Distributed Influence Maximization for Large-Scale Online Social Networks

Jing Tang
Data Science and Analytics Thrust
The Hong Kong University of Science and Technology
jingtang@ust.hk

Yuqing Zhu
School of Computer Science and Engineering
Nanyang Technological University
yuqing002@ntu.edu.sg

Xueyan Tang
School of Computer Science and Engineering
Nanyang Technological University
asxytang@ntu.edu.sg

Kai Han
School of Computer Science and Technology
University of Science and Technology of China
hankai@ustc.edu.cn

2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00011

***Abstract*—Thanks to billions of users in online social networks (OSNs), viral marketing becomes one of the most effective promotion channels for various new products or campaigns. Influence maximization is a classic problem in viral marketing, which has been extensively studied in the past two decades. Existing algorithms for influence maximization, however, mostly focus on single machine processing. To address the influence maximization problem on a massive scale, we design distributed algorithms via a cluster of machines, which can effectively speed up the computation while maintaining the state-of-the-art $(1-1/\mathrm{e}-\varepsilon)$-approximation guarantee. Our distributed algorithms consist of two building blocks: (i) distributed reverse influence sampling, and (ii) element-distributed maximum coverage. We carry out extensive experiments on real datasets with millions of nodes and billions of edges to demonstrate the scalability of our distributed algorithms for both influence maximization and maximum coverage. In particular, our distributed algorithms accelerate the state-of-the-art IMM algorithm by 31×–56× times using a machine with 64 cores.**

***Index Terms*—Influence maximization; Distributed algorithms; Online social networks**

I. Introduction

Online social networks (OSNs) attract billions of users to share information and communicate. According to statistics, Facebook has almost 2.9 billion monthly active users as of the second quarter of 2021 [52], which is more than one quarter of the world population. Leveraging the word-of-mouth effects, information can be disseminated widely and rapidly through OSNs. Viral marketing is such a typical application in which the demands for new products, campaigns, ideas and innovation are built up virally, i.e., propagating through OSNs as people make recommendations to their friends and followers [16]. Nowadays, thanks to the large number of users in OSNs, the market of OSN is growing steadily. Specifically, Oberlo [49] estimates that the total social media advertising spending by US companies is expected to exceed $\$44$ billion in 2021. Zenith [64] claims that social media ad spend is expected to grow by 25% annually to reach $\$137$ billion in 2021.

To initiate viral information propagation, a set of influential OSN users can serve as *seeds* who are usually offered a special promotion deal or even paid by advertisers. The number of seed users is normally constrained by the marketing costs (typically by a small constant k). Kempe et al. [33] are the first to formulate this problem as a constrained optimization problem referred to as *influence maximization* by introducing two basic diffusion models: *independent cascade (IC)* and *linear threshold (LT)* models. The influence maximization problem aims to identify a set of k seed users that can produce the largest expected size of influence cascade. Kempe et al. [33] show that influence maximization is NP-hard in general, and propose a simple and elegant greedy algorithm that can achieve $(1-1/\mathrm{e})$-approximation due to the submodularity and monotonicity [46] of the influence spread under the IC and LT models. Since then, a plethora of techniques have been proposed to improve the efficiency or effectiveness of influence maximization [1, 6, 9–13, 15, 18, 19, 21–23, 25, 29, 32, 37, 39, 42, 47, 50, 51, 54–56, 60, 61, 63, 66, 67].

One major challenge for influence maximization is that it is #P-hard to compute the exact influence spread under both the IC [10] and LT [11] models. Some sampling methods, such as Monte-Carlo simulation [33] and *reverse influence sampling* (RIS) [6], are used for influence estimation. The state-of-the-art influence maximization solutions [55, 60, 61], which can provide an approximation ratio of $(1-1/\mathrm{e}-\varepsilon)$, are based on the RIS approach. It is a double-edged sword to conduct sampling on a massive scale. Intuitively, more samples can help understand the information propagation better and thus make it easier to select the most influential users. On the other hand, it is highly challenging to process enormous datasets of samples. In particular, it may consume a huge amount of computational resources, e.g., CPU and memory, to estimate the influence spread for large-scale OSNs, which may not be feasible for a single regular machine.

To cope with massive amounts of data, a natural idea is to develop algorithms in a distributed manner to share the workload among multiple machines and accelerate the computation. Kim et al. [34] propose a scalable and parallelizable influence maximization algorithm, referred to as

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00011

Independent Path Algorithm (IPA), for the IC model. Chang et al. [7] propose another parallel algorithm, called Community-based Max Degree (CMD) algorithm, for the IC model. However, these algorithms are heuristics that have unbounded approximation ratio.

Some recent work [43, 48] has designed scalable algorithms for influence estimation of a given seed set that can be implemented in distributed computation frameworks such as MapReduce. These algorithms are based on Monte-Carlo simulation which must start from the seed set given. However, for influence maximization, the seed sets whose influence spreads need to be estimated are not known in advance and are dynamically determined by the seed selection process. Therefore, it is hard to apply these distributed influence estimation algorithms for influence maximization. In this paper, we aim to develop distributed algorithms with $(1-1/\mathrm{e}-\varepsilon)$-approximation for influence maximization.

We consider a distributed implementation of RIS. That is, each machine independently generates random *reverse reachable* (RR) sets. Based on RIS [6], influence maximization is transformed to maximum coverage. Specifically, consider the *set-element* paradigm of maximum coverage. For each RR set, its index is mapped to an *element*, while for each node, the indexes of all the RR sets containing this node are mapped to a *set*. Unlike the set-distributed maximum coverage (which distributes the *sets* to a cluster of machines) [3, 4, 35, 44, 45], distributed-RIS-based seed selection needs element-distributed maximum coverage (which instead distributes the *elements* to the machines). We develop a new framework for element-distributed maximum coverage, which can provide the exact $(1-1/\mathrm{e})$-approximation and hence return $(1-1/\mathrm{e}-\varepsilon)$-approximate solutions for influence maximization.

In summary, we make the following contributions:

- We devise a NEWGREEDI algorithm that can provide the exact $(1-1/\mathrm{e})$-approximation for maximum coverage in an element-distributed manner.
- We propose DIIMM that integrates distributed RIS and NEWGREEDI with the state-of-the-art IMM algorithm to return $(1-1/\mathrm{e}-\varepsilon)$-approximate solutions for influence maximization.
- We carry out extensive experiments on real datasets with millions of nodes and billions of edges to demonstrate the scalability of our distributed algorithms for both influence maximization and maximum coverage.

Organization. Section II defines the problem of influence maximization. Section III elaborates our distributed algorithms for influence maximization and maximum coverage. Section IV presents the experimental evaluation. Section V reviews the related work. Finally, Section VI concludes the paper.

II. PRELIMINARIES

This section introduces the concept of influence spread, and formally defines the problem of influence maximization. For ease of reference, Table I summarizes the notations that are frequently used.

TABLE I: Frequently used notations.

Notation	Description
$G=(V,E)$	a social network with node set V and edge set E
n, m	the numbers of nodes and edges in G, respectively
k	the number of seed nodes to select
ℓ	the number of distributed machines
ε, δ	the parameters for the approximation guarantee
$\sigma(S)$	the influence spread of a node set S
OPT	the maximum influence spread of any size-k node set S
$R, \mathcal{R}$	a random RR set and a set of RR sets
$\{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\}$	a set of RR sets distributed across ℓ machines
θ	the number of RR sets
$F_{\mathcal{R}}(S)$	the fraction of RR sets in $\mathcal{R}$ that are covered by S

A. Influence Spread

An OSN is abstracted as a directed graph $G=(V,E)$ with a set V of nodes and a set E of edges, where $|V|=n$ and $|E|=m$. The nodes represent users in social media and the edges represent connections among users, e.g., friendship on Facebook or followship on Twitter. For each directed edge $\langle u,v\rangle \in E$, we say that v is an *out-neighbor* of u, and u is an *in-neighbor* of v.

To facilitate discussion, we focus on two basic and widely adopted diffusion models, i.e., the *independent cascade (IC)* and *linear threshold (LT)* models [33]. In the IC and LT models, each edge $\langle u,v\rangle$ is associated with a propagation probability $p_{u,v}$, representing the probability for u to influence v. Given a set of seed nodes S, the influence propagation initiated by S is modeled as a discrete-time stochastic process as follows. Initially at time slot 0, the seed nodes S are *activated* and the other nodes are *inactive*. Suppose that node u is first activated at slot i, then u has a *single* chance to activate each outgoing neighbor v at time slot $i+1$, after which u remains active but cannot further activate any other nodes. This influence propagation process continues until no more inactive nodes can be activated. The difference between the IC and LT models lies in the details of node activation:

- *IC model.* When a node u first becomes activated, it has a *single* chance to activate each inactive neighbor v with a probability $p_{u,v}$. After that, u stops activating any other nodes.
- *LT model.* It requires the propagation probabilities to satisfy $\sum_{u\in N_v^{\mathrm{in}}} p_{u,v} \le 1$, where N_v^{in} denotes the set of node v's in-neighbors. Each node v uniformly and randomly selects a threshold λ_v from the interval $[0,1]$. An inactive node v becomes activated only if $\sum_{u\in A_v^{\mathrm{in}}} p_{u,v} \ge \lambda_v$, where $A_v^{\mathrm{in}} \subseteq N_v^{\mathrm{in}}$ denotes the set of v's in-neighbors that are activated.

Let $\sigma(S)$ denote the expected number of nodes activated by the seed set S via the above process, which is known as the *influence spread* of the seed set S.

Example 1. *Fig. 1 shows a social media and three of its possible influence propagation processes initiated by the seed set $\{v_1\}$. Under both the IC and LT models, nodes v_2 and v_3 are guaranteed to be activated at time slot 1 as $p_{v_1,v_2}=p_{v_1,v_3}=1$.*

(a) A Social Media (b) Possible Propagation (i) (c) Possible Propagation (ii) (d) Possible Propagation (iii)

Fig. 1: A social media and three of its possible influence propagation processes initiated by the seed set $\{v_1\}$.

