Efficient and Effective Algorithms for Revenue Maximization in Social Advertising

Kai Han
School of Computer Science and Technology / SuZhou Research Institute, University of Science and Technology of China
hankai@ustc.edu.cn

Benwei Wu
School of Computer Science and Technology, University of Science and Technology of China
wubenwei@mail.ustc.edu.cn

Jing Tang
School of Computing, National University of Singapore
isejtang@nus.edu.sg

Shuang Cui
School of Computer Science and Technology, University of Science and Technology of China
lakers@mail.ustc.edu.cn

Cigdem Aslay
Department of Computer Science, Aarhus University
cigdem@cs.au.dk

Laks V. S. Lakshmanan
Department of Computer Science, University of British Columbia
laks@cs.ubc.ca

ABSTRACT

We consider the revenue maximization problem in social advertising, where a social network platform owner needs to select seed users for a group of advertisers, each with a payment budget, such that the total expected revenue that the owner gains from the advertisers by propagating their ads in the network is maximized. Previous studies on this problem show that it is intractable and present approximation algorithms. We revisit this problem from a fresh perspective and develop novel efficient approximation algorithms, both under the setting where an exact influence oracle is assumed and under one where this assumption is relaxed. Our approximation ratios significantly improve upon the previous ones. Furthermore, we empirically show, using extensive experiments on four datasets, that our algorithms considerably outperform the existing methods on both the solution quality and computation efficiency.

ACM Reference Format:
Kai Han, Benwei Wu, Jing Tang, Shuang Cui, Cigdem Aslay, and Laks V. S. Lakshmanan. 2021. Efficient and Effective Algorithms for Revenue Maximization in Social Advertising. In *Proceedings of the 2021 International Conference on Management ofData (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3448016.3459243

1 INTRODUCTION

With the proliferation of Online Social Networks (OSNs) such as Facebook and Twitter, there emerge great opportunities for social network platform owners and advertisers to gain revenue through placing advertisements on OSNs. The availability of rich information in OSNs, such as user profiles, shared posts, and user behavioral features, brings a tremendous opportunity for personalized advertising, while the interactions between OSN users make it possible for advertisers to propagate their marketing messages to a large audience in short time. Due to these advantages, the paradigm of *social advertising* on OSNs has aroused great interest from both academia and industry. For example, statistics on social advertising across companies worldwide, published in a Hootsuite blogpost[1], include interesting trends, such as "27% of internet users say they find new products and brands through paid social ads".

Typically, social advertising is implemented by showing "promoted posts" in the news feed of OSN users, and these promoted posts can take various forms such as images, videos, and hyperlinks to advertisements. After a user sees a promoted post, she may react to it by performing a social action in the form of *comment*, *like*, or *reshare*. Once a user performs a social action, it is counted as an engagement with the advertisement, and the advertiser would pay a unit amount to the OSN platform owner for the engagement. This is a typical marketing paradigm known as Cost-Per-Engagement (CPE) advertising, where advertisers only pay when users engage with the advertisements.

As the users in OSNs often influence each other based on their social affinity, the promoted posts could propagate in the network as a result of the users' social actions.[2] Therefore, a large number of users could eventually engage with the advertisement through influence propagation, even when only a small number of "seed users" initially engage with the promoted posts inserted into their feed by the platform. Based on the above observation, it is possible to boost the revenue of social advertising by intelligently selecting a few most "influential" seed users for initial endorsements of the advertisements such that the total engagements through influence propagation, and hence the revenue of social advertising, are maximized. In this paradigm, seed users should be properly incentivized by the advertiser to ensure that they are actually engaged with the advertisement. In fact, the paradigm of "incentivized multi-hop social advertising" described above has already been implemented or experimented with in several major companies. For example,

SIGMOD '21, June 20–25, 2021, Virtual Event,China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3459243

[1] https://blog.hootsuite.com/social-media-advertising-stats/.
[2] It is reported that the promoted posts in Tumblr are reposted more than ten thousand times on average [5].

Youtube and Twitch currently pay content creators (influencers) selected as seeds, a portion of the revenue from a video ad included in the content [42, 55, 63]. In May 2020, Instagram rolled out a new feature, IGTV, providing similar monetization opportunities for influencers: IGTV ads appear when users click on IGTV videos in their feed. In these platforms, user actions (watching/clicking) are visible to their peers and the content propagates to the peers, and with them the ads. Instagram decides which ad will appear in which content, and shares 55% of the IGTV ad revenue with the content creator [50]. Similar situations also appear in FaceBook Stories, a social media application where users can share photos/videos with their friends, allowing the associated advertisements to naturally propagate virally in multi-hops [41, 54]. Due to a lot of successful stories on multi-hop advertising (a.k.a. "viral marketing" or "viral advertising") [3], it also has aroused significant interest from researchers in the area of advertising [22, 25, 37, 53].

In this work, we consider a social advertising scenario where there is a social networking platform owner (referred as the *host*) and a set of h advertisers who need social advertising service provided by the host. Each advertiser $i \in \{1, \ldots, h\}$ needs to propagate an advertisement in the social network and has a *budget* B_i to pay (i) the host for total engagements with their ad and (ii) the seed users for incentivizing them. As the products associated with different ads could be competitive, we assume that a seed user can endorse at most one ad. In such a scenario, the host is faced with a *revenue maximization* problem, i.e., how to select the seed users for each advertiser under the constraints described above, such that the revenue gained by the host is maximized.

Previous Work and Limitations. We discuss the related work on computational and social advertising and other related topics in Section 6 and provide here a brief comparison with the work that is most related to ours. Aslay et al. [5] studied revenue maximization for social advertising, and to our our knowledge, theirs is the only work that studies the problem within the "*incentivized*" social advertising framework, where seed users are paid monetary incentives. They study the problem under the *Topic-aware Independent Cascade (TIC)* model [9] as we do in our work, allowing ad-specific IC propagation parameters on each edge. They show that the revenue maximization problem in incentivized advertising corresponds to monotone[3] submodular[4] function maximization subject to a partition matroid constraint on the ads-to-seeds allocation, and submodular knapsack constraints on advertisers' budgets, generalizing the special case of the problem with a single submodular knapsack constraint [43, 44]. As in [43, 44], they propose cost-agnostic and a cost-sensitive greedy algorithms with provable approximation guarantees.

The approximation ratios of the above algorithms depend on the input social network instance and could be arbitrarily small, which can hurt the quality of the approximation achieved. Clearly, an approximation ratio that does not depend on the network instance, if possible, is more desirable. Second, the manner in which budget feasibility is ensured in their algorithms appeals to upper bounds on the expected spread, since it is #P-hard to compute the expected spread exactly. This has the consequence of making their algorithm "conservative", in that the seed allocation provided by the algorithms may end up *under-utilizing* the budget. Third, the experimental results in [5] reveal that the computational overheads of their algorithms can be significant. More detailed discussions on the results in [5] can be found in Section 2.2.

Contributions. Motivated by the limitations of the existing studies, we propose new approximation algorithms for the revenue maximization problem in social advertising that provide significantly improved approximation ratios, which are *independent of the network instance*. Furthermore, our algorithms are more efficient and achieve better revenue thanks to a combination of improved approximation guarantees and a bicriteria approximation strategy for better utilizing given advertiser budgets. We further elaborate on the need for bicriteria approximation and how it can be managed in practice in Sections 2.2 and 4.3.

More specifically, our major contributions include the following. First, we propose (Section 3) approximation algorithms with provable performance guarantees under the assumption that there is an *influence spread oracle*, which returns the exact influence spread of any set of seed nodes. Our algorithms are based on several novel methods combining a greedy node-selection strategy and binary search, which fully exploit the special structure of the revenue maximization problem. The approximation ratio λ of our algorithms is *independent* of the input social network and, depending on the number h of advertisers, is characterized as follows:

$$\lambda = \begin{cases} 1/3, & \text{if } h = 1; \\ \frac{1}{2(h+1)(1+\tau)}, & \text{if } h \in \{2, 3\}; \\ \frac{1}{(h+6)(1+\tau)}, & \text{if } h \geq 4. \end{cases} \quad (1)$$

Here τ is any number in $(0, 1)$, which reflects the trade-off between accuracy and efficiency of our revenue maximizing algorithm. Compared to the network-dependent approximation ratios proposed in [5] which could be arbitrarily bad, our approximation ratio λ is essentially a *constant* for a given h. We believe this improvement is highly-nontrivial and theoretically interesting.

Second, we extend our algorithms (Section 4) to the practical case where there is no exact influence spread oracle[5], by using novel adaptation of the notion of Reverse-Reachable Sets proposed by Borgs et al. [12]. We prove that our algorithms can achieve a $\lambda - \epsilon$ approximation ratio under the relaxed budget constraint of $(1 + \varrho)B_i$ for each advertiser i with high probability, where λ is the approximation ratio shown above, ϵ is any number in $(0, \lambda)$, and ϱ is any number in $(0, 1)$, which controls how much the budget is overshot. We discuss how budget overshoot can be managed in practice in Section 2.2.1.

Third, we evaluate the efficiency and effectiveness of our algorithms with extensive experiments on 4 social networks containing up to ~69M edges (Sec. 5). The experimental results show that our algorithms significantly outperform the algorithms in [5] on both the processing time and the achieved revenue under TIC model.

We provide the necessary preliminaries in the next section and formally define the problem studied. Section 6 discusses related work. Section 7 summarizes the paper and discusses interesting directions for future work. Additional proofs of some technical

[3] $f(S) \leq f(T)$ whenever $S \subseteq T \subseteq \mathcal{U}$.
[4] $f(T \cup \{x\}) - f(T) \leq f(S \cup \{x\}) - f(S)$ whenever $S \subseteq T \subseteq \mathcal{U}$ for $x \in \mathcal{U} \setminus T$.

[5] Computing the exact influence spread is #P-hard [17].

results and experimental results, omitted for lack of space, can be found in [36].

2 PRELIMINARIES

2.1 Problem Statement

Following the social advertising model in [5], we assume that there exist a set of h *advertisers* and a *host*, which is the owner of a social networking platform. The host owns a social network represented as a directed graph $G = (V, E)$, where V and E denote the sets of nodes (i.e., users) and edges in G respectively, with $|V| = n$ and $|E| = m$. Each advertiser i provides the host with an ad i, and the host is responsible to select a set of *seed users* $S_i \subseteq V$ to endorse ad i. It is assumed that each node $u \in V$ has a cost $c_i(u)$ to be "activated" to endorse ad i; an influence propagation process will be triggered to activate more nodes in V after the seed users in S_i are activated. Moreover, each activated node in the influence propagation process would bring revenue to the host as it engages with ad i. After the influence propagation process ends, advertiser i should pay an amount of money for:

(1) The incentive cost of activating the users in S_i, i.e., $c_i(S_i) = \sum_{u \in S_i} c_i(u)$; this amount is paid to the seed users in S_i.
(2) A cost-per-engagement amount $cpe(i)$ for each engagement with ad i during the influence propagation process as described above; this amount is paid to the host for its service.

Discussion. In our problem, we assume that the value of $cpe(i)$ is agreed upon between advertiser i and the host for each $i \in [h]$. This could happen in the scenarios, e.g., the host posts $cpe(i)$ to advertiser i as a "take-it-or-leave-it" price according to prior marketing studies on the advertised product, and advertiser i clearly would only accept this price if $cpe(i)$ is less than her/his "value-per-engagement", resulting in no negative utility of any part. We note that some excellent studies [33] consider a scenario where $cpe(i)$ is unknown and is determined by truthful auction mechanisms. The auction problem considered in these studies is orthogonal to ours.

We adopt a general model in which seed node costs can be any positive number obtained by any existing pricing strategy for social networks. For example, a simple strategy prices nodes based on their number of followers.[6] As another example, a recent study [72] proposed another pricing strategy based on the expected influence gains of candidate seeds.

Influence Propagation Model. We adopt the *Topic-aware Independent Cascade* (TIC) model proposed in [9] to characterize the influence propagation process for each ad i, as described below. At first, the set of "seed nodes" in S_i are activated at time 0. Afterwards, each node u newly activated at time $t-1$ has a single chance to activate each of its inactive out-neighbors v at time t, succeeding with probability $p^i_{(u,v)}$. The expected number of total activated nodes when the influence propagation ends is denoted by $\sigma_i(S_i)$ and is called the (expected) spread of S_i.

