approximate solution with at least* $1 - n^{-l}$ *probability.*

5.3 Put It Together

Given G, S, c, C_t, b, and two parameters ε and l, *MIC*$^+$ first decides the number of graph samples by Algorithm 5 and put it into θ. After that, *MIC* generates θ graph samples and uses Algorithm 4 to estimate the spread $\hat{\sigma}(s)$ for any $s \in S_{\neg t}$. Finally, *MIC* sorts $S_{\neg t}$ by the decreasing order of $\hat{\sigma}(s)$ and return the first b seeds as result. By Theorem 7, *MIC*$^+$ runs in $O((lnm \log n)/(b\varepsilon^2))$ time, and returns a $(1-\varepsilon)$-approximate solution with at least $1 - n^{-l}$ probability.

5.4 Theoretical Comparisons of *MIC*$^+$

Comparison with *MIC*. By Theorems 5 and 7, *MIC* and *MIC*$^+$ return the solutions with the same approximate guarantee in the same time complexity. But *MIC*$^+$ outperforms *MIC* by orders of magnitude in practice, due to the optimization in spread estimation and sampling phases.

For the spread estimation phase, the size of rsp-dag in *MIC*$^+$ is far smaller than the size of sp-dag in *MIC*. Consider rsp-dag in the original graph, it contains the shortest paths from S' (defined in rsp-dag) to x. Recall that sp-dag in *MIC* contains all shortest paths from S to the rest of the graph, and it follows that rsp-dag has a size far smaller than sp-dag.

For the sampling phase, the parameter θ of *MIC*$^+$ is much lower than the parameter r of *MIC* in practice. *MIC*$^+$ uses $\widehat{BPT}$ as a lower bound of OPT, while *MIC* uses b. By Definition 4, $\widehat{BPT}$ is an estimation of expected spread of b seeds, and it follows that $\widehat{BPT} \gg b$ for most graphs. This leads to $\theta \ll r$ by Lemma 4.

Comparison with baseline 1 (*BIM*). The efficiency of baseline *BIM* depends on the number of RR sets generated. The time complexity of the state-of-the-art algorithm to generate an RR set under the IC model is $O(n)$ [16], while *MIC*$^+$ needs $O(m)$. Because the baseline *BIM* does not take into account the competition among multiple campaigners in seed selection, it cannot provide a theoretical guarantee for estimating the influence spread under the MCIC model. Therefore, it lacks any theoretical guarantee for solving the influence countering problem.

Comparison with baseline 2 (*BGA*). Recall that baseline 2 runs in $O(|S| \cdot brm)$ time, which is much larger than the cost of *MIC*$^+$. It trivially set $r = 10000$ and this cannot achieve any approximate guarantees. To fairly compare the two algorithms, we devise the required number of simulations when the baseline has the same theoretical guarantee as *MIC*$^+$. The baseline needs a tighter error bound, as it uses b iterations to select the counter set and each iteration selects one seed to counter. Assume that *MIC*$^+$ has an error ε then the baseline requires an error ε/b to achieve the same approximate guarantee. We derive the following lemma by taking ε/b into ε in Lemma 4:

Lemma 9. *Assume that* r *satisfies* $r \geq (8b^2 + 2b\varepsilon) \cdot n \cdot \frac{l \log n + \log b}{OPT \cdot \varepsilon^2}$. *Then, for any countered set* A *of at most* b *seeds, the inequality holds with at least* $1 - n^{-l}$ *probability:* $|\hat{\sigma}(A) - \sigma(A)| < \frac{\varepsilon}{2b} \cdot OPT$.

By Lemma 9, the baseline returns a $(1-\varepsilon)$-approximate solution with at least $1 - n^{-l}$ probability when $r \geq (8b^2 + 2b\varepsilon) \cdot n \cdot \frac{l \log n + \log b}{b \cdot \varepsilon^2}$. Therefore, the baseline requires far more simulations than the number of graphs required by *MIC*$^+$, when they achieve the same approximate guarantee.

6 EXTENSIONS OF *MIC*$^+$

In this section, we devise an index for *MIC*$^+$ that supports seven types of updates in dynamic graphs, and extend *MIC*$^+$ to the triggering model (a generalization of the MCIC model).

6.1 Updating *MIC*$^+$ on Dynamic Graphs

The index for *MIC*$^+$ stores θ tuples $\mathcal{I} = \{T_1, \cdots, T_\theta\}$, in which $T_i = \{g_i, x_i, R_i, \hat{\sigma}_i\}$ contains a graph sample g_i, a vertex x_i uniformly sampled from $V(G)$, a rsp-dag R_i rooted at x_i in g_i, and the estimated spread $\hat{\sigma}_i(\cdot)$ in g_i. To initialize the index, we first determine the size θ by Algorithm 4 and then generate θ tuples and put them into the index. We implement it by executing Lines 1-3 of Algorithm 3 and it requires $O((lnm \log n)/(b\varepsilon^2))$ time.