In addition, there are three cases of influence propagation depending on whether and how v_4 is activated. Specifically, in case (i), as shown in Fig. 1(b), all the three nodes v_2, v_3, v_4 are directly activated by v_1 at time slot 1; in case (ii), as shown in Fig. 1(c), v_2 and v_3 are activated by v_1 at time slot 1, and v_4 is activated by v_2 or v_3 at time slot 2; and in case (iii), as shown in Fig. 1(d), v_2 and v_3 are activated by v_1 at time slot 1 while v_4 is not activated by any node.

Under the IC model, case (i) happens with a probability of 0.4 when v_1 successfully activates v_4. Case (ii) happens when v_1 does not activate v_4 at time slot 1 and either v_2 or v_3 activates v_4 at time slot 2. The probability for this case to happen is $(1-0.4)\times\big(1-(1-0.3)\times(1-0.2)\big)=0.264$, where $(1-0.3)\times(1-0.2)$ is the probability that neither v_2 nor v_3 activates v_4. Similarly, case (iii) happens with a probability of $(1-0.4)\times(1-0.3)\times(1-0.2)=0.336$. Consequently, the influence spread of $\{v_1\}$ can be calculated as $\sigma(\{v_1\})=0.4\times4+0.264\times4+0.336\times3=3.664$.

The probability for each propagation case under the LT model is different. Specifically, case (i) happens when the random threshold λ_{v_4} is in the range of $[0, 0.4]$, case (ii) happens when $0.4<\lambda_{v_4}\leq 0.9$, and case (iii) happens when $0.9<\lambda_{v_4}\leq 1$. This indicates that the probabilities for the three cases are 0.4, 0.5 and 0.1, respectively. As a consequence, $\sigma(\{v_1\})=0.4\times4+0.5\times4+0.1\times3=3.9$. □

B. Influence Maximization

Given a graph $G=(V,E)$ with a propagation probability $p_{u,v}$ for each edge $\langle u,v\rangle\in E$, the *influence maximization* problem [33] is to seek for a set S of k nodes that maximizes the influence spread $\sigma(S)$. Formally,

$$\arg\max_{|S|=k}\sigma(S). \tag{1}$$

It is NP-hard to maximize the influence spread in general [33]. Due to the submodularity and monotonicity of influence spread [33], a greedy hill-climbing algorithm can return a $(1-1/\mathrm{e}-\varepsilon)$-approximate solution with high probability [46], where ε is an error threshold parameter. A large amount of recent work [6, 55, 60, 61] aims to improve the efficiency for achieving the same approximation guarantee. These algorithms typically attempt to reduce the number of samples generated for estimating the influence spread as its exact value is #P-hard to compute under both the IC and LT models [10, 11]. Note that NP-hard (e.g., for decision problem) and #P-hard (e.g., for counting problem) are different. Specifically, influence maximization is NP-hard, even assuming the influence spread of any seed set can be computed in polynomial time.

To address the efficiency issue for influence maximization in large-scale OSNs, a natural idea is to develop distributed algorithms. Some recent work [43, 48] has designed scalable algorithms for influence estimation that can be implemented in distributed computation frameworks such as MapReduce. These algorithms generate samples in parallel by a group of machines to estimate the influence spread of a given seed set S in a distributed manner. Then, each machine returns an estimated influence spread of S to a master machine which gathers the information to calculate the final estimated influence spread of S. However, it is hard to directly apply these algorithms for distributed influence maximization. The main reason is that the seed sets that need to be evaluated in influence maximization are potentially large in number and they are not known in a priori. In other words, an influence maximization algorithm may need to query the estimated influence spread of any seed set S on the samples. One straightforward solution is to gather all the samples generated for influence estimation from a cluster of machines in one machine. After that, we can run the greedy seed selection algorithm on these samples in a single machine. Haque and Banerjee [28] naively use such a strategy. However, each machine may generate massive amount of samples so that it is impossible for a single machine to load all the data generated by a cluster of machines. Moreover, the communication cost is also expensive as collecting all the data from a group of machines to one machine can produce a huge amount of network traffic.

To address the above issues, in this paper, we aim to devise distributed seed selection algorithms where the samples used for estimating the influence spread are generated and stored on a group of distributed machines.

III. Distributed Algorithms for Influence Maximization

Computing the exact influence spread $\sigma(S)$ of a seed set S has been shown to be #P-hard for both the IC and LT models [10, 11]. To estimate the influence spread $\sigma(S)$ of a seed set S, some sampling methods have been proposed for unbiased estimation such as the naive Monte-Carlo simulation [33] and the advanced *reverse influence sampling* (RIS) approach [6]. These sampling methods can provide a bounded estimation error which is controllable. Using the sampling methods, at a high level, influence maximization algorithms consist of two phases:

1) **Sampling.** This phase generates a number of samples satisfying the approximation guarantee requirement.
2) **Seed Selection.** This phase applies a seed selection algorithm (which is usually a standard greedy algorithm

due to the submodularity [46]) to construct a size-k node set S that attempts to maximize the estimated influence spread on the samples generated.

In what follows, we show that both phases can be implemented in a distributed manner.

A. Distributed Reverse Influence Sampling

Kempe et al. [33] used the classical Monte-Carlo simulation to estimate the influence spread, which is very inefficient. To address the efficiency problem, Borgs et al. [6] introduced a novel RIS approach to dramatically facilitate the influence estimation.

Definition 1 (RR Set)**.** *A random reverse reachable (RR) set R for a graph G is generated by 1) first selecting a random node $v \in V$, 2) then sampling a random graph g from G according to the diffusion model, 3) finally taking the set of nodes in g that can reach v as R.*

Under the IC and LT models, a random RR set can be efficiently constructed. In particular, under the IC model, a random RR set R on G can be constructed in three steps:

1) Select a node v uniformly at random from V.
2) Perform a stochastic breadth first search (BFS) that starts from v and follows the incoming edges of each node. In particular, for each node u encountered during the BFS, we inspect the set E_u^{in} of incoming edges of u. For each edge $\langle u', u\rangle$ in E_u^{in}, we ignore the edge with $1 - p_{u',u}$ probability; with the other $p_{u',u}$ probability, we allow the BFS to traverse from u to u' (if u' has not been traversed).
3) Insert into R all nodes that are traversed during the stochastic BFS (including v).

Under the LT model, a random RR set R on G can also be generated in three steps:

1) Select a node v uniformly at random from V.
2) Generate a random walk from v that follows the incoming edges of each node. Specifically, at each step of the random walk, we examine the set N_u^{in} of in-neighbors of the current node u, and stop the walk at u with probability $1 - \sum_{w \in N_u^{\text{in}}} p_{u',u}$. With the other $\sum_{u' \in N_u^{\text{in}}} p_{u',u}$ probability, we sample a node u' from a distribution $f(u')$ on N_u^{in} with $f(u') \sim p_{u',u}$, and walk to u' (if u' has not been visited).
3) Insert all nodes visited in the random walk into R.

Example 2. *Consider the OSN in Fig. 1(a), and assume that v_4 is selected. Under the IC model, a random RR set $\{v_1, v_3, v_4\}$ may be constructed by traversing nodes v_1 and v_3 through edges $\langle v_1, v_4\rangle$ and $\langle v_3, v_4\rangle$ (with probability $0.2 \times 0.4 \times (1 - 0.3) = 0.056$). On the other hand, under the LT model, the same random RR set $\{v_1, v_3, v_4\}$ can only be generated by a path $v_4 \to v_3 \to v_1$.* □

Intuitively, for a random RR set generated from a randomly selected node v, a diffusion process starting from a node in the RR set should have a certain probability to activate v. The following lemma formalizes the connection between the influence spread and a random RR set.

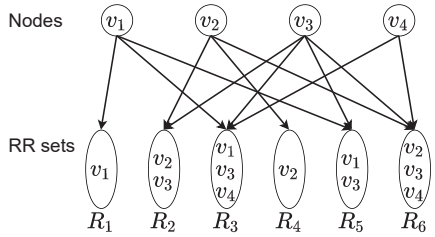


Fig. 2: RR sets and maximum coverage.

Lemma 1 ([6])**.** *Given a fixed seed set $S \subseteq V$, for a random RR set R,*

$$\sigma(S) = n \cdot \Pr[S \cap R \neq \emptyset], \tag{2}$$

where $n = |V|$ is the total number of nodes.

Lemma 1 indicates that the influence spread of a seed set S is proportional to the probability that S intersects with a random RR set. We say that a seed set S *covers* a random RR set R if S contains at least one node in R, i.e., $S \cap R \neq \emptyset$. Given a set $\mathcal{R}$ of random RR sets, let $\Lambda(S)$ denote the *coverage* of S in $\mathcal{R}$ which is the number of RR sets in $\mathcal{R}$ that is covered by S. Then, by Lemma 1, it is easy to see that $n \cdot \Lambda(S)/|\mathcal{R}|$ is an unbiased estimation of S's influence spread $\sigma(S)$. Thus, given a set of random RR sets generated for influence estimation, the influence maximization problem is transformed into a *maximum coverage* problem [17], which aims to select a set S of k nodes to cover the maximum number of RR sets. Specifically, the node set V and the RR sets (aka hyperedges) $\mathcal{R}$ form a hypergraph $\mathcal{H} = (V, \mathcal{R})$. For a node $v \in V$ and a hyperedge $R \in \mathcal{R}$, if $v \in R$, we say that v covers R. A set $S \subseteq V$ covers a hyperedge R if there is at least one node in S that covers R. Thus, maximizing the estimated influence on a set of RR sets is equivalent to selecting a subset $S \subseteq V$ that maximizes the number of RR sets in $\mathcal{R}$ covered by S.

Example 3. *Fig. 2 shows an example of 6 random RR sets generated, e.g., RR set R_3 contains nodes $\{v_1, v_3\}$, node v_1 covers RR sets R_1, R_3, R_5, and node set $\{v_1, v_4\}$ covers RR sets R_1, R_3, R_5, R_6. Suppose that we aim to identify a set of 2 nodes to cover the maximum number of RR sets. It is easy to see that selecting $\{v_1, v_2\}$ can cover all the 6 RR sets, which indicates that $\{v_1, v_2\}$ is the optimal solution.* □

We consider a simple distributed implementation of the sampling phase with ℓ machines available. If θ random RR sets are needed, each machine generates θ/ℓ RR sets concurrently. Note that the total generation time is determined by the longest one. However, when each machine generates a sufficiently large number of RR sets, due to the randomness, the difference in terms of generation time among all the machines is negligible as shall be formally analyzed later.