The activation probability $p^i_{(u,v)}$ associated with each edge $(u,v) \in E$ under the TIC model is defined as follows. Assume that there exist L latent *topics* for ads and users' interests, and there is a hidden random variable Z ranging over the L topics. The TIC model then maps ad i to a distribution $\phi_i(\cdot)$ over the L latent topics with $\phi_i(z) = \Pr[Z = z \mid i]$ and $\sum_{z=1}^{L} \phi_i(z) = 1$. The influence propagation in social advertising can be topic-dependent, i.e., user u's influence on user v may depend on the topic of the ad that is being propagated. In the TIC model, the probability that u can activate v for ad i (i.e., $p^i_{(u,v)}$) is defined as $p^i_{(u,v)} = \sum_{z=1}^{L} \phi_i(z) \cdot \hat{p}^z_{(u,v)}$, where $\hat{p}^z_{(u,v)}$ is the probability that u can activate v under latent topic z.

The Revenue Maximization (RM) Problem. Following the social advertising model, the host can gain an expected revenue of $\pi_i(S_i) = cpe(i) \cdot \sigma_i(S_i)$ from advertiser i, and the *total expected revenue* of the host is $\sum_{i \in [h]} \pi_i(S_i)$. The Revenue Maximization (RM) Problem aims to maximize this total expected revenue under the following constraints: (1) each advertiser i has a budget B_i for its total social ad spend, i.e., the total amount paid to the host and to the seed users in S_i; (2) each user in V can endorse at most one ad within a certain time window.[7] Formally, the RM problem is defined as follows.

Definition 2.1. The *Revenue Maximization* (RM) problem for social advertising aims to identify an optimal solution $\vec{S} = (S_1, \ldots, S_h)$ to the following optimization problem:

$$\begin{aligned} \textbf{Maximize} \quad & \sum\nolimits_{i \in [h]} \pi_i(S_i) \\ \textbf{s.t.} \quad & \pi_i(S_i) + c_i(S_i) \leq B_i, \ \forall i \in [h], \\ & S_i \cap S_j = \emptyset, \ i \neq j, \ \forall i, j \in [h]. \end{aligned}$$

It is well-known that the influence spread function $\sigma_i(\cdot)$ is monotone and submodular under the TIC model [9], so the revenue function $\pi_i(\cdot)$ for every $i \in [h]$ is also monotone and submodular. Aslay et al. [5] have shown that the RM problem is NP-hard.

We now introduce some notations. We use $\vec{S}$ to represent an allocation, i.e., a list of sets $(S_1, \ldots, S_h)$, and use $\vec{O} = (O_1, \ldots, O_h)$ to represent an optimal solution to the revenue maximization problem. We abuse these notations slightly by using $\vec{S}$ to also represent the set $\{(u,i) \mid u \in S_i \wedge i \in [h]\}$ (and also abuse $\vec{O}$ similarly), as these representations are essentially equivalent. For any set $M \subseteq V \times [h]$, we define $\pi(M) = \sum_{i \in [h]} \pi_i(M_i)$ where $M_i = \{u \in V \mid (u,i) \in M\}$, and we define $\text{OPT} = \pi(\vec{O})$. Furthermore, for any set function $f(\cdot)$, we use $f(X \mid Y) = f(X \cup Y) - f(Y)$ to denote the *marginal gain* of X with respect to Y. For example, we have $\pi_i(u \mid S_i) = \pi_i(S_i \cup \{u\}) - \pi_i(S_i)$ and $\pi((v,j) \mid \vec{S}) = \pi_j(v \mid S_j)$. Finally, we use $\zeta_i(u \mid S_i)$ to denote the *marginal rate* of node u upon seed set S_i for advertiser i, defined as the ratio of the marginal gain in revenue to the marginal gain in payment, i.e.,

$$\zeta_i(u \mid S_i) = \frac{\pi_i(u \mid S_i)}{c_i(u) + \pi_i(u \mid S_i)}. \tag{2}$$

2.2 Existing Solutions

To the best of our knowledge, only Aslay et al. [5] have addressed the revenue maximization problem in this framework. They show that the RM problem is essentially a *submodular maximization problem with a partition matroid and multiple submodular knapsack constraints*, and propose two approximation algorithms—Cost-Agnostic Greedy (i.e., CA-Greedy) and Cost-Sensitive Greedy (i.e.,

[6] Klear's survey [26] shows that brands pay, on average, 114 dollars for each video post on Instagram to nano-influencers (500∼5K followers) and 775 dollars per video to more powerful users (30K∼50K followers).

[7] Note that limiting the ads endorsed by a seed can increase the credibility for followers. The same constraint has been widely adopted to avoid undesirable situations, e.g., the same celebrity endorsing Nike and Adidas at the same time [5, 14, 33, 56].

CS-Greedy). Both algorithms iteratively select seed nodes under the budget constraint. In each iteration, CA-Greedy (resp. CS-Greedy) greedily selects an element (u, i) such that the marginal gain $\pi_i(u \mid S_i)$ (resp. the marginal rate $\zeta_i(u \mid S_i)$) is maximized.

Aslay et al. [5] prove that the CS-Greedy algorithm has an approximation ratio of

$$1 - \frac{R \cdot \rho_{max}}{R \cdot \rho_{max} + (1 - \max_{i \in [h]} \kappa_i)\rho_{min}}, \tag{3}$$

and that the approximation ratio of CA-Greedy follows from the result of Conforti et al. [24] for submodular maximization subject to an independence system

$$\left(1 - (1 - \kappa/R)^r\right)/\kappa. \tag{4}$$

The parameters $r, R, \kappa, \kappa_i, \rho_{min}, \rho_{max}$ in the above ratios all depend on the input social network and the detailed definitions of them can be found in [5].

Note that the approximation ratios in Eqn. (3) and (4) hold under the assumption that there is an influence spread oracle which can exactly evaluate $\sigma_i(\cdot)$. Given the #P-hardness of computing $\sigma_i(A)$ for any given $A \subseteq V$ [17], Aslay et al. [5] further propose algorithms TI-CARM and TI-CSRM, as practical versions of CA-Greedy and CS-Greedy, by extending TIM [68] for influence spread estimation.

They prove that these two algorithms can return a solution $\vec{S}$ satisfying the following performance bound:

$$\pi(\vec{S}) \geq \beta \cdot \mathrm{OPT} - \epsilon \cdot \sum\nolimits_{i \in [h]} cpe(i) \cdot \sigma_i(N_i), \tag{5}$$

where $N_i \subseteq V$ is a node set that maximizes $\sigma_i(N_i)$ under the condition that $|N_i|$ equals the estimated cardinality of the maximum allocation to ad i under the given budget. The value of β equals Eqn. (4) and Eqn. (3) for TI-CARM and TI-CSRM, respectively.

2.2.1 Limitations of the Existing Solutions. Unfortunately, the solutions provided in [5] suffer from the following major shortcomings: (i) The exact value of the approximation bounds in Eqn. (3)–(4) cannot be computed easily—there is no obvious way to calculate them in polynomial time. This is a direct consequence of the bounds depending on the network instance and the #P-hardness of influence spread computation. (ii) The theoretical approximation ratios of CA-Greedy and CS-Greedy could be arbitrarily small. To see this, consider the case $h = 1$ and a network $G = (V, E)$, with $|V| = n$ nodes. Assume that E contains an edge (u, v), where the out-degree of v is 0, and $p^1_{(u,v)} = 1$. On this instance, the approximation ratio of CS-Greedy is at most $\frac{c_1(v)}{R \cdot \rho_{max} + c_1(v)}$, which can be arbitrarily small as $\rho_{max}/c_1(v)$ can be arbitrarily large. CA-Greedy also has similar problems and Aslay et al. [5] actually indicate in their paper that the worst-case approximation ratio of CA-Greedy is $1/R$, which is in the order of $O(1/n)$. (iii) The algorithms in [5] incur large computational overheads by their implementation. In fact, experimental results reveal that the running time and memory consumption of the TI-CARM and TI-CSRM algorithms in [5] both grow drastically when ϵ gets small. Therefore, the work in [5] has to set ϵ to a relatively large number (e.g., 0.3) such that TI-CARM and TI-CSRM can handle a social network with 4.8M nodes using a computer equipped with 264GB memory. (iv) The manner in which budget feasibility is ensured by the TI-CARM and TI-CSRM appeals to upper bounds on the expected spread when using estimations from a sample. This results in their seed allocation under-utilizing the budget to a great extent for the sake of not violating budget constraints. Naturally, allowing the host to control how much the budget can be *overshot*, for the sake of fully-utilizing advertisers budgets, is more desirable as this would also imply higher revenue for the host. There can be an agreement between the host and an advertiser that specifies who would pay the excess amount when the budget is overshot: this might be the advertiser as they would be receiving more engagements to their ad compared with when their budget is under-utilized; *or* this might be the host preferring to give some "free service" to advertisers since they can fully earn B_i. Note that in the latter case, the host can simply control the amount of free service provided, using a parameter ϱ. In an extreme, the host could use $B_i/(1 + \varrho)$ as the input budget to the algorithms, thus canceling out the effect of the overshoot (details in Section 4). Our contributions address all four limitations.

3 SOLVING RM WITH AN ORACLE

In this section, we present algorithms for the revenue maximization problem under the assumption that there is an oracle to compute $\sigma_i(A)$ for any $i \in [h]$ and $A \subseteq V$. We will present algorithms without this assumption in the next section.

3.1 Algorithms for a Single Advertiser

We first consider the case where there is only one advertiser i. In this case, the RM problem defined in Definition 2.1 belongs to the class of submodular maximization problems with a *single submodular knapsack constraint*, introduced by Iyer et al. [44]. As in [5], the greedy approximation guarantees provided by Iyer et al. [43, 44] are instance dependent and could be arbitrarily small, and in the case of cost-sensitive approximation, the guarantee can be unbounded as they acknowledge. We now show that a simple Greedy algorithm (Algorithm 1) can achieve a constant $\frac{1}{3}$-approximation. The algorithm greedily selects a seed user with the maximum marginal rate from the input candidate set U, and adds u into S_i if the total cost of the currently selected nodes is no more than B_i. It adds u into D_i if u is the first node satisfying $c_i(S_i \cup \{u\}) > B_i$ (we call such a node u as a "stopple node"). Finally, Greedy returns one of S_i or D_i, whichever has the larger revenue.

Theorem 3.1. *When there is only one advertiser i, the* Greedy(V, i) *algorithm returns a solution $S^*_i \subseteq V$ to the revenue maximization problem with approximation ratio of* $1/3$.