The index can support the graph updates on edges, vertices, seeds, and propagation probability. After every graph update, we recompute the index size θ, create tuples if θ increases or stash if θ decreases, and update $\hat{\sigma}_i$ if R_i is changed.

U1. Edge Insertion (+*e*). Let $e = (u,v) \notin G$ be a new edge associated with probability p. For each tuple $T_i = \{g_i, x_i, R_i, \hat{\sigma}_i\}$, we insert the edge e into g_i with probability p, and update R_i if e is successfully inserted into g_i. The time complexity is $O(\theta \cdot m \cdot p_{u,v})$, as the insertion scans $O(\theta \cdot p_{u,v})$ indexes in expectation. Next, we detail the update of R_i (assume e is not yet inserted).

(i) $\underline{v \notin V(R_i)}$. We update R_i only if R_i is empty. Otherwise, the reverse path from x_i to v is not the shortest, thus $e^r = (v,u)$ never forms a reverse shortest path from x_i to v then to u.

(ii) $\underline{u \notin V(R_i) \text{ and } v \in V(R_i)}$. Let $old\_d = \delta^r_{g_i}(x_i, S)$ be the reverse distance from x_i to S, and $d = \delta^r_{g_i}(x_i, v) + 1 + \delta^r_{g_i}(u, S)$ be the reverse distance from x_i to v, then to u, and finally reaching S. (a) if $d < old\_d$, then we must rebuild R_i because d reduces the distance. (b) if $d = old\_d$, then we insert into R_i all reverse shortest paths from v to u then to S. (c) if $d > old\_d$ then R_i not changes.

(iii) $\underline{u, v \in V(R_i)}$. Let $d_u = \delta^r_{g_i}(x_i, u)$ and $d_v = \delta^r_{g_i}(x_i, v)$ be the reverse shortest distance from x_i to u and v respectively. (a) if $d_u \leq d_v$, then R_i is not changed. (b) if $du = dv + 1$, then we insert into R_i all reverse shortest path from v to u then to S. (c) if $d_u > d_v + 1$, then we rebuild R_i.

U2. Edge Removal (−*e*). Let $e = (u,v) \in G$ denote an edge to remove. For each tuple $T_i = \{g_i, x_i, R_i, \hat{\sigma}_i\}$, we update R_i if it contains

(u,v). Assume $d^-_u(R_i)$ is the number of vertices that points to u in the reverse shortest path DAG R_i.

(i) $d^-_u(R_i) > 1$. Not all reverse shortest paths from x_i to u contains (v,u), and the removal of (v,u) cannot change the distance. Therefore, we remove (v,u) from R_i.

(ii) $d^-_u(R_i) = 1$. Any reverse shortest path from x_i to u contains (v,u). If $d = \delta^r_{g_i}(x_i, S)$ is not increased after removing the edge, then we remove from R_i all reverse shortest paths from v to u, then finally reaching S. Otherwise, we have to rebuild R_i, as $\delta^r_{g_i}(x_i, S)$ changes.

U2 takes $O(\theta \cdot m)$ time which is the same as *MIC*$^+$, but the running time of U2 is much less than *MIC*$^+$ in practice.

U3. Edge Probability Updates (Δp). To update the associated probability of an edge from p_1 to p_2 in each tuple, we first remove the edge by U2 and then insert it back with probability p_2 by U1.

U4. Vertex Insertion (+v). We insert a new vertex v. For each tuple $T_i = \{g_i, x_i, R_i, \hat{\sigma}_i\}$, we assign v to x_i with probability $\frac{1}{|V(G)+1|}$. If x_i is successfully assigned to v, then we rebuild R_i accordingly. The running time of vertex insertion is $O(\theta \cdot m/n)$.

U5. Vertex Removal ($-v$). Let v be a vertex to remove. We first utilize U2 to remove all edges incident to v. Then, for each tuple $T_i = \{g_i, x_i, R_i, \hat{\sigma}_i\}$, if $x_i = v$ then we re-sample x_i and rebuild R_i for x_i. The running time is also $O(\theta \cdot m/n)$.

U6. Seed Additions (+s). Let s be a new seed (not yet added into S), and $d = \delta^r_{g_i}(x_i, s)$ be the reverse shortest distance from x_i to s. For each tuple $T_i = \{g_i, x_i, R_i, \hat{\sigma}_i\}$, if $d < \delta^r_{g_i}(x_i, S)$ then we rebuild R_i, if $d = \delta^r_{g_i}(x_i, S)$ then we insert into R_i all reverse shortest paths from x_i to s, otherwise, R_i is not changed.