B. New Framework for Distributed Maximum Coverage

Feige [17] has shown that there does not exist any $(1 - 1/\mathrm{e} + \varepsilon)$-approximation algorithm running in polynomial time for maximum coverage for any positive ε, unless $\mathrm{P} = \mathrm{NP}$. This indicates that the naive greedy method that selects a set (a

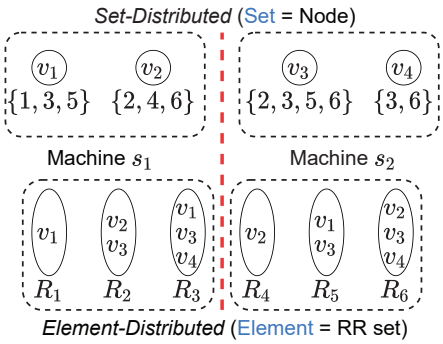


Fig. 3: Comparison between set-distributed approach and element-distributed approach.

seed in our context) covering the maximum possible number of new elements (RR sets in our context) in each iteration is the best approximation algorithm for maximum coverage in that it can return a $(1-1/\mathrm{e})$-approximate solution. In this paper, we utilize the greedy algorithm and implement it in a distributed manner.

We consider the *set-element* paradigm, where a *set* corresponds to a node and an *element* corresponds to an RR set sample. In this section, we analyze the deficiency of the existing *set-distributed* approach, and propose a novel *element-distributed* approach to cope with the context of influence maximization. Fig. 3 shows how the set-distributed approach and the element-distributed approach handle the simple example of RR sets.

1) Deficiency of Set-Distributed Approach: Maximum coverage is a typical application of submodular maximization. In the past few years, a significant amount of research has studied submodular maximization in a distributed fashion, such as GREEDYSCALING [35], GREEDI [45], PSEUDOGREEDY [44], RANDGREEDI [3], and PARALLELALG [4]. A key technique used in these studies is composable core-sets (or its randomized version) [44]. Table II summarizes the main results of the existing work in our context, where θ is the number of RR sets, ℓ is the number of machines, κ is the size of core-sets, and k is the size of the target seed set. We note that the existing composable core-sets approach cannot achieve the exact $(1-1/\mathrm{e})$-approximation unless a centralized machine has a copy of all the RR sets which degenerates to centralized maximum coverage.

More importantly, the composable core-sets approach uses the conventional *set-distributed* strategy, which is incompatible with distributed RIS. At a high level, the reason is that the former distributes the nodes in the OSN to each machine whereas the latter distributes the RR sets to each machine. In other words, the former is *set-distributed* whereas the latter is *element-distributed*. Specifically, to apply the composable core-sets approach, if a node v is assigned to a machine s_i, all the indexes $I(v)$ of those RR sets covered by v should also be managed by machine s_i. On the other hand, with the parallel sample generation described above, each machine s_i has the full information of all the RR sets generated by itself but it does not know whether a node v can cover any RR sets generated by other machines. Therefore, in distributed RIS, the nodes appearing in the same RR set R are recorded by one machine, while in composable core-sets, all the indexes of RR sets covered by the same node v must be recorded by one machine. This implies that the set-distributed method requires gathering all samples together. However, as discussed in Section II-B, gathering all the samples generated for influence estimation from a cluster of machines to one machine incurs some fundamental issues—(i) it might be impossible for a single machine to load all the data generated by a cluster of machines, and (ii) the communication cost is expensive as collecting all the data from a group of machines to one machine can produce a huge amount of network traffic.

TABLE II: Summary of the state of the art via (randomized) composable core-sets for distributed maximum coverage.

Algorithm	Rounds	Approximation
GREEDYSCALING [35]	$O\big(\log(\theta)/\varepsilon\big)$	$1-1/\mathrm{e}-\varepsilon$
GREEDI [45]	1	$\frac{(1-\mathrm{e}^{-\kappa/k})(1-1/\mathrm{e})}{\min\{\ell,k\}}$
PSEUDOGREEDY [44]	1	$0.545-O(\frac{1}{k})$
RANDGREEDI [3]	1	$\frac{1}{2}(1-1/\mathrm{e})$
PARALLELALG [4]	$O(1/\varepsilon)$	$1-1/\mathrm{e}-\varepsilon$

2) New Solution of Element-Distributed Approach: To remedy the deficiency of the set-distributed approach that all RR sets covered by a single node should be managed by a single machine, in this paper, we propose a novel *element-distributed* method NEWGREEDI (Algorithm 1) that is tailored for distributed RIS. NEWGREEDI initially labels all the RR sets as *uncovered* (line 2) and calculates the marginal coverage $\Delta(v)$ of every node $v \in V$ by aggregating the marginals from all machines (line 4). To implement the greedy algorithm efficiently (which is linear to the total size of hyperedges), we create a vector D of node lists to categorize each node by its coverage. Specifically, each node v is assigned to the list $D(\Delta(v))$ (line 5). We then scan the vector D in decreasing order of coverage and find the node u with the largest marginal coverage. To avoid unnecessary updates of D, we apply a lazy-update strategy that moves node u to a new list only when the recorded coverage is found outdated during the scan (lines 9–11). If the recorded coverage of u in D is up-to-date, we add u to S (line 12). The algorithm returns the solution S when k nodes are selected (line 13). Otherwise, it updates the marginal coverage for each node via a MapReduce-like procedure (lines 14–22). In the *map* stage, each machine s_i only updates the marginal coverages for those nodes affected by selecting u as a new seed. Specifically, let $\mathcal{R}_i$ denote the set of RR sets generated by machine s_i. Let $I_i(u)$ denote a set of indexes of RR sets that are covered by the new seed u on machine s_i, i.e., $I_i(u) := \{j\colon u \in R_{i,j} \wedge 1 \le j \le |\mathcal{R}_i|\}$, where $R_{i,j}$ is the j-th RR set in $\mathcal{R}_i$. We check through $I_i(u)$ for the RR sets newly covered by the new seed u. For each such RR set $R_{i,j}$, the marginal coverage of all the nodes in $R_{i,j}$ is to be decreased. Then, each machine s_i just needs to respond with a vector of tuples $(\langle v_1, \Delta_i(v_1)\rangle, \langle v_2, \Delta_i(v_2)\rangle, \dots)$ to indicate

Algorithm 1: NEWGREEDI($\{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\}, k$)

Input: A set $\{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\}$ of RR sets across ℓ machines, seed size k
Output: A size-k seed set S

1 **foreach** *machine* s_i **do**
2 label all the RR sets in $\mathcal{R}_i$ as *uncovered*;
3 record the coverage $\Delta_i(v)$ of every node v on machine s_i;
4 compute (at the master machine) $\Delta(v) \leftarrow \sum_{i=1}^{\ell} \Delta_i(v)$ for each v;
5 create a vector D such that $D(d)$ is a list of nodes with coverage d;
6 let $d^\star \leftarrow \max_{v \in V} \Delta(v)$;
7 **for** $d \leftarrow d^\star$ **to** 1 **do**
8 **foreach** *node* $u \in D(d)$ **do**
9 **if** $d > \Delta(u)$ **then** // outdated coverage
10 insert u to $D(\Delta(u))$;
11 **continue**;
12 add u into S;
13 **if** k *nodes are selected* **then return** solution S;
  // map stage
14 **foreach** *machine* s_i **do**
15 initialize an empty hash-map Δ_i;
16 **foreach** *RR set index* $j \in I_i(u)$ **do**
17 **if** *the RR set* $R_{i,j}$ *is uncovered* **then**
18 **foreach** *node* $v \in R_{i,j}$ **do**
19 **if** $v \notin \Delta_i$ **then** $\Delta_i(v) \leftarrow 0$;
20 $\Delta_i(v) \leftarrow \Delta_i(v) + 1$;
21 label $R_{i,j}$ as *covered*;
  // reduce stage
22 update $\Delta(v) \leftarrow \Delta(v) - \Delta_i(v)$ for every s_i and every $v \in \Delta_i$;

that the marginal coverage of each node v_j is to be decreased by $\Delta_i(v_j)$. It can dramatically save the traffic compared to directly responding with the new marginal coverages for all the nodes. Finally, in the *reduce* stage, it aggregates the changes collected from all the machines and updates the marginal coverages accordingly (line 22).

We show that the NEWGREEDI algorithm (Algorithm 1) can achieve the exact $(1 - 1/\mathrm{e})$-approximation guarantee for maximum coverage.

Lemma 2. *The* NEWGREEDI *algorithm returns a* $(1 - 1/\mathrm{e})$*-approximate solution in a distributed fashion.*

Proof. For each node v, let d_v denote the index of v in D, i.e., $v \in D(d_v)$. Due to the submodularity, we know that $d_v \geq \Delta(v)$ as the marginal coverage decreases along with the base set S. Thus, for each node u added to S (line 12) and any node $v \in V \setminus S$, $\Delta(u) = d_u \geq d_v \geq \Delta(v)$. In addition, let u_j be the j-th node selected and $S_j = \{u_1, \ldots, u_j\}$. Due to the monotonicity and submodularity, we have

$$\Lambda(S_j) + k \cdot \Delta(u_{j+1}) \geq \Lambda(S_j) + \sum\nolimits_{v \in S^\circ \setminus S_j} \Delta(v) \geq \Lambda(S^\circ),$$

where S° denotes the optimal solution. Rearranging it yields

$$\Lambda(S^\circ) - \Lambda(S_{j+1}) \leq (1 - 1/k) \cdot \big(\Lambda(S^\circ) - \Lambda(S_j)\big).$$

Recursively, we have $\Lambda(S^\circ) - \Lambda(S_k) \leq (1 - 1/k)^k \cdot \Lambda(S^\circ)$, which concludes the lemma of $\Lambda(S_k) \geq (1 - 1/\mathrm{e}) \cdot \Lambda(S^\circ)$. □

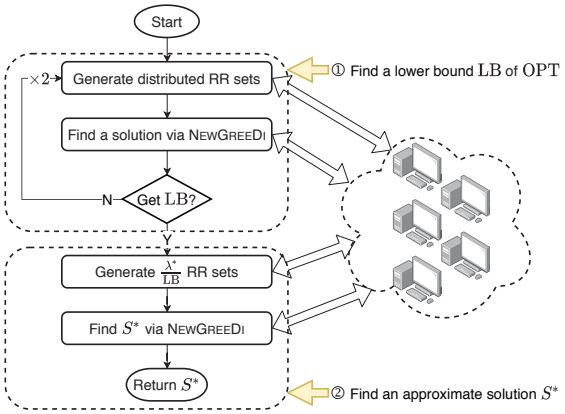


Fig. 4: Flowchart of DIIMM.

