Continuous Profit Maximization: A Study of Unconstrained Dr-Submodular Maximization

Jianxiong Guo and Weili Wu, *Senior Member, IEEE*

***Abstract*— Profit maximization (PM) is to select a subset of users as seeds for viral marketing in online social networks, which balances between the cost and the profit from influence spread. We extend PM to formulate a continuous PM under the general marketing strategies (CPM-MS) problem, whose domain is on integer lattices. The objective function of our CPM-MS is dr-submodular, but nonmonotone. It is a typical case of unconstrained dr-submodular maximization (UDSM) problem, and taking it as a starting point, we study UDSM systematically in this article, which is very different from those studied by existing researchers. First, we introduce the lattice-based double greedy algorithm, which can obtain a constant approximation guarantee. However, there is a strict and unrealistic condition that requiring the objective value is nonnegative on the whole domain or else no theoretical bounds. Thus, we propose a lattice-based iterative pruning technique. It can shrink the search space effectively, thereby greatly increasing the possibility of satisfying the nonnegative objective function on this smaller domain without losing approximation ratio. Then, to overcome the difficulty to estimate the objective value of CPM-MS, we adopt reverse sampling strategies and combine it with lattice-based double greedy, including pruning, without losing its performance but reducing its running time. The entire process can be considered as a general framework to solve the UDSM problem, especially for applying to social networks. Finally, we conduct experiments on several real data sets to evaluate the effectiveness and efficiency of our proposed algorithms.**

***Index Terms*— Approximation algorithm, continuous profit maximization (PM), dr-submodular maximization, integer lattice, sampling strategies, social networks.**

I. Introduction

ONLINE social networks (OSNs) were becoming more and more popular to exchange ideas and make friends gradually in recent years and accompanied by the rise of a series of social giants, such as Twitter, Facebook, Wechat, and LinkedIn. People tended to share what one sees and hears and discuss some hot issues on these social platforms instead of traditional ways. Many companies or advertisers exploited to spread their products, opinions, or innovations. By offering those influential users free or discounted samples, information can be spread across the whole network through word-of-mouth effect [1], [2]. Inspired by that, the influence maximization (IM) problem [3] was formulated, which selects a subset of users (seed set) to maximizing the expected follow-up adoptions (influence spread) for a given information cascade. In this Kempe *et al.*'s seminal work [3], IM was defined on the two basic discrete diffusion models, independent cascade model (IC-model) and linear threshold model (LT-model), and these two models can be generalized to the triggering model. Then, they proved that the IM problem is NP-hard and obtain a $(1 - 1/e)$-approximation [4] under the IC/LT-model by use of a simple hill-climbing in the framework of monotonicity and submodularity.

Since this seminal work, a plenty of related problems based on IM that were used for different scenarios have emerged [5], [6]. Among them, profit maximization (PM) [7]–[11] is the most representative and widely used one. Consider viral marketing for a given product, and the gain is the influence spread generated from our selected seed set in a social network. However, it is not free to activate those users in this seed set. For instance, in a real advertisement scenario, discounts and rewards are usually adopted to improve users' desire to purchase and stimulate consumption. Thus, the net profit is equal to the influence spread minus the expense of seed set, where more incentives do not imply more benefit. Tang *et al.* [9] proved that the objective function of PM is submodular, but not monotone. Before this, Kempe *et al.* [12] proposed the generalized marketing instead of the seed set. A marketing strategy is denoted by $\boldsymbol{x} \in \mathbb{Z}_+^d$ where the user u will be activated as a seed with probability $h_u(\boldsymbol{x})$. Thus, the seed set is not deterministic, but activated probabilistically according to a marketing strategy. In this article, we propose a continuous PM under the general marketing strategies (CPM-MS) problem, which aims to choose the optimal marketing vector $\boldsymbol{x}^* \preceq \boldsymbol{b}$ such that the net profit can be maximized. Each component $\boldsymbol{x}(i) \in \boldsymbol{x}$ stands for the investment to marketing action M_i. Actually, in order to promote their products, a company often adopts multiple marketing techniques, such as advertisements, discounts, cashback, and propagandas, whose effects are different to customers at different levels. Therefore, CPM-MS is much more generalized than traditional PM.