U7. Seed Deletions ($-s$). Let s be a seed to remove. For each tuple $T_i = \{g_i, x_i, R_i, \hat{\sigma}_i\}$, if s is the only seed in R_i, then we must rebuild R_i. Otherwise, R_i contains seeds other than s, and we can safely remove all shortest paths from x_i to s. U6 and U7 both run in $O(\theta \cdot m)$ time, but the running time of them is far less than *MIC*$^+$ in practice.

6.2 Extension to the Triggering Model

The triggering model is an influence propagation model that generalizes the independent cascade (IC) model [21], and it can be extended to a multi-campaigner setting, e.g., the K-LT model [18, 32]. The triggering model assumes that each vertex v is associated with a *triggering distribution* $\mathcal{T}(v)$ of a subset of v's in-neighbors. Each sample from $\mathcal{T}(v)$ is referred to as the *triggering set* of v. For a seed set S, the influence propagation from S under the multi-campaigner triggering model works as follows:

(1) For each $v \in V(G)$, we sample the triggering set of v from $\mathcal{T}(v)$ and remove any incoming edge of v that starts from a vertex outside $\mathcal{T}(v)$. Let G_{tr} be the resulting graph.
(2) We activate the vertices in S and propagate the diffusion on G_{tr}, which is the same as under the MCIC model.

The MCIC model and the multi-campaigner triggering model are different only in graph sampling. Therefore, when we extend *MIC* and *MIC*$^+$ to the multi-campaigner triggering model, we generate graph samples by the triggering model and leave the rest parts of the algorithms unchanged. Moreover, our algorithms may be applied to other diffusion models that require influence countering.

Table 2: Statistics of datasets

Dataset	n	m	d_{avg}	Type
Facebook	4,039	88,234	43.7	Undirected
Wiki	7,115	103,689	29.1	Directed
EmailAll	265,214	420,045	3.2	Directed
DBLP	317,080	1,049,866	6.6	Undirected
Stanford	281,903	2,312,497	16.4	Directed
Youtube	1,134,890	2,987,624	5.3	Undirected
LiveJournal	4,847,571	68,993,773	28.5	Directed
Orkut	3,072,441	117,185,083	38.1	Undirected

7 EXPERIMENTS

7.1 Experiment Setup

Datasets. The experiments are conducted on 8 datasets from SNAP [25]. Table 2 shows the statistics of the datasets, ordered by the number of edges, where d_{avg} is the average vertex degree.

To evaluate the algorithms comprehensively, we also test on undirected graphs. For undirected graphs, we convert each edge into a bi-directional edge. We find there is no clear difference between the performance on these graphs and the original directed graphs.

Algorithms. We evaluate baseline *BIM* (Section 3.5), baseline *BGA* (Section 3.6), basic approach *MIC* (Algorithm 1), and the improved approach *MIC*$^+$ (Algorithm 3). Moreover, we test *MIC-R* which is *MIC*$^+$ without BPT estimation (Section 5.2), i.e., we set $OPT = b$ when estimating the number of graph samples required.

Propagation Models. We consider two influence propagation models, that is, the MCIC model (see Section 3.1) and the multi-campaigner triggering model (see Section 6.2). We set the propagation probability by Weighted Cascade model: for each edge e points to a node v, we set $p_{u,v} = 1/d^-_v$ where d^-_v is the in-degree of v. This setting is widely adopted in existing studies [12, 20, 21].

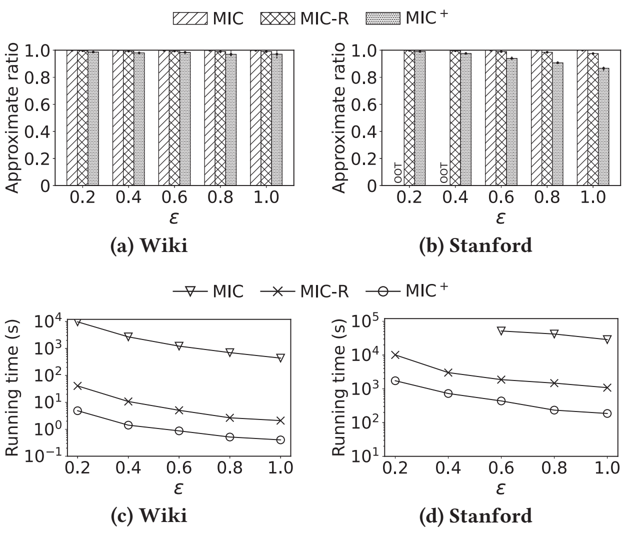
Parameter Settings. In each experiment, we repeat the method five times and report the average result. We terminate an algorithm if it cannot finish in 24 hours. Unless otherwise specified, we set $\varepsilon = 0.6$; set the number of seeds as 1% of the number of nodes, i.e., $\#seed = 0.01n$; and set the budget b as 10% of the number of seeds. To specify the seed set S, we find the set S of $\#seed$ nodes that maximize the influence [21], and then divide the seeds in S into five groups (each group adopts a company).