There are a total of k iterations in NEWGREEDI. In many applications, k is small (e.g., k is usually less than 100 for influence maximization to leverage the power of viral marketing), making this approach practical for MapReduce style computations.

C. Integrating with the State of the Art

There are several frameworks that are designed to provide the $(1 - 1/\mathrm{e} - \varepsilon)$-approximation guarantee with probability at least $1 - \delta$ for influence maximization, where $\varepsilon, \delta \in (0, 1)$ are user-specified parameters. Among these frameworks, IMM [61], SSA [47], OPIM-C [55], and SUBSIM [25] are the state-of-the-art ones. Our distributed RIS and NEWGREEDI approaches are compatible with all the aforementioned frameworks. By applying our distributed techniques, these frameworks can be horizontally scaled to handle OSNs of massive scale. In what follows, we elaborate how to implement IMM [61] in a distributed fashion using our proposed techniques. In particular, IMM consists of two phases, i.e., sampling and node selection. At a high level, IMM iteratively generates RR sets in the sampling phase and derives the size-k maximum coverage set in the node selection phase until a stopping condition is met to yield the solution with good estimation quality.

Fig. 4 overviews the integration of distributed RIS and NEWGREEDI with IMM [61], referred to as DIIMM. In a nutshell, DIIMM first derives a lower bound LB of OPT that is asymptotically tight, and then generates a number of $\theta = \lambda^* / \mathrm{LB}$ RR sets to construct the final solution S^*, where OPT is the maximum influence spread of any size-k node set and the choice of λ^* will be given shortly. The rationale behind is that a solution constructed with $\theta = \lambda^* / \mathrm{OPT}$ random

Algorithm 2: DIIMM$(G, \ell, k, \varepsilon, \delta)$

Input: Graph G, number of machines ℓ, seed size k, error threshold ε, and failure probability threshold δ

Output: A size-k seed set S^* that provides an approximation ratio of $(1 - 1/\mathrm{e} - \varepsilon)$ with probability at least $1 - \delta$

1 initialize $\mathcal{R}_i \leftarrow \emptyset$ for every machine s_i and LB $\leftarrow 1$;
2 let $\varepsilon' \leftarrow \sqrt{2} \cdot \varepsilon$ and δ' be as defined in (7);
3 **for** $t \leftarrow 1$ **to** $\log_2 n - 1$ **do**
4 let $x \leftarrow n/2^t$;
5 let $\theta_t \leftarrow \lambda'/x$, where λ' is as defined in (3);
6 each machine s_i adds $(\theta_t - \theta_{t-1})/\ell$ new RR sets to $\mathcal{R}_i$;
7 let $S_t \leftarrow$ NEWGREEDI$(\{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\}, k)$;
8 **if** $n \cdot F_{\mathcal{R}}(S_t) \geq (1 + \varepsilon') \cdot x$ **then**
9 LB $\leftarrow n \cdot F_{\mathcal{R}}(S_t)/(1 + \varepsilon')$;
10 **break**;
11 let $\theta \leftarrow \lambda^*/\mathrm{LB}$, where λ^* is as defined in (6);
12 each machine s_i adds $(\theta_t - \theta_{t-1})/\ell$ new RR sets to $\mathcal{R}_i$;
13 let $S^* \leftarrow$ NEWGREEDI$(\{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\}, k)$;
14 **return** S^*;

RR sets via the greedy approach for maximum coverage can provide an approximation guarantee of $1 - 1/\mathrm{e} - \varepsilon$ with at least $1 - \delta$ probability [61]. However, recall that it is NP-hard to obtain OPT. To ensure at least λ^*/OPT RR sets are generated, we find a tight lower bound LB of OPT and generate a number of λ^*/LB RR sets. Both stages are implemented in a distributed fashion utilizing our distributed RIS and NEWGREEDI approaches.

Algorithm 2 presents the pseudo-code of DIIMM. To find a tight lower bound LB of OPT (lines 1–10), DIIMM initially sets LB $= 1$, a parameter $\varepsilon' = \sqrt{2} \cdot \varepsilon$ and a parameter δ' as defined in (7), and iteratively increases the RR set number until a satisfactory solution is identified. Specifically, in the t-th iteration, a total number of $\theta_t = \lambda' \cdot 2^t/n$ RR sets (including the RR sets generated in the previous iterations) are generated across ℓ machines (lines 4–6), where

$$\lambda' = \frac{(2 + \frac{2}{3}\varepsilon') \cdot \left(\ln \binom{n}{k} + \ln(2/\delta') + \ln \log_2 n\right) \cdot n}{\varepsilon'^2}. \quad (3)$$

Based on these RR sets, a solution S_t is constructed via our NEWGREEDI algorithm (line 7). Let $F_{\mathcal{R}}(S_t)$ be the fraction of RR sets in $\mathcal{R}$ that are covered by S_t. When S_t's estimated influence spread $n \cdot F_{\mathcal{R}}(S_t)$ is no less than $(1 + \varepsilon') \cdot n/2^t$ (line 8), $n \cdot F_{\mathcal{R}}(S_t)/(1 + \varepsilon')$ is chosen as a lower bound of OPT (line 9). Tang et al. [61] show that such a lower bound LB is close to OPT with a high probability.

After a lower bound LB of OPT is obtained, DIIMM then generates a number of $\theta = \lambda^*/\mathrm{LB}$ random RR sets in a distributed manner and applies NEWGREEDI to construct the final solution S^* (lines 11–13). Specifically, λ^* is a function of k, n, ε, and δ such that

$$\alpha = \sqrt{\ln(2/\delta') + \ln 2}, \quad (4)$$

$$\beta = \sqrt{(1 - 1/\mathrm{e}) \cdot \left(\ln \binom{n}{k} + \ln(2/\delta') + \ln 2\right)}, \quad (5)$$

$$\lambda^* = 2n \cdot \left((1 - 1/\mathrm{e}) \cdot \alpha + \beta\right)^2 \cdot \varepsilon^{-2}, \quad (6)$$

$$\delta' \text{ is the root of } \lceil \lambda^* \rceil \cdot \delta' = \delta. \quad (7)$$

Note that in the original IMM algorithm [61], $\delta' = \delta$. However, Chen [8] has pointed out that there is a subtle issue in the martingale analysis of the IMM algorithm. Setting the parameter δ' as defined in (7) can fix the issue [8], which requires minor changes on the algorithm (line 2) and incurs a slight penalty on the running time of the algorithm.

Similar to IMM [8, 61], DIIMM can provide the following strong theoretical guarantees.

Theorem 1. *The* DIIMM *algorithm returns a* $(1 - 1/\mathrm{e} - \varepsilon)$*-approximate solution with at least* $1 - \delta$ *probability.*

Proof. For any size-k set S being independent of $\mathcal{R}$, we have

$$\begin{aligned} &\Pr[n \cdot F_{\mathcal{R}}(S) \geq (1 + \varepsilon') \cdot x \wedge \mathrm{OPT} < x] \\ &\leq \Pr[n \cdot F_{\mathcal{R}}(S) > \sigma(S) + \varepsilon' \cdot x \mid \mathrm{OPT} < x] \\ &\leq \mathrm{e}^{-\frac{(\varepsilon' x/n)^2 \lambda'/x}{(2 + 2\varepsilon'/3)\sigma(S)/n}} = \mathrm{e}^{-\frac{x \ln \delta''}{\sigma(S)}} \leq \delta'', \end{aligned}$$

where $\delta'' = \delta'/(2\binom{n}{k} \log_2 n)$ and the second inequality is due to martingale inequalities [61]. Similarly,

$$\begin{aligned} &\Pr[n \cdot F_{\mathcal{R}}(S) > (1 + \varepsilon') \cdot \mathrm{OPT} \wedge \mathrm{OPT} \geq x] \\ &\leq \Pr[n \cdot F_{\mathcal{R}}(S) > \sigma(S) + \varepsilon' \cdot \mathrm{OPT} \mid \mathrm{OPT} \geq x] \\ &\leq \mathrm{e}^{-\frac{(\varepsilon' \mathrm{OPT}/n)^2 \lambda'/x}{(2 + 2\varepsilon'/3)\sigma(S)/n}} = \mathrm{e}^{-\frac{\mathrm{OPT}^2 \ln \delta''}{\sigma(S) x}} \leq \delta''. \end{aligned}$$

In addition, OPT $<$ LB only if (i) condition $n \cdot F_{\mathcal{R}}(S_t) \geq (1 + \varepsilon') \cdot x$ is met in any iteration t such that OPT $< x$, or (ii) OPT $< \frac{n}{1+\varepsilon'} \cdot F_{\mathcal{R}}(S_t)$ when OPT $\geq x$, no matter what S_t is. Thus, since there are $\binom{n}{k}$ size-k node sets and the total iterations with OPT $< x$ is at most $\log_2 n - 1$, by the union bound, $\Pr[\mathrm{OPT} < \mathrm{LB}] \leq \binom{n}{k} \cdot \log_2 n \cdot \delta'' = \delta'/2 \leq \delta/2$.

It is known that when a fixed number θ of random RR sets are generated such that $\theta \geq \lambda^*/\mathrm{OPT}$, the greedy algorithm returns a $(1 - 1/\mathrm{e} - \varepsilon)$-approximate solution with probability at least $1 - \delta'/2$ [61]. By Lemma 2, our NEWGREEDI algorithm obtains the same solution as the greedy algorithm. In addition, when OPT $\geq$ LB so that $\theta \geq \lambda^*/\mathrm{OPT}$, we have $\Pr[\sigma(S^*) < (1 - 1/\mathrm{e} - \varepsilon) \cdot \mathrm{OPT}] \leq \lceil \lambda^* \rceil \cdot \delta'/2 = \delta/2$, as the possible setting of θ is an integer in the range of $[1, \lceil \lambda^* \rceil]$ (line 11).