3.2 Algorithms for Multiple Advertisers

In this section, we provide algorithms for the RM problem when the number of advertisers is more than one (i.e., $h > 1$). A straightforward approach is to apply the greedy selection rule in the CS-Greedy algorithm [5], i.e., selecting an element (u, i) at each step with its marginal rate being as large as possible. However, this approach could not have a "nice" (i.e., network independent) approximation ratio for the case of $h > 1$ due to the following reasons. In fact, we cannot guarantee that the marginal rate of (u, i) is no less than any unselected element in the optimal solution $\vec{O}$, because there could exist $(v, j) \in \vec{O}$ with a larger marginal rate but we have already selected (v, ℓ) (for certain $\ell \in [h]$), and hence we have to select (u, i) instead of (v, j). Due to this issue, applying the proof idea of

Algorithm 1: Greedy(U, i)

Input: Advertiser i, a set $U \subseteq V$ of candidate users;
Output: A subset of U selected as the seed nodes;

$U \leftarrow U - \{v \mid v \in U \wedge c_i(v) + \pi_i(v) > B_i\}$;
$S_i \leftarrow \emptyset; D_i \leftarrow \emptyset$;
while $U \neq \emptyset \wedge D_i = \emptyset$ **do**
  $u \leftarrow \arg\max_{v \in U} \zeta_i(v \mid S_i)$; $U \leftarrow U - \{u\}$;
  if $c_i(S_i \cup \{u\}) + \pi_i(S_i \cup \{u\}) \leq B_i$ **then** $S_i \leftarrow S_i \cup \{u\}$;
  else $D_i \leftarrow \{u\}$;
$S_i^* \leftarrow \arg\max_{X \in \{S_i, D_i\}} \pi_i(X)$; **return** S_i^*;

Algorithm 2: ThresholdGreedy(γ)

1 $M \leftarrow \{(v, j) : (v, j) \in V \times [h] \wedge c_j(v) + \pi_j(v) \leq B_j\}$;
2 **foreach** $j \in [h]$ **do** $S_j \leftarrow \emptyset$; $D_j \leftarrow \emptyset$; $A_j \leftarrow \emptyset$; $I \leftarrow \emptyset$;
3 **while** $M \neq \emptyset \wedge I \neq [h]$ **do**
4 $(u, i) \leftarrow \arg\max_{(v,j) \in M} \pi_j(v \mid S_j)$; $M \leftarrow M - \{(u, i)\}$;
5 **if** $\zeta_i(u \mid S_i \cup D_i) < \gamma / B_i \vee D_i \neq \emptyset$ **then continue**;
6 **if** $u \in \bigcup_{j \in [h]} S_j \cup D_j$ **then continue**;
7 **if** $c_i(S_i \cup \{u\}) + \pi_i(S_i \cup \{u\}) \leq B_i$ **then** $S_i \leftarrow S_i \cup \{u\}$;
8 **else** $D_i \leftarrow \{u\}$; $I \leftarrow I \cup \{i\}$;
9 **if** $|I| = 1$ **then**
10 $i \leftarrow$ the number in I; $A_i \leftarrow$ Greedy$(V - \bigcup_{j \in [h]} S_j, i)$;
11 **foreach** $j \in [h]$ **do** $S'_j \leftarrow \arg\max_{X \in \{S_j, D_j, A_j\}} \pi_j(X)$;
12 $\vec{S}^* \leftarrow$ Fill$(\vec{S}')$; $b \leftarrow |I|$;
13 **return** $\vec{S}^*, b$;

Theorem 3.1 does not lead to non-trivial bounds when $h > 1$. *This raises a major challenge.*

3.2.1 A Greedy Algorithm with a Threshold. Accordingly, we design a new greedy algorithm dubbed ThresholdGreedy (see Algorithm 2) which abandons the greedy selection rule used in CS-Greedy, but simply selects the node with the maximum marginal gain at each step (as in CA-Greedy). However, this has the drawback that a lot of nodes with large seeding costs may be selected, quickly depleting the budgets [8]. To address this problem, we set an additional rule that the marginal rate of any selected node should be no less than a given threshold γ (we will discuss in Section 3.2.2 how to set the value of γ). We next explain the details of ThresholdGreedy.

ThresholdGreedy uses $M \subseteq V \times [h]$ to denote all candidate elements to be selected and uses I to denote the set of advertisers whose budgets have been depleted by the already selected elements (i.e., elements in $\vec{S} \cup \vec{D}$). In each step, it removes an element (u, i) from M with the maximum marginal gain (Line 4), and adds u into S_i or D_i if and only if all of the three conditions are satisfied (Lines 5–8): (1) marginal rate of (u, i) is no less than γ / B_i; (2) the node u has not been assigned to any advertiser yet; (3) the budget of advertiser i has not been depleted by nodes already in $S_i \cup D_i$.

[8] Here is a toy example to illustrate the intuition. Suppose that u, v, w are nodes with highest singleton revenues 91, 50 and 45, respectively, and that there are no common nodes reached by them. Let the costs of u, v, w be 9, 3 and 2, respectively. Then, for a budget of 100, CA-Greedy would select u and exhaust the budget for a revenue of 91, while CS-Greedy would select v, w, obtaining a total revenue of 95.

Algorithm 3: Fill$(\vec{S})$

1 $M \leftarrow \{(v, j) : (v, j) \in V \times [h] \wedge c_j(v) + \pi_j(v) \leq B_j\}$;
2 **while** $M \neq \emptyset$ **do**
3 $(u, i) \leftarrow \arg\max_{(v,j) \in M} \zeta_j(v \mid S_j)$;
4 $M \leftarrow M - \{(u, i)\}$;
5 **if** $c_i(S_i \cup \{u\}) + \pi_i(S_i \cup \{u\}) \leq B_i \wedge u \notin \bigcup_{j \in [h]} S_j$ **then**
6 $S_i \leftarrow S_i \cup \{u\}$;
7 **return** $\vec{S} = (S_1, \ldots, S_h)$;

This process terminates either when M is empty or when $|I| = h$. As in Algorithm 1, the "stopple node" for each i is stored in D_i.

After this greedy procedure terminates, the budgets of the advertisers in I must have been depleted by the nodes in $\bigcup_{i \in I} S_i \cup D_i$. If there is only one such advertiser i, we call the Greedy algorithm again to find a node set A_i (Line 10), and the revenue of A_i can help to derive the approximation ratio of ThresholdGreedy (details can be found in the proof of Theorem 3.2 in [36]). Next, the ThresholdGreedy algorithm sets S'_j to be the one in $\{S_j, A_j, D_j\}$ with the largest revenue for all $j \in [h]$ (Line 11). Finally, the function Fill is called (Line 12) to select more seed nodes for the advertisers whose budgets have not been depleted by the elements in $\vec{S}'$, and function Fill greedily selects nodes with the maximum marginal rate until the budgets of all advertisers are depleted. After that, the ThresholdGreedy algorithm returns the final solution $\vec{S}^*$.

The performance bound of ThresholdGreedy is shown in Theorem 3.2. Roughly speaking, the main idea in the proof is to classify the elements in the optimal solution into several categories according to their marginal rates with respect to the elements selected by ThresholdGreedy; we then bound the "revenue loss" caused by missing the elements in each category by γ or the revenue of the solution returned by ThresholdGreedy.

Theorem 3.2. *Suppose that Algorithm* ThresholdGreedy(γ) *returns* $(\vec{S}^*, b)$. *Then we have:*

$$\pi(\vec{S}^*) \geq \begin{cases} b \cdot \gamma / 2, & \text{if } b \geq 2; \\ \max\left\{\frac{1}{6}(\mathrm{OPT} - h \cdot \gamma), \frac{\gamma}{2}\right\}, & \text{if } b = 1; \\ \frac{1}{2}(\mathrm{OPT} - h \cdot \gamma), & \text{if } b = 0. \end{cases}$$

3.2.2 Searching for a Good Threshold. It can be seen from Theorem 3.2 that the approximation quality of ThresholdGreedy is affected by the threshold γ. Specifically, if γ is small, then the ThresholdGreedy algorithm could select more elements with large marginal gain. On the other hand, if γ is large, then ThresholdGreedy could select more elements with large marginal rate. Therefore, we propose a novel binary-search process (see Algorithm 4) to find an appropriate γ to strike a balance and find a good approximation ratio.

The Search algorithm takes two parameters $\tau \in (0, 1)$ and $b_{min} \in \{1, 2\}$ as input, and maintains an interval $[\gamma_1, \gamma_2]$ which is initialized to $[0, (1 + \tau)\gamma_{max}]$ and is halved at each iteration during binary-search, where γ_{max} is defined as

$$\gamma_{max} = \max\left\{B_j \cdot \zeta_j(v \mid \emptyset) : v \in V, j \in [h]\right\}. \quad (6)$$

Intuitively, if $\gamma > \gamma_{max}$, then no nodes would be selected by ThresholdGreedy; if $\gamma = 0$, then ThresholdGreedy would select nodes purely based on their marginal gains without considering

Algorithm 4: Search(τ, b_{min})

1 $\gamma_2 \leftarrow (1+\tau)\gamma_{max}$; $\gamma_1 \leftarrow 0$; $\mathcal{Q} \leftarrow \emptyset$; $\gamma \leftarrow \gamma_1$;
2 $\vec{T}_1^* \leftarrow \emptyset$; $\vec{T}_2^* \leftarrow \emptyset$; $b_1 \leftarrow 0$; $b_2 \leftarrow 0$;
3 **repeat**
4 $(\vec{T}, b) \leftarrow$ ThresholdGreedy(γ); $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\vec{T}\}$;
5 **if** $b \geq b_{min}$ **then** $(\vec{T}_1^*, b_1, \gamma_1) \leftarrow (\vec{T}, b, \gamma)$;
6 **else** $(\vec{T}_2^*, b_2, \gamma_2) \leftarrow (\vec{T}, b, \gamma)$;
7 $\gamma \leftarrow (\gamma_1 + \gamma_2)/2$;
8 **until** $\big((1+\tau)\gamma_1 \geq \gamma_2\big) \vee \big(\gamma_2 \leq \min_{i\in[h]} cpe(i)/(h+6)\big)$;
9 $\vec{S}^* \leftarrow \arg\max_{\vec{T}\in\mathcal{Q}} \pi(\vec{T})$;
10 **return** $\vec{S}^*, (\vec{T}_1^*, b_1, \gamma_1), (\vec{T}_2^*, b_2, \gamma_2)$;

their costs. The Search algorithm starts searching from $\gamma = 0$, such that it can test as many thresholds as possible in the binary search process for the purpose of maximizing the revenue. The input parameter b_{min} is a threshold used to guide the searching direction in binary search. For example, if γ is too large, then it is very likely that no advertisers would deplete their budgets and hence Line 4 returns $b < b_{min}$, which implies that we should try a smaller γ (see Line 6). More detailed explanations can be found in the sequel.

Throughout the searching process, Search maintains two solutions $(\vec{T}_1^*, b_1)$ and $(\vec{T}_2^*, b_2)$ such that $(\vec{T}_1^*, b_1)$ = ThresholdGreedy(γ_1) and $(\vec{T}_1^*, b_2)$ = ThresholdGreedy(γ_2), and adds $\vec{T}_1^*$ and $\vec{T}_2^*$ into the set $\mathcal{Q}$ during the search process. Note that b_1 (resp. b_2) represents the number of advertisers whose budgets would be depleted by the nodes selected by ThresholdGreedy under threshold γ_1 (resp. γ_2). The binary-search process stops when the length of $[\gamma_1, \gamma_2]$ is sufficiently small, and the Search algorithm returns the solution in $\mathcal{Q}$ that has the maximum revenue. Roughly speaking, the reason that Search can return a solution with large revenue is that, it keeps adjusting the interval $[\gamma_1, \gamma_2]$ to ensure that $b_1 \geq b_{min}$ and $b_2 < b_{min}$. Therefore, when the Search algorithm stops, if γ_1 is sufficiently large, then we can guarantee that $\pi(\vec{T}_1^*) \geq b_{min}\gamma_1/2$ is also sufficiently large following Theorem 3.2; on the other hand, if γ_1 is small, then γ_2 should also be small thanks to the stopping condition of Search, so we can guarantee that OPT $- h\gamma_2$ hence $\pi(\vec{T}_2^*)$ is sufficiently large according to Theorem 3.2. The parameter $b_{min} \in \{1, 2\}$ is just used to control the solution quality according to the above explanation and Theorem 3.2. Based on these ideas, we show in Theorem 3.3 that Search($\tau, 2$) achieves a good performance ratio.

Theorem 3.3. Search($\tau, 2$) *returns a solution* $\vec{S}^*$ *satisfying* $\pi(\vec{S}^*) \geq \frac{1}{(h+6)(1+\tau)}$OPT.

By an argument analogous to the proof of Theorem 3.3, we get:

Theorem 3.4. Search($\tau, 1$) *returns a solution* $\vec{S}^*$ *satisfying* $\pi(\vec{S}^*) \geq \frac{1}{2(h+1)(1+\tau)}$OPT.

3.3 Putting It Together

Theorems 3.1–3.4 imply that we can make optimizations based on the number of advertisers h. As $h + 6 \leq 2(h+1)$ when $h \geq 4$, we design an algorithm RM_with_Oracle (Algorithm 5) to get the best performance. The following theorem is immediate:

Algorithm 5: RM_with_Oracle(τ)

1 **if** $h = 1$ **then return** Greedy(V, 1);
2 **if** $2 \leq h \leq 3$ **then return** Search(τ, 1);
3 **if** $h \geq 4$ **then return** Search(τ, 2);

Theorem 3.5. *Given any* $\tau \in (0, 1)$, RM_with_Oracle(τ) *can return a solution to the RM problem with the approximation ratio* λ, *where* $\lambda = \frac{1}{3}$ *for* $h = 1$, $\lambda = \frac{1}{2(h+1)(1+\tau)}$ *for* $h \in \{2, 3\}$, *and* $\lambda = \frac{1}{(h+6)(1+\tau)}$ *for* $h \geq 4$.

