Efficient Algorithm for Budgeted Adaptive Influence Maximization: An Incremental RR-set Update Approach

Technical Report

ABSTRACT

Given a graph G, a cost associated with each node, and a budget B, the budgeted influence maximization (BIM) aims to find the optimal set S of seed nodes that maximizes the influence among all possible sets such that the total cost of nodes in S is no larger than B. Existing solutions mainly follow the non-adaptive idea, i.e., determining all the seeds before observing any actual diffusion. Due to the absence of actual diffusion information, they may result in unsatisfactory influence spread. Motivated by the limitation of existing solutions, in this paper, we make the first attempt to solve the BIM problem under the adaptive setting, where seed nodes are iteratively selected after observing the diffusion result of the previous seeds. We design the first practical algorithm which achieves an expected approximation guarantee by probabilistically adopting a cost-aware greedy idea or a single influential node. Further, we give an optimized version to improve its practical performance.

Besides, the scalability issues of the adaptive IM-related problems still remain open. It is because they usually involve multiple rounds (e.g., equal to the number of seeds) and in each round, they have to construct sufficient new reverse-reachable set (RR-set) samples such that the claimed approximation guarantee can actually hold. However, this incurs prohibitive computation, imposing limitations on real applications. To solve this dilemma, we propose an incremental update approach. Specifically, it maintains extra construction information when building RR-sets, and then it can quickly correct a problematic RR-set from the very step where it is first affected. As a result, we recycle the RR-sets at a small computational cost, while still providing correctness guarantee. Finally, extensive experiments on large-scale real graphs demonstrate the superiority of our algorithms over baselines in terms of both influence spread and running time.

1 INTRODUCTION

With the massive proliferation of social networks, viral marketing has become one of the most effective strategies in information dissemination, product promotion, and ideology formation, etc., by mining the network value of users [9]. In many applications, it is a common wish that the influence spread could be maximized when recruiting only a small subset of users as the seeds. In response to this practical demand, Kempe et al. first formalize the *influence maximization* (IM) problem in [26] and have thereafter inspired a plethora of research in this field [1, 3, 4, 13, 20, 30, 37, 38, 51].

The above efforts mainly belong to the non-adaptive category, where seed users are selected all at once, implying that the marketing company has to determine all the seed users before observing any diffusion results. In practice, some seeds, such as fake influencers [12], may fall significantly short of the predicted spread. Hence, companies tend to adopt the adaptive approach that alternates between seed selection and influence diffusion. Let us consider a case study released by the influencer marketing agency IZEA, where the famous fashion company Guess engaged them to raise awareness of their eyewear collection on Instagram [23] in 2019. To accomplish this task, IZEA progressively recruited 31 influencers over a span of at least 45 days. For instance, Virginia Varinelli (with 484k followers) posted her content on October 1, followed by André Hamann (with 990k followers) on October 9, Nadya Hasan (with 323k followers) on October 20, and Anita Místico (with 90k followers) on October 30. Further, Rocío Camacho (with 820k followers) was invited to enhance the influence on November 15. As promoted on IZEA's website, their team provides detailed analytics and reports that help understand how the campaign is performing and make adjustments to ensure the best possible results.

In addition, in real-world scenarios, the costs to invite influencers are often different from one another. Typically, the cost is measured by the number of followers an influencer has. It is reported that a nano influencer (1000–10,000 followers) on Instagram costs \$10–100 for each post, while a mega influencer (1,000,000+ followers) costs over \$10,000 for each post [39]. Accordingly, clients are usually recommended to determine their budget for the influencer marketing campaign, depending on their marketing objectives [24]. This is also reflected in the case studies released by IZEA. For example, Visit Tampa Bay, the official tourism organization for the city of Tampa, Florida, launched an influencer marketing campaign to attract more visitors to the Bay via IZEA. As shown in [25], the campaign is carefully conducted under a tight budget of \$66,250 provided by Visit Tampa Bay. Other influencer marketing agencies, such as Flaminijoy [11] and Linqia [31], also invite their clients to determine the budget for campaigns. This naturally motivates us to study the influence maximization problem under the budgeted adaptive setting (i.e. the budgeted adaptive IM problem, BAIM), aiming to provide efficient algorithms for practical viral marketing with theoretical guarantees.

To maximize the spread under a budget constraint, an intuitive greedy strategy may be considered for solution, which adaptively selects the node with maximum influence per unit cost as the seed. However, such a seemingly promising strategy provides no constant approximation guarantee. To illustrate, let us consider the case where a graph consists of node u_1 with influence $2 \cdot \epsilon$ and cost ϵ, node u_2 with influence 1 and cost 1, and all the other nodes with costs larger than 1. Given budget $B = 1$, according to the greedy strategy, we should select u_1 first, and then no other node can be taken in any more. It is clear that the optimal solution should be node u_2 with influence spread 1. Since ϵ can be an arbitrarily small value, we know that the greedy strategy achieves no constant approximation guarantee. To solve this problem, we develop the first practical framework MAPLE[1] for the BAIM problem in Section 3, which achieves an expected approximation guarantee by probabilistically adopting one of two sub-policies (i.e., CaGreedy and MIS). To improve the practical performance, we further propose

[1] Budgeted Adaptive Influence Maximization via Probabilistic Strategy.

an optimized version of MAPLE by providing the bounds for the expected influence of the two sub-policies and selecting the better one for execution. Especially, the bound for CaGreedy is derived by drawing support from another non-adaptive policy, since it is intricate to directly calculate a bound for an adaptive policy.

The scalability issues of the adaptive IM-related problems also deserve our efforts, as pinpointed in [21, 35]. The mechanism of adaptivity naturally asks the algorithm to work in a round-wise fashion: select the seeds one by one, each followed a diffusion observation. In each round of seed selection, to ensure the quality of the selected seed, it has to generate sufficient *new* RR-sets, instead of directly reusing the *old* ones in the previous rounds, to avoid estimation biases (which we will explain in Section 4.1). Typically, the number of the seeding rounds can be as large as several hundred [15, 17, 21], which incurs prohibitive computational costs. For example, with the state-of-the-art RR-set sampling technique [15, 16], it still takes about 1.8×10^4 seconds to find out 500 seeds under the uniform cost setting for the medium-scale dataset Livejournal with 4.8 million nodes. In Section 4, we manage to fill this gap by developing an incremental update approach. Specifically, we keep the unaffected RR-sets unchanged, and revise the affected RR-sets by resuming the sampling process from the very first activated node. In this way, we could recycle the RR-sets inherited from the previous rounds for influence estimation. Further theoretical analyses also validate the correctness of our technique.

We summarize our main contributions as follows.

- We design the first practical algorithm MAPLE that provides expected approximation guarantee for the BAIM problem. To achieve this, we develop the CASE[2] algorithm which selects a seed node with desirable guarantees in each seeding round.
- To improve the practical performance of MAPLE, we propose an optimized version, which attempts to compare the influence of its two sub-policies and adopt the larger one for execution.
- We further devise an incremental update approach such that we could recycle the old RR-sets to improve efficiency. This update approach can be easily extended to other adaptive IM-related problems, addressing the open problem of scalability.
- Finally, extensive experiments are carried out to evaluate the performance of our algorithms. The results show that our algorithms achieve significantly larger influence spread than the baselines and the incremental update method can reduce the running time by up to 50x.

2 PRELIMINARIES

The social network is denoted as a directed graph $G = (V, E)$, where the users and relationships are encoded by V and E respectively with $|V| = n$ and $|E| = m$. Each edge $e = (u, v)$ is associated with some probability $p(e) \in [0, 1]$ and each node is associated with a cost $c(v)$. We use the notation V_B to represent the set of nodes where each node has a cost no greater than B, namely $V_B = \{v \in V | c(v) \le B\}$. Further, for each edge $e = (u, v) \in E$, we say node u is the in-neighbor of node v, and node v is the out-neighbor of node u. Let $N_{\text{in}}(u)$ be the set of in-neighbors of node u, and $N_{\text{in}}(u, i)$ be the i-th in-neighbor of node u. The order of the in-neighbors does not affect the influence spread or the running time. Therefore, to facilitate the explanation of our algorithm, we assume that the in-neighbors of each node are sorted in ascending order of node IDs.

Table 1: Frequently used notations.

Notation	Description
$G = (V, E)$	a directed graph consists of node set V and edge set E
n, m	the number of nodes and edges, respectively
S	the set of nodes that are selected as seeds
B	the total budget
$c(v)$	the cost of selecting node v as a seed node
$\mathcal{R}$	a collection of RR-sets
π	an adaptive strategy of seed selection
ϕ	a realization instance
ψ	a partial realization instance
ω	random sources determining which node is selected
$\sigma_{\text{avg}}(S)$	the expected influence spread of the seed set S (usually used in non-adaptive setting)
$\sigma_{\text{avg}}(\pi)$	the expected influence spread of the policy π

For readers' convenience, we summarize the notations frequently used in the paper in Table 1.

2.1 Influence Propagation and Realization

Influence Propagation. In IM research, two diffusion models, i.e., the *independent cascade (IC)* and *linear threshold (LT)* model [26], are widely adopted to depict the influence propagation process, and thus are taken into account by our work. For ease of exposition, we will mainly focus on the IC model like [21, 41], while our results could be easily extended to the LT model. Specifically, in the IC model, given a set of seed nodes S, the diffusion process expands in discrete steps as follows. Initially, at step 0, only the seed nodes in S are activated and the remaining nodes are inactivated. Then, at each step $i > 0$, each node (e.g., u) that is newly activated in step $i - 1$ has a *single* chance to activate its inactive out-neighbors (e.g., v), succeeding with probability $p(u, v)$. After that, the node u could not make another attempt anymore. And the process continues until no node could be further activated. When the diffusion terminates, the total number of users activated, denoted by $\sigma(S)$, is referred to as the *influence spread* of the seed set S.

Realization. Next, we introduce the concept of realization under the IC model. To elaborate, for each edge $e = (u, v)$ with probability $p(e)$, we independently flip a biased coin which shows the head with probability $p(e)$, to determine the state of edge e (namely, *live* or *blocked*). If it is head, then edge e is live, indicating node u will influence v once u becomes activated; otherwise, edge e is blocked. A *realization* ϕ is the set of states of all edges in the graph. We use $\phi(e) = 1$ (resp. $\phi(e) = 0$) to represent an edge e is live (resp. blocked) under a realization ϕ. Then, we know that given a realization ϕ, the influence spread $\sigma(S, \phi)$ of a seed set S is the number of nodes that are reachable from S through the live edges in ϕ. Note that, the probability that ϕ happens is

$$p_\phi = \prod_{e:\phi(e)=1} p(e) \prod_{e:\phi(e)=0} (1 - p(e)).$$

[2] Cost-aware Node Selection with Expected Approximate Guarantee.

Then, let Φ be a random realization. We could define the *expected influence spread* of seed set S as

$$\sigma_{\text{avg}}(S) := \mathbb{E}_{\Phi}[\sigma(S,\Phi)] = \sum_{\phi} p_{\phi} \cdot \sigma(S,\phi).$$

2.2 Budgeted Adaptive Influence Maximization

Different from non-adaptive influence maximization that decides the seed set ahead of propagation, adaptive IM selects seeds in an iterative fashion: in each round, based on previous diffusion results which have been observed, a seed is selected according to some strategy, and then we make an observation of influence diffusion on the graph after activating the seed. Here we adopt the *full-adoption feedback model* [14]. Given realization ϕ, the activation state of node u, denoted as $\phi(u)$, is the statuses (i.e., live or blocked) of all edges that would be explored after we activate u. Mathematically, we model $\phi(u)$ as a function $\phi_u : E \longrightarrow \{0, 1, ?\}$, where $\phi_u(e) = 0$, $\phi_u(e) = 1$, and $\phi_u(e) = ?$ means activating u has revealed that e is blocked/live, and has not revealed the status of e, respectively, for edge $e \in E$ [14]. In addition, the set of all possible activation states in the graph is denoted as O.

Partial Realization. Next, we introduce the concept of *partial realization* ψ which represents the observations that we have made so far. It is a relation $\psi \subseteq V \times O$ such that $\{(u,o) : \psi(u) = o\}$. Furthermore, we use $\text{dom}(\psi)$ to denote the domain of ψ, namely $\text{dom}(\psi) = \{u : \exists o, (u,o) \in \psi\}$. We say a partial realization ψ is *consistent* with a realization ϕ (denoted as $\phi \sim \psi$) if for any node $u \in \text{dom}(\psi)$, we always have $\psi(u) = \phi(u)$. Furthermore, we say ψ is a subrealization of ψ' (denoted as $\psi \subseteq \psi'$) if $\text{dom}(\psi) \subseteq \text{dom}(\psi')$ and there exists at least one realization ϕ such that both ψ and ψ' are consistent with ϕ.

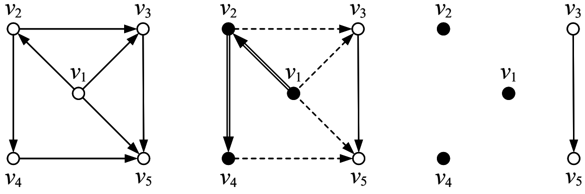
Further, we introduce the concept of *residual graph.* Given a partial realization ψ, we can construct a subgraph of G, called residual graph, by removing all activated nodes in ψ as well as their incident (incoming and outgoing) edges from graph G.

Policy. In general, the strategy for selecting the next seed in the adaptive IM problem is called a *policy*, denoted as π. Let $\pi(\psi)$ be the node selected by π under partial realization ψ, and $\mathcal{E}(\pi,\phi)$ be the set of seed nodes selected by π under realization ϕ. Then, we know the influence spread of policy π under realization ϕ is $\sigma(\mathcal{E}(\pi,\phi),\phi)$, i.e., the total number of nodes reachable from the seeds $\mathcal{E}(\pi,\phi)$. By taking the expectation with respect to the distribution of ϕ, the expected influence spread of policy π is defined as

$$\sigma_{\text{avg}}(\pi) := \mathbb{E}_{\Phi}[\sigma(\mathcal{E}(\pi,\Phi),\Phi)].$$

Given the same partial realization, a policy π is *deterministic*, if it always selects the same seed node; while a *randomized* policy may return nodes following certain distribution. Like [21], we use ω to represent all possible randomness brought by the randomized policy. When we write down $\pi(\omega)$, it represents a randomized policy. According to [21], we know that a deterministic policy is a special randomized policy without randomness, and any randomized policy can be constructed by a convex combination of deterministic policies. Therefore, the following conclusion holds [21]: for any randomized policy $\pi(\omega)$ and any deterministic policy π',

$$\max_{\pi} \mathbb{E}_{\omega}\left[\sigma_{\text{avg}}(\pi(\omega))\right] = \max_{\pi'} \sigma_{\text{avg}}(\pi').$$



(a) Graph G (b) Partial realization (c) Residual graph

Figure 1: Example of a policy and the partial realization.

BAIM. In the classic IM problem, all nodes have a uniform cost, which, however, as discussed in Section 1, may be unable to reflect real-world situations. Motivated by this, we define the budgeted adaptive influence maximization (BAIM) problem as follows:

Definition 2.1 (BAIM). Given an input graph $G = (V,E)$, a budget constraint B, and a cost function $c : V \longrightarrow \mathbb{Q}^+$, where $\mathbb{Q}^+$ is the set of positive rational numbers, BAIM aims to find a policy that maximizes the expected influence spread within budget B. Formally,

$$\begin{aligned} \max \quad & \sigma_{\text{avg}}(\pi) \\ \text{s.t.} \quad & \sum_{v \in \mathcal{E}(\pi,\phi)} c(v) \le B, \quad \text{for all } \phi. \end{aligned}$$

Example 2.2. To illustrate, let us consider Fig. 1 as an example. Given the graph $G = (V,E)$, with each edge of the same probability 0.5 and each node of the same cost 1, we aim to maximize the influence spread with an adaptive policy π. Suppose the budget constraint is $B = 2$. In the first round, the partial realization is obviously $\psi_0 = \emptyset$. By invoking π, we select the influential node $\pi(\emptyset) = v_1$ as the first seed. After diffusion, assume (v_1,v_2) and (v_2,v_4) are live while (v_1,v_3), (v_1,v_5), (v_2,v_3) and (v_4,v_5) are blocked. Thus, nodes v_2 and v_4 are activated. Then, the resultant observation of edges and nodes forms a partial realization ψ_1, as shown in Fig. 1(b). In the second round, we only need to consider the subgraph in Fig. 1(c) induced by the inactivated nodes v_3 and v_5. Clearly, the policy would reasonably yield $\pi(\psi_1) = v_3$ as the seed to maximize the influence. Finally, the budget is used up and the policy terminates.

Adaptive Monotonicity & Adaptive Submodularity. Given arbitrary partial realization (i.e., observation) ψ, to select a seed node, we often need to evaluate the influence increment that each node (e.g., v) could bring about in expectation, i.e., the conditional marginal benefit of v, which is formally defined as

$$\Delta(v \mid \psi) := \mathbb{E}_{\Phi\sim\psi}[\sigma(\text{dom}(\psi) \cup \{v\},\Phi) - \sigma(\text{dom}(\psi),\Phi)].$$

Similarly, the conditional marginal benefit of a policy π based on ψ could be written as

$$\Delta(\pi \mid \psi) := \mathbb{E}_{\Phi\sim\psi}[\sigma(\text{dom}(\psi) \cup \mathcal{E}(\pi,\Phi),\Phi) - \sigma(\text{dom}(\psi),\Phi)].$$

On this basis, we could define the adaptive monotonicity and adaptive submodularity as follows.