Putting it together, the lemma is proved. □

The main difference between DIIMM and IMM lies in the way that they generate RR sets to construct the solution. In particular, DIIMM generates RR sets (lines 6 and 12) and constructs the solution (lines 7 and 13) in parallel by a group of ℓ machines. Recall that NEWGREEDI initially calculates the marginal coverage $\Delta(v)$ of every node $v \in V$ by aggregating the marginals from all machines (line 4 of Algorithm 1), which may require each machine to directly

respond with the coverages for all the nodes. Alternatively, whenever NEWGREEDI is called after generating new RR sets, each machine s_i can initially respond with a vector of tuples $(\langle v_1, \Delta_i(v_1)\rangle, \langle v_2, \Delta_i(v_2)\rangle, \dots)$ to indicate the coverage $\Delta_i(v_j)$ of each node v_j over the newly generated RR sets. Then, for each node v, aggregating its marginals in such tuples as well as its coverage for the previously generated RR sets gives v's new coverage. As DIIMM calls NEWGREEDI multiple times (lines 7 and 13), this approach is able to further save communication traffic.

D. Complexity Analysis

In what follows, we analyze the communication complexity and time complexity of the NEWGREEDI (Algorithm 1) and DIIMM (Algorithm 2) algorithms.

Communication Complexity. We consider a master-slave paradigm of distributed framework. In particular, for each call of NEWGREEDI, each slave machine needs to respond to the master machine with a vector of tuples $(\langle v_1, \Delta_i(v_1)\rangle, \langle v_2, \Delta_i(v_2)\rangle, \dots)$ for k times. The length of such a vector is at most n, where n is the number of nodes in G. Thus, the communication complexity of NEWGREEDI for each slave (resp. the master) machine is $O(kn)$ (resp. $O(\ell kn)$, where ℓ is the number of machines). In addition, the communication overhead of DIIMM is determined by the call of NEWGREEDI. There are at most $\log_2 n$ calls of NEWGREEDI by DIIMM. Therefore, the total communication complexity of DIIMM for each slave (resp. the master) machine is $O(kn \log_2 n)$ (resp. $O(\ell kn \log_2 n)$).

Time Complexity. We first analyze the time complexity of NEWGREEDI. It is easy to see that for each hyperedge R, the link between a node v and an RR set R can be visited at most twice, i.e., one for forward visit (line 16 of Algorithm 1) and the other for backward visit (line 18 of Algorithm 1). Thus, the time complexity of NEWGREEDI for machine s_i is linear to the total size of its hyperedges, i.e., $O(\sum_{R\in\mathcal{R}_i}|R|)$. Let EPS be the expected size of an RR set, which is given in the following lemma.

Lemma 3. *Under the* triggering model *[33] (which generalizes both the IC and LT models), the expected size of a random RR set is*

$$\mathrm{EPS} = \frac{1}{n}\sum_{v\in V}\sigma(\{v\}), \tag{8}$$

where n is the number of nodes.

Proof. For any given node v and a random RR set R, by Lemma 1, we have

$$\sigma(\{v\}) = n \cdot \Pr[\{v\} \cap R \neq \emptyset]. \tag{9}$$

Thus,

$$\frac{1}{n}\sum_{v\in V}\sigma(\{v\}) = \sum_{v\in V}\Pr[\{v\}\cap R\neq\emptyset] = \sum_{R}(\Pr[R]\cdot|R|) = \mathrm{EPS}.$$

Hence, the lemma is proved. □

Lemma 3 shows that the expected size of a random RR set is the average influence spread of the nodes in V. When the RR sets $\mathcal{R}$ are randomly and uniformly distributed to ℓ machines (i.e., $|\mathcal{R}_i| = |\mathcal{R}|/\ell$), NEWGREEDI requires each machine to take an expected time of $O(\mathrm{EPS}|\mathcal{R}|/\ell)$.

The computation overhead of DIIMM is incurred by (i) the generation of RR sets, and (ii) the execution of NEWGREEDI. The time required to generate a random RR set R is $O(w(R))$, where $w(R)$ denotes the number of edges in G that point to the nodes in R. Observe that $w(R) \geq |R|$. Thus, the time complexity of DIIMM relies on the generation of RR sets. Let EPT be the expected value of $w(R)$. Previous work [60] shows that the expected time required to generate a random RR set is $O(\frac{m}{n}\mathbb{E}[\sigma(\{v^*\})])$, where n is the number of nodes, m is the number of edges, and v^* is a node selected randomly from V with probabilities proportional to their in-degrees. On the other hand, similar to the analysis of IMM in [61], when $\delta \leq 1/2$, DIIMM generates an expected number of $O\big((k\ln n + \ln(1/\delta))n\varepsilon^{-2}/\mathrm{OPT}\big)$ RR sets. In addition, $\mathbb{E}[\sigma(\{v^*\})] \leq \mathrm{OPT}$. Putting it all together with Wald's equation [62] shows that DIIMM requires each machine to take an expected time of $O\big((k\ln n + \ln(1/\delta))(n+m)\varepsilon^{-2}/\ell\big)$.

In addition to the expected time complexity, we show in the following that the workload is asymptotically balanced across a group of distributed machines. The analysis is based on the concept of martingales [14] that can circumvent the dependency of RR sets. A martingale is a sequence of random variables $X_0, X_1, X_2, \dots$ with finite means such that the conditional expectation of X_i given $X_0, X_1, \dots, X_{i-1}$ is equal to X_{i-1}. The following lemma gives the concentration bounds for martingales.

Lemma 4 ([14])**.** *Let $X_0, X_1, X_2, \dots$ be a martingale satisfying $X_i - X_{i-1} \leq a_i + M$ and $\mathrm{Var}[X_i \mid X_0, X_1, \dots, X_{i-1}] \leq \sigma_i^2$ for any $i \in [1, T]$, where $\mathrm{Var}[\cdot]$ denotes the variance of a random variable. Then, for any $\gamma > 0$, we have*

$$\Pr\left[X_T - \mathbb{E}[X_T] \geq \gamma\right] \leq \mathrm{e}^{-\frac{\gamma^2}{2(\sum_{i=1}^{T}(\sigma_i^2+a_i^2))+M\gamma/3)}}.$$

According to Lemma 4, we have the following corollary.

Corollary 1. *Let $R_1, R_2, \dots$ be a sequence of random RR sets. Let* EPS *be the expected size of a random RR set. For any $\gamma > 0$, we have*

$$\Pr\left[\sum_{j=1}^{T}|R_j| \geq T\,\mathrm{EPS} + \gamma\right] \leq \mathrm{e}^{-\frac{\gamma^2}{2n(T\,\mathrm{EPS}+\gamma/3)}}, \tag{10}$$

$$\Pr\left[\sum_{j=1}^{T}|R_j| \leq T\,\mathrm{EPS} - \gamma\right] \leq \mathrm{e}^{-\frac{\gamma^2}{2nT\,\mathrm{EPS}}}. \tag{11}$$

Proof. Let $Y_0 = 0$ and $Y_i = \sum_{j=1}^{i}(|R_j| - \mathrm{EPS})$. Since each $|R_j|$ is a random variable with a mean of EPS, it holds that $\mathbb{E}[Y_i] = 0$ and $\mathbb{E}[Y_i \mid Y_0, \dots, Y_{i-1}] = Y_{i-1}$, which indicates that $Y_0, Y_1, Y_2, \dots$ is a martingale. In addition,

$$Y_i - Y_{i-1} = |R_i| - \mathrm{EPS},$$

and $\mathrm{Var}[Y_i \mid Y_0, \dots, Y_{i-1}] = \mathrm{Var}[|R_i|] = \mathbb{E}[|R_i|^2] - \mathbb{E}^2[|R_i|]$.

Since $|R_i| \leq n$, we have

$$Y_i - Y_{i-1} \leq n - \text{EPS},$$
$$\text{and} \quad \text{Var}[Y_i \mid Y_0, \ldots, Y_{i-1}] \leq n\,\text{EPS} - \text{EPS}^2.$$

By Lemma 4, we have

$$\begin{aligned}&\Pr\left[\sum_{j=1}^{T}|R_j| \geq T\,\text{EPS} + \gamma\right]\\ &= \Pr\left[Y_T \geq \mathbb{E}[Y_T] + \gamma\right]\\ &\leq \mathrm{e}^{-\frac{\gamma^2}{2(T(n\,\text{EPS} - \text{EPS}^2) + (n - \text{EPS})\gamma/3)}} \leq \mathrm{e}^{-\frac{\gamma^2}{2n(T\,\text{EPS} + \gamma/3)}}.\end{aligned}$$

Similarly, $-Y_0, -Y_1, -Y_2, \ldots$ is a martingale satisfying

$$(-Y_i) - (-Y_{i-1}) = \text{EPS} - |R_i| \leq \text{EPS},$$
$$\text{and } \text{Var}[-Y_i \mid -Y_0, \ldots, -Y_{i-1}] \leq n\,\text{EPS} - \text{EPS}^2.$$

When $\gamma > T\,\text{EPS}$, the inequality of $\sum_{j=1}^{T}|R_j| \leq T\,\text{EPS} - \gamma$ cannot hold. Meanwhile, when $\gamma \leq T\,\text{EPS}$, by Lemma 4,

$$\begin{aligned}&\Pr\left[\sum_{j=1}^{T}|R_j| \leq T\,\text{EPS} - \gamma\right]\\ &= \Pr\left[-Y_T \geq \mathbb{E}[-Y_T] + \gamma\right]\\ &\leq \mathrm{e}^{-\frac{\gamma^2}{2(T(n\,\text{EPS} - \text{EPS}^2) + \text{EPS}\,\gamma/3)}} \leq \mathrm{e}^{-\frac{\gamma^2}{2nT\,\text{EPS}}}.\end{aligned}$$

Hence, the proof is completed. □

Consider that $\gamma = \varepsilon T\,\text{EPS}$ for $\varepsilon \in (0, 1)$. Then, according to Corollary 1, we have In particular, we can obtain that

$$\Pr\left[\sum_{j=1}^{T}|R_j| \geq (1+\varepsilon)T\,\text{EPS}\right] \leq \mathrm{e}^{-\frac{\varepsilon^2 T\,\text{EPS}}{2n(1+\varepsilon/3)}},$$
$$\text{and} \quad \Pr\left[\sum_{j=1}^{T}|R_j| \leq (1-\varepsilon)T\,\text{EPS}\right] \leq \mathrm{e}^{-\frac{\varepsilon^2 T\,\text{EPS}}{2n}}.$$

This shows that the total size of hyperedges is in the range of $[1-\varepsilon, 1+\varepsilon]$ times the expectation with high probability, which implies the time complexity of NEWGREEDI for each machine is asymptotically equal to its expected time complexity.