We compute the average influence spread over 10^5 rounds of MCIC diffusion as the ground truth of influence spread. We use this method when comparing the influence spread of different methods, which is widely used in prior works [42, 52].

Environments. We perform experiments on a CentOS Linux serve (Release 7.5.1804) with Quad-Core Intel Xeon CPU (E5-2640 v4 @ 2.20GHz) and 128G memory. All algorithms are implemented in C++17. The code is compiled with g++ 10.2.1 under O3 optimization.

7.2 Experiment Results

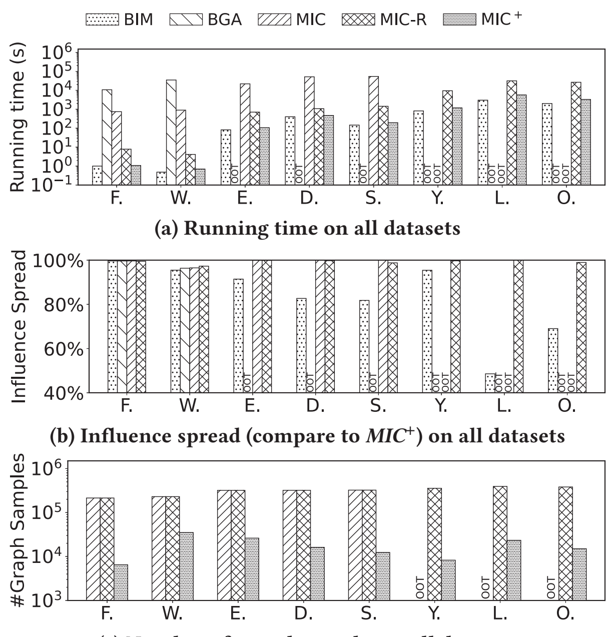
Exp. 1: Varying ε. We measure the efficiency of a method by running time, and the effectiveness by the approximate ratio of the result, i.e., the ratio between the influence increase led to by the algorithm and *OPT* which is the largest increase in influence spread for any possible countered set A. We obtain *OPT* as follows.



(a) Wiki (b) Stanford

(c) Wiki (d) Stanford

Figure 7: Vary ε (Exp. 1)



(a) Running time on all datasets

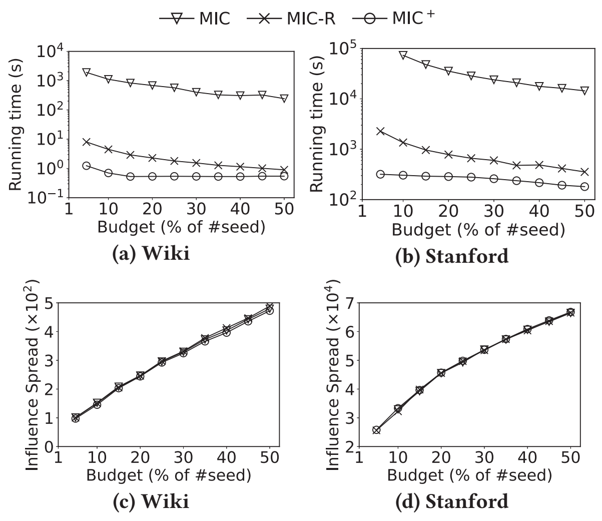
(b) Influence spread (compare to *MIC*$^+$) on all datasets

(c) Number of sample graphs on all datasets

Figure 8: Overall performance (Exp. 2)

First, we feed $l = 1, \varepsilon = 0.01$ to *MIC*$^+$, and obtain a set A in return. After that, we compute the influence spread of A by Monte Carlo simulations, and assign the resulting spread to OPT.

We vary ε from 0.2 to 1, and plot the performance of methods in Figure 7. The approximate ratio for *MIC-R* and *MIC*$^+$ are both larger than 0.9 when $\varepsilon \leq 0.8$, and the error is much smaller than ε. *MIC* slightly outperforms *MIC*$^+$ in approximate ratio, while it is up to 3 orders of magnitude slower (e.g., on `Wiki` dataset). Taking into account the above, we set $\varepsilon = 0.6$ by default for the experiments.