Definition 2.3. Given realization distribution p_{ϕ}, a function $\sigma(\cdot)$ is said to be adaptive monotone, if for any ψ and $v \in V \setminus \text{dom}(\psi)$, we have the inequality that $\Delta(v \mid \psi) \ge 0$. Further, the function $\sigma(\cdot)$ is said to be adaptive submodular, if for any $\psi \subseteq \psi'$ and $v \in V \backslash \text{dom}(\psi')$, it holds that $\Delta(v \mid \psi) \ge \Delta(v \mid \psi')$.

Algorithm 1: SelectAndVerify

```
   Input: $G = (V, E)$, approximation ratio $\alpha$.
   Output: A seed set $S$.
1  Compute the initial number $r_0$ and maximum number $r_{\max}$ of RR-sets;
2  $\mathcal{R}_1 \leftarrow \emptyset, \mathcal{R}_2 \leftarrow \emptyset, r \leftarrow r_0$;
3  do
4      Insert new random RR-sets into $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively until $|\mathcal{R}_1| = |\mathcal{R}_2| = r$;
5      $r \leftarrow 2r$;
6      Select a seed set $S \subseteq V$ based on $\mathcal{R}_1$;
7      Compute a lower bound $Q^l(S)$ of the quality score of $S$ based on $\mathcal{R}_2$;
8      Compute an upper bound $Q^u(S^o)$ of the quality score of the optimal solution $S^o$ based on $\mathcal{R}_2$ (or $\mathcal{R}_1$);
9  while $\left(Q^l(S) < \alpha \cdot Q^u(S^o) \textit{ and } r \le r_{\max}\right)$;
10 return $S$;
```

By Lemma 2.4, we see that the influence function $\sigma(\cdot)$ possesses the adaptive monotonicity and adaptive submodularity.

Lemma 2.4. *[14] The influence function $\sigma(\cdot)$ is adaptive monotone and adaptive submodular.*

2.3 Existing Solutions

Reverse-Reachable Sets. Since the seminal work of Borgs et al. [6], reverse-reachable sets (RR-sets) have been playing an essential role in influence spread estimation. To generate a random RR-set under the IC model, we need to follow two steps: (i) sample a node v uniformly at random from V; (ii) perform a *stochastic BFS* (i.e., Breadth First Search) from v in the reverse directions of edges. Specifically, for each frontier node u in the BFS process, it can successfully discover (namely, reversely activate) its in-neighbor w with probability $p(w, u)$, where $p(w, u)$ is the propagation probability of edge (w, u); otherwise we ignore w. The set of all discovered nodes forms an RR-set, denoted as R. Node v is called the *target node* of R. Further, Borgs et al. establish the following lemma that acts as the cornerstone of influence estimation.

Lemma 2.5. *[6] Let S be a seed set and R be a random RR-set, then*

$$\sigma_{avg}(S) = n \cdot \mathbb{P}[S \cap R \neq \emptyset].$$

We say a seed set S *covers* an RR set R if $S \cap R \neq \emptyset$, and the coverage of S in a collection $\mathcal{R}$ of RR-sets is the number of RR-sets covered by S (denoted as $\Lambda_{\mathcal{R}}(S)$). Then, according to Lemma 2.5, we see that $n \cdot \frac{\Lambda_{\mathcal{R}}(S)}{|\mathcal{R}|}$ is an unbiased estimation of seed set S's influence.

Based on the above idea, a number of estimation techniques are proposed [22, 42–44], among which the SelectAndVerify framework (see Alg. 1) exhibits the best performance and is thus widely applied in IM problems [5, 15, 16, 21, 42]. Specifically, SelectAndVerify maintains two collections of random RR-sets, $\mathcal{R}_1$ and $\mathcal{R}_2$. Then, in each iteration $\mathcal{R}_1$ is applied to select a seed set S, and $\mathcal{R}_2$ is then applied to verify the quality of S. Once S satisfies the desired approximation ratio, the algorithm stops immediately; otherwise, it doubles the size of $\mathcal{R}_1$ and $\mathcal{R}_2$, and repeats the above process until the stop condition is met. In the adaptive/non-adaptive IM problem under the uniform cost setting, the quality score of a seed set S is measured by the coverage of S on the collection of random RR-sets. It can be explained by Lemma 2.5: the higher influence spread a seed set has, the more RR-sets it tends to cover.

It should be pointed out that the seed sets returned by a RR-set-based algorithm may be different if we invoke it multiple times (all of the seed sets satisfy the desired approximation guarantee). That is, there exists randomness for the returned seed set.

Budgeted Non-adaptive IM. The state-of-the-art solution for the budgeted IM problem under the *non-adaptive* setting is IMAGE, proposed by Bian et al. [5]. IMAGE follows the aforementioned SelectAndVerify framework, and improve its efficiency by applying a tighter bound for the upper bound $Q^u(S^o)$ (see Line 8 in Alg. 1). To explain, recall that in the SelectAndVerify framework, the algorithm can terminate as soon as $Q^l(S) \geq \alpha \cdot Q^u(S^o)$, which indicates that at this moment, the expected influence of the selected seed set S has reached the desired approximation ratio α. Thus, given the same $Q^l(S)$, a tighter (i.e., smaller) bound $Q^u(S^o)$ could stop the algorithm earlier (with fewer iterations), saving the computational cost. IMAGE further accelerates the seed selection phase by utilizing a threshold greedy strategy [2], which takes $d_{\max} \cdot (1 - \xi)^i$ (where $d_{\max}$ is the maximum influence-to-cost ratio, and ξ is the error parameter) as the threshold in the i-th iteration, and then includes the nodes whose marginal influence-to-cost ratio is greater than the threshold. As a result, IMAGE could add more than one node in a single iteration, thus improving its efficiency. Although IMAGE is highly efficient such that it can answer the budgeted IM problem on billion-scale graphs within half an hour, under the non-adaptive setting the seed nodes are selected all at once before making any observation on the actual influence spread. Such a fixed strategy fails to take advantage of the previous spreading results when selecting the next seed node and thus is inferior in maximizing the total influence spread.

Adaptive IM with Uniform Cost. The adaptive uniform-cost IM problem aims to find k seed nodes in an adaptive manner to maximize the expected influence spread. It is a special case of our BAIM problem, by associating each node with a unit cost, and the budget as $B = k$. Huang et al. [21] propose the first practical framework, AdaptGreedy, providing a user-specified approximation guarantee. They prove that if in each round the selected seed node satisfies the expected α-approximate guarantee, then the final solution will achieve an expected approximate guarantee of $(1 - e^{-\alpha})$. Then, they adopt the SelectAndVerify framework to develop an algorithm EPIC, which returns a seed node with expected approximate guarantee. The instantiation of AdaptGreedy with EPIC is called EptAIM [21].

However, there exist two main drawbacks in EptAIM. First, the setting of all nodes having an equal cost is unrealistic. In social networks, it usually costs more to invite users with hundreds of thousands of followers than invite users with dozens of followers to join the campaign. Thus, the cost-aware setting should be considered for real applications. Second, as EptAIM takes k rounds to select k seed nodes and in each round, it needs to sample new RR-sets from scratch, as discussed in the introduction, which incurs huge computational cost. By [21], we know the scalability of the adaptive IM is still an open problem, deserving our attention.

Algorithm 2: MAPLE

```
Input: Graph $G = (V, E)$, approximation parameters $\alpha$ and
       $\beta$, budget $B$.
Output: A seed set $S$.
$p \leftarrow \frac{\beta}{1+\beta}$;
$r \leftarrow \mathrm{rand}(0, 1)$;
if $p \geq r$ then
    $S \leftarrow \mathrm{CaGreedy}(G, \alpha, B)$;
else
    $S \leftarrow \mathrm{MIS}(G, \beta, B)$;
end
return $S$;
```

3 OUR SOLUTION

Before presenting our solution, we introduce some useful concepts. The first one is *Expected Approximate Cost-aware Greedy Policy*:

Definition 3.1 (Expected Approximate Cost-aware Greedy Policy). A randomized policy π is an expected α-approximate cost-aware greedy policy if for any partial realization ψ, the selected node $\pi(\omega, \psi)$ always satisfies:

$$\mathbb{E}_{\omega}\left[\frac{\Delta(\pi(\omega,\psi) \mid \psi)}{c(\pi(\omega,\psi))}\right] \geq \alpha \cdot \max_{u} \frac{\Delta(u \mid \psi)}{c(u)}.$$

Then, we define two types of policy truncation operations, which would be applied when analyzing our solution. Given a policy π, let $v_{(i)}$ be the i-th node selected by π, and the accumulated cost of the first k nodes be $c(\pi, k) = \sum_{i=1}^{k} c(v_{(i)})$. Then we can give definitions of *Strict Policy Truncation* and *Probabilistic-rounding Policy Truncation* as follows.

Definition 3.2 (Strict Policy Truncation). The strict truncation of policy π to budget t, denoted as $\pi_{\leq t}$, is a policy that performs exactly the same as π, except that $\pi_{\leq t}$ only selects the first k nodes where k satisfies $c(\pi, k) \leq t$ and $c(\pi, k+1) > t$.

Definition 3.3 (Probabilistic-rounding Policy Truncation). The probabilistic-rounding truncation of policy π to budget t, denoted as π_t, is a randomized policy that performs exactly the same as π, except that π_t selects the first k nodes for certainty, and includes the $(k+1)$-th node independently with probability $\frac{t-c(\pi,k)}{c(v_{(k+1)})}$, where k satisfies $c(\pi, k) \leq t$ and $c(\pi, k+1) > t$.

From the above definitions, we know that the total cost of the strict truncation $\pi_{\leq B}$ (resp., probabilistic-rounding truncation π_B) of a policy π is not larger than (resp., equal to) budget B.

3.1 The MAPLE Framework

Under the budgeted setting, at first glance, we could greedily select those nodes with a high influence-to-cost ratio. However, as discussed in the introduction, in the worst case, the last node, which happens to be highly influential at a heavy cost, might be excluded to avoid exceeding the budget, resulting in no approximation guarantee. To tackle this issue, we integrate the ratio-first (i.e. influence-to-cost) strategy with the influence-first strategy, such that the mixed algorithm could provide expected approximation guarantee.

Procedure CaGreedy(G, α, B)

```
$S \leftarrow \emptyset$;
$G_1 \leftarrow G$;
for $i = 1, 2, 3, \ldots$ do
    $u^* \leftarrow CASE(G_i, \alpha, B)$;
    if $c(S) + c(u^*) \leq B$ then
        $S \leftarrow S \cup \{u^*\}$;
    else
        return $S$;
    end
    Observe an influence spread from $u^*$ in $G_i$;
    Remove the newly activated nodes from $G_i$ and obtain
      the residual graph $G_{i+1}$;
end
```

Procedure MIS(G, β, B)

```
Find a node $v \in V_B$ such that
    $\mathbb{E}_{\omega}(\sigma_{\mathrm{avg}}(\{v\})) \geq \beta \cdot \max_{u \in V_B} \sigma(u)$;
return $\{v\}$;
```

Specifically, our MAPLE framework follows a randomized structure that probabilistically adopts one of two sub-policies: cost-aware greedy (CaGreedy) and maximum-influence singleton (MIS). The pseudo-code of MAPLE is summarized in Alg. 2. To elaborate, given the approximation parameters α and β, we specify the probability to run CaGreedy and MIS as $p = \frac{\beta}{1+\beta}$ and $1 - p$ respectively. By comparing p with a random number r uniformly sampled from the interval $[0, 1]$, we determine which policy to adopt.

Suppose $p > r$, then the MAPLE framework invokes CaGreedy, which is based on the cost-aware greedy idea. It runs in a round-wise manner: in each round, a node u^*, whose influence per unit cost on the current residual graph G_i is at least α times that of the maximum one, is selected as the next seed. This is achieved by our CASE algorithm, which will be presented in Section 3.2. Then, we check whether the total cost would exceed B if u^* is added to the seed set S. If the answer is no, then we can safely insert u^* into S without violating the budget constraint, otherwise, we drop it out and terminate the algorithm. As u^* is included as a seed node, we then observe the influence spread starting from u^* in G_i, namely, the nodes reachable via live edges in G_i. To maximize the influence spread, it is not beneficial to repeatedly reach the same activated nodes in subsequent rounds. Therefore, at the end of each round, the newly activated nodes are removed from G_i, creating a new residual graph G_{i+1}. The algorithm repeats until we meet a node returned by CASE that exceeds the budget constraint. From the above discussion, we know CaGreedy is a strict truncation of the expected α-approximate cost-aware greedy policy.

If $p \leq r$, the MAPLE framework would execute the policy MIS which selects at most one seed. First, we do not need to consider the nodes whose cost is larger than B, since it is meaningless to take any node that violates the budget constraint. If V_B (recall that $V_B = \{v \in V | c(v) \leq B\}$) is not empty, then we try to find a node $v \in V_B$ such that in expectation (with respect to ω) v has an expected influence spread at least β times that of the optimal node. To achieve this end, we could directly invoke the algorithm EptAIM [21] by setting $k = 1$, where the implementation details are omitted to save space. Finally, node v is returned. When k=1, EptAIM is also a non-adaptive algorithm. EptAIM differs from standard IM algorithm by the type of approximation ratio they provide. EptAIM offers an expected approximation ratio, while the standard IM algorithm provides a worst-case approximation ratio, thus with more computational costs.

In the following, we will provide a theoretical analysis for the MAPLE framework. Before that, we present a useful conclusion for the expected approximate cost-aware greedy policy.

Theorem 3.4. *Given $\alpha \in [0, 1]$, budget $B \in \mathbb{Q}^+$, where $\mathbb{Q}^+$ is the set of positive rational numbers, and cost function $c : V \longrightarrow \mathbb{Q}^+$, let $\pi^c(\omega)$ be an expected α-approximate cost-aware greedy policy, and $\pi^*(\omega)$ be any policy. For any $0 \leq t \leq B$ $(t \in \mathbb{Q}^+)$, we have*

$$\mathbb{E}_\omega\left[\sigma_{avg}(\pi_t^c(\omega))\right] \geq \left(1 - e^{-\alpha t/B}\right) \cdot \mathbb{E}_\omega\left[\sigma_{avg}(\pi_B^*(\omega))\right].$$

The main idea of the proof for the above theorem is as follows: we first utilize the adaptive submodularity to derive that for two arbitrary policies π_B^* and π_t, it holds that $\sigma_{\text{avg}}\left(\pi_B^*\right) - \sigma_{\text{avg}}(\pi_t) \leq B \cdot \mathbb{E}_\Phi\left[\max_v \frac{\Delta(v|\Psi(\pi_t,\Phi))}{c(v)}\right]$, where the r.h.s term is budget B times the expectation of the maximum influence-to-cost ratio under the partial realization of π_t. Then, by the definition of cost-aware greedy policy, its gain for each unit cost is derived as: $\mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_{t+1}^c(\omega)\right)\right] - \mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_t^c(\omega))\right] \geq \frac{\alpha}{B} \cdot \left[\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_B^*(\omega))\right] - \mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_t^c(\omega))\right]\right]$. Further, rewriting the above inequality as a recurrence relation, and using the fact $1 - x \leq e^{-x}$ for any $x > 0$, we prove the theorem. The detailed proof could be found in the appendix.

Theorem 3.5. *The MAPLE framework $\pi_{\leq B}(\omega)$ achieves an approximation ratio of $\frac{\beta}{1+\beta} \cdot (1 - e^{-\alpha})$. That is,*

$$\mathbb{E}_\omega\left[\sigma_{avg}(\pi_{\leq B}(\omega))\right] \geq \frac{\beta}{1+\beta} \cdot \left(1 - e^{-\alpha}\right) \cdot \sigma_{avg}(\pi_{\leq B}^o).$$

where $\pi_{\leq B}^o = \arg\max_{\pi'_{\leq B}} \sigma_{avg}(\pi'_{\leq B})$ is the optimal policy.

Proof. It is obvious that $\pi_{\leq B}^o$ does not contain any node whose cost is greater than B, since $\pi_{\leq B}^o$ should be a feasible solution. Then, without loss of generality, we assume the cost of each node $v \in V$ satisfies $c(v) \leq B$.

Let $\pi^c(\omega)$ be an expected α-approximate cost-aware greedy policy. Note that policy CaGreedy is the strict truncation of $\pi^c(\omega)$, and thus could be denoted as $\pi_{\leq B}^c(\omega)$. Let $\pi_B^c(\omega)$ be the probabilistic-rounding truncation of $\pi^c(\omega)$. From the definitions of policy truncation, for any fixed ω, the only difference between $\pi_{\leq B}^c(\omega)$ and $\pi_B^c(\omega)$ lies in that $\pi_B^c(\omega)$ may admit the last node with some probability. Therefore, by the adaptive submodularity, we have

$$\sigma_{\text{avg}}(\pi_B^c(\omega)) \leq \sigma_{\text{avg}}\left(\pi_{\leq B}^c(\omega)\right) + \max_u \sigma_{\text{avg}}(\{u\}).$$

Then, by taking the expectation over the randomness of ω in both sides, we could derive

$$\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_B^c(\omega))\right] \leq \mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_{\leq B}^c(\omega))\right] + \max_u \sigma_{\text{avg}}(\{u\}). \tag{1}$$

On the other hand, according to Theorem 3.4 , by setting $t = B$, policy $\pi_B^c(\omega)$ achieves an approximation of $(1 - e^{-\alpha})$:

$$\begin{aligned}\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_B^c(\omega))\right] &\geq \left(1 - e^{-\alpha}\right) \cdot \max_{\pi'_B} \sigma_{\text{avg}}(\pi'_B) \\ &\geq \left(1 - e^{-\alpha}\right) \cdot \max_{\pi'_{\leq B}} \sigma_{\text{avg}}\left(\pi'_{\leq B}\right) \end{aligned}\tag{2}$$

where the second inequality is due to that for any policy π, its probabilistic-rounding truncation π_t admits the last node with some probability, and thus its expected influence will be not smaller than that of the strictly truncated policy $\pi_{\leq t}$.