Note that τ reflects a trade-off between accuracy and efficiency: the Search algorithm needs $O(\log \frac{h\gamma_{max}}{\min_{i\in[h]} cpe(i)})$ iterations in the worst case and needs $O(\log \frac{1}{\tau})$ iterations in most cases, while the approximation ratio improves with smaller τ. In practice, τ can be set as a small number (e.g., $\tau = 0.1$) while our algorithm still runs very fast, as our experiments in Section 5 demonstrate.

4 SOLVING RM WITHOUT AN ORACLE

In this section, we present algorithms for the RM problem without the influence spread oracle assumed in last section. Our algorithms are based on a novel adaptation to Reverse Reachable Sets [12] and several efficient sampling techniques with guaranteed performance bounds for the RM problem.

4.1 Reverse Reachable Sets

The concept of Reverse Reachable Set (RR-set) was first proposed in [12] for the Independent Cascade (IC) influence model. Given a social network G with each edge (u, v) associated with an influence propagation probability $p_{u,v}$ under the IC model, a random RR-set R is generated by first selecting a node $v \in V$ uniformly at random, and then setting R as the set of nodes in V that are reverse-reachable from v in a random graph generated by independently removing each edge $(u, v) \in E$ with probability $1 - p_{u,v}$. Given any node set A and a random RR-set R, we can define a random variable $X(A, R)$ such that $X(A, R) = 1$ when A intersects R and $X(A, R) = 0$ otherwise. Borgs et al. [12] show that the influence spread of A under the IC model equals $n \cdot \mathbb{E}[X(A, R)]$, and $\mathbb{E}[X(A, R)]$ can be estimated in an unbiased manner by the empirical mean $\sum_{R\in\mathcal{R}} X(A, R)/|\mathcal{R}|$ based on concentration bounds, where $\mathcal{R}$ is a set of generated RR-sets.

4.2 A New Method for Generating RR-Sets

In our problem, we need to design a method to estimate $\pi(\vec{S}) = \sum_{i\in[h]} cpe(i) \cdot \sigma_i(S_i)$ for any solution $\vec{S} = (S_1, \ldots, S_h)$ to the RM problem. According to Section 4.1, a straightforward idea for estimating $\pi(\vec{S})$ is to generate a set $\mathcal{R}_i$ of random RR-sets for each advertiser $i \in [h]$ with $|\mathcal{R}_1| = |\mathcal{R}_2| = \cdots = |\mathcal{R}_h|$, such that $\sigma_i(S_i)$ can be estimated using $\mathcal{R}_i$ for each $i \in [h]$. However, the estimation accuracy of this method is unsatisfactory, as the random variables in $\{X(S_i, R) : i \in [h] \wedge R \in \mathcal{R}_i\}$ have h different distributions while the concentration bounds are generally sharper when the considered random variables are identically distributed. To overcome this hurdle, we propose a *uniform sampling* method for generating a random RR-set, as described below:

(1) Sample a random advertiser $i \in [h]$ with probability proportional to $cpe(i)$.

(2) Generate an RR-set R for advertiser i selected in the first step using the edge probability $p^i_{(u,v)}$ for each edge $(u,v) \in E$.

Given $\vec{S} = (S_1, \ldots, S_h)$ and a random RR-set R generated as above, we define a random variable $\Lambda(\vec{S}, R)$ such that $\Lambda(\vec{S}, R) = 1$ if R is generated for certain advertiser $j \in [h]$ and $S_j \cap R \neq \emptyset$, and $\Lambda(\vec{S}, R) = 0$ otherwise. Let $\Gamma = \sum_{i \in [h]} cpe(i)$. Then it follows that:

Lemma 4.1. $\pi(\vec{S}) = n\Gamma \cdot \mathbb{E}[\Lambda(\vec{S}, R)]$.

Given a set $\mathcal{R}$ of random RR-sets generated by using the uniform sampling method described above, Lemma 4.1 suggests that $\widetilde{\pi}(\vec{S}, \mathcal{R}) = n\Gamma \cdot \sum_{R \in \mathcal{R}} \Lambda(\vec{S}, R)/|\mathcal{R}|$ is an unbiased estimation of $\pi(\vec{S})$. Moreover, as the random variables in $\{\Lambda(\vec{S}, R) \colon R \in \mathcal{R}\}$ follow the same distribution, it is possible to use sharper concentration bounds to improve the estimation accuracy. Similarly, we also have $\widetilde{\pi}_i(S_i, \mathcal{R}) = n\Gamma \cdot \sum_{R \in \mathcal{R}} \Lambda(S_i, R)/|\mathcal{R}|$ as an unbiased estimation of $\pi_i(S_i)$ for any $i \in [h]$, where $\Lambda(S_i, R) = \min\{|S_i \cap R|, 1\}$ if R is generated for advertiser i, and $\Lambda(S_i, R) = 0$ otherwise.

4.3 One-Batch Sampling

With the uniform sampling method described above, a simple *one-batch* sampling algorithm can be used to address the RM problem: we first generate a set $\mathcal{R}$ of RR-sets, then call the RM_with_Oracle algorithm with the function $\pi_i(\cdot)$ replaced by $\widetilde{\pi}_i(\cdot, \mathcal{R})$ and the budget B_i replaced by $(1 + \varrho/2)B_i$ for all $i \in [h]$, where $\varrho \in (0, 1)$ is an input parameter for bicriteria approximation. Note that we have to seek a bicriteria approximation for the RM problem as the optimal solution $\vec{O}$ may violate the budget constraint in the sampling space due to the sampling error. Therefore, it is hopeless to find an approximate solution as we can only derive an approximation ratio through sampling, unless we under-utilize the available budget, based on estimates and concentration bounds.[9] The following theorem shows that such a one-batch algorithm can return an approximate solution to the RM problem when $|\mathcal{R}|$ is sufficiently large:

Theorem 4.2. *The one-batch approach described above can return a solution $\vec{S}^*$ satisfying $c_i(S^*_i) + \pi_i(S^*_i) \leq (1 + \varrho)B_i$ for all $i \in [h]$ and $\pi(\vec{S}^*) \geq (\lambda - \epsilon)\mathrm{OPT}$ with probability at least $1 - \delta$, as long as $|\mathcal{R}| \geq \theta_{max} = \max\{\hat{\theta}_{max}, \bar{\theta}_{max}\}$ by setting*

$$\hat{\theta}_{max} = \frac{2n}{\epsilon^2}\left(\lambda\sqrt{\ln\frac{4}{\delta}} + \sqrt{\lambda\left(\ln\frac{4}{\delta} + \sum_{i \in [h]} \mu_i \ln\frac{en}{\mu_i}\right)}\right)^2,$$

$$\bar{\theta}_{max} = \frac{8n\Gamma(1+\varrho)}{\varrho^2 B_{min}}\left(\ln\frac{4h}{\delta} + \mu\ln\frac{en}{\mu}\right),$$

where μ_i is the maximum number of nodes that can be selected by advertiser i without exceeding the relaxed budget of $(1 + \varrho)B_i$, $\mu = \max\{\mu_i \colon i \in [h]\}$, and $B_{min} = \min\{B_i \colon i \in [h]\}$.

The proof of Theorem 4.2 (provided in [36]) is highly non-trivial compared to the existing results for the traditional influence maximization problem [67, 68], as the RM problem is more complex. In a nutshell, the proof of Theorem 4.2 shows that, when the number of RR-sets in $\mathcal{R}$ is sufficiently large (i.e., $\geq \theta_{max}$), the one-batch algorithm can achieve the claimed performance guarantee because all the following conditions simultaneously hold with high probability:

[9] Recall, the effect of budget overshoot can be canceled out by the host using a "corrected" budget $B_i := B_i/(1 + \varrho)$, if desired (see Section 2.2.1).

Algorithm 6: RM_without_Oracle$(\epsilon, \delta, \tau, \varrho)$

1 $\lambda \leftarrow$ the approximation ratio shown in Theorem 3.5;
2 $\delta' \leftarrow \delta/4$ and compute θ_{max} by replacing δ with δ' in $\hat{\theta}_{max}$ and $\bar{\theta}_{max}$ defined in Theorem 4.2;
3 $\theta_0 \leftarrow \frac{4n\Gamma(2+\varrho/3)}{\varrho^2 B_{min}} \ln\frac{h}{\delta'}$; $t_{\max} \leftarrow \lceil \log_2 \frac{\theta_{max}}{\theta_0} \rceil$; $q \leftarrow \ln\frac{h+2}{\delta' t_{\max}}$;
4 generate two sets $\mathcal{R}_1$ and $\mathcal{R}_2$ of random RR sets, with $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_0$;
5 **while** *true* **do**
6 $\quad \vec{S}^*, (\vec{T}^*_1, b_1, \gamma_1), (\vec{T}^*_2, b_2, \gamma_2) \leftarrow$ RM_with_Oracle(τ) with function $\pi_i(\cdot)$ replaced by $\widetilde{\pi}_i(\cdot, \mathcal{R}_1)$ and B_i replaced by $(1 + \frac{\varrho}{2})B_i$ for all $i \in [h]$;
7 $\quad z \leftarrow$ SeekUB$(\vec{S}^*, \vec{T}^*_1, b_1, \gamma_1, \vec{T}^*_2, b_2, \gamma_2, b_{min}, \lambda, \mathcal{R}_1)$;
8 $\quad$ *Feasible* $\leftarrow$ True; $B_{min} \leftarrow \min\{B_i \colon i \in [h]\}$;
9 $\quad$ **foreach** $i \in [h]$ **do**
10 $\quad\quad UB(S^*_i) \leftarrow \left(\sqrt{\frac{\widetilde{\pi}_i(S^*_i, \mathcal{R}_2)|\mathcal{R}_2|}{n\Gamma} + \frac{q}{2}} + \sqrt{\frac{q}{2}}\right)^2 \cdot \frac{n\Gamma}{|\mathcal{R}_2|}$;
11 $\quad\quad$ **if** $UB(S^*_i) > (1 + \varrho)B_i - c_i(S^*_i)$ **then** *Feasible* $\leftarrow$ False;
12 $\quad LB(\vec{S}^*) \leftarrow \left(\left(\sqrt{\frac{\widetilde{\pi}(\vec{S}^*, \mathcal{R}_2)|\mathcal{R}_2|}{n\Gamma} + \frac{2q}{9}} - \sqrt{\frac{q}{2}}\right)^2 - \frac{q}{18}\right) \cdot \frac{n\Gamma}{|\mathcal{R}_2|}$;
13 $\quad UB(\vec{O}) \leftarrow \left(\sqrt{\frac{z|\mathcal{R}_1|}{n\Gamma} + \frac{q}{2}} + \sqrt{\frac{q}{2}}\right)^2 \cdot \frac{n\Gamma}{|\mathcal{R}_1|}$;
14 $\quad \beta \leftarrow LB(\vec{S}^*)/UB(\vec{O})$;
15 $\quad$ **if** $(\beta \geq \lambda - \epsilon \wedge$ *Feasible*$) \vee |\mathcal{R}_1| \geq \theta_{max}$ **then return** $\vec{S}^*$;
16 $\quad$ double the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ with new random RR sets;

(i) The optimal solution $\vec{O}$ is budget-feasible in the "sampling space", i.e., $c_i(O_i) + \widetilde{\pi}_i(O_i, \mathcal{R}) \leq (1 + \varrho/2)B_i$ for all $i \in [h]$.
(ii) The approximate solution $\vec{S}^*$ is "almost" budget-feasible, i.e., $c_i(S^*_i) + \pi_i(S^*_i, \mathcal{R}) \leq (1 + \varrho)B_i$ for all $i \in [h]$.
(iii) The approximate solution $\vec{S}^*$ satisfies the $\lambda - \epsilon$ approximation ratio, i.e., $\pi(\vec{S}^*) \geq (\lambda - \epsilon)\mathrm{OPT}$.