After formulating our CPM-MS problem, we discuss its properties first. We show that the CPM-MS problem is NP-hard, and given a marketing vector $\boldsymbol{x}$, it is #P-hard to compute the expected profit exactly. Because of the difficulty to compute the expected profit, we give an approximate method that needs to run Monte Carlo (MC) simulations on a constructed graph. Then, we prove that the objective function of CPM-MS problem is dr-submodular, but not monotone. Extended from set function to vector function on integer

Manuscript received August 18, 2020; revised December 10, 2020 and January 31, 2021; accepted February 20, 2021. Date of publication March 10, 2021; date of current version May 28, 2021. This work was supported in part by the National Science Foundation under Grant 1747818 and Grant 1907472. *(Corresponding author: Jianxiong Guo.)*

The authors are with the Department of Computer Science, Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: jianxiong.guo@utdallas.edu).

Digital Object Identifier 10.1109/TCSS.2021.3061452

lattice, the dr-submodularity has a diminishing return property. For the unconstrained submodular maximization (USM), Buchbinder *et al.* [13] proposed a randomized double greedy algorithm that can achieve a tight (1/2)-approximation ratio. To our CPM-MS problem, we are able to consider it as a case of unconstrained dr-submodular maximization (UDSM) inspired by USM. Here, we introduce a lattice-based double greedy algorithm for the UDSM, and a (1/2)-approximation can be obtained as well if the objective value is nonnegative. The marketing vector $\boldsymbol{x}$ is defined on $\mathbf{0} \preceq \boldsymbol{x} \preceq \boldsymbol{b}$, and thus, this approximation can be guaranteed only when the sum of objective values on $\mathbf{0}$ and $\boldsymbol{b}$ is not less than zero, which is hard to be satisfied in the real applications. Imagine to offer all marketing actions full investments, is it still profitable? The answer is no. To overcome this defect, we design a lattice-based iterative pruning technique. It shrinks the searching space gradually in an iterative manner, and then, we initialize our lattice-based double greedy with this smaller searching space. The objective values on this smaller space are very likely to be nonnegative, thereby increasing the applicability of our algorithm's approximation. Even if we can use MC simulations to estimate the expected profit, its time complexity is too high. Here, based on the reverse influence sampling (RIS) [14]–[17], we design an unbiased estimator for the profit function, which can estimate the objective value of a given marketing vector accurately. Next, we take this estimator as our new objective function, combine with lattice-based pruning and double greedy algorithm, and propose the DG-IP-RIS algorithm eventually. It guarantees to obtain a $(1/2 - \varepsilon)$-approximation under a weak condition, whose time complexity is improved significantly. Finally, we conduct several experiments to evaluate the superiority of our proposed DG-IP-RIS algorithm to other heuristic algorithms and compare their running times, which supports the effectiveness and efficiency of our approaches strongly.

Organization: Section II discusses the related work. Section III formulates our main problem. The properties and computability of our CPM-MS problem are presented in Section IV. Section V is the main contributions, including lattice-based double greedy and pruning algorithms. Section VI analyzes the time complexity and designs speedup algorithms based on sampling strategies. Experiments and discussions are presented in Section VII, and finally, Section VIII draws the conclusion.

II. Related Work

A. Influence Maximization

Kempe *et al.* [3] formulated IM to a combinatorial optimization problem. Given a seed set, Chen *et al.* [18], [19] proved that computing its exact influence spread under the IC-model and LT-model, respectively, are #P-hard, and they designed two heuristic algorithms that can solve IM problem under the IC-model [18] and LT-model [19], which reduces the computation overhead effectively. Borgs *et al.* [14] took RIS to estimate the influence spread first, and subsequently, a lot of researchers utilized RIS to design efficient algorithms with $(1 - 1/e - \varepsilon)$-approximation. Tang *et al.* [15] proposed TIM/TIM+ algorithms, which were better than Brogs *et al.*'s IM method regardless of accuracy and time complexity. Then, they developed a more efficient algorithm, IMM [16], based on martingale analysis. Nguyen *et al.* [20] designed SSA/D-SSA and claimed it reduces the running time significantly without losing approximation ratio, but still be doubted by other researchers. Recently, Tang *et al.* [17] created an OPIM-C, which can be terminated at any time and get a solution with its approximation guarantee.