Let $\pi^s(\omega)$ be policy MIS. From the pseudo-code, we have

$$\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi^s(\omega))\right] \geq \beta \cdot \max_u \sigma_{\text{avg}}(\{u\}). \tag{3}$$

From inequalities (1) and (3), we obtain

$$\begin{aligned}&\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_B^c(\omega))\right] \leq \mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_{\leq B}^c(\omega)\right)\right] + \frac{\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi^s(\omega))\right]}{\beta} \\ &= \frac{1+\beta}{\beta} \cdot \left(\frac{\beta}{1+\beta} \cdot \mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_{\leq B}^c(\omega)\right)\right] + \frac{1}{1+\beta} \cdot \mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi^s(\omega))\right]\right) \\ &= \frac{1+\beta}{\beta} \cdot \mathbb{E}_\omega[\pi_{\leq B}(\omega)] \end{aligned}\tag{4}$$

where the second equality holds since Alg. 2 runs CaGreedy with probability $\frac{\beta}{1+\beta}$, and runs MIS with probability $\frac{1}{1+\beta}$.

Finally, combining inequalities (2) and (4), we prove

$$\begin{aligned}\mathbb{E}_\omega[\pi_{\leq B}(\omega)] &\geq \frac{\beta}{1+\beta} \cdot \mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_B^c(\omega))\right] \\ &\geq \frac{\beta}{1+\beta} \cdot \left(1 - e^{-\alpha}\right) \cdot \max_{\pi'_{\leq B}} \sigma_{\text{avg}}\left(\pi'_{\leq B}\right).\end{aligned}$$

This completes the proof. □

3.2 CASE Algorithm

Recap that policy CaGreedy needs to find a seed node whose influence per unit cost in expectation is at least α times the maximum one. To achieve this end, we develop the CASE algorithm which returns a seed node satisfying the desired expected approximation guarantee. The pseudo-code of CASE is shown in Alg. 3.

At a high level, CASE follows the SelectAndVerify framework. In Lines 1-3, we first initialize a few parameters such that the correctness of the output could be guaranteed (please refer to the proof of Lemma 3.6). Then we begin to generate RR-sets for seed selection with a while loop. In each round, we first select the node u^* whose coverage (in $\mathcal{R}_1$) per unit cost is the largest one as the candidate (see Line 7). Then, to verify the quality of the candidate u^*, we compute a lower bound $Q^l(u^*)$ (see Line 9) for its expected influence-to-cost ratio, namely $\mathbb{E}_\omega\left[\frac{\sigma_{\text{avg}}(\{u^*\})}{c(u^*)}\right]$. Meanwhile, an upper bound $Q^u(v^o)$ of the optimal node v^o's influence-to-cost ratio is further calculated (see Line 10). If the quality of u^* is good enough such that the approximation satisfies $Q^l(u^*) \geq \alpha \cdot Q^u(v^o)$, then the algorithm terminates immediately; otherwise, it will double the size of $\mathcal{R}_1$ and $\mathcal{R}_2$ (unless reaching the maximum size $r_{\max}$).

By Lemma 3.6, we show that the CASE algorithm could achieve an expected α-approximate guarantee.

Lemma 3.6. *Given residual graph $G_i = (V_i, E_i)$, approximation parameter $\alpha \in [0, 1)$ and budget B, the CASE algorithm could return a seed node u^* such that it satisfies*

$$\mathbb{E}_\omega\left[\frac{\sigma_{avg}(\{u^*\})}{c(u^*)}\right] \geq \alpha \cdot \max_{v \in V_B} \frac{\sigma_{avg}(\{v\})}{c(v)}, \tag{5}$$

where $V_B = \{v \in V_i \mid c(v) \leq B\}$.

The detailed proof could be found in the appendix. Let $v^o = \arg\max_{v \in V_B} \frac{\sigma_{\text{avg}}(\{v\})}{c(v)}$. Intuitively, as CASE terminates due to $Q^l(u^*) \geq \alpha \cdot Q^u(v^o)$, by taking the expectation with respect to the randomness of ω, we have

$$\mathbb{E}_\omega\left[\frac{\sigma_{\text{avg}}(u^*)}{c(u^*)}\right] \geq \mathbb{E}_\omega\left[Q^l(u^*)\right] \geq \mathbb{E}_\omega\left[\alpha \cdot Q^u(v^o)\right]$$
$$\geq \alpha \cdot \mathbb{E}_\omega\left[\frac{\Lambda_{\mathcal{R}_2}(v^o)}{c(v^o)} \cdot \frac{n_i}{r}\right] = \alpha \cdot \frac{\sigma_{\text{avg}}(v^o)}{c(v^o)}$$

where the first inequality is due to the Chernoff concentration bound, the second inequality is due to the stopping condition, the third inequality is due to the definition of $Q^u(v^o)$, and the last equality is due to the fact $\mathbb{E}_\omega\left[\frac{\Lambda_{\mathcal{R}_2}(v^o)}{c(v^o)} \cdot \frac{n_i}{r}\right] = \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}$. On the other hand, as the influence of each node is at least one (i.e., itself), by concentration bounds, we can estimate the expected influence of each node with approximation guarantee when $r \geq r_{\max}$. Further, we prove that in this case, the selected u^* satisfies inequality (5).

In this paper, the upper bound $Q^u(v^o)$ only needs to satisfy $Q^u(v^o) \geq \frac{\Lambda_{\mathcal{R}_2}(v^o)}{c(v^o)} \cdot \frac{n_i}{r}$. By taking the expectation, we can guarantee that $\mathbb{E}_\omega[Q^u(v^o)] \geq \mathbb{E}_\omega\left[\frac{\Lambda_{\mathcal{R}_2}(v^o)}{c(v^o)} \cdot \frac{n_i}{r}\right] = \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}$. Therefore, it is sufficient to provide an *expected* approximation guarantee. It is noted that the upper bound expression for $Q^u(v^o)$ in this paper differs from Equation (8) in [42] due to the different types of approximation guarantees required. In [42], the bound $Q^u(S^o)$ guarantees that $Q^u(S^o)$ is no smaller than the expected influence $\sigma_{\text{avg}}(S^o)$ with high probability, providing a worst-case approximation guarantee with high probability. As to $Q^l(u^*)$, since we may select different u^*, we must ensure that $Q^l(u^*) \leq \frac{\sigma_{\text{avg}}(\{u^*\})}{c(u^*)}$ holds, such that after taking the expectation, we can derive $\mathbb{E}_\omega\left[Q^l(u^*)\right] \leq \mathbb{E}_\omega\left[\frac{\sigma_{\text{avg}}(\{u^*\})}{c(u^*)}\right]$. This is why the concentration bound is still utilized in the expression of $Q^l(u^*)$.

Let $c_{\max} = \max_{v \in V} c(v)$ and $c_{\min} = \min_{v \in V} c(v)$. Then we can derive the time complexity of CASE.

Lemma 3.7. *Given the approximation ratio α, the algorithm CASE runs in $O\Big(\frac{c_{\max}}{c_{\min}} \cdot \frac{\log n_i \cdot (n_i + m_i)}{(1-\alpha)^2}\Big)$ expected time.*

The proof of Lemma 3.7 is deferred to the appendix.

By the fact that we select at most $B/c_{\min}$ seed nodes, we derive that the time complexity of CaGreedy is $O\Big(\frac{B}{c_{\min}} \cdot \frac{c_{\max}}{c_{\min}} \cdot \frac{\log n \cdot (n+m)}{(1-\alpha)^2}\Big)$ with Lemma 3.7. According to [21], given the approximation parameter β, as we run only one round, the time complexity of MIS is $O\Big(\frac{m+n}{(1-\beta)^2} \cdot \log n\Big)$. Recall that MAPLE probabilistically adopts CaGreedy or MIS. The time complexity of MAPLE is dominated by CaGreedy, and thus is $O\Big(\frac{B}{c_{\min}} \cdot \frac{c_{\max}}{c_{\min}} \cdot \frac{\log n \cdot (n+m)}{(1-\alpha)^2}\Big)$ if we set $\alpha \approx \beta$.

Algorithm 3: CASE

```
Input: $G_i = (V_i, E_i)$, approx. parameter $\alpha$, budget $B$
Output: A seed node $u^*$
$\mathcal{R}_1 \leftarrow \emptyset$, $\mathcal{R}_2 \leftarrow \emptyset$;
$\delta \leftarrow \frac{1}{n_i}$, $\epsilon_a \leftarrow \frac{1-\delta}{\alpha} - 1$, $r_{\max} \leftarrow \left\lceil \frac{(2+2\epsilon_a/3)n_i}{\epsilon_a^2} \cdot \ln \frac{n_i}{\delta} \right\rceil + 1$;
$r \leftarrow \frac{1}{2} \cdot \ln \frac{n_i}{\delta}$, $a \leftarrow \ln \frac{1}{\delta}$;
do
    $r \leftarrow 2 \cdot r$;
    Insert new random RR-sets into $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively
      until $|\mathcal{R}_1| = |\mathcal{R}_2| = r$;
    $u^* \leftarrow \arg\max_{v \in V_B} \frac{\Lambda_{\mathcal{R}_1}(v)}{c(v)}$;
    $\Lambda^l_{\mathcal{R}_2}(u^*) \leftarrow \left(\sqrt{\Lambda_{\mathcal{R}_2}(u^*) + \frac{2a}{9}} - \sqrt{\frac{a}{2}}\right)^2 - \frac{a}{18}$;
    $Q^l(u^*) \leftarrow \Lambda^l_{\mathcal{R}_2}(u^*) \cdot \frac{n_i}{r \cdot c(u^*)} - \delta \cdot \frac{n_i}{c(u^*)}$;
    $Q^u(v^o) \leftarrow \max_{v \in V_B} \frac{\Lambda_{\mathcal{R}_2}(v)}{c(v)} \cdot \frac{n_i}{r}$;
while $\left(Q^l(u^*) < \alpha \cdot Q^u(v^o)\ and\ r < r_{\max}\right)$;
return $u^*$;
```

3.3 Improve Practical Influence Spread

In Section 3.1, we develop the algorithm MAPLE, which manages to provide the expected approximation guarantee. However, the practical performance may not be that satisfactory due to the probabilistic execution strategy. To explain, as $\beta \in (0, 1)$, the probability $\frac{1}{1+\beta}$ of invoking procedure MIS will be greater than 1/2. Therefore, the practical performance of MAPLE largely depends on MIS. However, MIS only selects at most one node, thus likely to produce an influence spread much smaller than CaGreedy which could select numerous seed nodes with a high influence-to-cost ratio.

To overcome the above deficiency, we find it encouraging to compare the influences of the two policies before performing probabilistic execution. When CaGreedy is expected to have a larger influence than MIS, we could execute CaGreedy with absolute certainty to improve the practical influence spread. Otherwise, we continue to adopt the two policies probabilistically. In practice, it is relatively easy to estimate the influence of MIS, which could be realized by applying the RR-set technique. However, it is almost unrealistic to estimate the influence of CaGreedy $\pi^c_{\leq B}(\omega)$ in advance since due to its adaptive nature, the seed nodes are selected one by one according to the propagation results of the previous seed set.

To circumvent this hurdle, we could resort to a non-adaptive policy π^{na} as a bridge. That is, with π^{na}, we establish a lower bound for the expected influence of $\pi^c_{\leq B}(\omega)$.

Let S_{na} denote the seed set returned by π^{na}, and $\sigma^l(S_{\text{na}})$ be a lower bound of its expected influence spread. Further, let v^o still be the node with maximum influence spread, and $\sigma^u(\{v^o\})$ be an upper bound of its influence spread. The relationship between S_{na} and $\pi^c_{\leq B}(\omega)$ is given by Lemma 3.8.

Lemma 3.8. *Given approximation parameter α, $\pi^c_{\leq B}(\omega)$ satisfies*

$$\mathbb{E}_\omega\left[\sigma_{avg}\Big(\pi^c_{\leq B}(\omega)\Big)\right] \geq (1 - e^{-\alpha}) \cdot \sigma^l(S_{na}) - \sigma^u(\{v^o\}).$$

At a high level, Lemma 3.8 can be proved by Theorem 3.4 (let π^*_B be a non-adaptive policy returning seeds S_{na}) and the fact that for any policy, the spread difference between its strictly truncated version and probabilistic-rounding version is at most $\sigma_{\mathrm{avg}}(\{v^o\})$. The detailed proof is deferred to the appendix.

Then, Theorem 3.9 shows that if the non-adaptive proxy is better than MIS by a constant factor, the approximation performance of MAPLE could be elevated to $\frac{1}{2} \cdot (1 - e^{-\alpha})$.

Theorem 3.9. *If S_{na} satisfies $(1 - e^{-\alpha}) \cdot \sigma^l(S_{na}) \geq 2\sigma^u(\{v^o\})$, then by running CaGreedy with probability one, it holds for MAPLE*

$$\mathbb{E}_\omega\left[\sigma_{avg}(\pi_{\leq B}(\omega))\right] \geq \frac{1}{2}(1 - e^{-\alpha}) \cdot \sigma_{avg}(\pi^o_{\leq B}),$$

where $\pi^o_{\leq B}$ is the optimal policy.

The detailed proof is deferred to the appendix. Note that Lemma 3.8 indicates that to find a tight lower bound for CaGreedy, we should find a seed set S_{na} with as high expected influence as possible. To this end, we can utilize the existing algorithm IMAGE [5] designed for the budgeted non-adaptive IM problem. Furthermore, to calculate the bounds $\sigma^l(S_{\mathrm{na}})$ and $\sigma^u(\{v^o\})$, we make use of sufficient RR-sets to obtain a probabilistic result. That is, the bounds $\sigma^l(S_{\mathrm{na}})$ and $\sigma^u(\{v^o\})$ hold with high probability (say $1 - \frac{1}{n}$). Therefore, we can summarize the optimized MAPLE (denoted as MAPLE+) as follows:
(i) invoke IMAGE to get a seed set S_{na} and calculate the bounds $\sigma^l(S_{\mathrm{na}})$ and $\sigma^u(\{v^o\})$ with high probability;
(ii) if $(1 - e^{-\alpha}) \cdot \sigma^l(S_{\mathrm{na}}) \geq 2\sigma^u(\{v^o\})$, execute CaGreedy with probability one; if $\sigma^l(S_{\mathrm{na}}) \geq \sigma^u(\{v^o\})$, execute CaGreedy with probability $\frac{\beta}{1+\beta}$, and return S_{na} with $\frac{1}{1+\beta}$; otherwise, run CaGreedy with probability $\frac{\beta}{1+\beta}$, and run MIS with $\frac{1}{1+\beta}$.

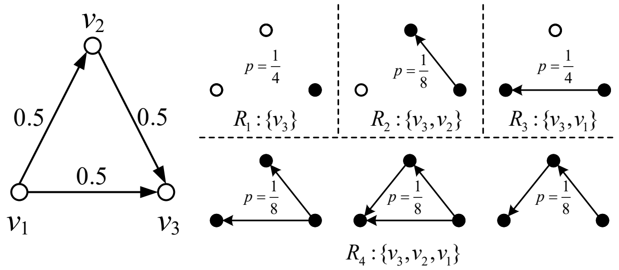
From the proof of Theorem 3.5, we can realize that the optimized version MAPLE+ has the same expected approximation guarantee as the basic version MAPLE with high probability.

Time Complexity. Note that MAPLE+ needs to first run IMAGE and MIS and relies on their output to make a decision about which algorithm to invoke. according to [5], given the approximation parameter α, the time complexity of IMAGE is $O\Big(\frac{n+m}{(1-\alpha)^2} \cdot (\log n + \frac{B}{c_{\min}})\Big)$. From the time complexity of IMAGE, MIS, and MAPLE, we can immediately know that in spite of extra costs, MAPLE+ still has the same complexity as MAPLE, namely $O\Big(\frac{B}{c_{\min}} \cdot \frac{c_{\max}}{c_{\min}} \cdot \frac{\log n \cdot (n+m)}{(1-\alpha)^2}\Big)$.

4 INCREMENTAL UPDATES ON RR-SETS

4.1 Estimation Biases

In Section 3.1, we have developed the MAPLE framework to solve the BAIM problem. Recall that MAPLE involves multi-round seed selection, and in each round, it has to generate sufficient RR-sets to guarantee the quality of the selected seed node, which incurs huge computational costs. To improve efficiency, one may consider a strategy of reusing RR-sets. Specifically, at the very beginning, a collection $\mathcal{R}$ of random RR-sets is generated, and we select the first seed node $u_{(1)}$ based on $\mathcal{R}$. Then by making an observation starting from $u_{(1)}$, we remove from $\mathcal{R}$ those RR-sets that contain any activated node and use the remaining RR-sets in $\mathcal{R}$ to select the second seed node $u_{(2)}$. With such a strategy, those RR-sets that are not covered by the active nodes (observed so far) are left to be reused in the subsequent rounds, thus, saving computational costs of re-generating them from scratch.



(a) Graph G (b) Possible RR-sets from v_3

Figure 2: Example of estimation biases.