Roughly speaking, Condition (i) ensures that the optimal solution is comparable to the approximate solution $\vec{S}^*$, as we require $\forall i \in [h] \colon c_i(S^*_i) + \widetilde{\pi}_i(S^*_i, \mathcal{R}) \leq (1 + \varrho/2)B_i$ in searching $\vec{S}^*$; Conditions (ii)–(iii) ensure that $\vec{S}^*$ is a valid bi-criteria approximate solution.

4.4 Progressive Sampling

Theorem 4.2 implies that θ_{max} is an upper bound on the required number of RR-sets for guaranteed performance. In this section, we propose a progressive sampling algorithm in Algorithm 6 that generates fewer RR-sets in practice without compromising the performance guarantee. The design of Algorithm 6 is similar in spirit to the OPIM-C framework in [65] for the influence maximization problem, but it involves more complex operations as RM problem has more stringent requirements on bounding the sampling errors.

Instead of generating θ_{max} RR-sets in one batch, Algorithm 6 first generates two sets of RR-sets (i.e., $\mathcal{R}_1$ and $\mathcal{R}_2$) with $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_0$, where θ_0 is much smaller than θ_{max}. It then uses $\mathcal{R}_1$ as the input to the one-batch algorithm to find a solution $\vec{S}^*$ (Line 6). Afterwards, it tests whether Conditions (i)–(iii) listed in Section 4.3 can be satisfied by $\vec{S}^*$, $\mathcal{R}_1$ and $\mathcal{R}_2$ with high probability. Specifically, Line 3 ensures that $|\mathcal{R}_1|$ is sufficiently large such that Condition (i) can be satisfied;

Lines 9–11 check whether Condition (ii) can be satisfied, where $UB(S_i^*)$ is an upper bound of $\pi_i(S_i^*)$ computed using the concentration bounds; Line 15 checks whether Condition (iii) is satisfied (i.e., whether $LB(\vec{S}^*)/UB(\vec{O}) \geq \lambda - \epsilon$), where $LB(\vec{S}^*)$ and $UB(\vec{O})$ are lower bound and upper bound of $\pi(\vec{S}^*)$ and $\pi(\vec{O})$ with high probability, respectively. When all the three conditions are satisfied, the solution $\vec{S}^*$ is returned immediately. Otherwise, the algorithm doubles the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ and repeats the above process until a satisfying solution is returned or the number of generated RR-sets reaches θ_{max} (Lines 15–16). Although RM_without_Oracle may theoretically generate θ_{max} RR-sets in the worst case, our experimental results in Section 5 show that it runs very fast in practice.

A key challenge in Algorithm 6 is that we need to make the upper bound $UB(\vec{O})$ and lower bound $LB(\vec{S}^*)$ as tight as possible, so that the condition in Line 15 can be met more easily, making the algorithm more likely to generate fewer RR-sets and stop early.

Although it is relatively easy to get $LB(\vec{S}^*)$ based on concentration bounds, finding a tight $UB(\vec{O})$ is non-trivial as $\vec{O}$ is unknown. To address this problem, Algorithm 6 calls the SeekUB function to find an upper bound z of $\widetilde{\pi}(\vec{O}, \mathcal{R}_1)$ (Line 7), which is further used to derive $UB(\vec{O})$ based on concentration bounds (Line 13). The SeekUB function adopts a novel method to find a tight upper bound of $\widetilde{\pi}(\vec{O}, \mathcal{R}_1)$ based on the special binary-search process of the Search algorithm. More specifically, as $\widetilde{\pi}(\cdot, \mathcal{R}_1)$ is also a monotone submodular function, ThresholdGreedy called by the Search algorithm also satisfies Theorem 3.2 in the sampling space with OPT replaced by $\widetilde{\pi}(\vec{O}, \mathcal{R}_1)$, which can be used for SeekUB to derive an upper bound of $\widetilde{\pi}(\vec{O}, \mathcal{R}_1)$. For example, when the Search algorithm returns $(\vec{T}_2^*, b_2, \gamma_2)$ with $b_2 = 0$, we can know from Theorem 3.2 that $\widetilde{\pi}(\vec{T}_2^*, \mathcal{R}_1) \geq \frac{1}{2}(\widetilde{\pi}(\vec{O}, \mathcal{R}_1) - h \cdot \gamma_2)$, so $2\widetilde{\pi}(\vec{T}_2^*, \mathcal{R}_1) + h\gamma_2$ is an upper bound of $\widetilde{\pi}(\vec{O}, \mathcal{R}_1)$ and it could be tighter than the naive upper bound of $\widetilde{\pi}(\vec{S}^*, \mathcal{R}_1)/\lambda$. Based on all the methods decribed above, we can get the following theorem:

Theorem 4.3. RM_without_Oracle$(\epsilon, \delta, \tau, \theta_0, \theta_{max})$ *returns a solution* $\vec{S}^*$ *satisfying* $c_i(S_i^*) + \pi_i(S_i^*) \leq (1+\varrho)B_i$ *for all* $i \in [h]$ *and* $\pi(\vec{S}^*) \geq (\lambda - \epsilon)\mathrm{OPT}$ *with probability at least* $1 - \delta$ *for any* $\delta \in (0, 1)$, *where* λ *is the approximation ratio shown in Theorem 3.5.*

The intuition is that as in the proof of Theorem 4.2, the proof of Theorem 4.3 similarly shows that the conditions (i)–(iii) given in Section 4.3 can be satisfied by RM_without_Oracle with high probability. The major difference is that, since RM_without_Oracle adopts a "trial-and-error" approach and returns an approximate solution immediately in each trial if it judges that the current solution has already satisfied the performance guarantee, we need to show that the total probability of RM_without_Oracle making wrong judgements in all the trials is no more than δ. We prove this by using concentration bounds [36].

Time complexity. We provide the theoretical time complexity of RM_without_Oracle (RMA), and of the algorithms of [5], left open in [5]. RMA has an expected time complexity of $O\big(\frac{m\sum_{i\in[h]}\mathbb{E}[\pi_i(\{v^*\})](\ln\frac{1}{\delta}+n\ln h)}{\epsilon^2 B_{min}}\big)$, where v^* denotes a random node selected from V with probability proportional to its in-degree. Both algorithms of [5] on the other hand have a time complexity of $O\big(\frac{n(1+\frac{\ln 1/\delta}{\ln n})(m+n)\ln n}{\epsilon^2}\big)$. These results show that the running time of RMA is dominated by the factor mn while the running time of the algorithms of [5] are dominated by the factor $n(m+n)$, translating to the superiority of the RMA algorithm in terms of asymptotic worst-case running time. The proofs can be found in [36].

Algorithm 7: SeekUB$(\vec{S}^*, \vec{T}_1^*, b_1, \gamma_1, \vec{T}_2^*, b_2, \gamma_2, b_{min}, \lambda, \mathcal{R}_1)$

1 **if** $h = 1$ **then return** $\widetilde{\pi}(\vec{S}^*, \mathcal{R}_1)/\lambda$;
2 **if** $b_1 < b_{min}$ **then** $z \leftarrow 6\widetilde{\pi}(\vec{T}_2^*, \mathcal{R}_1)$;
3 **if** $b_1 \geq b_{min} \wedge \vec{T}_2^* \neq \emptyset$ **then**
4 **if** $b_2 = 0$ **then** $z \leftarrow 2\widetilde{\pi}(\vec{T}_2^*, \mathcal{R}_1) + h\gamma_2$;
5 **if** $b_2 = 1$ **then** $z \leftarrow 6\widetilde{\pi}(\vec{T}_2^*, \mathcal{R}_1) + h\gamma_2$;
6 **if** $b_1 \geq b_{min} \wedge \vec{T}_2^* = \emptyset$ **then** $z \leftarrow \widetilde{\pi}(\vec{T}_1^*, \mathcal{R}_1)/\lambda$;
7 **return** $\min\{z, \widetilde{\pi}(\vec{S}^*, \mathcal{R}_1)/\lambda\}$;

Discussion. We note that although Algorithm 6 borrows some ideas from the OPIM-C framework [65], it embodies necessary and nontrivial extensions (e.g., the sampling method in Sec. 4.2) because OPIM-C was originally designed for the simpler Influence Maximization (IM) problem. We also note that a recent study [34] on the IM problem also used the OPIM-C framework, but it presented an algorithm, dubbed SUBSIM, to accelerate the generation of a single RR-set, which can also be used by RMA. Another useful extension for improving the empirical performance of RMA is as follows. Before RMA returns a solution $\vec{S}^*$, check whether the ratio of $\widetilde{\pi}(\vec{S}^*, \mathcal{R}_2)$ to $\widetilde{\pi}(\vec{S}^*, \mathcal{R}_1)$ is too small (e.g., less than 80%); if so, then generate more RR-sets to enlarge the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ by a constant factor (e.g., 15×), and then repeat the solution-seeking process as before to find a new solution $\vec{S}^c$ using the new collections of RR-sets $\mathcal{R}_1$ and $\mathcal{R}_2$. Finally, return $\vec{S}^c$ if it also satisfies the stopping condition in Line 15, and otherwise return $\vec{S}^*$. Clearly, such an extension does not affect the theoretical performance bound of RMA, and it enables RMA to potentially output a solution with an empirical (instance-dependent) approximation ratio better than $\lambda - \epsilon$ (and also minimize biases), by possibly generating more RR-sets.

5 PERFORMANCE EVALUATION

In this section, we compare the performance of our algorithm with the state-of-the-art algorithms proposed in [5]. The performance of considered algorithms is evaluated in terms of their revenue, seeding costs and running time. All algorithms are implemented using C++ and all experiments are run on a Linux server with Intel Xeon 2.20GHz CPU and 192GB memory.

5.1 Experimental Setting

Datasets. We use several public datasets in our experiments. Flixster [5] is from a social movie rating network website (www.flixster.com), where each node represents a user and two users are connected by a directed edge if they are friends, both rating the same movies. LastFM [8] is a social network where people can specify their interests on music types and make friends. As both Flixster and LastFM have action logs that record users' activities of rating movies or music (i.e., "a log of past propagation" in [9]), we use the method provided in [9] to learn the topic-dependent influence probabilities (i.e., $\hat{p}^z_{u,v}$) on each edge $(u, v) \in E$. We also follow the settings in [5] to set the default numbers $L = 10$ and

Table 1: Datasets

Dataset	$\|V\|$	$\|E\|$	Type
Lastfm	1.3K	14.7K	directed
Flixster	30K	425K	directed
DBLP	317K	1.05M	undirected
LiveJournal	4.8M	69M	directed

Table 2: Advertiser budgets and CPE values

Dataset	Budgets			CPEs		
	mean	max	min	mean	max	min
Lastfm	320	1200	100	1.5	2	1
Flixster	10.1K	20K	6K	1.5	2	1

$h = 10$ for the Flixster and LastFM datasets, and use the same topic distributions as that in [5] for the Flixster dataset. The topic distributions used for LastFM are learned from its action logs. As a result, more than 95% (resp. 77%) of the influence probabilities generated for Flixster (resp. LastFM) are positive.

We also use the DBLP and LiveJournal Datasets to test the scalability of the implemented algorithms. DBLP [52] is a collaboration network where each node represents an author and co-authors are adjacent in the network. LiveJournal [52] is a free on-line blogging community where users declare friendship with each other. These two datasets are also used in [5]. The details of our datasets used in the experiments are listed in Table 1.

Seed Incentive Models. Similar to [5], we use three seed incentive models (i.e., node seeding cost models) in the experiments. Given a fixed constant $\alpha > 0$ and any pair $(u, i) \in V \times [h]$, these models set the cost of node u for advertiser i as follows:

- Linear incentive model: the cost of u is proportional to its influence spread, i.e., $c_i(u) = \alpha \cdot \sigma_i(\{u\})$.
- QuasiLinear incentive model: the cost of u is a quasi-linear function, i.e., $c_i(u) = \alpha \cdot \sigma_i(\{u\}) \ln(\sigma_i(\{u\}))$.
- SuperLinear incentive model: the cost of u is a quadratic function of its influence spread. i.e., $c_i(u) = \alpha \cdot (\sigma_i(\{u\}))^2$.