(a) Wiki **(b) Stanford**

(c) Wiki **(d) Stanford**

Figure 9: Vary the countering budget (Exp. 3)

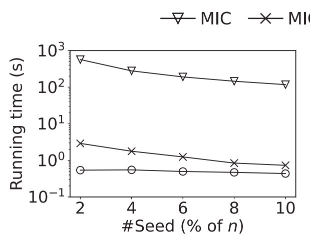
Exp. 2: Overall Performance. Figure 8 reports the performance of different algorithms. According to Figure 8a, *MIC* outperforms *BGA* in efficiency by 1 order of magnitude, and *MIC*$^+$ can further improve the efficiency by 3 orders compared with *MIC*. Besides, *MIC-R* is slower than *MIC*$^+$ by up to one order of magnitude, due to the optimization technique of *MIC*$^+$.

Figure 8b compares the influence spread of algorithms. Because our *MIC*$^+$ algorithm outperforms other algorithms in influence spread, we report the result percentage compared with *MIC*$^+$ for each algorithm. Although *BIM* has a close time cost with *MIC*$^+$, the influence spread of *BIM* is much smaller, e.g., in the Livejournal dataset, *BIM* returns only 48.57% of the influence spread achieved by *MIC*$^+$. Overall, *BIM* and *BGA* (i.e., the two baselines) have limitations in either effectiveness or efficiency, while our *MIC*$^+$ provides state-of-the-art performance.

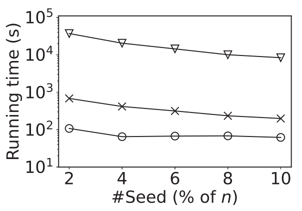
Figure 8c shows the number of graph samples θ required for the methods. The parameter θ decides the running time, and a small θ implies that the algorithm achieves the same approximate guarantees with a less number of samples. Compared with *MIC* and *MIC-R*, the number of samples required for *MIC*$^+$ is about one order of magnitude smaller. This shows that the technique of *MIC*$^+$ effectively reduces θ and the running time.

Exp. 3: Varying the Budget. Figure 9 reports the performance of algorithms, varying the budget b from 5% to 50% of the number of seeds. *MIC*$^+$ outperforms *MIC* and *MIC-R* in running time by 2-3 orders of magnitude, meanwhile, the running time of each method is stable across different budgets. The influence spread of the three methods (i.e., *MIC*, *MIC-R*, and *MIC*$^+$) is close, and the spread increases with the increase of budget. The running time of *MIC* and *MIC*$^+$ both decrease as the budget increases. The reason is that *BPT* will become larger as the budget b increases, and this leads to a smaller number of graph samples θ as well as reduced running time. Therefore, the budget will slightly affect the running time of our methods, and it can largely affect the influence spread.

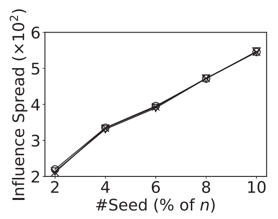
Exp. 4: Varying the Number of Seeds. Next, we vary the number of seeds from 2% to 10% of the number of vertices, and report the



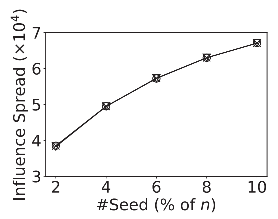
(a) Wiki



(b) Stanford



(c) Wiki



(d) Stanford

Figure 10: Vary the number of seeds (Exp. 4)

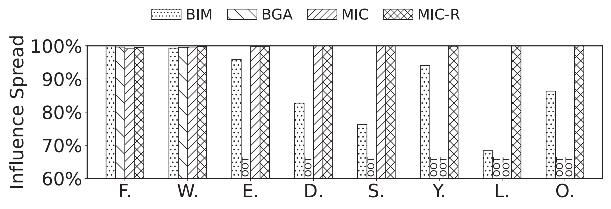
Table 3: Performance on dynamic graphs (Exp. 5)

D	Index		Update time (s)						
	Time (s)	Size (G)	$+e$	$-e$	Δp	$+v$	$-v$	$+s$	$-s$
F.	0.717	0.08	0.02	0.01	0.03	0.01	0.01	0.19	0.04
W.	1.452	0.07	0.07	0.01	0.09	0.01	0.01	0.56	0.02
E.	543.5	0.31	0.35	0.20	0.68	0.01	0.12	25.9	0.06
D.	270.5	0.31	0.49	0.02	0.07	0.01	0.01	5.50	0.13
S.	196.71	0.46	0.05	0.02	0.07	0.01	0.01	5.88	0.22
Y.	1232.8	0.81	0.04	0.02	0.06	0.01	0.01	26.8	0.10
L.	5733.7	9.88	0.09	0.06	0.14	0.05	0.05	450	0.25
O.	3635.3	14.9	0.27	0.16	0.52	0.02	0.01	194	0.18