At first glance, such an idea is promising for efficiency. However, it will cause estimation biases. To illustrate, let us consider a simple example in Figure 2. Given the toy graph $G = (V, E)$ consisting of three nodes v_1, v_2, v_3 and three edges $(v_1, v_2), (v_1, v_3), (v_2, v_3)$, with each edge associated with equal probability 0.5, we attempt to estimate the probability of v_1 and v_2 influencing v_3. To this end, we reverse all the edges and select v_3 as the target node to perform reverse propagation. Then, it is easy to enumerate all possible propagation patterns and calculate the corresponding occurrence probabilities. As shown in Figure 2(b), there are six propagation results, leading to four types of RR-sets, that is, $R_1 = \{v_3\}$, $R_2 = \{v_3, v_2\}$, $R_3 = \{v_3, v_1\}$ and $R_4 = \{v_3, v_2, v_1\}$, each existing with probability 0.25, 0.125, 0.25, 0.375 respectively. Then, according to the rationale of RR-sets, the probability of v_1 influencing v_3 is equal to the frequency that v_1 exists in the RR-sets generated from v_3, which is $0.25 + 0.375 = 0.625$ due to R_3 and R_4. Similarly, we could derive the probability v_2 influencing v_3 to be 0.5. After a simple verification, we can find that the estimation so far is correct. Next, assume node v_1 is selected as the seed and we further assume only v_1 itself is activated. Then we delete RR-sets R_3 and R_4, since they are covered by v_1, leaving a quarter of RR-sets being R_1 type and an eighth of RR-sets being R_2 type. On this basis, we further estimate the probability of v_2 influencing v_3 which would be $\frac{0.125}{0.25+0.125} = \frac{1}{3}$, contradicting with the actual probability 0.5.

Thus, we can see that the naive idea of deleting intersected RR-sets would incur evident biases in adaptive influence estimation. Currently, there exists no approach to correct such estimation biases [21]. To tackle this problem, a redeeming method is to reconstruct all the RR-sets in each round, which however would incur prohibitive time consumption, especially in large graphs.

4.2 Incremental Approach

Our goal is to find an efficient way to maintain the index of RR-sets, such that it can be still used for influence estimation without biases in the subsequent rounds, thus saving computational costs. To this end, we propose an incremental update approach. We say an RR-set is *polluted* if it contains at least one activated node. At a high level, we revise those polluted RR-sets to make them clean again, such that the integration of newly updated RR-sets and unpolluted ones can accurately estimate the influence spread. In the following,

Algorithm 4: RR-Set-Gene(G_i, v)

```
Input: Graph $G_i = (V_i, E_i)$, target node $v$.
Output: RR-set $R$ (in an array) and predecessor array $H$.
$R \leftarrow \emptyset, H \leftarrow \emptyset; discovered[u] \leftarrow false, \forall u \in V_i$;
$R$.append($v$) ;            /* append $v$ to the end of $R$ */
$H$.append($-1$);                  /* root node */
$discovered[v] \leftarrow true$;
$k \leftarrow 0$;
while $k < |R|$ do
    $f \leftarrow R[k]$ ;          /* the frontier of BFS */
    Probe($f, k, R, H, 0$);
    $k \leftarrow k + 1$;
end
```

Procedure Probe(f, k, R, H, pos)

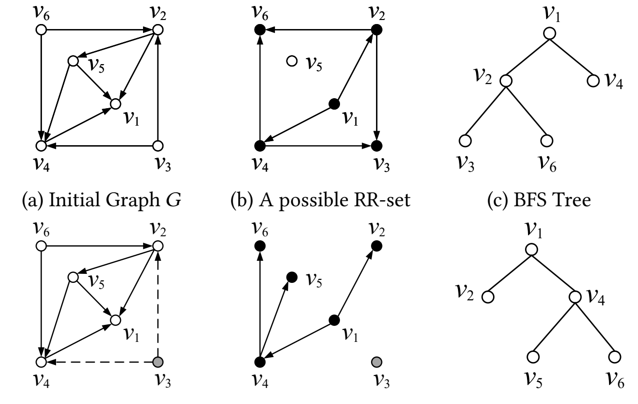
```
for $l = pos, pos + 1, \dots, |N_{in}(f)| - 1$ do
    $w \leftarrow N_{\text{in}}(f, l)$ ;         /* the $l$-th in-neighbor */
    if $rand() \leq p(w, f)$ and $discovered[w] \neq true$ then
        $R$.append($w$);
        $H$.append($k$);
        $discovered[w] \leftarrow true$;
    end
end
```

we first present the new RR-set generation method, and then we demonstrate how to update a polluted RR-set.

RR-Set Generation. Recalling Section 2.3, we know that an RR-set R is generated by first selecting a target node v, then performing a stochastic BFS from v along reversed edges and finally taking the activated nodes as R. Naturally, the nodes in R form a BFS tree where each parent-child relationship indicates the child node is activated by the parent node. To help identify which nodes are affected by the newly active node, we would like to keep track of the activation relationships, and thus associate with each R a predecessor array H, which records the parent information of each node in R. We use an array to store R, where the elements are ordered according to when it is discovered. Further, without loss of generality, we assume the traversal of neighbors during BFS follows a fixed order (e.g., ascending order of node IDs).

The pseudo-code of our RR-set generation process is presented in Alg. 4. After invoking the generation algorithm, we obtain an array-formatted RR-set R and an H, where $R[i]$ is the i-th discovered node, and it is first discovered by node $R[H[i]]$. To help comprehension, we further present example 4.1 for a vivid explanation.

Example 4.1. Let us consider the toy graph in Fig. 3(a). First, we need to reverse all the edge directions to facilitate RR-set derivation. A possible RR-set is illustrated in Fig. 3(b), where a blacked (resp. hollow) circle represents an activated (resp. unactivated) node. A solid line is used to represent a successful activation attempt. In contrast, if an activation attempt fails, we remove the corresponding edge. The generation procedure of the RR-set is as follows: First, node v_1 is selected as the target node. Accordingly, we have $R = \{v_1\}$, $H = \{-1\}$ and only $discovered[v_1] = true$. To start the stochastic BFS, we explore the in-neighbors of v_1 in the ascending order of node ID, that is $\{v_2, v_4, v_5\}$. Assume v_2 and v_4 are successfully activated in sequence, while v_5 is not activated. Then, the RR-set is updated as $R = \{v_1, v_2, v_4\}$ and the corresponding $H = \{-1, 0, 0\}$, since both v_2 and v_4 are activated by v_1 and the index of v_1 in R is 0. In addition, v_2 and v_4 are labeled as discovered. In the next round, we take out $R[1] = v_2$ to probe its neighbors v_3 and later v_6 which are both activated. Accordingly, we have $R = \{v_1, v_2, v_4, v_3, v_6\}$ and $H = \{-1, 0, 0, 1, 1\}$, since the index of their trigger node v_2 is 1 in R. Also, the labels of v_2 and v_4 become $true$. To proceed, we attempt to probe the neighbors of v_4 but to find v_3 and v_6 both have been discovered. Thus, we skip the subsequent two rounds. Meanwhile, all nodes in R have been probed and the generation process hence terminates. Resultantly, we obtain the corresponding BFS tree in Fig. 3(c), where each edge embodies the activation relationship.



(a) Initial Graph G (b) A possible RR-set (c) BFS Tree

(d) Residual Graph G' (e) Revised RR-set (f) BFS Tree

Figure 3: An example of RR-set generation and update.

RR-Set Update. Let us consider a polluted RR-set R. By recalling the stochastic BFS process that generates R, we could see that the nodes discovered after newly activated nodes (e.g., v) are not guaranteed to be active in the new residual graph without v, while the nodes discovered before v are actually not affected. Motivated by this observation, we strive to design an update mechanism that consists of three steps: (i) find out the first node (say v) that is newly active in the polluted RR-set R; (ii) find out the node (say u) that reversely discovers v; (iii) then skip edge (u, v) and redo the subsequent BFS process based on the current residual graph.

The detailed pseudo-code could be found in Alg. 5. To begin with, we attempt to find out the first node that becomes active in R (i.e., the active node with minimal index), from which the RR-set begins to be contaminated. Let it be $R[k]$. If $R[k]$ is exactly the first node in R, then we just invoke RR-Set-Gene to generate a fresh new RR-set as a replacement. Otherwise, we continue. By recalling the BFS process that induces R, we could note that the probing on nodes added to R before $R[k]$ are actually not affected. Thus, we could simply reserve these nodes and assign the first $k - 1$ elements in R and H to R' and H' directly. Accordingly, the first $k - 1$ nodes in R' are marked as discovered. Next, the treatment of the k-th node is a little more sophisticated. Let the k-th node $R[k]$ be v. From the procedure RR-set-Gene, we know it is $R[H[k]]$ that first activates

Algorithm 5: RR-Set-Update

```
Input: Residual graph $G_i = (V_i, E_i)$, RR-set $R$, observation
       $\psi_i$, corresponding predecessor array $H$.
Output: Revised RR set $R'$ and corresponding $H'$.
1  $R' \leftarrow \emptyset$; $H' \leftarrow \emptyset$; $discovered[u] \leftarrow false, \forall u \in V_i$;
2  $k \leftarrow \mathrm{argmin}_{0 \le k \le |R|} \psi_i(R[k]) = 1$;
3  if $k = 0$ then
4      return RR-Set-Gene($G_i$, $UniformSample(V_i)$);
5  end
6  $R' \leftarrow$ truncate $R$ to length $k$;
7  $H' \leftarrow$ truncate $H$ to length $k$;
8  for $w \in R'$ do
9      $discovered[w] \leftarrow true$;
10 end
11 $v \leftarrow R[k]$, $j \leftarrow H[k]$, $u \leftarrow R[j]$;
12 $pos \leftarrow$ find the $pos$ such that $N_{\mathrm{in}}(u, pos) = v$;
13 Probe($u, j, R', H', pos + 1$);
14 $j \leftarrow j + 1$;
15 while $j < |R'|$ do
16     $f \leftarrow R'[j]$;
17     Probe($f, j, R', H', 0$);
18     $j \leftarrow j + 1$;
19 end
20 return $R', H'$;
```

v during the BFS process. Let this node be u (see Line 11). Then, we find the position of v (i.e., pos) in the in-neighbors of u, and continue to probe u's neighbors from the starting point $pos + 1$ (i.e., the in-neighbor of u exactly after v), since previous neighbors are not affected and have been added to R'. After the subsequent neighbors have been probed, the discovered ones are added to R'. Then, we take out $R'[j+1]$ to continue probing and updating, until all nodes in R' have been probed.

Example 4.2. As shown in Fig. 3(d), node v_3 becomes active in some round (to illustrate, we mark v_3 with a gray color and represent its incident edges (v_3, v_2) and (v_3, v_4) with dotted lines.), yielding the residual graph G'. Then, the previous RR-set $R = \{v_1, v_2, v_4, v_3, v_6\}$ and $H = \{-1, 0, 0, 1, 1\}$ obtained on G are polluted by v_3. Note that v_3 is also the first active node in R, resulting in $k = 3$. Thus, we initialize R' as $\{v_1, v_2, v_4\}$ and H' as $\{-1, 0, 0\}$. Meanwhile, the labels of nodes in R' are set to be $true$. By looking up H, we know it is v_2 (i.e., $R[H[3]]$) that discovers v_3. Then, we locate the position of v_3 in $N_{\mathrm{in}}(v_2) = \{v_3, v_6\}$ as $pos = 0$. We skip v_3 and probe the next neighbor $v_6 = N_{\mathrm{in}}(v_2, 1)$ again. Assume v_6 is not discovered by the probing, then no element needs to be added to R' and H'. After finishing probing v_2's neighbors, we continue with the node right behind v_2 (i.e., node $v_4 = R'[H[3] + 1] = R'[2]$) whose neighbors v_5 and v_6 have not been probed in the residual graph yet. Assume both v_5 and v_6 are reversely activated. Then, the RR-set is updated as $R' = \{v_1, v_2, v_4, v_5, v_6\}$ and the corresponding predecessor array is $H' = \{-1, 0, 0, 2, 2\}$. The labels of v_5 and v_6 also become $true$. Finally, since all nodes have been discovered, no more probing will be carried out. Thus R is updated as $R' = \{v_1, v_2, v_4, v_5, v_6\}$.

Complexity. In Alg. 5, in the worst case, we scan the RR-set array R, and we may redo the stochastic BFS process at the first node. Therefore, the update has the same average cost of generating a new random RR-set. As to space consumption, we store the extra predecessor array, whose length is exactly the same as that of the RR-set array. Therefore, we need twice space of the vanilla version.

4.3 Correctness Analysis

In this section, we first give a theoretical analysis of the correctness of our update strategy and then discuss how to integrate it into the SelectAndVerify framework to improve efficiency.

Let $\mathcal{R}$ be a collection of random RR-sets obtained from the initial graph $G = (V, E)$. We further assume that $\mathcal{R}$ is independent of both the selected seed set and the subsequent realization observation. Given the current residual graph $G' = (V', E')$ in some round, we apply our update method (i.e., Alg. 5) to revise those polluted RR-sets in $\mathcal{R}$, leaving the clean RR-sets unchanged. The revised collection of RR-sets is denoted as $\mathcal{R}'$. For comparison, we also generate a collection of fresh random RR-sets $\mathcal{R}''$ directly based on graph G', with $|\mathcal{R}''| = |\mathcal{R}'|$. Lemma 4.3 indicates that the revised collection of RR-sets $\mathcal{R}'$ actually obtains the same distribution as a fresh collection of RR-sets $\mathcal{R}''$. Its proof is deferred to the appendix.

Lemma 4.3. *The set of RR-sets $\mathcal{R}'$ has the same distribution as $\mathcal{R}''$.*

To further demonstrate the correctness of the update technique, we may consider again the example of Fig. 2, trying to correct the estimation biases which can be found in B.1 in the appendix.

Recall that CASE can be viewed as a specific instance of the SelectAndVerify framework by instantiating the lower/upper bounds. Thus, in the following, we consider the general case of how to incorporate the update technique (i.e., Alg. 5) into the SelectAndVerify framework. Specifically, the new SelectAndVerify is as follows: *(i)* At the beginning, we invoke Alg. 5 to correct $\mathcal{R}_1$ and $\mathcal{R}_2$ which just took effect in the previous round; *(ii)* Then we use $\mathcal{R}_1$ to select a seed set S, and $\mathcal{R}_2$ to verify S's quality. If S satisfies $Q^l(S) \ge \alpha \cdot Q^u(S^o)$, we can terminate the process; *(iii)* If S fails to satisfy $Q^l(S) \ge \alpha \cdot Q^u(S^o)$, we generate new random RR-sets based on the current residual graph such that the size of $\mathcal{R}_1$ and $\mathcal{R}_2$ is double, and go back to Step (ii). *(iv)* When the number of newly generated RR-sets reaches $r_{\max}$ (a parameter required by the SelectAndVerify framework), we can select a seed set S based on only these new RR-sets, and then terminate the algorithm with S as the final answer.

In the following, we analyze the correctness of the SelectAndVerify framework equipped with our update approach. Before proceeding, we first analyze the collection $\mathcal{R}_1$ as we cannot apply Lemma 4.3. To elaborate, recall that we select the seed set based on the RR-sets in $\mathcal{R}_1$. This implies that the selected seed set is slightly dependent on the RR-sets in $\mathcal{R}_1$, which violates the independence prerequisite in Lemma 4.3. As a result, we are uncertain whether $\mathcal{R}_1$ will produce a good S. However, Lemma 4.4 allows us to be optimistic about the quality of the seed set selected by $\mathcal{R}_1$ due to the convergence property it exhibits. The proof of Lemma 4.4 can be found in the appendix.

Lemma 4.4. *Given a seed set S, the estimated influence of S based on $\mathcal{R}_1$ will converge to its expected influence.*

Now, we give the following lemma to conclude the correctness of SelectAndVerify integrated with our update strategy, whose proof is also deferred to the appendix.

Lemma 4.5. *The SelectAndVerify framework integrated with the update algorithm returns a seed set with the same approximation guarantee as the vanilla SelectAndVerify framework.*

Specifically, if SelectAndVerify is instantiated as our CASE, then it is the expected α-approximation guarantee. Note that the pseudocode of Alg. 3 (i.e., CASE) does not incorporate the update strategy. If the update strategy is applied, we need to make some minor modifications to ensure that up to $r_{\max}$ new RR-sets are generated in the worst-case scenario.

Time Complexity. The new SelectAndVerify framework first updates random RR-sets inherited from the previous round. As discussed in Section 4.2, the cost of updating an RR-set is bounded by that of generating a new one. Besides, the maximum number of RR-sets needed is still $r_{\max}$. Thus, the worst-case time complexity of our new SelectAndVerify framework only increases by a constant factor.

In terms of expected time complexity, due to the lack of independence in the set $\mathcal{R}_1$, the application of the Chernoff bound to derive its expected time complexity becomes unfeasible. To address this challenge, we propose a minor adjustment: we select the second seed set candidate, denoted as S', which relies entirely on the newly generated RR-sets. Subsequently, we assess the quality of both seed sets, S (chosen based on a combination of updated RR-sets and newly generated RR-sets) and S'. The algorithm terminates when either S or S' satisfies the stopping condition. As a result, the algorithm terminates no later than an algorithm devoid of the updating strategy (i.e., only S'). The extra cost incurred by the second seed set S' is dominated by the overhead associated with generating the newly generated RR-sets. Therefore, we deduce that the expected time complexity of the enhanced SelectAndVerify framework, incorporating the updating strategy, remains equivalent to that of the original version lacking updates.

5 RELATED WORKS

Domingos and Richardson [9] are pioneers in leveraging the word-of-mouth effect to promote the purchasing behavior in social networks. Then, Kempe et al. further formalize the well-known influence maximization problem, and invent the elegant greedy algorithm that provides constant theoretical guarantee $(1 - 1/e)$ [26]. Since then, a plethora of research [1, 3, 4, 13, 18, 20, 30, 48, 51, 53] has been conducted to advance the field from various perspectives.

It is worth noting that the budgeted influence maximization problem has attracted much attention due to its fine agreement with realistic scenarios. Notably, based on a greedy idea, Kuhller et al. [27] first establish an approximation ratio of $\frac{1}{2}(1 - 1/\sqrt{e})$ for the maximum coverage problem and claim the ratio could be refined to $(1 - 1/\sqrt{e})$. Further, [32] and [33] attempt to extend the idea to influence maximization and succeed the approximation ratio. However, a later work [50] refutes the results in [27] and therefore [32] [33] and revises it to be $\frac{1}{2}(1 - 1/e)$, which conforms with previous analysis of Krause and Guestrin [28]. Especially, the recent work [5] elevates the approximation ratio to $(1 - \beta)(1 - 1/e)$ and improves the experimental efficiency by over 10 times. Besides, a line of research advances previous works by combining the budgeted setting [19] [36] [47] [52].