Baseline Algorithms. As mentioned in Section 2.2, only Aslay et al. [5] have addressed the revenue maximization problem considered in this paper and proposed TI-CSRM and TI-CARM that can be implemented in practice. Therefore, we use TI-CSRM and TI-CARM to compare with our algorithm RM_without_Oracle (RMA for short) (Section 4.4).

Parameter Settings. In all the experiments, we set $\epsilon = 0.02$, $\delta = 1/n$, $\varrho = 0.1$ and $\tau = 0.1$ for our RMA algorithm unless otherwise stated, so RMA always returns an approximate solution with non-negative approximation ratios, according to Theorem 4.3. However, TI-CSRM and TI-CARM algorithms cannot terminate successfully on all the four datasets due to memory issues and high running time when ϵ is set to 0.02, where ϵ is the parameter in Eqn. (5). Therefore, we follow the same setting in [5] to set their parameter $\epsilon = 0.1$ for the Flixster and LastFM datasets, and set $\epsilon = 0.3$ for DBLP and LiveJournal. In all experiments, for fair comparison, we set the budget input to each advertiser in TI-CSRM and TI-CARM to $(1 + \varrho)\times$ the budget to the same advertiser in the RMA algorithm, due to the consideration that RMA is a bi-criteria approximation algorithm. With this implicit rule, we will only cite the budget setting of TI-CSRM and TI-CARM in the experiments. Note that this budget setting is practically equivalent to the setting that the budget used by RMA is $(1+\varrho)^{-1}$ fraction of that used by TI-CSRM and TI-CARM.

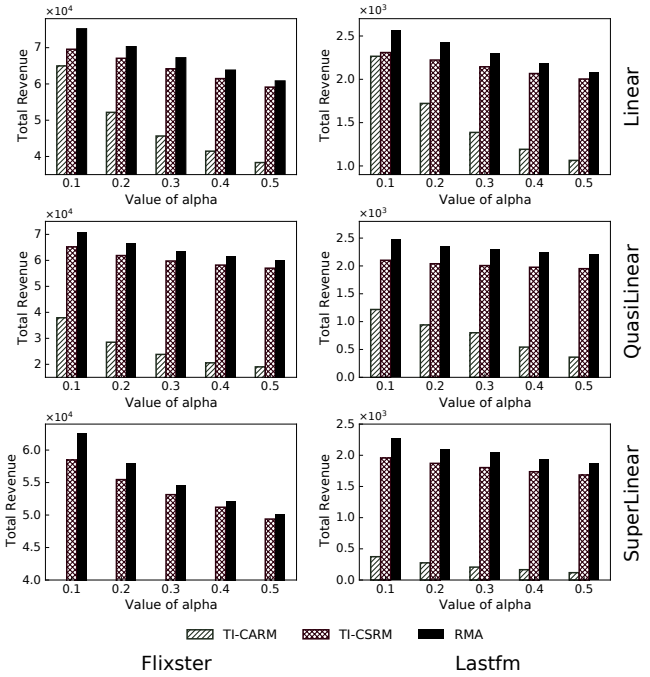

Figure 1: Total revenue as a function of α, on `Flixster`(left) and `Lastfm`(right), for linear (top), quasi-linear (middle), and super-linear (bottom) incentive models.

In all our experiments, we measure the revenue of the implemented algorithms by using 10^7 RR-sets, generated independently of the considered algorithms. Alternatively, we could measure the revenue using (the much slower) Monte-Carlo simulations, but we found in the experiments that the accuracy of these methods does not have a noticeable difference as long as the samples used are independent of the considered algorithms. Besides conducting experiments under the implicit parameter settings described above, we will also study the impact of varying these parameters in Sec. 5.2.2.

5.2 Experimental Results

5.2.1 General Comparisons. In this section, we compare the algorithms under different node cost models on LastFM and Flixster datasets. Following the experimental settings in [5], the advertisers are assigned heterogeneous budgets and CPE values for seed selection in TI-CARM and TI-CSRM, as shown in Table 2. We first plot the the revenue performance of TI-CARM, TI-CSRM and the RMA algorithm in Fig. 1 under the linear, quasilinear and superlinear cost modes, with varying levels of the parameter α. It can be seen that the revenue of all algorithms decreases when α increases, which can be explained by the fact that the costs of all nodes increase with α, so fewer seed nodes can be selected by all algorithms when α gets larger. The results in Fig. 1 also reveal that our RMA algorithm consistently outperforms TI-CARM and TI-CSRM algorithms under all three seed incentive models and for different values of α. Specifically, the RMA algorithm can achieve up to 15.81× (resp. 17.68%) gain on the revenue compared to TI-CARM (resp. TI-CSRM). Moreover, we observe that TI-CARM performs very poorly under the superlinear cost model. The intuition is that TI-CARM greedily selects elements purely based on their marginal gains while neglecting the costs, so it may quickly meet an element violating

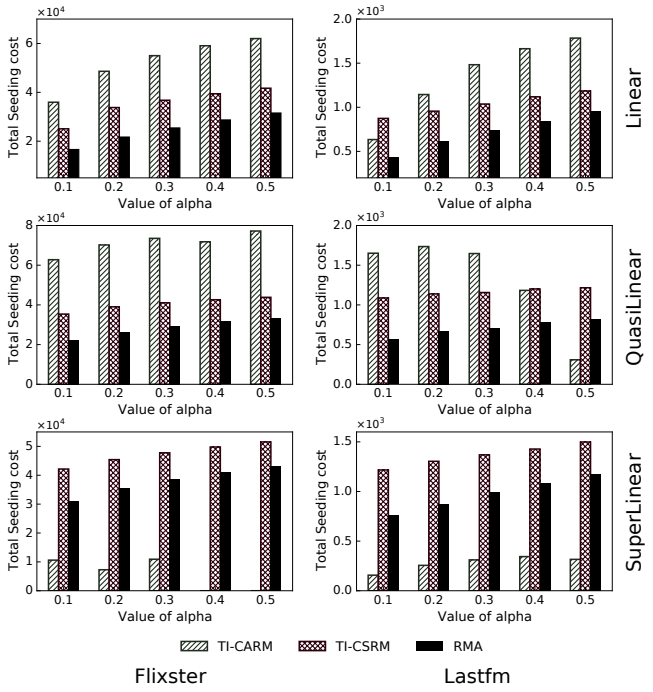

Figure 2: Total seeding cost as a function of α, on `Flixster`(left) and `Lastfm`(right), for linear (top), quasi-linear (middle), and super-linear (bottom) incentive models.

the budget constraint and hence terminate with very few seeds selected. This situation is analogous to choosing items without considering their costs in the traditional knapsack problem. These results demonstrate the effectiveness of our methods used in RMA for revenue optimization.

In Fig. 2, we plot the total seed costs (i.e., the amount paid by all advertisers for incentivizing seeds) assigned by different algorithms. RMA produces consistently lower seeding cost than TI-CSRM and the lowest cost for most cases under the linear and quasilinear models. Under the superlinear cost model, TI-CARM achieves very low seeding costs, which can be explained by a reason similar to that described above for the revenue performance of TI-CARM. These experiments show that RMA achieves the best revenue performance. Its seeding cost is always lower than TI-CSRM. While TI-CARM has lower costs in some settings (especially under the superlinear model), that does not translate to better revenue.

In Fig. 3, we study the impact of α on the number of seeds selected by the implemented algorithms under the linear cost model, where the other settings are the same with Fig. 1. It can be seen that the seed size decreases when α increases, as the node costs increase and hence fewer nodes can be selected under given budgets. While the seed set sizes for RMA and TI-CSRM are comparable, TI-CARM only manages to select very few seeds.

In Table 3, we show the running time of the implemented algorithms under the linear cost model. It can be seen that RMA runs faster than TI-CARM and TI-CSRM under all the settings of α (1.04× to 38.5× faster), which can be explained by the fact that our RMA algorithm has leveraged the sampling algorithms proposed in Section 4 to reduce the number of RR-sets to be generated, while still achieving the guaranteed performance ratio. The experimental

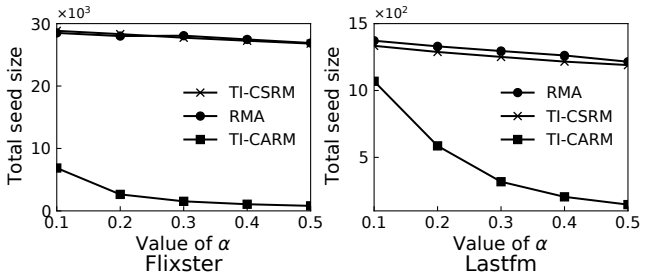

Figure 3: Impact of α on seed size

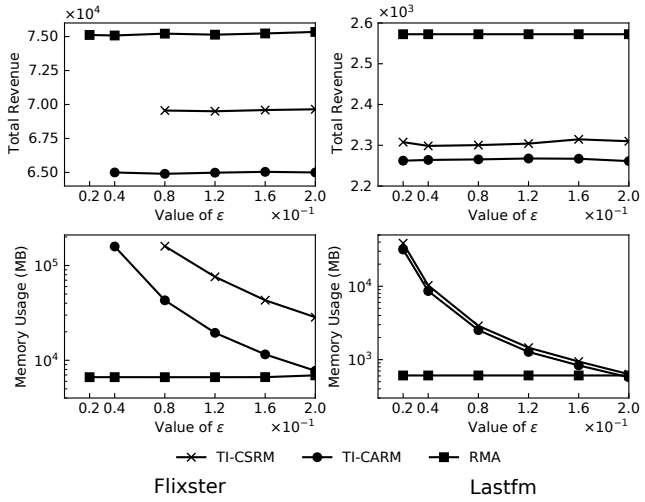

Figure 4: The impact of ϵ on revenue and memory usage

Table 3: Running time (seconds) under linear cost model

`Flixster`	$\alpha = 0.1$	0.2	0.3	0.4	0.5
RMA	736	738	664	695	681
TI-CARM	3803	1609	1074	880	710
TI-CSRM	16255	18798	25572	24109	23473
`Lastfm`	$\alpha = 0.1$	0.2	0.3	0.4	0.5
RMA	26	25	23	24	23
TI-CARM	108	91	75	65	56
TI-CSRM	130	147	145	152	153

results on the running time of the implemented algorithms under the other cost models are qualitatively similar and hence are omitted due to space constraints. It is also noted that [5] exhibits faster running time than that shown in Table 3, because only half of the budgets shown in Table 2 of [5] were used in their experiments, resulting in fewer selected nodes and hence faster running time.