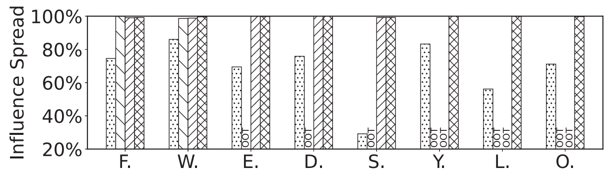
results in Figure 10. Our *MIC*$^+$ is consistently faster than *MIC* and *MIC-R*, achieving a speedup of 2-3 orders of magnitude in running time, despite they have the same theoretical time complexity. The influence spread of *MIC*, *MIC-R*, and *MIC*$^+$ is close. Note that, when the number of seeds becomes large, the budget b also increases as we have $b = 0.1\#seed$. As a result, the influence spread of methods increases and the running time decreases slightly, according to the analysis in Section 5.2. In short, the number of seeds has a limited effect on the running time of our methods, while this parameter can largely affect the influence spread of the results.

Exp. 5: Performance of Dynamic Algorithm. We evaluate the index construction of *MIC*$^+$ and the seven update operations on the index (see Section 6.1), then report the running time in Table 3. Compared with the running time of *MIC*$^+$ (Exp. 2), the index construction time is within the same order of magnitude. Note that a naive dynamic algorithm is to re-run *MIC*$^+$ after each update. Compared with this algorithm, our dynamic algorithm can speed up the update by up to 4 orders of magnitude. Overall, the index construction of *MIC*$^+$ incurs an overhead that is close to running *MIC*$^+$, while the update cost over *MIC*$^+$'s index is fairly small compared with re-running *MIC*$^+$.

To test edge insertion and removal (U1-U2), we remove 100 random edges from the graph, and then insert them back. We use a similar testing method for vertex (resp. seed) insertion and removal U4-U5 (resp. U6-U7). When testing edge probability change (U3),



(a) Influence spread (compare to *MIC*$^+$) on all datasets (`Degree`)



(b) Influence spread (compare to *MIC*$^+$) on all datasets (`Random`)

Figure 11: Different seed selection strategies (Exp. 6)

we select a random edge (u, v) from the graph, and then convert its associated propagation probability to either $p \times 2$ or $p/2$ at random.

Exp. 6: Seed Selection Strategies. By default, we adopt the `Max` strategy for seed selection, i.e., we select the seeds by the greedy influence maximization method. Here, we evaluate the performance with other vertex selection methods: (i) `Deg` selects the vertices with the highest degrees as the seeds for countering, and (ii) `Random` randomly chooses the seeds for countering. Figure 11 shows that the influence spread of *MIC*$^+$ is larger than other algorithms by up to 3.45 times for all the results returned within the time limit. We find that *BIM* is more sensitive to different seed selection methods because it does not consider the competition of multi-campaigners in seed selection and the selected seeds have varied qualities.

8 CONCLUSION

To the best of our knowledge, we are the first to study the problem of influence countering. We prove that the problem is #P-complete and its influence computation is #P-hard. Despite the theoretical hardness, we propose two novel algorithms *MIC* and *MIC*$^+$ to solve the problem with approximation guarantees and practical efficiency. Both *MIC* and *MIC*$^+$ run in $O((lnm \log n)/(b\varepsilon^2))$ time and return a $(1-\varepsilon)$-approximate solution with at least $1 - n^{-l}$ probability. In addition, we propose a well-designed index for *MIC*$^+$ that can handle graphs with frequent updates. The experimental results show that our algorithms are effective and efficient. For future work, we plan to study the problem under different influence strategies and design a system to facilitate successful influence competitions.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China (62002073, U2241211, U20B2046) and the Guangdong Basic and Applied Basic Research Foundation (2023A1515012603). Deming Chu is supported by the scholarship of the China Scholarship Council (No. 202006140012). We thank the anonymous reviewers for their insightful suggestions.

REFERENCES

[1] Zahra Aghaee, Mohammad Mahdi Ghasemi, Hamid Ahmadi Beni, Asgarali Bouyer, and Afsaneh Fatemi. 2021. A survey on meta-heuristic algorithms for the influence maximization problem in the social networks. *Computing* (2021), 2437–2477.

[2] Akhil Arora, Sainyam Galhotra, and Sayan Ranu. 2017. Debunking the Myths of Influence Maximization: An In-Depth Benchmarking Study. In *SIGMOD*. ACM, 651–666.

[3] Suman Banerjee, Mamata Jenamani, and Dilip Kumar Pratihar. 2020. A survey on influence maximization in a social network. *Knowl. Inf. Syst.* (2020), 3417–3455.