Similarly, for DIIMM, we have

$$\Pr\left[\sum_{j=1}^{\theta} w(R_j) \geq (1+\varepsilon)\theta\,\text{EPT}\right] \leq \mathrm{e}^{-\frac{\varepsilon^2\theta\,\text{EPT}}{2m(1+\varepsilon/3)}},$$
$$\text{and} \quad \Pr\left[\sum_{j=1}^{\theta} w(R_j) \geq (1-\varepsilon)\theta\,\text{EPT}\right] \leq \mathrm{e}^{-\frac{\varepsilon^2\theta\,\text{EPT}}{2m}}$$

Thus, the computation time of DIIMM is concentrated to its expected time within a factor of $[1-\varepsilon, 1+\varepsilon]$ with high probability.

IV. EXPERIMENTS

This section experimentally evaluates the scalability of our distributed algorithms for influence maximization and maximum coverage. We implement our algorithms using Open MPI[1] on a cluster of machines consisting of 17 work nodes each with an Intel Core 2.7GHz CPU (Skylake) and 48GB memory and connected to other nodes via 1Gbps switch. One machine works as the master and the others work as slaves. We also carry out experiments on a large-memory server with multiple CPU cores implemented using Microsoft MPI[2]. The server is equipped with an Intel Xeon 2.2GHz CPU, 12TB memory and 80 cores. All the algorithms are implemented in C++.

[1] https://www.open-mpi.org/

TABLE III: Datasets.

Dataset	#nodes	#edges	Type	Avg. degree
Facebook	4.0K	88.2K	Undirected	43.7
Google+	107.6K	13.7M	Directed	254.1
LiveJournal	4.8M	69.0M	Directed	28.5
Twitter	41.7M	1.5G	Directed	70.5

A. Experimental Setup

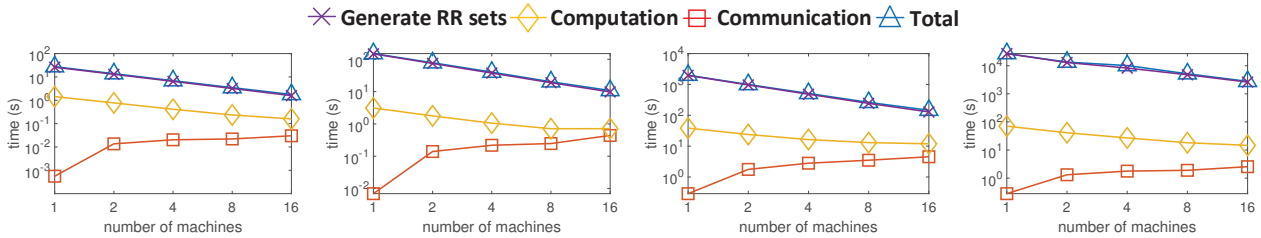
Datasets. We evaluate our algorithm on four real datasets with up to millions of nodes and billions of edges including Facebook, Google+, LiveJournal and Twitter. The Facebook, Google+ and LiveJournal datasets are available at [38], and the Twitter dataset is obtained from [36]. Table III shows the details of each dataset.

Algorithms. For influence maximization, the state-of-the-art algorithms, including IMM [61], SSA [47], OPIM-C [55] and SUBSIM [25], can all be implemented in a distributed manner using our approaches. To demonstrate the scalability of our distributed algorithms, we mainly report the results of distributed implementation of IMM, referred to as DIIMM. In particular, we measure the running time of DIIMM under different numbers of machines or cores. Note that no matter how many machines or cores are used, the influence spread of DIIMM is the same as that of IMM [61]. Thus, we do not show the influence spread in the experiments. Moreover, to show the general applicability of our approach, we also report the results of distributed implementation of SUBSIM [25] under the IC model.

For maximum coverage, we compare our NEWGREEDI algorithm against GREEDI [45]—a state-of-the-art composable core-sets approach. In GREEDI, each slave machine maintains an equal partition of all nodes and greedily selects κ nodes to submit to the master machine, and then the master machine merges the selected nodes from all partitions to yield a final set of size k. We set $\kappa = k$ in our implementation such that GREEDI achieves an approximation guarantee of $\frac{(1-1/\mathrm{e})^2}{\min\{\ell,k\}}$ [45]. For fair comparison, NEWGREEDI and GREEDI start with element-distributed data and set-distributed data respectively in favor of their schemes.

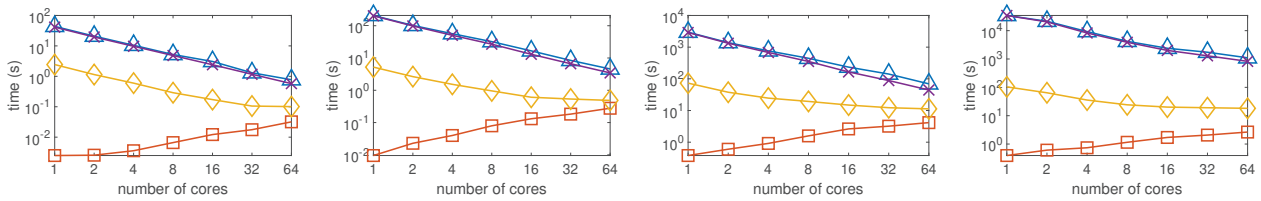
Parameter Settings. For influence maximization, we use both the IC and LT diffusion models. We set the propagation probability $p_{u,v}$ of each edge $\langle u, v\rangle$ to the reciprocal of v's in-degree, which is a setting frequently adopted by existing studies [10, 33, 61]. By default, we set the seed set size $k = 50$, the error threshold $\varepsilon = 0.01$ and the failure probability $\delta = 1/n$. For each parameter setting, we run each algorithm for 10 times and report the average measurement.

[2] https://docs.microsoft.com/en-us/message-passing-interface/microsoft-mpi



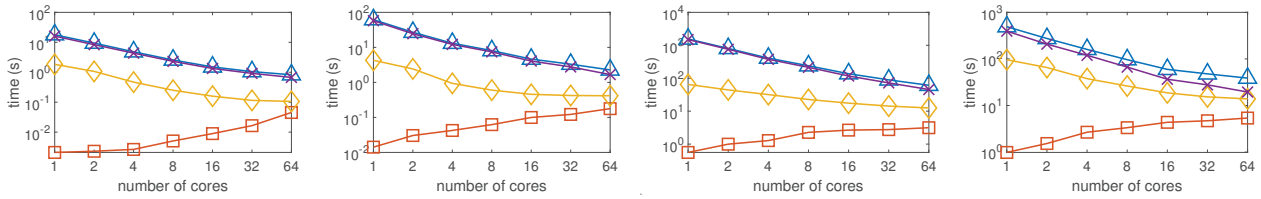
(a) Facebook (b) Google+ (c) LiveJournal (d) Twitter

Fig. 5: Running time of DiIMM for IM under IC model using a cluster of machines.



(a) Facebook (b) Google+ (c) LiveJournal (d) Twitter

Fig. 6: Running time of DiIMM for IM under IC model using a multi-core server.



(a) Facebook (b) Google+ (c) LiveJournal (d) Twitter

Fig. 7: Running time of distributed SUBSIM for IM under IC model using a multi-core server.

TABLE IV: The size of RR sets under the IC model.

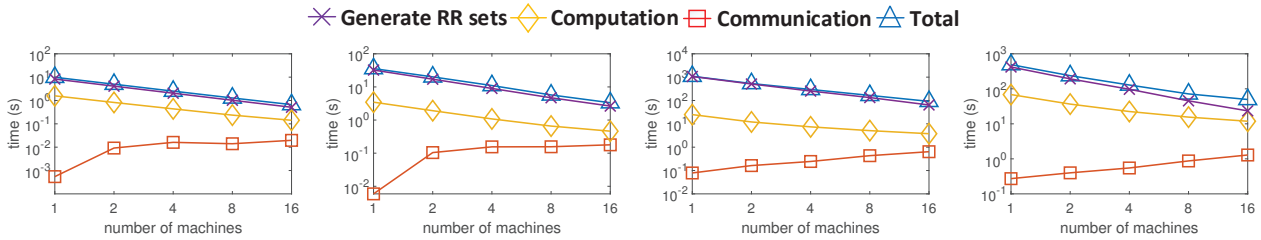
Dataset	#RR sets	Total size
Facebook	8.2M	70.8M
Google+	37.7M	118.3M
LiveJournal	215.6M	2.2G
Twitter	31.5M	558.5M

For maximum coverage, we consider each graph $G = (V, E)$ as a collection of $|V|$ sets over $|V|$ elements with a total size of $|E|$, where the $|V|$ and $|E|$ values of each dataset are given in Table III. Specifically, the universal node set V is mapped to the ground set of elements V, and the collection of all the neighbors of a node u in G is mapped to a hyperedge N_u. Our aim is to find k users from V with the maximum size of their neighbor union. We set $k = 50$ by default.

B. Results for Influence Maximization

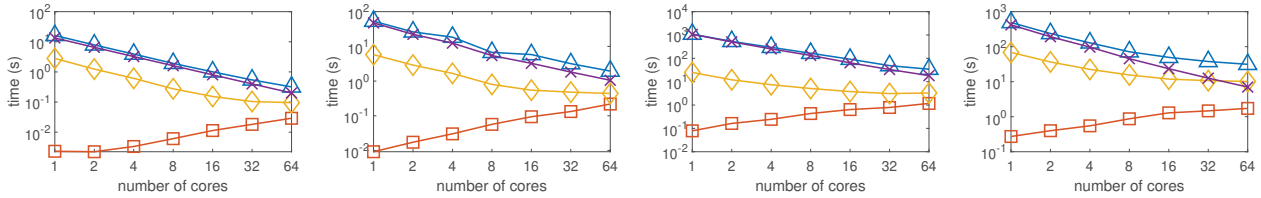
IC Model. We first evaluate our algorithms for influence maximization. Table IV shows the number of RR sets and the total size of RR sets generated under the IC model for the datasets tested.