As one may note, the above studies mainly belong to the non-adaptive category, where seeds are selected before diffusion. Differently, Golovin and Krause initialize the concept of adaptive submodularity in [14], where the seed selection is based on the observation of previous diffusion results, and prove the performance guarantee of adaptive greedy policy to be still $(1 - 1/e)$. Due to its sensible mechanism, the adaptive idea soon becomes widely applied in the research community [10] [21] [40] [45] [49]. Under the uniform cost model (i.e., each node has equal cost), Vaswani et al. [46] propose a node-level feedback model, different from the edge-level feedback considered in [14]. Vaswani et al. further show that if the time horizon is unbounded, the adaptive greedy strategy can achieve a $(1 - e^{-1/\alpha\gamma})$ approximation guarantee. However, their solution is unable to deal with skewed-cost scenarios.

Under the skewed cost model (i.e., the cost of each node may not be equal), Srinivasan [34] gives a policy with approximation guarantee, based on a differential analysis technique. However, it requires that the expected influence of each node can be exactly computed, which is impractical since it is a #P-hard problem [8]. Chen et al. studies the adaptive IM problem under the expected budgeted constraint in [7]. However, their algorithms do not offer any approximation guarantee for the strictly budgeted constraint investigated in this paper. Furthermore, their Theorem 7 claims an approximation guarantee for their greedy policy. However, the proof contains an evident reasoning gap. Specifically, they claim that Theorem 7 holds due to their Theorem 3, Lemma 1, and Theorem 6 [21]. However, Theorem 3 applies only to the non-adaptive budgeted case, and Theorem 6 from [21] applies only to the uniform-cost case. Thus, their combination fails to derive Theorem 7.

To be scalable to huge networks, both adaptive and non-adaptive algorithms demand efficient and precise estimation of user influences. In this regard, the seminal work [26] applies the Monte Carlo simulations for estimation, which however requires a time complexity of $\Omega(kmn{\cdot}\text{poly}(1/\epsilon))$ to maintain the $(1-1/e-\epsilon)$ approximation ratio. Then, Borgs et al. reduce the time to $O(k(m+n)\log^2 n/\epsilon^3)$ by proposing the reverse sampling technique in their ground-breaking work [6]. Tang et al. further present TIM/TIM$^+$ and IMM in [44] and [43] respectively, improving the running time to a near optimum of $O(k(m + n)\log n/\epsilon^2)$. Thereafter, another milestone is laid by Tang et al. [42] by proposing OPIM-C, which follows a SelectAndVerify structure and exhibits superior empirical effectiveness than previous algorithms. Recently, Guo et al. design SUBSIM to ameliorate the sampling of a single RR-set and the treatment of sizeable RR-sets [15] [16]. However, the above techniques are mainly devised for non-adaptive algorithms, which if directly applied to the adaptive case would incur evident estimation bias, as explained in Section 4.1. Thus, we develop the tailored incremental update technique for adaptive influence estimation, which could act as a general approach for subsequent research on adaptive IM.

Table 2: Datasets. ($K = 10^3, M = 10^6, B = 10^9$)

Name	n	m	Avg Degree	Type
Epinions	282K	2.3M	13.4	directed
DBLP	655K	2.0M	6.1	undirected
LiveJournal	4.8M	69.0M	28.5	directed
Twitter	41.7M	1.5B	35.3	directed

6 EXPERIMENTS

In this section, we experimentally evaluate our solution MAPLE and the incremental update strategy against alternatives. All experiments are conducted on a Linux server clocked at 2.3GHz, and all codes are implemented in C++ and compiled with -O3.

6.1 Experimental Settings

Algorithms. Recall that our MAPLE (see Alg. 2) is a probabilistic approach, which invokes CaGreedy with $\frac{\beta}{1+\beta}$ probability, and MIS with$\frac{1}{1+\beta}$ probability. To measure the expected influence of MAPLE, we run these two algorithms separately to get their influence spread and then report a weighted sum. Further, we add the prefix "I-" if the incremental update strategy is applied. When implementing MIS, we make a small modification to help it make full use of the budget. That is, after obtaining a single qualified node, we continue to select nodes with high influence (based on the maximum coverage criterion) until the budget runs out. Since the influence spread is monotonic with respect to the seed set, the performance guarantee of MAPLE still holds with the modified MIS. For comparison, we also include the state-of-the-art solution IMAGE [5] devised for the non-adaptive BIM problem. Further, the adaptive solution EptAIM [21] designed for the classic IM problem is also adapted as a baseline. Besides, for all algorithms, we use the subset sampling technique (SUBSIM) proposed in [15] to improve their efficiency. We also include the algorithm SAG [7] for comparison, which returns a seed set with total costs equal to B in expectation. Note that SAG possibly provides a seed set whose cost sum exceeds the budget B.

Datasets. The algorithms are run on four widely used datasets, available on SNAP [29]. Their statistic information is shown in Table 2. Especially, Epinions, DBLP, and Livejournal are used in [21]. The dataset Twitter is the largest one, containing billions of edges. Following [21], we generate 20 random realizations for each dataset and report the average influence spread of each algorithm.

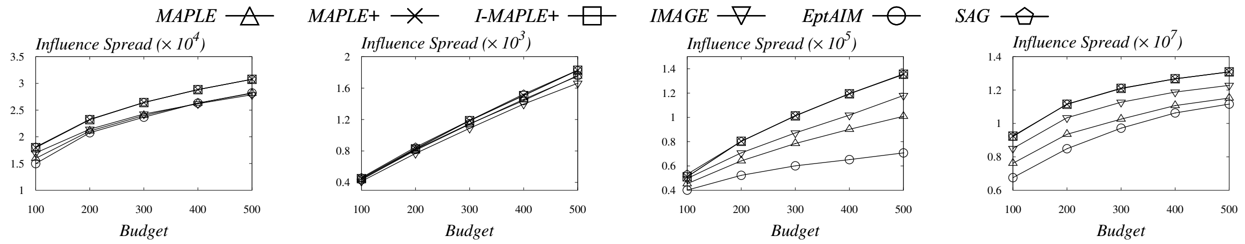
Cost Model. When evaluating our algorithms under the skewed cost setting, we consider two types of cost models: degree-based cost model and random cost model. *(i)* In the degree-based cost model, all nodes are endowed with a basic cost $c_{\text{basic}} = 1$ to reflect these scenarios, where users are often seeded by offering a piece of product in viral marketing. Following [5], we further impose an extra cost that is linear with the in-degree, to capture the fact that influential users with more followers (i.e., in-neighbors) tend to demand a higher reward $c(u) = c_{\text{basic}} + d_{\text{in}}(u) \cdot c_r$, where c_r mimics the payment for having each follower and $d_{\text{in}}(u)$ is the in-degree of u. We set c_r to be 0.01 just like [5]. The budget B is varied from 100 to 500. *(ii)* In the random cost model [32], the cost of each node is randomly selected from the range $[1.0, 10.0]$. It allows us to explore the performance of our algorithms under scenarios where a higher expected influence does not necessarily correspond to a higher cost.

Parameter Setting. First, following most existing IM works [10, 16, 42, 48], the diffusion probability of each edge (v, u) is set to be $\frac{1}{d_{\text{in}}(u)}$ under the IC model. Second, regarding the MAPLE framework, the approximation parameters α for CaGreedy and β for MIS are fixed to be $\alpha = 0.5$ and $\beta = 0.8$. The ratio α is given a relatively smaller value, since CaGreedy runs in a round-wise fashion and thus a tighter approximation guarantee would incur much higher computational cost. For EptAIM and SAG, we set their deviation $\epsilon = 0.5$. For the non-adaptive solution IMAGE, we set the deviation to be $\epsilon = 0.5$ and the failure probability to be $\delta = \frac{1}{n}$.

6.2 Experimental Performance

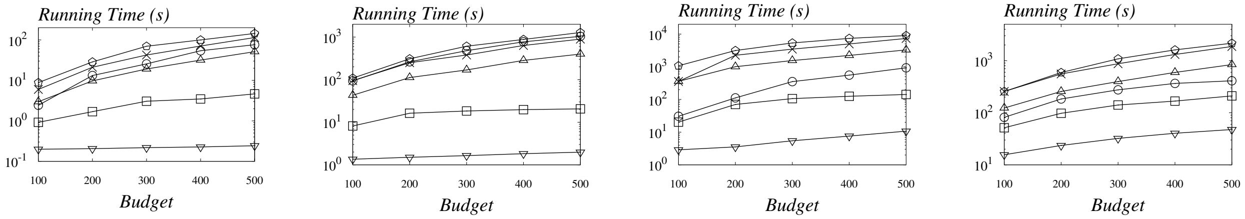
Influence Spread. We first evaluate the performance of influence spread for each algorithm under the degree-based cost model. The experimental results are shown in Fig. 4. The first observation is that MAPLE+ (i.e., the optimized version of MAPLE), and I-MAPLE+ (i.e., the MAPLE+ with the incremental update method), and SAG provide the best performance in terms of influence spread over all four datasets. In particular, MAPLE+ achieves around 16% larger spread compared with the second best algorithm IMAGE on Livejournal. The second observation is that MAPLE gives an inferior performance than IMAGE. To explain, recall that MAPLE is a weighted sum of CaGreedy and MIS. When MIS suffers from low influence spread, it would encumber the overall performance of MAPLE. As for EptAIM, since the seed node is simply selected based on the maximum influence criterion without considering the actual cost, its influence spread is relatively smaller just as expected. The third observation from Fig. 4 is that the curves of I-MAPLE+ are completely overlapped by that of MAPLE+ for all four datasets. It shows that our incremental update strategy can provide a selected seed set with identical quality, demonstrating its effectiveness. Finally, we observe that the curves of MAPLE+/I-MAPLE+ are visually consistent with those of SAG. This is expected as all three methods rely on the cost-aware greedy strategy for seed node selection. However, SAG lacks any performance guarantee for the strictly budgeted setting. Besides, it is likely to return an over-budget solution, where the cost of the seed set exceeds the given budget. For example, in this set of experiments, SAG returned over-budget results in 43%, 52%, 45%, and 55% invocations on datasets Epinions, DBLP, LiveJournal, and Twitter, respectively. In real scenarios of influencer marketing, it implies that the total expense spent by the influencer marketing agency exceeds the budget set by the customer, violating the purpose of budgeting.

Running Time. Next, we evaluate the efficiency of each algorithm in terms of running time under degree-based cost model. The experimental results are shown in Fig. 5. First, we can observe that all the adaptive algorithms (i.e., MAPLE, MAPLE+, I-MAPLE+, EptAIM, and SAG) take much more time than the non-adaptive algorithm IMAGE. It can be explained that adaptive algorithms need to run many rounds and naturally, take much more time to complete. The second observation is that I-MAPLE+ runs much faster than MAPLE+ over all four datasets, demonstrating the superiority of our incremental strategy in efficiency. Specifically, on dataset Livejournal (resp. DBLP), I-MAPLE+ achieves 50x (resp. 40x) speedup when budget



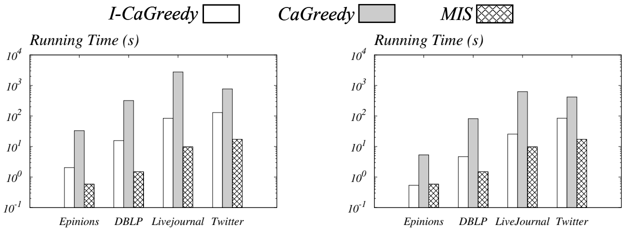
(a) Epinions (b) DBLP (c) LiveJournal (d) Twitter

Figure 4: Varying budget B: Average influence spread of each algorithm under degree-based cost model.



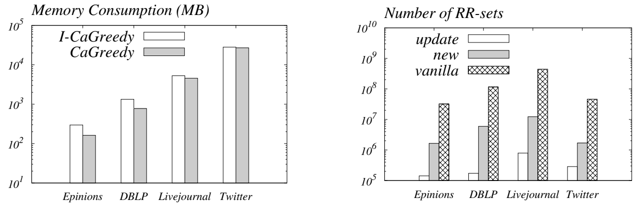
(a) Epinions (b) DBLP (c) LiveJournal (d) Twitter

Figure 5: Varying budget B: Running time of each algorithm under degree-based cost model.



(a) Parameter $\alpha = 0.5$ (b) Parameter $\alpha = 0.2$

Figure 6: Running time of sub-policies with $B = 200$.



(a) Memory Consumption (b) Statistics of RR-sets

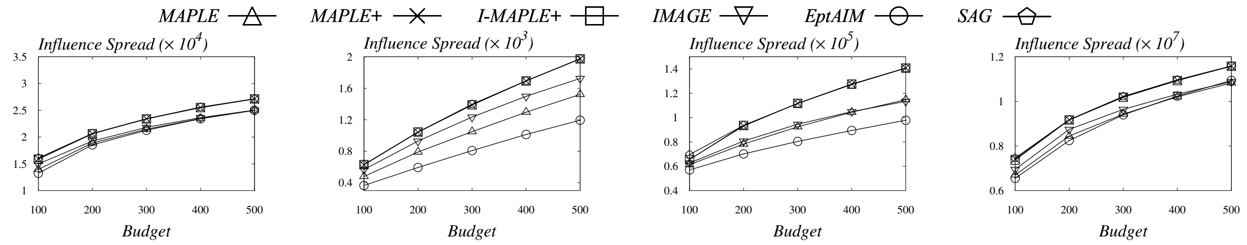
Figure 7: Memory consumption and statistics of RR-sets.

$B = 500$. Third, as can be seen, in dataset Twitter, the running time of EptAIM is close to that of I-MAPLE. This is because EptAIM tends to select the nodes with higher costs as seeds, and thus may run fewer rounds than I-MAPLE. In addition, compared with SAG, our I-MAPLE+ also consistently outperforms SAG by up to 60x on dataset LiveJournal with $B = 500$.

Furthermore, for a better illustration of our algorithms, we also report the running time of each sub-policy (i.e., CaGreedy, I-CaGreedy and MIS) under budget $B = 200$ in Fig. 6, by setting α as 0.2 and 0.5 respectively. As we observe, MIS is much more efficient than CaGreedy/I-CaGreedy when $\alpha = 0.5$, since MIS runs only one round. As α becomes 0.2 (smaller than 0.5), both CaGreedy and I-CaGreedy run faster due to looser approximation requirements.

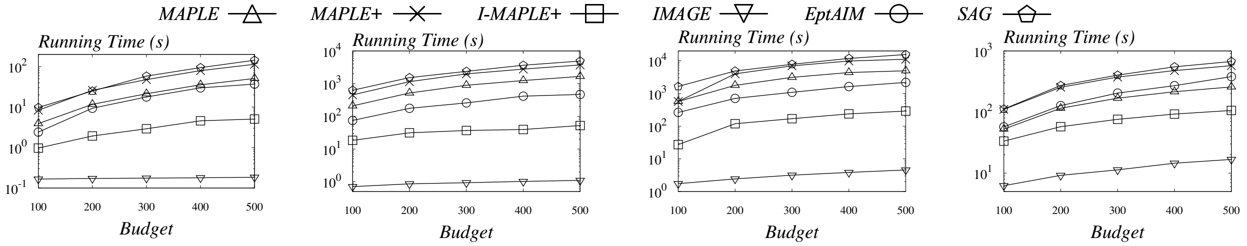
Random Cost Model. We further evaluate our algorithms under random cost model. The performance of our algorithms under this model is similar to that of the degree-based cost model. As shown in Fig. 8, when considering influence spread, both our MAPLE+/I-MAPLE+ and the baseline SAG consistently perform the best on all datasets. In particular, they have around 23% (resp. 14%) larger spread against the second best algorithm IMAGE on dataset LiveJournal (resp. DBLP) when $B = 500$. As for running time, our I-MAPLE+ algorithm requires significantly less running time compared to other adaptive algorithms, shown in Fig. 9. Specifically, I-MAPLE+ takes only 50 (resp. 300) seconds, while MAPLE+/SAG take more than 4000 (resp. 12000) seconds on DBLP (resp. Livejournal) with $B = 500$. It demonstrates its effectiveness and efficiency under the random cost model.

Uniform Cost. We also test our incremental update approach in the classic adaptive IM (AIM) problem, where all nodes have the same cost. As for the solution, we integrate the incremental update strategy with the state-of-the-art algorithm, EptAIM [21]. The augmented algorithm is denoted as I-EptAIM, since the incremental update strategy is applied. Following [21], the size of the seed set k varies from 100 to 500 and the approximate ratio ϵ is set to be 0.5. For comparison under this setting, the non-adaptive solution SUBSIM [15] is also included for evaluation. The influence spread and running time are shown in Fig. 10 and Fig. 11 respectively. First, we find from the two figures that our incremental update strategy could significantly reduce the running time by up to two orders (see dataset DBLP when $k = 500$), while still providing a seed node with equal influence spread. Compared with EptAIM/I-EptAIM, the non-adaptive solution SUBSIM is the most efficient one, which yields a seed set within one second in most cases, at the cost of low influence



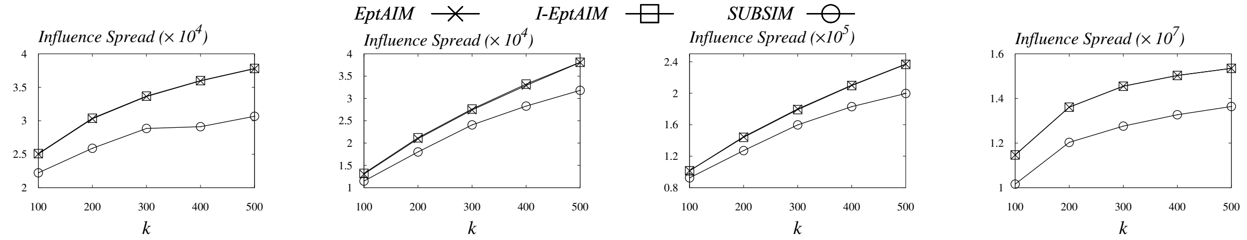
(a) Epinions (b) DBLP (c) LiveJournal (d) Twitter

Figure 8: Varying budget B: Average influence spread of each algorithm under random cost model.