[4] Glenn S. Bevilacqua and Laks V. S. Lakshmanan. 2022. A fractional memory-efficient approach for online continuous-time influence maximization. *VLDB J.* 31, 2 (2022), 403–429.

[5] Shishir Bharathi, David Kempe, and Mahyar Salek. 2007. Competitive Influence Maximization in Social Networks. In *Internet and Network Economics, Third International Workshop, WINE (Lecture Notes in Computer Science, Vol. 4858)*. Springer, 306–311.

[6] Kaan Bingöl, Bahaeddin Eravci, Çagri Özgenç Etemoglu, Hakan Ferhatosmanoglu, and Bugra Gedik. 2019. Topic-Based Influence Computation in Social Networks Under Resource Constraints. *IEEE Trans. Serv. Comput.* 12, 6 (2019), 970–986.

[7] Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Brendan Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *SODA*. 946–957.

[8] Allan Borodin, Yuval Filmus, and Joel Oren. 2010. Threshold Models for Competitive Influence in Social Networks. In *Internet and Network Economics - 6th International Workshop, WINE (Lecture Notes in Computer Science, Vol. 6484)*. Springer, 539–550.

[9] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. 2011. Limiting the spread of misinformation in social networks. In *WWW*. 665–674.

[10] Tim Carnes, Chandrashekhar Nagarajan, Stefan M. Wild, and Anke van Zuylen. 2007. Maximizing influence in a competitive social network: a follower's perspective. In *ICEC (ACM International Conference Proceeding Series, Vol. 258)*. ACM, 351–360.

[11] Shuo Chen, Ju Fan, Guoliang Li, Jianhua Feng, Kian-Lee Tan, and Jinhui Tang. 2015. Online Topic-Aware Influence Maximization. *Proc. VLDB Endow.* 8, 6 (2015), 666–677.

[12] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*. ACM, 1029–1038.

[13] Pedro M. Domingos and Matthew Richardson. 2001. Mining the network value of customers. In *KDD*. ACM, 57–66.

[14] Sanjeev Goyal and Michael J. Kearns. 2012. Competitive contagion in networks. In *STOC*. ACM, 759–774.

[15] Long Guo, Dongxiang Zhang, Gao Cong, Wei Wu, and Kian-Lee Tan. 2017. Influence Maximization in Trajectory Databases. In *ICDE*. IEEE Computer Society, 27–28.

[16] Qintian Guo, Sibo Wang, Zhewei Wei, and Ming Chen. 2020. Influence Maximization Revisited: Efficient Reverse Reachable Set Generation with Bound Tightened. In *SIGMOD*. 2167–2181.

[17] Xinran He and David Kempe. 2013. Price of Anarchy for the N-Player Competitive Cascade Game with Submodular Activation Functions. In *WINE (Lecture Notes in Computer Science, Vol. 8289)*. Springer, 232–248.

[18] Xinran He, Guojie Song, Wei Chen, and Qingye Jiang. 2012. Influence Blocking Maximization in Social Networks under the Competitive Linear Threshold Model. In *SIAM*. SIAM / Omnipress, 463–474.

[19] Keke Huang, Sibo Wang, Glenn S. Bevilacqua, Xiaokui Xiao, and Laks V. S. Lakshmanan. 2017. Revisiting the Stop-and-Stare Algorithms for Influence Maximization. *Proc. VLDB Endow.* 10, 9 (2017), 913–924.

[20] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. IRIE: Scalable and Robust Influence Maximization in Social Networks. In *ICDM*. 918–923.

[21] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD*. ACM, 137–146.

[22] Arijit Khan, Benjamin Zehnder, and Donald Kossmann. 2016. Revenue maximization by viral marketing: A social network host's perspective. In *ICDE*. IEEE Computer Society, 37–48.

[23] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. 2006. Structure and evolution of online social networks. In *KDD*. ACM, 611–617.

[24] Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*. ACM, 177–187.

[25] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[26] Guoliang Li, Shuo Chen, Jianhua Feng, Kian-Lee Tan, and Wen-Syan Li. 2014. Efficient location-aware influence maximization. In *SIGMOD*. ACM, 87–98.

[27] Hui Li, Sourav S. Bhowmick, Jiangtao Cui, Yunjun Gao, and Jianfeng Ma. 2015. GetReal: Towards Realistic Selection of Influence Maximization Strategies in Competitive Networks. In *SIGMOD*. ACM, 1525–1537.

[28] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. 2018. Influence Maximization on Social Graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* 30, 10 (2018), 1852–1872.

[29] Konstantinos Liontis and Evaggelia Pitoura. 2016. Boosting Nodes for Improving the Spread of Influence. *CoRR* abs/1609.03478 (2016).