Fig. 5 shows the running times, including the overall execution time and the breakdown of generation time of RR sets, computation time and communication time, on various datasets under the IC model when different numbers of machines are used. Note that the running time is plotted in *logscale*. Note also that the state-of-the-art IMM algorithm is compared as a baseline where 1 machine is used. As expected, the total running time almost reduces in inverse proportion to the number of cores. Specifically, when 4 machines are used, the speedup ratio is around **3.5×** upon the vanilla IMM algorithm, and when 16 machines are used, the speedup ratio becomes around **14×**. In fact, the major running time is spent for generating RR sets. This is because the number of edges traversed by the stochastic BFS is much more than the number of nodes traversed (i.e., the total size of all RR sets generated) according to the sampling procedure given in Section III-A, where the former determines the generation time of RR sets and the latter determines the computation time and communication time. Since our method generates RR sets via distributed RIS, it can save time in inverse proportion to the number of machines. In addition, we also explore the computation time and communication time of our NewGreeDi algorithm (after the RR sets are generated) on various graphs under the IC model. The results also show an inversely proportional trend for the computation time while the communication time increases along with the number of machines. Compared with the computation time, the communication time is usually an order of magnitude less, which is negligible.



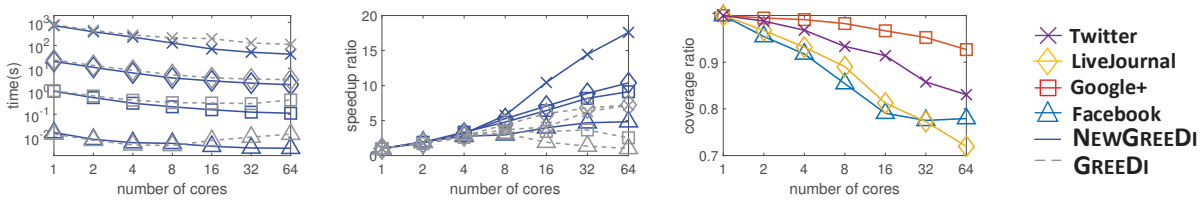
(a) Facebook (b) Google+ (c) LiveJournal (d) Twitter

Fig. 8: Running time of DiIMM for IM under LT model using a cluster of machines.



(a) Facebook (b) Google+ (c) LiveJournal (d) Twitter

Fig. 9: Running time of DiIMM for IM under LT model using a multi-core server.



(a) Running time (b) Speedup ratio (c) Coverage ratio: $\frac{\text{GreeDi}}{\text{NewGreeDi}}$

Fig. 10: Results for maximum coverage using a multi-core server.

Fig. 6 shows the results for the large-memory server with multiple cores. Note that the number of cores is also shown in *logscale*. The results are generally similar to those for a cluster of machines. We observe that when 64 cores are used, the speedup ratios upon the vanilla IMM algorithm are **56×**, **45×**, **43×** and **31×** on the Facebook, Google+, LiveJournal and Twitter datasets respectively, which indicates that the parallel efficiency of our distributed techniques are generally good. In addition, we also obverse that the communication overhead is still significantly less than the computation time even when 64 cores are used. These results demonstrate the scalability of proposed distributed algorithms for influence maximization.

Fig. 7 shows the results of distributed implementation of SUBSIM using the large-memory server with multiple cores. SUBSIM utilizes a subset sampling technique to improve the efficiency of RR set generation so that the overall running time is significantly reduced. As expected, our distributed implementation of SUBSIM achieves a similar speedup ratio to that of DiIMM over IMM, which demonstrates the general applicability of our approach.

LT Model. Fig. 8 and Fig. 9 show the results of running time under the LT model. The performance trends observed from Fig. 5 and Fig. 6 are generally applicable to Fig. 8 and Fig. 9 as well. The major differences lie in that the total running time under the LT model is shorter than that under the IC model under the same setting (see Fig. 8 and Fig. 9) as it takes less time to generate a random RR set under the LT model than that under the IC model. (We have described how to efficiently construct a random RR set under the IC and LT models in Section III-A, which is also mentioned and analyzed in previous work [1, 55]).

Remark. A key difference of the state-of-the-art RIS-based influence maximization algorithms, including IMM [61], SSA [47], OPIM-C [55] and SUBSIM [25], lies in the number of RR sets generated or sampling procedure for each RR set. As a result, our distributed techniques are expected to achieve a similar speedup ratio as of IMM for other sequential algorithms such as SSA, OPIM-C and SUBSIM, as evidenced by Fig. 7.

C. Results for Maximum Coverage

We further run experiments of maximum coverage to compare our NewGreeDi algorithm against a state-of-the-art composable core-sets approach of GreeDi. Fig. 10(a) shows that our NewGreeDi algorithm can be easily applied to data at a massive scale when a number of cores collaborate together to process the job. In particular, we observe from Fig. 10(b) that on the four datasets, the speedup ratio is around 3.5× upon the standard sequential greedy algorithm when 4 cores are used. Furthermore, when 64 cores are used, the speedup ratio becomes 18× on the Twitter dataset and 10× on the

LiveJournal and Google+ datasets respectively, whereas the speedup ratio on the Facebook datasets is slightly lower, as the running time is very short, i.e., less than 0.01 second. The parallel efficiency is slightly reduced when more cores are used, since the communication time increases along with the number of cores but the total running time is still dominated by the computation time. Moreover, we can see that compared with GREEDI, our method shows a better parallel efficiency with shorter running time and remarkably larger speedup ratio when multiple cores are used. In addition, we also plot the coverage ratio of GREEDI to our NEWGREEDI algorithm in Fig. 10(c). Note that our method can always obtain the same coverage as the centralized greedy method. As can be seen, the coverage achieved by GREEDI decreases with the growing number of cores and is significantly less than our NEWGREEDI algorithm especially when a large number of cores are used. These results demonstrate the parallel efficiency and effectiveness of our element-distributed NEWGREEDI algorithm.

V. RELATED WORK

Domingos and Richardson [16] are the first to study viral marketing in OSNs from an algorithmic perspective. Kempe et al. [33] then formulate influence maximization as a discrete optimization problem that targets at finding a size-k seed set with the largest influence spread. They propose a $(1 - 1/\mathrm{e} - \varepsilon)$-approximation greedy algorithm for several influence diffusion models (e.g., the de facto IC and LT models), by utilizing Monte-Carlo simulation. Since then, there has been considerable research on improved algorithms for influence maximization [1, 6, 9–13, 15, 18, 19, 21–23, 25, 29, 32, 37, 39, 42, 47, 50, 51, 54–56, 60, 61, 63, 66, 67]. In particular, a long line of work [9–11, 15, 23, 32, 34, 39, 42, 54, 56, 63] develops heuristics that are lightweight for influence estimation at the cost of foregoing worst-case guarantees. In contrast, more recent work [6, 25, 47, 55, 60, 61] focuses on designing advanced $(1 - 1/\mathrm{e} - \varepsilon)$-approximation algorithms, leveraging reverse influence sampling (RIS) [6]. In this paper, we aim to implement RIS-based algorithms in a distributed fashion that can horizontally scale to OSNs of massive scale. Our distributed algorithms can be easily adapted to accelerate the greedy heuristics for many influence-based applications beyond influence maximization, such as targeted influence maximization focusing on activating targeted users [40], multi-objective influence maximization considering multiple emphasized groups [20], budgeted influence maximization where each node is associated with a distinct cost [5, 39], revenue maximization optimizing the total payment of advertisers to the OSN owner [2, 27], profit maximization combining the benefit of influence spread with the cost of seed selection or information propagation [53, 57, 58], seed minimization with a given amount of influence spread to achieve [24, 41, 65], and their variants under the adaptive setting [26, 30, 31, 59]. In particular, similar to influence maximization, the solutions to the aforementioned problems also (i) generate RR sets, and (ii) construct a seed set via a greedy search, which can be accelerated by our proposed approach.

Another line of related work lies in distributed maximum coverage. In fact, in RIS-based influence maximization, for each RR set, its index is mapped to an element. For each node, the indexes of all the RR sets containing this node are mapped to a set. In the past few years, a large body of research has studied distributed maximum coverage (or its generalized form, i.e., submodular maximization) [3, 4, 35, 44, 45]. These studies are based on the composable core-sets (or its randomized version) technique [44] that distributes the sets to a cluster of machines, referred to as *set-distributed* maximum coverage. Applying set-distributed approach,unfortunately, is incompatible with distributed RIS, which incurs some critical issues as analyzed in Section III-B1. Instead, we propose a new framework tailored for *element-distributed* maximum coverage since in RIS-based influence maximization, each element (RR set) is maintained by one machine whereas each set (node) may appear multiple times in different machines. In addition, our element-distributed maximum coverage algorithm NEWGREEDI in Algorithm 1 can ensure the exact $(1 - 1/\mathrm{e})$-approximation which cannot be achieved by any set-distributed maximum coverage algorithm so far unless it degenerates to the centralized version.

VI. CONCLUSION

In this paper, we have implemented the advanced RIS-based influence maximization algorithms in a distributed fashion. Our distributed algorithms consist of two phases: (i) a sampling phase that generates a number of samples satisfying the approximation guarantee requirement in a distributed manner, and (ii) a seed selection phase that is equivalent to element-distributed maximum coverage. Our NEWGREEDI algorithm ensures $(1 - 1/\mathrm{e})$-approximation for element-distributed maximum coverage. In addition, our DIIMM algorithm that integrates distributed RIS and NEWGREEDI with the state-of-the-art IMM algorithm returns $(1 - 1/\mathrm{e} - \varepsilon)$-approximate solutions for influence maximization. With extensive experiments on real datasets with up to millions of nodes and billions of edges, we show that our solutions can horizontally scale to address influence maximization and maximum coverage with massive data. In fact, our distributed techniques can also accelerate other sequential algorithms for influence maximization, such as SSA, OPIM-C and SUBSIM, by a similar rate as of DIIMM to IMM. Moreover, the greedy algorithms for many influence-based applications, e.g., targeted/multi-objective/budgeted influence maximization, revenue maximization, profit maximization, seed minimization, etc., can be implemented in a distributed manner via our approaches to further improve their efficiency.