5.2.2 Impact of Parameters. Fig. 4 compares the revenue and memory consumption of implemented algorithms by varying ϵ from 0.02 to 0.2 in Flixster and LastFM, using the linear cost model, with $\alpha = 0.1$. The results reveal that the revenue of RMA does not vary much and it consistently outperforms both TI-CSRM and TI-CARM over the range of values of ϵ considered. The reason is as follows. Although RMA has a theoretical approximation ratio of $\lambda - \epsilon$ (Theorem 4.2) which is affected by ϵ, this ratio is just worst case and the actual performance ratio of RMA on specific datasets could be much better. Indeed, we observed in our experiments that RMA can practically achieve an approximation ratio β, where β is even larger

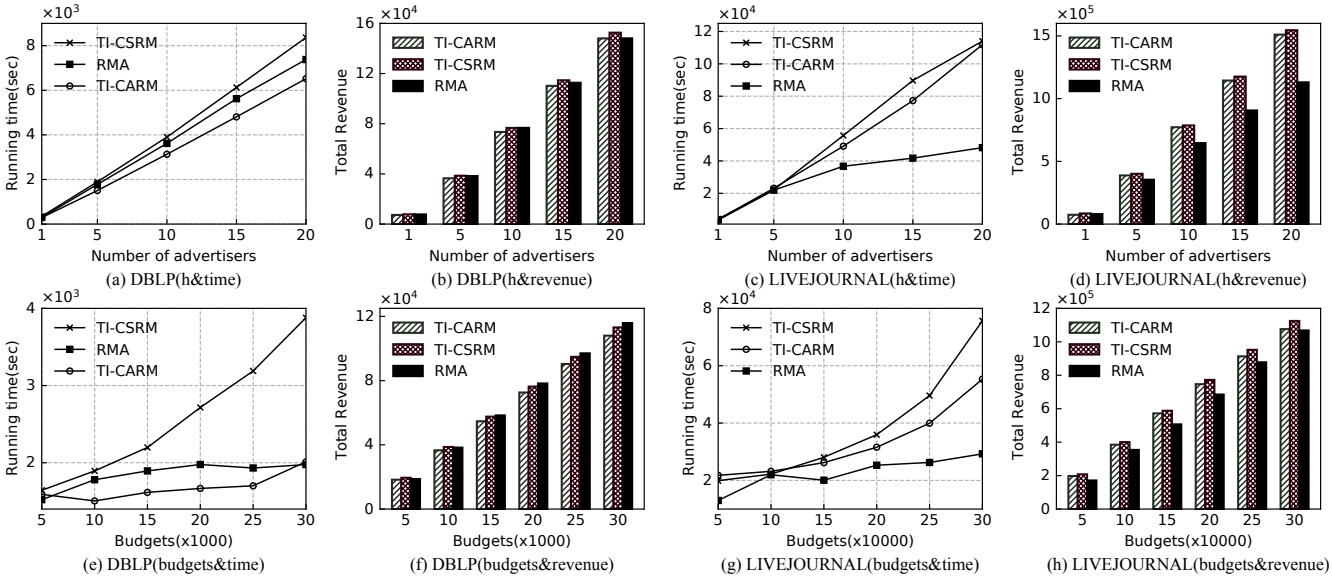

Figure 5: Running time and total revenue of RM, TI-CARM and TI-CSRM on `DBLP` and `LiveJournal`

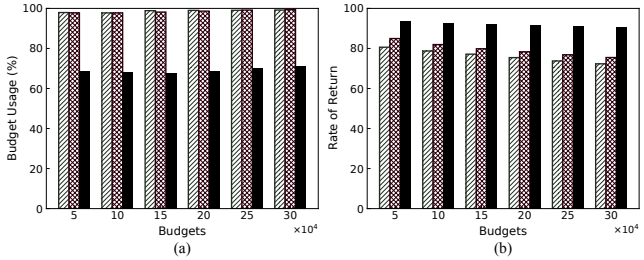

Figure 6: Budget usage and rate of return of RMA, TI-CARM, TI-CSRM on LiveJournal

than λ in most cases, so its revenue performance is quite "robust" to the variation of ϵ due to the stopping rule in Line 15 of Algorithm 6. However, Fig. 4 also shows that the memory consumption of TI-CARM and TI-CSRM significantly increases due to the large number of generated RR-sets with decreasing ϵ, which eventually causes memory overflow problems on the Flixster dataset when $\epsilon \leq 0.02$ (for TI-CARM) or $\epsilon \leq 0.04$ (for TI-CSRM).

5.2.3 Scalability Test. In this section, we follow the same settings as those in [5] to test the scalability of the implemented algorithms on DBLP and LiveJournal datasets. As there are no action logs for these two datasets, we cannot use the method proposed in [9] to learn the influence probabilities. So we follow [5] to use the Weighted-Cascade model, i.e., setting $p^i_{u,v} = 1/|N^{in}(v)|$ for all $i \in [h]$ and $(u, v) \in E$, where $N^{in}(v)$ denotes the set of in-neighbors of node v. We also follow [5] to use the Linear incentive model with $\alpha = 0.2$ and identical budgets input to all advertisers. Note that advertisers usually have heterogeneous budgets and the influence propagation depends on specific topics/items in practice (like the settings adopted in Fig. 1). So the settings of uniform budgets and Weighted-Cascade model in this section are less practical and are only used for a fair comparison with [5].

In Figs. 5(a)–(d), we compare the running time and revenue of the implemented algorithms by scaling the number of advertisers h from 1 to 20, where the budget of each advertiser is set to 10K (100K) for DBLP (resp. for LiveJournal). In Figs. 5(e)–(h), we compare the algorithms by scaling the budget of advertisers, with the number of advertisers fixed to 5. Notice that fixing number of advertisers and scaling up budgets is similar to increasing number of advertisers with a fixed budget, w.r.t. the number of seeds selected.

It can be seen from Fig. 5 that RMA runs faster than TI-CARM and TI-CSRM in almost all cases. On DBLP, all three algorithms attain almost the same revenue. However, on LiveJournal, although RMA still runs faster than TI-CSRM/TI-CARM, it achieves smaller revenue than TI-CSRM/TI-CARM. We explain the reason below. Note that the budget used by RMA is $(1+\varrho)^{-1}$ times that used by TI-CSRM and TI-CARM, where we set $\varrho = 0.1$. Although this does not affect RMA's superiority in most cases (as shown in other figures), it could occasionally make TI-CSRM/TI-CARM perform better on revenue in some networks (depending on network structures and propagation models). This happens especially when the total budget is large, causing a large budget overshoot for a given ϱ. We analyzed the *rate of actual budget usage* $(\pi(\vec{S}) + \sum_{i\in[h]} c_i(S_i))/(\sum_{i\in[h]} B_i)$ and the *rate of return* $\pi(\vec{S})/[\pi(\vec{S}) + \sum_{i\in[h]} c_i(S_i)]$ of all algorithms for Fig. 5(h): results shown in Fig. 6 (results for Fig. 5(d) are similar). It can be seen from Fig. 6 that, RMA uses smaller budgets than TI-CSRM/TI-CARM, while its rate of return is clearly higher. This implies that RMA is much more "profitable" than TI-CSRM/TI-CARM, which could be important from a practical point of view.

5.2.4 Studying the Scenario with a Holistic Demand. In this section, we consider a practical scenario where social advertising demands are controlled holistically. An advertiser budget B_i includes seeding costs as well as user engagements. Thus, $B_i/(cpe(i) * n)$ is the maximum percentage of user engagements the advertiser can expect, so we can regard this as a proxy for the demand from advertiser i. We use $M = \sum_{i\in[h]} M_i$ to denote the total demand of a social advertising market, where $M_i = B_i/(n \cdot cpe(i))$. For simplicity, we assume $cpe(i) = 1$ for all $i \in [h]$. In Fig. 7(a)-(b), we use the Flixster dataset to study the impact of M on revenue and total seed cost under the linear cost model, where we set $h = 10$, $\alpha = 0.1$, and the individual

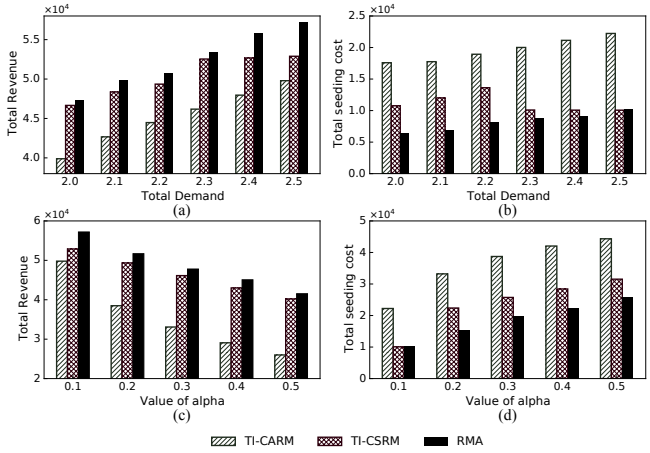

Figure 7: Comparing revenue and seed cost under the case where the advertising demand is controlled holistically.

demands $M_i : i \in [h]$ are all randomly generated such that they sum to M. The results in Fig. 7(a)-(b) show that, the revenues of all algorithms increase with M, as more elements can be selected under a larger demand, while RMA always achieves a better revenue with smaller seed costs than the other algorithms, for all values of M tested. In Fig. 7(c)-(d), we further study the relationship between parameter α and total revenue and seeding cost, for a fixed total demand $M = 2.5$, with all other settings being the same as those in Fig. 7(a)-(b). The results show that (i) RMA outperforms the other algorithms again and (ii) the revenue of all algorithm decreases when α increases, since the node costs increase with α, causing fewer seed nodes to be selected. In summary, the results in Fig. 7 demonstrate that from the perspective of total advertising demand, RMA still exhibits a performance superior to the baselines.

6 RELATED WORK

Influence Maximization. Kempe et al. [46] study influence maximization (IM) where the aim is to select k seed nodes in a social network such that the expected spread is maximized. They propose a simple greedy algorithm with $(1 - 1/e - \epsilon)$ approximation, but their algorithms are based on Monte-Carlo sampling hence have high time complexity. Since then, there has been considerable research on improving algorithms for IM [4, 11, 12, 17–21, 23, 28–32, 34, 40, 45, 49, 51, 58, 60–62, 65, 67–71]. In particular, Borgs et al. [12] proposed Reverse-Reachable Sets that can efficiently estimate influence spread with accuracy guarantee, based on which several studies [34, 60, 65, 67, 68] propose more efficient algorithms for IM while still achieving $(1 - 1/e - \epsilon)$-approximation. Moreover, several variants of IM have been studied, such as topic-aware [9, 16], competition [10, 57], adaptive solutions [35, 38, 64]. However, these studies concentrate on seed selection for submodular optimization with a single cardinality or knapsack constraint.

Social Advertising. Compared to influence maximization, the studies on social advertising are relatively few. Chalermsook et al. [14] study the revenue maximization problem for a social network platform with multiple advertisers, where each advertiser has a cardinality constraint on their seed set. The presence of cardinality constraint on considerably simplifies the problem, while its absence in our setting poses a significant challenge. Aslay et al. [6] study the regret minimization problem in social advertising, where the regret is defined as the the difference between the advertisers' budgets and the expected revenue achieved by social advertising. Their setting do not consider seed user costs. Alon et al. [2] and Abbassi et al. [1] investigate the channel allocation and user ordering problems in social advertising, respectively. Both works do not consider viral propagation. Moreover, none of these studies [1, 2, 6, 14] consider seed user costs. Some recent work [39, 66] focus on profit maximization combining the benefit of influence spread with the cost of seed selection or information propagation, albeit in a single advertiser setting. Due to the differences in problem definitions, techniques developed for these problems are inapplicable to our RM problem. The work closest to ours is by Aslay et al. [5]. As discussed in Section 2.2, their solutions have several limitations. In this paper, we develop efficient approximation algorithms that are theoretically and empirically superior to their solutions on the solution quality and computational efficiency.

Submodular Optimization. The theory of submodular optimization has been extensively studied. Nemhauser et al. [59] and Khuller et al. [47] investigate the submodular function maximization problem under a single cardinality or knapsack constraint. Other studies [7, 13, 15, 24, 27, 48] propose submodular optimization algorithms under more complex constraints such as matroid and multilinear constraints. However, as indicated in [5], our problem is intrinsically a submodular maximization problem under a matroid constraint and multiple submodular knapsack constraints, which are more complex than the constraints in the previous proposals: although we may use an independence system to model our constraints as in [5], the resulting approximation could be instance dependent and arbitrarily small as that in [5, 44]. In contrast, our algorithms achieve a much better approximation ratio which is independent of the network instance, by exploiting the special problem structure of social advertising.

7 CONCLUSION

We have studied the revenue maximization problem in social advertising where multiple advertisers pay the social network platform to disseminate their ads. Previous work on this problem presents algorithms with weak approximation ratios, and they incur large computational overheads in practice. We provide algorithms with significantly better approximation ratios, which can be efficiently implemented. We also conduct extensive experiments using four public datasets to compare our algorithms with the existing ones, and the experimental results demonstrate the superiority of our algorithms both on the running time and on the revenue gained by the social network platform, which our algorithm achieves at considerably less seed incentive cost compared to previous algorithms. For future work, we aim to take into account the natural competitive and complementary relationships between different propagating entities from multiple advertisers, e.g., competition between iPhone 11 and Samsung Galaxy S20, and complementarity between iPhone 11 and Apple Watch.

ACKNOWLEDGEMENT

Lakshmanan's research was supported in part by a discovery grant and a discovery accelerator supplement grant from NSERC (Canada).