[30] Bo Liu, Gao Cong, Dong Xu, and Yifeng Zeng. 2012. Time Constrained Influence Maximization in Social Networks. In *ICDM*. IEEE Computer Society, 439–448.

[31] Bo Liu, Gao Cong, Yifeng Zeng, Dong Xu, and Yeow Meng Chee. 2014. Influence Spreading Path and Its Application to the Time Constrained Social Influence Maximization Problem and Beyond. *IEEE Trans. Knowl. Data Eng.* 26, 8 (2014), 1904–1917.

[32] Wei Lu, Francesco Bonchi, Amit Goyal, and Laks V. S. Lakshmanan. 2013. The bang for the buck: fair competitive viral marketing from the host perspective. In *KDD*. ACM, 928–936.

[33] Wei Lu, Wei Chen, and Laks V. S. Lakshmanan. 2015. From Competition to Complementarity: Comparative Influence Diffusion and Maximization. *Proc. VLDB Endow.* 9, 2 (2015), 60–71.

[34] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press.

[35] Hung T. Nguyen, Thang N. Dinh, and My T. Thai. 2018. Revisiting of 'Revisiting the Stop-and-Stare Algorithms for Influence Maximization'. In *CSoNet (Lecture Notes in Computer Science, Vol. 11280)*. Springer, 273–285.

[36] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. 2016. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. In *SIGMOD*. ACM, 695–710.

[37] Naoto Ohsaka. 2020. The Solution Distribution of Influence Maximization: A High-level Experimental Study on Three Algorithmic Approaches. In *SIGMOD*. ACM, 2151–2166.

[38] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2016. Dynamic Influence Analysis in Evolving Networks. *Proc. VLDB Endow.* 9, 12 (2016), 1077–1088.

[39] Binghui Peng. 2021. Dynamic influence maximization. In *NeurIPS 2021*. 10718–10731.

[40] Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. 2018. Online Processing Algorithms for Influence Maximization. In *SIGMOD*. ACM, 991–1005.

[41] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence Maximization in Near-Linear Time: A Martingale Approach. In *SIGMOD*. ACM, 1539–1554.

[42] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: near-optimal time complexity meets practical efficiency. In *SIGMOD*. ACM, 75–86.

[43] Dimitris Tsaras, George Trimponias, Lefteris Ntaflos, and Dimitris Papadias. 2021. Collective Influence Maximization for Multiple Competing Products with an Awareness-to-Influence Model. *Proc. VLDB Endow.* 14, 7 (2021), 1124–1136.

[44] Leslie G. Valiant. 1979. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.* 8, 3 (1979), 410–421.

[45] Sharan Vaswani, Branislav Kveton, Zheng Wen, Mohammad Ghavamzadeh, Laks V. S. Lakshmanan, and Mark Schmidt. 2017. Model-Independent Online Learning for Influence Maximization. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 3530–3539.

[46] Yanhao Wang, Qi Fan, Yuchen Li, and Kian-Lee Tan. 2017. Real-Time Influence Maximization on Dynamic Social Streams. *Proc. VLDB Endow.* 10, 7 (2017), 805–816.

[47] Yanhao Wang, Yuchen Li, Ju Fan, and Kian-Lee Tan. 2018. Location-aware Influence Maximization over Dynamic Social Streams. *ACM Trans. Inf. Syst.* 36, 4 (2018), 43:1–43:35.

[48] Jiadong Xie, Fan Zhang, Kai Wang, Xuemin Lin, and Wenjie Zhang. 2023. Minimizing the Influence of Misinformation via Vertex Blocking. In *39th IEEE International Conference on Data Engineering, ICDE 2023*. IEEE, 789–801.

[49] Jiadong Xie, Fan Zhang, Kai Wang, Jialu Liu, Xuemin Lin, and Wenjie Zhang. 2023. Influence Minimization via Blocking Strategies. *CoRR* abs/2312.17488 (2023). arXiv:2312.17488

[50] Miao Xie, Qiusong Yang, Qing Wang, Gao Cong, and Gerard de Melo. 2015. DynaDiffuse: A Dynamic Diffusion Model for Continuous Time Constrained Influence Maximization. In *AAAI*. AAAI Press, 346–352.

[51] Yipeng Zhang, Yuchen Li, Zhifeng Bao, Baihua Zheng, and H. V. Jagadish. 2021. Minimizing the Regret of an Influence Provider. In *SIGMOD*. ACM, 2115–2127.

[52] Yuqing Zhu, Jing Tang, Xueyan Tang, and Lei Chen. 2021. Analysis of Influence Contribution in Social Advertising. *Proc. VLDB Endow.* 15, 2 (2021), 348–360.