ACKNOWLEDGMENT

Jing Tang's work is partially supported by HKUST(GZ) under a Startup Grant. Xueyan Tang's work is partially supported by Singapore Ministry of Education Academic Research Fund Tier 2 under Grant T2EP20121-0026. Kai Han's work is partially supported by National Natural Science Foundation of China (NSFC) under Grant No. 62172384 and by Alibaba Group through Alibaba Innovative Research Program.

References

[1] A. Arora, S. Galhotra, and S. Ranu. Debunking the myths of influence maximization: An in-depth benchmarking study. In *Proc. ACM SIGMOD*, pages 651–666, 2017.

[2] C. Aslay, F. Bonchi, L. V. Lakshmanan, and W. Lu. Revenue maximization in incentivized social advertising. *Proc. VLDB Endowment*, 10(11):1238–1249, 2017.

[3] R. Barbosa, A. Ene, H. L. Nguyen, and J. Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *Proc. ICML*, pages 1236–1244, 2015.

[4] R. Barbosa, A. Ene, H. L. Nguyen, and J. Ward. A new framework for distributed submodular maximization. In *Proc. IEEE FOCS*, pages 645–654, 2016.

[5] S. Bian, Q. Guo, S. Wang, and J. X. Yu. Efficient algorithms for budgeted influence maximization on massive social networks. *Proc. VLDB Endowment*, 13(9):1498–1510, 2020.

[6] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proc. SODA*, pages 946–957, 2014.

[7] Y. Chang, H. Huang, Q. Liu, and X. Jia. Scalable and parallel processing of influence maximization for large-scale social networks. In *Proc. BIGCOM*, pages 183–192, 2017.

[8] W. Chen. An issue in the martingale analysis of the influence maximization algorithm IMM. arXiv preprint, https://arxiv.org/abs/1808.09363, 2018.

[9] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proc. ACM KDD*, pages 199–208, 2009.

[10] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. ACM KDD*, pages 1029–1038, 2010.

[11] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proc. IEEE ICDM*, pages 88–97, 2010.

[12] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng. Staticgreedy: Solving the scalability-accuracy dilemma in influence maximization. In *Proc. ACM CIKM*, pages 509–518, 2013.

[13] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng. IMRank: Influence maximization via finding self-consistent ranking. In *Proc. ACM SIGIR*, pages 475–484, 2014.

[14] F. Chung and L. Lu. Concentration inequalities and martingale inequalities: A survey. *Internet Mathematics*, 3(1):79–127, 2006.

[15] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proc. ACM CIKM*, pages 629–638, 2014.

[16] P. Domingos and M. Richardson. Mining the network value of customers. In *Proc. ACM KDD*, pages 57–66, 2001.

[17] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[18] S. Galhotra, A. Arora, S. Virinchi, and S. Roy. Asim: A scalable algorithm for influence maximization under the independent cascade model. In *Proc. WWW Companion*, pages 35–36, 2015.

[19] S. Galhotra, A. Arora, and S. Roy. Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In *Proc. ACM SIGMOD*, pages 743–758, 2016.

[20] S. Gershtein, T. Milo, and B. Youngmann. Multi-objective influence maximization. In *Proc. EDBT*, pages 145–156, 2021.

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

[22] A. Goyal, W. Lu, and L. V. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proc. WWW Companion*, pages 47–48, 2011.

[23] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Proc. IEEE ICDM*, pages 211–220, 2011.

[24] A. Goyal, F. Bonchi, L. V. Lakshmanan, and S. Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social Network Analysis and Mining*, 3(2):179–192, 2013.

[25] Q. Guo, S. Wang, Z. Wei, and M. Chen. Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *Proc. ACM SIGMOD*, pages 2167–2181, 2020.

[26] K. Han, K. Huang, X. Xiao, J. Tang, A. Sun, and X. Tang. Efficient algorithms for adaptive influence maximization. *Proc. VLDB Endowment*, 11(9):1029–1040, 2018.

[27] K. Han, B. Wu, J. Tang, S. Cui, C. Aslay, and L. V. Lakshmanan. Efficient and effective algorithms for revenue maximization in social advertising. In *Proc. ACM SIGMOD*, pages 671–684, 2021.

[28] M. Haque and D. S. Banerjee. Enhancing influence maximization in social networks using parallel reverse reachability set computations. In *Proc. IC3*, pages 1–6, 2018.

[29] K. Huang, S. Wang, G. Bevilacqua, X. Xiao, and L. V. S. Lakshmanan. Revisiting the stop-and-stare algorithms for influence maximization. *Proc. VLDB Endowment*, 10(9): 913–924, 2017.

[30] K. Huang, J. Tang, K. Han, X. Xiao, W. Chen, A. Sun, X. Tang, and A. Lim. Efficient approximation algorithms for adaptive influence maximization. *The VLDB Journal*, 29(6):1385–1406, 2020.

[31] K. Huang, J. Tang, X. Xiao, A. Sun, and A. Lim. Efficient approximation algorithms for adaptive target profit maximization. In *Proc. IEEE ICDE*, pages 649–660, 2020.

[32] K. Jung, W. Heo, and W. Chen. IRIE: Scalable and robust

influence maximization in social networks. In *Proc. IEEE ICDM*, pages 918–923, 2012.

[33] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. ACM KDD*, pages 137–146, 2003.

[34] J. Kim, S.-K. Kim, and H. Yu. Scalable and parallelizable processing of influence maximization for large-scale social networks. In *Proc. IEEE ICDE*, pages 266–277, 2013.

[35] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. In *Proc. ACM SPAA*, pages 1–10, 2013.

[36] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *Proc. WWW*, pages 591–600, 2010.

[37] J. R. Lee and C. W. Chung. A fast approximation for influence maximization in large social networks. In *Proc. WWW Companion*, pages 1157–1162, 2014.

[38] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2014.

[39] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proc. ACM KDD*, pages 420–429, 2007.

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

[41] C. Long and R. C.-W. Wong. Minimizing seed set for viral marketing. In *Proc. IEEE ICDM*, pages 427–436, 2011.

[42] W.-X. Lu, P. Zhang, C. Zhou, C. Liu, and L. Gao. Influence maximization in big networks: An incremental algorithm for streaming subgraph influence spread estimation. In *Proc. IJCAI*, pages 2076–2082, 2015.

[43] B. Lucier, J. Oren, and Y. Singer. Influence at scale: Distributed computation of complex contagion in networks. In *Proc. ACM KDD*, pages 735–744, 2015.

[44] V. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proc. ACM STOC*, pages 153–162, 2015.

[45] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Proc. NeurIPS*, pages 2049–2057, 2013.

[46] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294, 1978.

[47] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proc. ACM SIGMOD*, pages 695–710, 2016.

[48] H. T. Nguyen, T. P. Nguyen, T. N. Vu, and T. N. Dinh. Outward influence and cascade size estimation in billion-scale networks. In *Proc. ACM SIGMETRICS*, pages 63–63, 2017.

[49] Oberlo. Social media ad spend in the US (2017–2024), 2021.

[50] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *Proc. AAAI*, pages 138–144, 2014.

[51] G. Song, X. Zhou, Y. Wang, and K. Xie. Influence maximization on large-scale mobile social network: A divide-and-conquer method. *IEEE Transactions on Parallel and Distributed Systems*, 26(5):1379–1392, 2015.

[52] Statista. Number of monthly active Facebook users worldwide as of 2nd quarter 2021, September 2021. URL https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/.

[53] J. Tang, X. Tang, and J. Yuan. Profit maximization for viral marketing in online social networks. In *Proc. IEEE ICNP*, pages 1–10, 2016.

[54] J. Tang, X. Tang, and J. Yuan. Influence maximization meets efficiency and effectiveness: A hop-based approach. In *Proc. IEEE/ACM ASONAM*, pages 64–71, 2017.

[55] J. Tang, X. Tang, X. Xiao, and J. Yuan. Online processing algorithms for influence maximization. In *Proc. ACM SIGMOD*, pages 991–1005, 2018.

[56] J. Tang, X. Tang, and J. Yuan. An efficient and effective hop-based approach for inluence maximization in social networks. *Social Network Analysis and Mining*, 8(1):10, 2018.

[57] J. Tang, X. Tang, and J. Yuan. Profit maximization for viral marketing in online social networks: Algorithms and analysis. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1095–1108, 2018.

[58] J. Tang, X. Tang, and J. Yuan. Towards profit maximization for online social network providers. In *Proc. IEEE INFOCOM*, pages 1178–1186, 2018.

[59] J. Tang, K. Huang, X. Xiao, L. V. Lakshmanan, X. Tang, A. Sun, and A. Lim. Efficient approximation algorithms for adaptive seed minimization. In *Proc. ACM SIGMOD*, pages 1096–1113, 2019.

[60] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proc. ACM SIGMOD*, pages 75–86, 2014.

[61] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proc. ACM SIGMOD*, pages 1539–1554, 2015.

[62] A. Wald. *Sequential Analysis*. Wiley, 1947.

[63] X. Wang, Y. Zhang, W. Zhang, X. Lin, and C. Chen. Bring order into the samples: A novel scalable method for influence maximization. *IEEE Transactions on Knowledge and Data Engineering*, 29(2):243–256, 2017.

[64] Zenith. Globally, social media ad spend forecast to overtake paid search this year. *Marketingcharts*, 2021.

[65] P. Zhang, W. Chen, X. Sun, Y. Wang, and J. Zhang. Minimizing seed set selection with probabilistic coverage guarantee in a social network. In *Proc. ACM KDD*, pages 1306–1315, 2014.

[66] C. Zhou, P. Zhang, J. Guo, X. Zhu, and L. Guo. UBLF:

An upper bound based approach to discover influential nodes in social networks. In *Proc. IEEE ICDM*, pages 907–916, 2013.

[67] C. Zhou, P. Zhang, J. Guo, and L. Guo. An upper bound based greedy algorithm for mining top-k influential nodes in social networks. In *Proc. WWW Companion*, pages 421–422, 2014.