REFERENCES

[1] Zeinab Abbassi, Aditya Bhaskara, and Vishal Misra. 2015. Optimizing Display Advertising in Online Social Networks. In *Proc. WWW*. 1–11.

[2] Noga Alon, Iftah Gamzu, and Moshe Tennenholtz. 2012. Optimizing budget allocation among channels and influencers. In *Proc. WWW*. 381–388.

[3] Animalz. 2020. *Viral marketing: What it is and how to make it work for you.* Retrieved August 12, 2020 from https://adespresso.com/blog/viral-marketing-examples-tips/

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

[5] Çigdem Aslay, Francesco Bonchi, Laks V. S. Lakshmanan, and Wei Lu. 2017. Revenue Maximization in Incentivized Social Advertising. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1238–1249, arXiv:1612.00531.

[6] Çigdem Aslay, Wei Lu, Francesco Bonchi, Amit Goyal, and Laks V. S. Lakshmanan. 2015. Viral Marketing Meets Social Advertising: Ad Allocation with Minimum Regret. *Proceedings of the VLDB Endowment* 8, 7 (2015), 822–833.

[7] Ashwinkumar Badanidiyuru and Jan Vondrák. 2014. Fast algorithms for maximizing submodular functions. In *Proc. ACM-SIAM SODA*. 1497–1514.

[8] Nicola Barbieri and Francesco Bonchi. 2014. Influence maximization with viral product design. In *Proc. SIAM SDM*. 55–63.

[9] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2012. Topic-aware social influence propagation models. In *ICDM*. 81–90.

[10] Shishir Bharathi, David Kempe, and Mahyar Salek. 2007. Competitive Influence Maximization in Social Networks. In *Proc. WINE*. 306–311.

[11] Song Bian, Qintian Guo, Sibo Wang, and Jeffrey Xu Yu. 2020. Efficient algorithms for budgeted influence maximization on massive social networks. *Proceedings of the VLDB Endowment* 13, 9 (2020), 1498–1510.

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

[13] Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2011. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM J. Comput.* 40, 6 (2011), 1740–1766.

[14] Parinya Chalermsook, Atish Das Sarma, Ashwin Lall, and Danupon Nanongkai. 2015. Social Network Monetization via Sponsored Viral Marketing. In *Proc. ACM SIGMETRICS*. 259–270.

[15] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. 2014. Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes. *SIAM J. Comput.* 43, 6 (2014), 1831–1879.

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

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

[18] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient Influence Maximization in Social Networks. In *Proc. ACM KDD*. 199–208.

[19] Wei Chen, Yifei Yuan, and Li Zhang. 2010. Scalable Influence Maximization in Social Networks Under the Linear Threshold Model. In *Proc. IEEE ICDM*. 88–97.

[20] Suqi Cheng, Huawei Shen, Junming Huang, Wei Chen, and Xueqi Cheng. 2014. IMRank: Influence Maximization via Finding Self-consistent Ranking. In *Proc. ACM SIGIR*. 475–484.

[21] Suqi Cheng, Huawei Shen, Junming Huang, Guoqing Zhang, and Xueqi Cheng. 2013. StaticGreedy: Solving the Scalability-accuracy Dilemma in Influence Maximization. In *Proc. ACM CIKM*. 509–518.

[22] Gary K. Clark. 2020. *1000 Social Media Marketing Secrets: Viral Advertising and Personal Brand Secrets to Grow Your Business with YouTube, Facebook, Instagram - Become an Influence with Over One Million Followers.* Aprilis Publishing LLC. https://www.amazon.com/1000-Social-Media-Marketing-Secrets/dp/164745042X

[23] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. 2014. Sketch-Based Influence Maximization and Computation: Scaling Up with Guarantees. In *Proc. ACM CIKM*. 629–638.

[24] Michele Conforti and Gérard Cornuéjols. 1984. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete applied mathematics* 7, 3 (1984), 251–274.

[25] Robert Denares. 2020. *Instagram Marketing For Viral Influence: Proven Secrets To Build Personal Branding For Business Advertising And 10x Your Followers To Make Money Online: Easy Influencer Guide Beginners Friendly!* Independently published. https://www.amazon.com/Instagram-Marketing-Viral-Influence-Advertising/dp/B086MM2HNP

[26] Blake Droesch. 2019. *How much are brands paying influencers?* Retrieved July 16, 2019 from https://www.emarketer.com/content/how-much-are-brands-paying-influencers

[27] Yuval Filmus and Justin Ward. 2012. A Tight Combinatorial Algorithm for Submodular Maximization Subject to a Matroid Constraint. In *Proc. IEEE FOCS*. 659–668.

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

[29] Sainyam Galhotra, Akhil Arora, Srinivas Virinchi, and Shourya Roy. 2015. ASIM: A Scalable Algorithm for Influence Maximization Under the Independent Cascade Model. In *Proc. WWW Companion*. 35–36.

[30] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. 2011. A Data-based Approach to Social Influence Maximization. *Proceedings of the VLDB Endowment* 5, 1 (2011), 73–84.

[31] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. 2011. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In *Proc. WWW Companion*. 47–48.

[32] Amit Goyal, Wei Lu, and Laks V. S. Lakshmanan. 2011. SIMPATH: An Efficient Algorithm for Influence Maximization Under the Linear Threshold Model. In *Proc. IEEE ICDM*. 211–220.

[33] Tobias Grubenmann, Reynold CK Cheng, and Laks VS Lakshmanan. 2020. TSA: A truthful mechanism for social advertising. In *Web Search and Data Mining (WSDM)*. 214–222.

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

[35] Kai Han, Keke Huang, Xiaokui Xiao, Jing Tang, Aixin Sun, and Xueyan Tang. 2018. Efficient Algorithms for Adaptive Influence Maximization. *Proceedings of the VLDB Endowment* 11, 9 (2018), 1029–1040.

[36] Kai Han, Benwei Wu, Jing Tang, Shuang Cui, Çigdem Aslay, and Laks V. S. Lakshmanan. 2021. Efficient and Effective Algorithms for Revenue Maximization in Social Advertising (Technical Report). https://www.dropbox.com/s/losewmcsmynjk39/RevMaxFullVersion.pdf.

[37] Itai Himelboim and Guy J Golan. 2019. A social networks approach to viral advertising: The role of primary, contextual, and low influencers. *Social Media+ Society* 5, 3 (2019), 2056305119847516.

[38] 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. *The VLDB Journal* (2020).

[39] Keke Huang, Jing Tang, Xiaokui Xiao, Aixin Sun, and Andrew Lim. 2020. Efficient Approximation Algorithms for Adaptive Target Profit Maximization. In *Proc. IEEE ICDE*. 649–660.

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

[41] Andrew Hutchinson. 2016. *Facebook adds new tools to amplify word-of-mouth recommendations, boost response.* Retrieved October 19, 2016 from https://www.socialmediatoday.com/social-business/facebook-adds-new-tools-amplify-word-mouth-recommendations-boost-response

[42] Instagram. 2020. *Doing more to support creators on instagram.* Retrieved May 27, 2020 from https://about.instagram.com/blog/announcements/supporting-creators-on-instagram

[43] Rishabh Krishnan Iyer. 2015. *Submodular optimization and machine learning: Theoretical results, unifying and scalable algorithms, and applications.* Ph.D. Dissertation.

[44] Rishabh K. Iyer and Jeff A. Bilmes. 2013. Submodular Optimization with Submodular Cover and Submodular Knapsack Constraints. In *Proc. NIPS*. 2436–2444.

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

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

[47] Samir Khuller, Anna Moss, and Joseph Naor. 1999. The Budgeted Maximum Coverage Problem. *Inform. Process. Lett.* 70, 1 (1999), 39–45.

[48] Ariel Kulik, Hadas Shachnai, and Tami Tamir. 2009. Maximizing submodular set functions subject to multiple linear constraints. In *Proc. ACM-SIAM SODA*. 545–554.

[49] Jong Ryul Lee and Chin Wan Chung. 2014. A Fast Approximation for Influence Maximization in Large Social Networks. In *Proc. WWW Companion*. 1157–1162.

[50] Paige Leskin. 2020. *Instagram will let influencers make money off ads on IGTV content, following in the footsteps of YouTube's revenue-sharing model.* Retrieved May 28, 2020 from https://www.businessinsider.com/instagram-influencers-monetize-igtv-ads-revenue-sharing-content-creators-badges-2020-5

[51] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective Outbreak Detection in Networks. In *Proc. ACM KDD*. 420–429.

[52] Jure Leskovec and Andrej Krevl. 2014. SNAP datasets: Stanford large network dataset collection. *URL: http://snap.stanford.edu/* (2014).

[53] Yuping Liu-Thompkins, Ewa Maslowska, Yuqing Ren, and Hyejin Kim. 2020. Creating, metavoicing, and propagating: A road map for understanding user roles in computational advertising. *Journal of Advertising* 49, 4 (2020), 394–410.

[54] Jon Loomer. 2013. *Everything you need to know about facebook sponsored stories.* Retrieved June 3, 2013 from https://www.jonloomer.com/facebook-sponsored-stories/

[55] Taylor Lorenz. 2020. *Instagram wants its influencers to make more money.* Retrieved May 27, 2020 from https://www.nytimes.com/2020/05/27/style/instagram-influencer-monetization-live-igtv.html
[56] Wei Lu, Francesco Bonchi, Amit Goyal, and Laks VS Lakshmanan. 2013. The bang for the buck: Fair competitive viral marketing from the host perspective. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 928–936.
[57] Wei Lu, Wei Chen, and Laks VS Lakshmanan. 2015. From Competition to Complementarity: Comparative Influence Diffusion and Maximization. *Proceedings of the VLDB Endowment* 9, 2 (2015), 60–71.
[58] Wei-Xue Lu, Peng Zhang, Chuan Zhou, Chunyi Liu, and Li Gao. 2015. Influence Maximization in Big Networks: An Incremental Algorithm for Streaming Subgraph Influence Spread Estimation. In *Proc. IJCAI*. 2076–2082.
[59] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming* 14, 1 (1978), 265–294.
[60] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. 2016. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-Scale Networks. In *Proc. ACM SIGMOD*. 695–710.
[61] Naoto Ohsaka. 2020. The Solution Distribution of Influence Maximization: A High-Level Experimental Study on Three Algorithmic Approaches. In *Proc. ACM SIGMOD*. 2151–2166.
[62] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2014. Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations. In *Proc. AAAI*. 138–144.
[63] Arielle Pardes. 2020. *Instagram will (finally) pay influencers.* Retrieved May 27, 2020 from https://www.wired.com/story/instagram-finally-pay-influencers-badges-igtv-ads/
[64] 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 *Proc. ACM SIGMOD*. 1096–1113.
[65] Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. 2018. Online Processing Algorithms for Influence Maximization. In *SIGMOD*. 991–1005.
[66] Jing Tang, Xueyan Tang, and Junsong Yuan. 2018. Profit Maximization for Viral Marketing in Online Social Networks: Algorithms and Analysis. *IEEE Transactions on Knowledge and Data Engineering* 30, 6 (2018), 1095–1108.
[67] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*. 1539–1554.
[68] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*. 75–86.
[69] Xiaoyang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin, and Chen Chen. 2017. Bring Order into the Samples: A Novel Scalable Method for Influence Maximization. *IEEE Transactions on Knowledge and Data Engineering* 29, 2 (2017), 243–256.
[70] Chuan Zhou, Peng Zhang, Jing Guo, and Li Guo. 2014. An Upper Bound Based Greedy Algorithm for Mining Top-k Influential Nodes in Social Networks. In *Proc. WWW Companion*. 421–422.
[71] Chuan Zhou, Peng Zhang, Jing Guo, Xingquan Zhu, and Li Guo. 2013. UBLF: An Upper Bound Based Approach to Discover Influential Nodes in Social Networks. In *Proc. IEEE ICDM*. 907–916.
[72] Yuqing Zhu, Jing Tang, and Xueyan Tang. 2020. Pricing influential nodes in online social networks. *Proceedings of the VLDB Endowment (PVLDB)* 13, 10 (2020), 1614–1627.