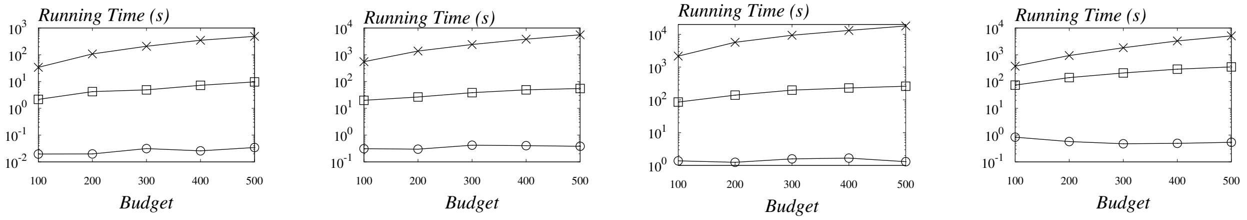
(a) Epinions (b) DBLP (c) LiveJournal (d) Twitter

Figure 9: Varying budget B: Running time of each algorithm under random cost model.



(a) Epinions (b) DBLP (c) LiveJournal (d) Twitter

Figure 10: Varying k: Average influence spread of each algorithm under uniform cost model.



(a) Epinions (b) DBLP (c) LiveJournal (d) Twitter

Figure 11: Varying k: Average running time of each algorithm under uniform cost model.

spread. On dataset Epinions (resp. Livejournal), ExpAim/I-EptAIM returns a seed set with 23% (resp. 18%) more influence spread than SUBSIM, since adaptive solutions could make wiser seed selection based on observation of previous diffusion results.

Memory Footprint. We further report the memory footprint of CaGreedy and I-CaGreedy when budget $B = 200$, shown in Fig. 7 (a). As can be seen, for all datasets, the memory footprint increment caused by our incremental update strategy is insignificant (less than twice). For example, I-CaGreedy costs only 5% more memory for dataset Twitter than CaGreedy. It can be explained that in addition to storing the generated RR-sets, CaGreedy also maintains some necessary information in the memory, such as the adjacent matrix of the input graph and the activated status of each node. It exhibits that our incremental update strategy is space-efficient.

Number of RR-sets. Recall that in our update strategy, we first revise the polluted RR-sets, and then select a seed for quality verification. If the quality is unsatisfactory, we will continue to generate new RR-sets to further refine the selection quality. As shown in

Fig. 7(b), we report the total number of polluted RR-sets and the newly generated RR-sets under the setting of budget $B = 200$ in I-CaGreedy. For comparison, we also report the total number of RR-sets generated by CaGreedy without the incremental update strategy, labeled as *vanilla*. From the figure, the number of newly generated RR-sets in I-CaGreedy is at least one order smaller than that in the vanilla CaGreedy. Besides, the number of polluted RR-sets is also significantly less than that of the newly generated RR-sets. These observations can visually explain why our incremental update approach is such efficient.

7 CONCLUSION

In this paper, we make the first attempt to solve the BIM problem under the adaptive setting and propose the first practical algorithm MAPLE by randomly switching between the cost-aware greedy selection and the highly influential node. It is proven to achieve an expected approximation guarantee. Based on RR-set estimation, the CASE algorithm is designed to return a seed node with approximation guarantee. We also develop an optimized MAPLE+ to further improve its practical spread performance. Moreover, we propose an incremental update technique to recycle the old RR-sets generated in the previous rounds, which significantly improves efficiency. Finally, extensive experiments are conducted to demonstrate that our algorithms outperform the baselines.

REFERENCES

[1] Cigdem Aslay, Francesco Bonchi, Laks VS Lakshmanan, and Wei Lu. 2017. Revenue Maximization in Incentivized Social Advertising. *PVLDB* 10, 11 (2017).

[2] Ashwinkumar Badanidiyuru and Jan Vondrák. 2014. Fast algorithms for maximizing submodular functions. In *SODA*. 1497–1514.

[3] Prithu Banerjee, Wei Chen, and Laks VS Lakshmanan. 2020. Maximizing social welfare in a competitive diffusion model. *PVLDB* 14, 4 (2020), 613–625.

[4] Shishir Bharathi, David Kempe, and Mahyar Salek. 2007. Competitive influence maximization in social networks. In *WINE*. 306–311.

[5] Song Bian, Qintian Guo, Sibo Wang, and Jeffrey Xu Yu. 2020. Efficient Algorithms for Budgeted Influence Maximization on Massive Social Networks. *PVLDB* 13, 9 (2020), 1498–1510.

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

[7] Tiantian Chen, Jianxiong Guo, and Weili Wu. 2022. Adaptive multi-feature budgeted profit maximization in social networks. *SNAM* 12, 1 (2022), 164.

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

[9] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *SIGKDD*. 57–66.

[10] Chen Feng, Luoyi Fu, Bo Jiang, Haisong Zhang, Xinbing Wang, Feilong Tang, and Guihai Chen. 2020. Neighborhood matters: Influence maximization in social networks with limited access. *IEEE Trans. Knowl. Data Eng.* (2020).

[11] Flaminjoy. 2022. Influencer Marketing Budgets. https://flaminjoy.com/blog/influencer-marketing-budget/.

[12] Forbes. 2021. One In Four Influencers Bought Fake Followers. https://www.forbes.com/sites/forbestechcouncil/2022/09/09/new-study-one-in-four-influencers-bought-fake-followers/?sh=3e3471056843.

[13] Sainyam Galhotra, Akhil Arora, and Shourya Roy. 2016. Holistic Influence Maximization: Combining Scalability and Efficiency with Opinion-Aware Models. In *SIGMOD*. 743–758.

[14] Daniel Golovin and Andreas Krause. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.* 42 (2011), 427–486.

[15] 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.

[16] Qintian Guo, Sibo Wang, Zhewei Wei, Wenqing Lin, and Jing Tang. 2022. Influence Maximization Revisited: Efficient Sampling with Bound Tightened. *ACM Trans. Database Syst.* 47, 3 (2022), 12:1–12:45.

[17] Kai Han, Keke Huang, Xiaokui Xiao, Jing Tang, Aixin Sun, and Xueyan Tang. 2018. Efficient Algorithms for Adaptive Influence Maximization. *PVLDB* 11, 9 (2018), 1029–1040.

[18] Kai Han, Chaoting Xu, Fei Gui, Shaojie Tang, He Huang, and Jun Luo. 2018. Discount allocation for revenue maximization in online social networks. In *MobiHoc*. 121–130.

[19] Shuo Han, Fuzhen Zhuang, Qing He, and Zhongzhi Shi. 2014. Balanced seed selection for budgeted influence maximization in social networks. In *PAKDD*. 65–77.

[20] Xinran He, Ke Xu, David Kempe, and Yan Liu. 2016. Learning influence functions from incomplete observations. *NeurIPS* 29 (2016).

[21] Keke Huang, Jing Tang, Kai Han, Xiaokui Xiao, Wei Chen, Aixin Sun, Xueyan Tang, and Andrew Lim. 2020. Efficient approximation algorithms for adaptive influence maximization. *VLDBJ* 29, 6 (2020), 1385–1406.

[22] Keke Huang, Sibo Wang, Glenn Bevilacqua, Xiaokui Xiao, and Laks VS Lakshmanan. 2017. Revisiting the stop-and-stare algorithms for influence maximization. *PVLDB* 10, 9 (2017), 913–924.

[23] IZEA. 2021. GUESS Eyewear. https://izea.com/resources/case-studies/guess-eyewear.

[24] IZEA. 2021. The Ultimate Guide to Influencer Marketing. https://izea.com/resources/influencer-marketing/.

[25] IZEA. 2021. Visit Tampa Bay. https://izea.com/resources/case-studies/visit-tampa-bay.

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

[27] Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Inf. Process. Lett.* 70, 1 (1999), 39–45.

[28] Andreas Krause and Carlos Guestrin. 2005. *A note on the budgeted maximization of submodular functions*. Citeseer.

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

[30] Yuchen Li, Ju Fan, George Ovchinnikov, and Panagiotis Karras. 2019. Maximizing multifaceted network influence. In *ICDE*. 446–457.

[31] LINQIA. 2019. How Much Budget Should You Spend on Influencer Marketing? https://www.linqia.com/insights/how-much-budget-should-you-spend-on-influencer-marketing/.

[32] Huy Nguyen and Rong Zheng. 2013. On budgeted influence maximization in social networks. *IEEE J. Sel. Areas Commun.* 31, 6 (2013), 1084–1094.

[33] Hung T Nguyen, Thang N Dinh, and My T Thai. 2016. Cost-aware targeted viral marketing in billion-scale networks. In *INFOCOM*. 1–9.

[34] Srinivasan Parthasarathy. 2020. Adaptive Submodular Maximization under Stochastic Item Costs. In *COLT*. 3133–3151.

[35] Binghui Peng and Wei Chen. 2019. Adaptive Influence Maximization with Myopic Feedback. *arXiv preprint arXiv:1905.11663* (2019).

[36] Canh V Pham, My T Thai, Hieu V Duong, Bao Q Bui, and Huan X Hoang. 2018. Maximizing misinformation restriction within time and budget constraints. *J. Combin. Optim.* 35, 4 (2018), 1202–1240.

[37] Lior Seeman and Yaron Singer. 2013. Adaptive seeding in social networks. In *FOCS*. 459–468.

[38] Chonggang Song, Wynne Hsu, and Mong Li Lee. 2016. Targeted influence maximization in social networks. In *CIKM*. 1683–1692.

[39] Shopify Staff. 2023. Influencer Marketing Prices: How Much Should You Pay. https://www.shopify.com/blog/influencer-pricing.

[40] Lichao Sun, Weiran Huang, Philip S Yu, and Wei Chen. 2018. Multi-round influence maximization. In *SIGKDD*. 2249–2258.

[41] Jing Tang, Keke Huang, Xiaokui Xiao, Laks V. S. Lakshmanan, Xueyan Tang, Aixin Sun, and Andrew Lim. 2019. Efficient Approximation Algorithms for Adaptive Seed Minimization. In *SIGMOD*. 1096–1113.

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

[43] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*. 1539–1554.

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

[45] Guangmo Tong, Weili Wu, Shaojie Tang, and Ding-Zhu Du. 2016. Adaptive influence maximization in dynamic social networks. *IEEE/ACM Trans. Netw.* 25, 1 (2016), 112–125.

[46] Sharan Vaswani and Laks VS Lakshmanan. 2016. Adaptive influence maximization in social networks: Why commit when you can adapt? *arXiv preprint arXiv:1604.08171* (2016).

[47] Ke Xu and Kai Han. 2018. Cost-Aware Targeted Viral Marketing with Time Constraints in Social Networks. In *CollaborateCom*. 75–91.

[48] Yu Yang, Xiangbo Mao, Jian Pei, and Xiaofei He. 2016. Continuous influence maximization: What discounts should we offer to social network users?. In *SIGMOD*. 727–741.

[49] Jing Yuan and Shao-Jie Tang. 2017. Adaptive discount allocation in social networks. In *MobiHoc*. 1–10.

[50] Ping Zhang, Zhifeng Bao, Yuchen Li, Guoliang Li, Yipeng Zhang, and Zhiyong Peng. 2018. Trajectory-driven influential billboard placement. In *SIGKDD*. 2748–2757.

[51] Yipeng Zhang, Yuchen Li, Zhifeng Bao, Baihua Zheng, and HV Jagadish. 2021. Minimizing the regret of an influence provider. In *SIGMOD*. 2115–2127.

[52] Yapu Zhang, Xianliang Yang, Suixiang Gao, and Wenguo Yang. 2019. Budgeted profit maximization under the multiple products independent cascade model. *Access* 7 (2019), 20040–20049.

[53] Yuqing Zhu, Jing Tang, and Xueyan Tang. 2020. Pricing influential nodes in online social networks. *PVLDB* 13, 10 (2020), 1614–1627.

A MISSING PROOFS

A.1 Proof of Theorem 3.4

Let τ represent the random sources caused by using the probabilistic-rounding strategy. We use $\Delta_\tau(\pi \mid \psi)$ to denote the expectation of $\Delta(\pi \mid \psi)$ over the randomness of the probabilistic-rounding method, i.e.,

$$\Delta_\tau(\pi \mid \psi) = \mathbb{E}_\tau[\Delta(\pi \mid \psi)].$$

Lemma A.1. *Assume we have made observation ψ. Let π^* be any policy. Then we have*

$$\Delta_\tau\big(\pi^* \mid \psi\big) \leq \mathbb{E}_\tau\big[c\big(\pi^* \mid \psi\big)\big] \cdot \max_v \frac{\Delta(v \mid \psi)}{c(v)},$$

where the expectation $\mathbb{E}_\tau[\cdot]$ is taken over the probabilistic-rounding randomness.

Proof. From Lemma 37 in [14], we have

$$\Delta\big(\pi^* \mid \psi\big) \leq c\big(\pi^* \mid \psi\big) \cdot \max_v \frac{\Delta(v \mid \psi)}{c(v)}.$$

Then, on both sides of the above inequality, by taking an expectation over τ, we could derive the lemma immediately. □

Here we introduce the concept of policy concatenation[14], which will be used in our proof. Given two policies π_1 and π_2, the *policy concatenation*, denoted as $\pi_1@\pi_2$, is a policy which first runs the policy π_1, and then runs π_2 from a fresh start, neglecting the information obtained from the invocation of π_1.

Lemma A.2. *Given budget B, for any adaptive policy π^* and any adaptive policy π_t truncated at t, for $0 \leq t \leq B$, we have*

$$\sigma_{avg}\big(\pi^*_B\big) - \sigma_{avg}(\pi_t) \leq B \cdot \mathbb{E}_\Phi\left[\max_v \frac{\Delta(v \mid \Psi(\pi_t, \Phi))}{c(v)}\right] \tag{6}$$

Proof. We first derive the following formula from the l.h.s. of Eq. (6)

$$\begin{aligned}
&\sigma_{\mathrm{avg}}\big(\pi^*_B\big) - \sigma_{\mathrm{avg}}(\pi_t) \leq \sigma_{\mathrm{avg}}\big(\pi^*_B@\pi_t\big) - \sigma_{\mathrm{avg}}(\pi_t)\\
&= \mathbb{E}_\Phi\big[\mathbb{E}_\tau\big[\sigma\big(S\big(\pi^*_B@\pi_t, \Phi\big), \Phi\big) - \sigma(S(\pi_t, \Phi), \Phi)\big]\big] \qquad (7)\\
&= \mathbb{E}_{\Psi_t}\big[\mathbb{E}_\tau\big[\Delta\big(\pi^*_B \mid \Psi_t\big)\big]\big],
\end{aligned}$$

where the first inequality is due to the adaptive submodularity of $\sigma(\cdot)$, while the next two equalities are due to the definition of $\sigma_{\mathrm{avg}}(\cdot)$ and $\Delta(\cdot)$. The notation Ψ_t is defined as a random partial realization obtained after running π_t.

Since the cost spends by π^*_B is at most B in expectation, by applying Lemma A.1 to formula (7) we could derive

$$\begin{aligned}
\sigma_{\mathrm{avg}}\big(\pi^*_B\big) - \sigma_{\mathrm{avg}}(\pi_t) &\leq B \cdot \mathbb{E}_{\Psi_t}\left[\max_v \frac{\Delta(v \mid \Psi_t)}{c(v)}\right]\\
&= B \cdot \mathbb{E}_\Phi\left[\max_v \frac{\Delta(v \mid \Psi(\pi_t, \Phi))}{c(v)}\right].
\end{aligned}$$

This completes the proof. □

Theorem A.3 (Theorem 3.4). *Given $\alpha \in [0, 1]$, budget $B \in \mathbb{Q}^+$, where $\mathbb{Q}^+$ is the set of positive rational numbers, and cost function $c : V \longrightarrow \mathbb{Q}^+$, let $\pi^c(\omega)$ be an expected α-approximate cost-aware greedy policy, and $\pi^*(\omega)$ be any policy. For any $0 \leq t \leq B$ $(t \in \mathbb{Q}^+)$, we have*

$$\mathbb{E}_\omega\big[\sigma_{avg}\big(\pi^c_t(\omega)\big)\big] \geq \Big(1 - e^{-\alpha t/B}\Big) \cdot \mathbb{E}_\omega\big[\sigma_{avg}\big(\pi^*_B(\omega)\big)\big].$$

Proof. First, assume we have a finite set of positive rational numbers, $\{q_1, q_2, \ldots, q_L\}$. By the definition of rational number, each q_i can be represented as a fraction $\frac{p}{q}$, where $p \in \mathbb{N}$ and $q \in \mathbb{N}$. It is not difficult to realize that there exists a scaling factor $M \in \mathbb{N}$ such that $q_i \cdot M \in \mathbb{N}$ for all $i \in \{1, 2, \ldots, L\}$. Therefore, we can scale $B, t, c(v)$ (where $v \in V$) to integers by multiplying a proper scaling factor. Without loss of generality, we consider $B \in \mathbb{N}, t \in \mathbb{N}$ and $c(v) \in \mathbb{N}$.

We can fix the randomness of ω such that we make π^* and π^c deterministic on which node to be selected as the next node (it should be noted that if the budget is not enough, we may admit it with a certain probability).

$$\begin{aligned}
&\mathbb{E}_\omega\big[\sigma_{\mathrm{avg}}\big(\pi^c_{t+1}(\omega)\big) - \sigma_{\mathrm{avg}}\big(\pi^c_t(\omega)\big)\big]\\
=&\mathbb{E}_\omega\big[\mathbb{E}_\Phi\big[\Delta_\tau\big(\pi^c_{t+1}(\omega) \mid \Psi_t(\omega, \Phi)\big)\big]\big] \qquad (8)\\
=&\mathbb{E}_\Phi\big[\mathbb{E}_\omega\big[\Delta_\tau\big(\pi^c_{t+1}(\omega) \mid \Psi_t(\omega, \Phi)\big)\big]\big],
\end{aligned}$$

where the second equality holds since the randomness of ω and the distribution of realization Φ are independent.

To facilitate subsequent analyses, we next fix the random realization Φ to an arbitrary explicit realization ϕ, with $\mathbb{P}(\Phi = \phi) = p_\phi$. Accordingly, the random partial realization (i.e., observation) $\Psi_t(\omega, \Phi)$ in each round becomes the deterministic observation ψ^ϕ_t, since the all edges' states are evident under a full realization ϕ. Then,

$$\begin{aligned}
&\mathbb{E}_\omega\left[\Delta_\tau\Big(\pi^c_{t+1}(\omega) \mid \psi^\phi_t\Big)\right]\\
&\geq \mathbb{E}_\omega\left[\alpha \cdot \max_v \frac{\Delta\Big(v \mid \psi^\phi_t\Big)}{c(v)}\right] = \alpha \cdot \mathbb{E}_\omega\left[\max_v \frac{\Delta\Big(v \mid \psi^\phi_t\Big)}{c(v)}\right], \qquad (9)
\end{aligned}$$

where the inequality can be derived from the definition of the expected α-approximate policy, and the fact that $c(v) \in \mathbb{N}$ for all $v \in V$. On this basis, we could further make derivation from Equation (8) as follows

$$\begin{aligned}
&\mathbb{E}_\omega\big[\sigma_{\mathrm{avg}}\big(\pi^c_{t+1}(\omega)\big) - \sigma_{\mathrm{avg}}\big(\pi^c_t(\omega)\big)\big]\\
&\geq \alpha \cdot \mathbb{E}_\Phi\left[\mathbb{E}_\omega\left[\max_v \frac{\Delta(v \mid \Psi_t(\omega, \Phi))}{c(v)}\right]\right]\\
&= \alpha \cdot \mathbb{E}_\omega\left[\mathbb{E}_\Phi\left[\max_v \frac{\Delta(v \mid \Psi_t(\omega, \Phi))}{c(v)}\right]\right]\\
&\geq \frac{\alpha}{B} \cdot \mathbb{E}_\omega\big[\sigma_{\mathrm{avg}}\big(\pi^*_B(\omega)\big) - \sigma_{\mathrm{avg}}\big(\pi^c_t(\omega)\big)\big]. \qquad (10)
\end{aligned}$$

Specifically, the first inequality holds since Equation (9) holds for arbitrary ϕ and the expectation is actually a linear combination over all possible realizations. The second inequality could be obtained by taking an expectation over ω on both sides of Inequality (6) in Lemma A.2.

Let $\Gamma_t = \mathbb{E}_\omega\Big[\sigma_{\mathrm{avg}}\Big(\pi^*_B(\omega)\Big)\Big] - \mathbb{E}_\omega\big[\sigma_{\mathrm{avg}}\big(\pi^c_t(\omega)\big)\big]$. Then Inequality (10) becomes $\Gamma_t - \Gamma_{t+1} \geq \frac{\alpha}{B} \cdot \Gamma_t$. Therefore,

$$\Gamma_t \leq (1 - \alpha/B) \cdot \Gamma_{t-1} \leq (1 - \alpha/B)^t \cdot \Gamma_0 \leq e^{-\alpha t/B} \cdot \Gamma_0 \tag{11}$$

where the last inequality is due to $1 - x \leq e^{-x}$ for any $x > 0$.

Besides, since π_0^c could not select any seed due to its zero budget, we have $\mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_0^c(\omega)\right)\right] = 0$. Then, plugging it into Eq. (11), we prove

$$\mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_t^c(\omega)\right)\right] \geq \left(1 - e^{-\alpha t/B}\right) \cdot \mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_B^*(\omega)\right)\right].$$

□

A.2 Proof of Lemma 3.6

For ease of notation, we write $\zeta_{G_i}(v) = \frac{\sigma_{\text{avg}}(\{v\})}{c(v)}$, where the symbol G_i is omitted when the context is clear. Further, let v^o be the node with maximal influence to cost ratio, namely $v^o = \text{argmax}_{v \in V_B} \frac{\sigma_{\text{avg}}(\{v\})}{c(v)}$.

Let E_1 and E_2 denote the following two events respectively:

$$\begin{aligned} &E_1 : Q^l(u^*) \geq \alpha \cdot Q^u(v^o), \\ &E_2 : r \geq r_{\max}. \end{aligned}$$

To prove the theorem, we only need to show that when either of E_1 or E_2 happens, ineuqality 5 will hold.

Event E_1. By Chernoff bound, we know that with $1 - \delta$ probability, the following inequality holds

$$\sigma_{\text{avg}}(\{u^*\}) \geq \frac{\Lambda^l_{\mathcal{R}_2}(u^*)}{r} \cdot n_i.$$

On the other hand, in the case of failure (happens with probability δ), the influence spread of u^* is bounded by $\sigma_{\text{avg}}(\{u^*\}) > 1$ (a seed activates itself at least). Thus, we could derive a lower bound:

$$\begin{aligned} \mathbb{E}_\omega\left[\zeta(u^*)\right] &= \mathbb{E}_\omega\left[\frac{\sigma_{\text{avg}}(\{u^*\})}{c(u^*)}\right] \geq (1-\delta) \cdot \mathbb{E}_\omega\left[\frac{\Lambda^l_{\mathcal{R}_2}(u^*)}{r} \cdot \frac{n_i}{c(u^*)}\right] \\ &\geq \mathbb{E}_\omega\left[\frac{\Lambda^l_{\mathcal{R}_2}(u^*) \cdot n_i}{r \cdot c(u^*)} - \frac{\delta \cdot n_i}{c(u^*)}\right] = \mathbb{E}_\omega\left[Q^l(u^*)\right]. \quad (12) \end{aligned}$$

where the second inequality is due to $\Lambda^l_{\mathcal{R}_2}(u^*)/r \leq 1$ by definition.

Besides, as $Q^u(v^o) = \max_{v \in V_B} \frac{\Lambda_{\mathcal{R}_2}(v)}{c(v)} \cdot \frac{n_i}{r}$, we have

$$\frac{\Lambda_{\mathcal{R}_2}(v^o)}{c(v^o)} \cdot \frac{n_i}{r} \leq \max_{v \in V_B} \frac{\Lambda_{\mathcal{R}_2}(v)}{c(v)} \cdot \frac{n_i}{r} = Q^u(v^o).$$

Therefore, when event E_1 happens, we have

$$Q^l(u^*) \geq \alpha \cdot Q^u(v^o) \geq \alpha \cdot \frac{\Lambda_{\mathcal{R}_2}(v^o)}{c(v^o)} \cdot \frac{n_i}{r}.$$

Taking expectation on both sides of the above inequality with regard to the randomness of ω and combining inequality (12), we get

$$\mathbb{E}_\omega\left[\zeta(u^*)\right] \geq \alpha \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}$$

where we use the fact that $\sigma_{\text{avg}}(\{v^o\}) = \mathbb{E}_\omega\left[\mathcal{R}_2(v^o) \cdot \frac{n_i}{r}\right]$. Thus, we obtain equation (5).

Event E_2. First we derive the following result: given $\epsilon_a > 0$, $\delta \in (0, 1)$, and

$$r_{\max} \geq \frac{(2 + 2\epsilon_a/3)n_i}{\epsilon_a^2} \cdot \left(\ln \frac{n_i}{\delta}\right),$$

if we generate a set $\mathcal{R}$ of random RR-sets with $|\mathcal{R}| \geq r_{\max}$, and select a seed node u^*, then with $1 - \delta$ probability it holds that,

$$\zeta(u^*) \geq \frac{1}{1 + \epsilon_a} \cdot \frac{\Lambda_{\mathcal{R}}(v^o) \cdot n_i}{|\mathcal{R}|} \cdot \frac{1}{c(v^o)}. \quad (13)$$

The derivation is as follows: according to Chernoff bound, for any node v, we have

$$\begin{aligned} &\mathbb{P}\left[\frac{\Lambda_{\mathcal{R}}(v) \cdot n_i}{|\mathcal{R}|} \geq (1 + \epsilon_a) \cdot \sigma_{\text{avg}}(\{v\})\right] \\ &\leq \exp\left(-\frac{\epsilon_a^2}{2 + 2\epsilon_a/3} \cdot \frac{|\mathcal{R}| \cdot \sigma_{\text{avg}}(\{v\})}{n_i}\right) \\ &\leq \delta/n_i. \end{aligned}$$

By the union bound (there exists n_i possible cases), we immediately get that with $1 - \delta$, it holds that

$$\sigma_{\text{avg}}(\{u^*\}) \geq \frac{1}{1 + \epsilon_a} \cdot \Lambda_{\mathcal{R}}(u^*) \cdot \frac{n_i}{|\mathcal{R}|}.$$

Then, divided by $c(u^*)$, it becomes

$$\begin{aligned} \zeta(u^*) = \frac{\sigma_{\text{avg}}(\{u^*\})}{c(u^*)} &\geq \frac{1}{1 + \epsilon_a} \cdot \frac{\Lambda_{\mathcal{R}}(u^*)}{c(u^*)} \cdot \frac{n_i}{|\mathcal{R}|} \\ &\geq \frac{1}{1 + \epsilon_a} \cdot \frac{\Lambda_{\mathcal{R}}(v^o)}{c(v^o)} \cdot \frac{n_i}{|\mathcal{R}|} \end{aligned}$$

where the second inequality comes from that we select u^* with maximal ratio of coverage to cost. Thus, we get Equation (13).

Let $1 - \epsilon = \frac{1}{1 + \epsilon_a}$, which yields $\epsilon_a = \frac{\epsilon}{1 - \epsilon}$. Equation (13) becomes

$$\zeta(u^*) \geq (1 - \epsilon) \cdot \frac{\Lambda_{\mathcal{R}}(v^o) \cdot n_i}{|\mathcal{R}|} \cdot \frac{1}{c(v^o)}.$$

On the other hand, recall that Equation (13) holds with probability $(1 - \delta)$. To make sure that

$$\mathbb{E}_\omega\left[\zeta(u^*)\right] \geq \alpha \cdot \mathbb{E}_\omega\left[\frac{\Lambda_{\mathcal{R}}(v^o)}{c(v^o)} \cdot \frac{n_i}{|\mathcal{R}|}\right] = \alpha \cdot \zeta(v^o),$$

where the second equality is due to $\zeta(v^o) = \mathbb{E}_\omega\left[\frac{\Lambda_{\mathcal{R}}(v^o)}{c(v^o)} \cdot \frac{n_i}{r}\right]$, we should have

$$(1 - \delta) \cdot (1 - \epsilon) = \alpha.$$

Therefore, $\epsilon = 1 - \frac{\alpha}{1 - \delta}$. Then, by plugging it into ϵ_a, we finally have $\epsilon_a = \frac{1 - \delta}{\alpha} - 1$. It completes the proof of Event E_2.

A.3 Proof of Lemma 3.7

In the proof of Theorem 5 in [21], as EPIC always finishes earlier than OPIM-C [42], the time complexity of EPIC is immediately derived based on the result of OPIM-C. Following a similar idea, we first give a slightly modified version of CASE, called m-CASE, which replaces the computation of $Q^u(v^o)$ with

$$Q^u(v^o) \leftarrow \max_{v \in V_B} \left(\sqrt{\Lambda_{\mathcal{R}_2}(v) + \frac{a}{2}} + \sqrt{\frac{a}{2}}\right)^2 \cdot \frac{n_i}{r \cdot c(v)}$$

It is easy to know the $Q^u(v^o)$ in m-CASE is larger than the $Q^u(v^o)$ in CASE. Therefore, when m-CASE stops, CASE must also stop. The relationship of CASE and m-CASE is analogous to that of EPIC and OPIM-C.

In the following, we will show the time complexity of m-CASE. the main idea of our proof follows the framework of Theorem 3 in [5].

Let $v_o = \arg\max_{v \in V_B} \frac{\sigma_{\text{avg}}(\{v\})}{c(v)}$. As the context is clear, we omit the subscript of n_i or m_i. First, we first prove two following lemmas.

Lemma A.4. *Let θ be the size of the set of random RR-sets. Given the error threshold, the failure probability δ_1, and*

$$\theta_1 = \frac{2 \cdot n \cdot \ln(1/\delta_1)}{\sigma_{avg}(\{v^o\}) \cdot \epsilon_1^2}$$

if $\theta \geq \theta_1$, then we have $\frac{n}{\theta} \cdot \frac{\Lambda(v^o)}{c(v^o)} \geq (1-\epsilon) \cdot \frac{\sigma_{avg}(\{v^o\})}{c(v^o)}$ with at least $1 - \delta_1$ probability.

Proof. According to Chernoff concentration bound,

$$\begin{aligned}
&\mathbb{P}\left[\frac{n}{\theta} \cdot \frac{\Lambda(v^o)}{c(v^o)} \leq (1-\epsilon_1) \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}\right] \\
&= \mathbb{P}\left[\frac{n}{\theta} \cdot \Lambda(v^o) \leq (1-\epsilon_1) \cdot \sigma_{\text{avg}}(\{v^o\})\right] \\
&\leq \mathbb{P}\left[\Lambda(v^o) - \frac{\theta}{n} \cdot \sigma_{\text{avg}}(\{v^o\}) \leq -\epsilon_1 \cdot \frac{\theta}{n} \cdot \sigma_{\text{avg}}(\{v^o\})\right] \\
&\leq \exp\left(-\frac{\epsilon_1^2 \cdot \theta \sigma_{\text{avg}}(\{v^o\})}{2n}\right) \leq \delta_1.
\end{aligned}$$

□

Let u^* be the node selected by the greedy strategy. When θ is sufficiently large, we have

$$\frac{n}{\theta} \cdot \frac{\Lambda(u^*)}{c(u^*)} \geq \frac{n}{\theta} \cdot \frac{\Lambda(v^o)}{c(v^o)} \geq (1-\epsilon_1) \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}. \tag{14}$$

Lemma A.5. *Given $\delta_2, \epsilon_1 \leq \epsilon$, $\tau_c = c_{\min}/c(v^o)$ and*

$$\theta_2 = \frac{2n \cdot (\ln n + \ln \frac{1}{\delta_2})}{\sigma_{avg}(\{v^o\}) \cdot \tau_c \cdot (\epsilon - \epsilon_1)^2},$$

if (14) holds and $\theta \geq \theta_2$, we have $\frac{\sigma_{avg}(\{u^*\})}{c(u^*)} \geq (1-\epsilon) \cdot \frac{\sigma_{avg}(\{v^o\})}{c(v^o)}$ with at least $1 - \delta_2$ probability.

Proof. Let u_b be an arbitrary node. We say u_b is bad, if $\frac{\sigma_{\text{avg}}(\{u_b\})}{c(u_b)} \leq (1-\epsilon) \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}$. Let $\epsilon_2 = \epsilon - \epsilon_1$. According to the Chernoff bound and the definition of bad u_b,

$$\begin{aligned}
&\mathbb{P}\left[\frac{n}{\theta} \cdot \frac{\Lambda(u_b)}{c(u_b)} - \frac{\sigma_{\text{avg}}(\{u_b\})}{c(u_b)} \geq \epsilon_2 \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}\right] \\
&= \mathbb{P}\left[\frac{n}{\theta} \cdot \Lambda(u_b) - \sigma_{\text{avg}}(\{u_b\}) \geq \epsilon_2 \cdot \frac{c(u_b)}{c(v^o)} \cdot \sigma_{\text{avg}}(\{v^o\})\right] \\
&\leq \mathbb{P}\left[\frac{n}{\theta} \cdot \Lambda(u_b) - \sigma_{\text{avg}}(\{u_b\}) \geq \epsilon_2 \cdot \tau_c \cdot \sigma_{\text{avg}}(\{v^o\})\right] \\
&\leq \exp\left(-\frac{\epsilon_2^2 \cdot \tau_c \cdot \sigma_{\text{avg}}(\{v^o\})}{2n \cdot (1-\epsilon) + \frac{2}{3}\epsilon_2 \cdot n}\right) \\
&\leq \exp\left(-\frac{\epsilon_2^2 \cdot \sigma_{\text{avg}}(\{v^o\})}{2n} \cdot \theta\right) \\
&\leq \delta_2/n.
\end{aligned}$$

As there exist at most n bad nodes, by union bound, we know the failure probability is δ_2. Therefore, with $1 - \delta_2$ probability, u^* is not a bad node, and thus,

$$\begin{aligned}
\frac{\sigma_{\text{avg}}(\{u^*\})}{c(u^*)} &\geq \frac{n}{\theta} \cdot \frac{\Lambda(u^*)}{c(u^*)} - \epsilon_2 \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)} \\
&\geq (1-\epsilon_1) \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)} - \epsilon_2 \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)} \\
&= (1-\epsilon) \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}.
\end{aligned}$$

where the second inequality is due to (14). □

According to Lemmas A.4 and A.5, when $\theta \geq \max(\theta_1, \theta_2)$, we have $\frac{\sigma_{\text{avg}}(\{u^*\})}{c(u^*)} \geq (1-\epsilon) \cdot \frac{\sigma_{\text{avg}}(\{v^o\})}{c(v^o)}$ with at least $1 - \delta_1 - \delta_2$ probability. Set $\delta_1 = \delta_2 = \frac{1}{2n}$ and let $\theta_1 \approx \theta_2$. With this setting, we can further have the following conclusion: Given the error parameter ϵ, and

$$\theta^* = \frac{4n \cdot \ln(6/n)}{\sigma_{\text{avg}}(\{v^o\}) \cdot \epsilon^2} \cdot \left(1 + \sqrt{\frac{c(v^o)}{c_{\min}}}\right)^2,$$

when $|\mathcal{R}| = \theta^*$, m-CASE terminates with $1 - \frac{1}{n}$ probability. According to this conclusion, it is easy to get when $|\mathcal{R}| = 2^i \cdot \theta^*$, the probability that m-CASE does not stop is at most $\frac{6}{(6n)^{2^i}}$. Therefore, we can derive that the expected number of RR-sets could be bounded by

$$\begin{aligned}
\mathbb{E}[|\mathcal{R}|] &\leq (1 - \frac{1}{n}) \cdot \theta^* + \sum_{i=1}^{\infty} 2^i \theta^* \cdot \frac{6}{(6n)^{2^i}} \\
&\leq \theta^* + (\theta^* + \theta^*/2 + \theta^*/2^2 + \theta^*/2^3 + \dots) \\
&\leq 3\theta^*.
\end{aligned}$$

Therefore, we conclude this lemma:

Lemma A.6. *The expected RR-sets number is $O\left(\frac{n \cdot \log n}{\sigma_{avg}(\{v^o\}) \cdot \epsilon^2} \cdot \frac{c(v^o)}{c_{\min}}\right)$ the algorithm m-CASE terminates.*

According to [43], the expected running cost is $\frac{m}{n} \cdot \mathbb{E}[\sigma_{\text{avg}}(\{v^*\})]$, where v^* is a randomly selected node with probabilities proportional to their in-degree. Therefore, $\frac{\mathbb{E}[\sigma_{\text{avg}}(\{v^*\}]}{c_{\max}} \leq \mathbb{E}[\frac{\sigma_{\text{avg}}(\{v^*\}}{c_(v^*)}] \leq \frac{\sigma_{\text{avg}}(\{v^o\}}{c_(v^o)}$. From Lemma A.6, we further derive that the time complexity of m-CASE is $O\left(\frac{(m+n) \cdot \log n}{\epsilon^2} \cdot \frac{c_{\max}}{c_{\min}}\right)$.

Finally, as aforementioned, CASE always stops no later than m-CASE as CASE has a looser upper bound $Q^u(v^o)$. Therefore, it completes the proof of Lemma 3.7.

A.4 Proof of Lemma 3.8

As the π^* in Theorem 3.4 can be any policy, surely its conclusion also holds for non-adaptive ones. Thus, by setting $t = B$ in Theorem 3.4, we have

$$\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_B^c(\omega))\right] \geq \left(1 - e^{-\alpha}\right) \cdot \sigma_{\text{avg}}(S_{\text{na}}) \geq \left(1 - e^{-\alpha}\right) \cdot \sigma^l(S_{\text{na}}) \tag{15}$$

where π_B^c is the probabilistic-rounding version of $\pi_{\leq B}^c(\omega)$.

On the other hand, compared with any strictly truncated policy, its probabilistic-rounding version may admit one more node with some probability. This induces

$$\begin{aligned}
\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_B^c(\omega))\right] &\leq \mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_{\leq B}^c(\omega)\right)\right] + \sigma_{\text{avg}}(\{v^o\}) \\
&\leq \mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi_{\leq B}^c(\omega)\right)\right] + \sigma^u(\{v^o\}). \qquad (16)
\end{aligned}$$

Combining Inequalities (15) and (16) and rearranging the terms, we complete the proof.

A.5 Proof of Theorem 3.9

When S_{na} satisfies $(1-e^{-\alpha})\cdot\sigma^l(S_{\text{na}}) \ge 2\sigma^u(\{v^o\})$, by Lemma 3.8, we immediately obtain $\mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi^c_{\le B}(\omega)\right)\right] \ge \sigma^u(\{v^o\})$. Then, Equation (16) becomes

$$\begin{aligned}\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi^c_B(\omega))\right] &\le \mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi^c_{\le B}(\omega))\right] + \sigma_{\text{avg}}(\{v^o\}) \\ &\le 2\cdot\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi^c_{\le B}(\omega))\right] \qquad (17)\end{aligned}$$

One the other hand, since MAPLE directly runs CaGreedy with probability one, we have $\mathbb{E}_\omega\left[\sigma_{\text{avg}}(\pi_{\le B}(\omega))\right] = \mathbb{E}_\omega\left[\sigma_{\text{avg}}\left(\pi^c_{\le B}(\omega)\right)\right]$. Then by Theorem 3.4, we can immediately prove the theorem.

A.6 Proof of Lemma 4.3

First, we fix node v as the target node. Let R' be a random RR-set starting from v on residual graph G'. Assume that the ordered edges we explore when generating R' is $\{e'_1, e'_2, \ldots, e'_r\}$; that is, we explore e'_i ahead of e'_{i+1}. Let s'_i be the status (i.e., live or blocked) of the i-th edge, where $s'_i = 1$ if it is live, otherwise $s'_i = 0$. We use the notation (e'_i, s'_i) to represent that the status of e'_i is s'_i. By the definition of IC model, we know that

$$\mathbb{P}[R' \text{ is sampled on } G'] = \prod_{i=1}^{r} \mathbb{P}[(e'_i, s'_i)].$$

We say edge (u, w) is a crossing edge if $w \in V'$ and $u \in V\backslash V'$ (namely, u has been activated in the previous rounds while w is still inactive). Now we sample a random RR-set R starting from target node v on graph G. When generating R, we explore these edges $\{e_1, e_2, e_3, \ldots\}$ with statuses $(s_1, s_2, s_3, \ldots)$. we assume after exploring k edges, we encounter the first crossing edge e_{k+1}. If we want to get the RR-set R' after the update algorithm, we should have: for $i = 1, 2, \ldots, k$, it satisfies that $e_i = e'_i$ and $s_i = s'_i$. Otherwise, after the updating step, we would never get R'. We say such an R is a candidate of R'. Thus,

$$\mathbb{P}[R \text{ is a candidate of } R'] = \prod_{i=1}^{k} \mathbb{P}[(e_i, s_i)] = \prod_{i=1}^{k} [(e'_i, s'_i)].$$

Let $\mathcal{E}$ denote the event that R becomes R' after the correcting phase. Then, for the edge e_{k+1}, we consider two cases:

CASE 1: it is blocked (i.e., $s_{k+1} = 0$). Here we assume there is only one crossing edge. We do not need to re-do the BFS process. As we finally get R', it indicates that $e_{i+1} = e'_i$ and $s_{i+1} = s'_i$ for $i = k, \ldots, r$. Therefore, the probability is

$$\begin{aligned}\mathbb{P}[\mathcal{E} \wedge (e_{k+1}, 0)] &= \prod_{i=1}^{r+1} \mathbb{P}[(e_i, s_i)] \\ &= \prod_{i=1}^{r} \mathbb{P}[(e'_i, s'_i)] \cdot \mathbb{P}[(e_{k+1}, 0)].\end{aligned}$$

Case 2: it is live (i.e., $s_{k+1} = 1$). In this case, we have to re-do the BFS process. To make R become R', it should have the same activating outcome as R' when re-doing the BFS process. Therefore, the probability is

$$\begin{aligned}\mathbb{P}[\mathcal{E} \wedge (e_{k+1}, 1)] &= \prod_{i=1}^{k} \mathbb{P}[(e_i, s_i)] \cdot \mathbb{P}[(e_{k+1}, 1)] \cdot \prod_{i=k+1}^{r} \mathbb{P}[(e'_i, s'_i)] \\ &= \prod_{i=1}^{r} \mathbb{P}[(e'_i, s'_i)] \cdot \mathbb{P}[(e_{k+1}, 1)].\end{aligned}$$

According to the law of total probability, we further get

$$\begin{aligned}\mathbb{P}[\mathcal{E}] &= \mathbb{P}[\mathcal{E} \wedge (e_{k+1}, 0)] + \mathbb{P}[\mathcal{E} \wedge (e_{k+1}, 1)] \\ &= \prod_{i=1}^{r} \mathbb{P}[(e'_i, s'_i)] = \mathbb{P}[R' \text{ is sampled on } G']\end{aligned}$$

where $\mathbb{P}[(e_{k+1}, 0)] + \mathbb{P}[(e_{k+1}, 1)] = 1$.

If there are multiple crossing edges, we can analyze each one individually in a similar manner as above to prove its correctness.

As for independence, because the RR-sets in $\mathcal{R}$ are independently generated, and the selected seed set and the subsequent realization observation, and the revising process are also independent of $\mathcal{R}$, we thus conclude that the RR-sets in $\mathcal{R}'$ are independent. Therefore, we have proved Lemma 4.3.

We can understand the correctness of this lemma from the perspective of RR-set construction. Let $\mathcal{A}$ be the set $\{v \mid v \in V\backslash V'\}$. Then with the rejection technique, we can use the topology of G to construct an RR-set R' for graph G' in the following manner: (i) first we uniformly sample a node $v \in V$ from graph G. If $v \in \mathcal{A}$, then we reject v and re-sample a new node until $v \notin \mathcal{A}$; (ii) Then we conduct a stochastic BFS from the target node v. When the current frontier successfully discovers a node u which $u \in \mathcal{A}$, it should reject u and continue to probe the next in-neighbor. The correctness of such an RR-set construction comes from the fact that each in-neighbor is probed independently. That is, we may do unnecessary works: explore an active in-neighbor w ($w \in \mathcal{A}$) of node u ($u \in V'$), and then reject w. However, in spite of wasting computational resources, it has no effect on the result of exploring other nodes in V'. From the above construction approach, we can consider the set $\mathcal{R}'$ to be unfinished work without rejection. What we need to do is to restart the BFS of a polluted RR-set from the step where it should have made a rejection. Therefore, our update strategy can achieve the goal of returning a set of random RR-sets for graph G'.

A.7 Proof of Lemma 4.4

Let $G = (V, E)$ (resp. $G' = (V', E')$) be the initial graph (resp. the current residual graph) and $S \subset V'$ be any seed set in V'.

Recall that $\mathcal{R}_1$ includes two parts: one is the RR-sets inherited from the previous rounds (denoted as $\mathcal{R}^h_1$) and the other is the newly generated RR-sets on G' (denoted as $\mathcal{R}^g_1$). Let n_h be the size of $\mathcal{R}^h_1$, namely $n_h = |\mathcal{R}^h_1|$, and n_g be the size of $\mathcal{R}^g_1$. Let $\sigma_{\text{avg}}(S)$ be the expected influence of S. Let $\hat{\sigma}_h(S)$ (resp. $\hat{\sigma}_g(S)$) be the estimated influence of S based on h (resp. $\mathcal{R}^g_1$). Assume $\hat{\sigma}_h(S) = \sigma_{\text{avg}}(S) + \Delta$. Then the estimated influence of S based on $\mathcal{R}_1$ is:

$$\hat{\sigma}_{\mathcal{R}_1}(S) = \frac{n_h}{n_h + n_g}\cdot\hat{\sigma}_h(S) + \frac{n_g}{n_h + n_g}\cdot\hat{\sigma}_g(S).$$

As n_g increases, $\hat{\sigma}_g(S) \to \sigma_{\text{avg}}(S)$. Thus,

$$\begin{aligned}\hat{\sigma}_{\mathcal{R}_1}(S) &\approx \frac{n_h}{n_h + n_g} \cdot \left[\sigma_{\text{avg}}(S) + \Delta\right] + \frac{n_g}{n_h + n_g} \cdot \sigma_{\text{avg}}(S) \\ &= \sigma_{\text{avg}}(S) + \frac{n_h}{n_h + n_g} \cdot \Delta \\ &\to \sigma_{\text{avg}}(S).\end{aligned}$$

In practice, because the RR-sets in $\mathcal{R}_1^h$ are generated and corrected independently, $\hat{\sigma}_h(S)$ tends to be close to $\sigma_{\text{avg}}(S)$. To explain, if the size of $\mathcal{R}_1^h$ is infinite, at this moment, the collection $\mathcal{R}_1^h$ can accurately capture its underlying distribution of influence propagation on the initial graph G, and the selected seed set is determined by the structure of G, rather than our sampled results. Then after updating $\mathcal{R}_1^h$ accordingly, the revised $\mathcal{R}_1^h$ can still accurately capture the distribution on the residual graph G'.

A.8 Proof of Lemma 4.5

For ease of expression, we use the notation $\mathcal{R}_1^h$ and $\mathcal{R}_2^h$ be the sets of RR-sets inherited from the previous invocation of SelectAndVerify (it should be pointed out that $\mathcal{R}_1^h$ and $\mathcal{R}_2^h$ include the unchanged RR-sets). We use $\mathcal{R}_1^g$ and $\mathcal{R}_2^g$ to denote the sets of newly generated RR-sets. Then, we can write $\mathcal{R}_1 = \mathcal{R}_1^h \cup \mathcal{R}_1^g$ and $\mathcal{R}_2 = \mathcal{R}_2^h \cup \mathcal{R}_2^g$.

Recall that in SelectAndVerify we utilize the set $\mathcal{R}_2$ to verify whether the selected seed set S satisfies α-approximation guarantee (Specifically for the algorithm CASE in this paper, it is expected α-approximation ratio). In order to accomplish this, it is essential that the influence estimation based on $\mathcal{R}_2$ is unbiased and the RR-sets in $\mathcal{R}_2$ are independent, which allows us to apply the Chernoff-like concentration bound. Lemma 4.3 establishes that $\mathcal{R}_2^h$ follows an identical distribution to that of a fresh collection of RR-sets. Thus, we can further conclude that $\mathcal{R}_2 = \mathcal{R}_2^h \cup \mathcal{R}_2^g$ also fulfill the aforementioned requirements and produce a correct verification result.

Next, we need to ensure that when the algorithm terminates, $\mathcal{R}_1$ can produce a seed set S that satisfies the verification criterion of $\mathcal{R}_2$. The convergence demonstrated in Lemma 4.4 implies that the estimated influence spread of a seed set closely approximates its true influence, thereby enabling us to identify a high-quality seed set (e.g., one with a high influence-to-cost ratio). If the selected seed set can pass the verification test of $\mathcal{R}_2$, we have successfully obtained a solution with a specified approximation guarantee, and the algorithm can terminate.

Stepping back for a moment, in the worst-case scenario, the seed set selected by $\mathcal{R}_1$ fails to satisfy the specified approximation guarantee. In such cases, we may generate up to $r_{\max}$ additional RR-sets (i.e., $|\mathcal{R}_1^g| = r_{\max}$). We can then utilize only the set $\mathcal{R}_1^g$ to obtain a seed set. This number of randomly independent RR-sets is sufficient to provide the desired approximation guarantee (e.g., as outlined in the correctness analysis of the CASE algorithm in Section 3.2). Therefore, our updated SelectAndVerify algorithm is capable of returning a seed set that satisfies the desired approximation guarantee, even in the worst-case scenario.

B EXAMPLE

Example B.1. Likewise v_1 is assumed to be activated, thus leaving RR-sets R_3 and R_4 polluted, while R_1 and R_2 are still clean and could be directly reused. To revise, let us first consider R_3 which exists with probability 0.25 and holds the predecessor array $H_3 = \{-1, 0\}$. Observing that $v_1 = R_3[1]$ is the first active node, we preserve $R_3[0]$ and re-probe its next neighbor $v_2 = N_{\text{in}}(R[H[0]], 1)$, following an ascending order of node indexes. With equal probability, the RR-set R_3 becomes $R_1 = \{v_3\}$ or $R_2 = \{v_3, v_2\}$, thus raising the probability of R_1 to 0.375 and R_2 to 0.25 respectively.

Regarding R_4, note that our technique asks the traversal of neighbors and the resultant RR-sets to be ordered. Thus, the previous RR-set R_4 actually includes two sub-categories: $R_{4a} = \{v_3, v_2, v_1\}$ with $H_{4a} = \{-1, 0, 1\}$ and $R_{4b} = \{v_3, v_1, v_2\}$ with $H_{4b} = \{-1, 0, 0\}$. Then, let us update R_{4a} first. After truncating R_{4a} at v_1, R'_{4a} becomes $\{v_3, v_2\}$ (i.e., R_2), which already contains all the nodes in the residual graph. Thus, no more probing is needed. Next, to revise R_{4b}, we begin with $R'_{4b} = \{v_3\}$ by truncation. By looking up H_{4b}, we know that the trigger node of v_1 is $v_3 = H_{4b}[0]$. Then, we proceed to re-probe v_3's next neighbor v_2 by the ascending index order, resulting in $R'_{4b} = R_1 = \{v_3\}$ and $R'_{4b} = R_2 = \{v_3, v_2\}$ with equal probability. Thus after revision, all R_{4a} together with half of R_{4b} will be added to R_2, and the other half of R_{4b} will be added to R_1.

To summarize, recall that R_{4a} and R_{4b} exist with probability 0.125 and 0.25 respectively. It is easy to see that the probabilities of both $R_1 = \{v_3\}$ and $R_2 = \{v_3, v_2\}$ are raised to 0.5. As a result, node v_2 covers all R_2 and thus influences v_3 with probability 0.5, coincident with the activation probability $p(v_2, v_3) = 0.5$ in Fig. 2.