B. Profit Maximization

Domingos and Richardson [1] and Richardson and Domingos [2] studied viral marketing systematically first. Lu and Lakshmanan, [7] distinguished between influence and actual adoption and designed a decision-making process to explain how to adopt a product. Zhang *et al.* [8] studied the problem of distributing a limited budget across multiple products such that maximizing total profit. Tang *et al.* [9] analyzed and solved the USM problem by double greedy algorithm thoroughly with PM as background and proposed iterative pruning technique, which is different from our pruning process, because our objective function is defined on the integer lattice. Tong *et al.* [10] considered the coupon allocation in the PM problem and designed efficient randomized algorithms to achieve $(1/2 - \varepsilon)$-approximation with high probability. Guo *et al.* [11] proposed a budgeted coupon problem whose domain is constrained and provided a continuous double greedy algorithm with a valid approximation.

C. (Dr-)Submodular Maximization

The PM problem is submodular but not monotone, which is a case of USM problem [13], [21]. Feige *et al.* [21] pointed out that there is no approximate solution existing unless giving a nonnegative objective function, and they developed a deterministic local search which (1/3)-approximation and a randomized local search with (2/5)-approximation for maximizing nonnegative submodular function. Buchbinder *et al.* [13] optimized it to (1/2)-approximation further with much lower computational complexity by a randomized double greedy algorithm. Enlightened by the diminishing return on set function, Soma and Yoshida [22], [23] created a new concept that is dr-submodularity defined on integer lattice. Bian *et al.* [24] studied to maximize the nonmonotone continuous dr-submodular function under general down-closed convex constraints used in social networks, and Chen *et al.* [25] proposed a continuous IM problem and designed an efficient $(1 - 1/e - \varepsilon)$-approximate algorithm by use of its monotonicity and dr-submodularity. Guo *et al.* [26] designed a sandwich approximation framework to solve a monotone non-dr-submodular maximization problem.

When the nonnegativity of objective function in our CPM-MS problem cannot be satisfied, it is hard to maximize a nonmonotone dr-submodular function with an acceptable approximation. The hardness is how to design an effective method to satisfy the nonnegativity as much as possible. Thus, we propose the latticed-based double greedy algorithm and pruning techniques, whose process and mathematical induction are different from [9] and [27] because set functions and vector functions have different characteristics. Vector functions are more common but more complex. Then, we improve their scalability by using RIS to estimate objective values. This is the summary of our contribution.

III. Problem Formulation

In this section, we provide some preliminaries to the rest of this article and formulate our continuous PM under the general marketing strategies (CPM-MS).

A. Influence Model

An OSN can be abstracted as a directed graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of n nodes (users) and $E = \{e_1, e_2, \ldots, e_m\}$ is the set of m edges (relationship between users). We default $|V| = n$ and $|E| = m$ given $G = (V, E)$. For each directed edge $(u, v) \in E$, we say that v is an outgoing neighbor of u and u is an incoming neighbor of v. For any node $u \in V$, let $N^-(u)$ denote its set of incoming neighbors and $N^+(u)$ denote its set of outgoing neighbors. In the process of influence diffusion, we consider that a user is active if she accepts (is activated by) the information cascade from her neighbors or she is selected as a seed successfully. To model the influence diffusion, Kempe *et al.* [3] proposed two classical models, IC-model and LT-model.

Let $S \subseteq V$ be a seed set and $S_i \subseteq V$ be the set of all active nodes at time step t_i. The influence diffusion initiated by S can be represented by a discrete-time stochastic process. At time step t_0, all nodes in S are activated, so we have $S_0 := S$. Under the IC-model, there is a diffusion probability $p_{uv} \in (0, 1]$ associated with each edge $(u, v) \in E$. We set $S_i := S_{i-1}$ at time step t_i $(t \geq 1)$ first; then, for each node $u \in S_{i-1} \backslash S_{i-2}$, activated first at time step t_{i-1}, it have one chance to activate each of its inactive outgoing neighbor v with probability p_{uv}. We add v into S_i if u activates v successfully at t_i. Under the LT-model, each edge $(u, v) \in E$ has a weight b_{uv}, and each node $v \in V$ has a threshold θ_v sampled uniformly in $[0, 1]$ and $\sum_{u \in N^-(v)} b_{uv} \leq 1$. We set $S_i := S_{i-1}$ at time step t_i $(t \geq 1)$ first; then, for each inactive node $v \in V \backslash S_{i-1}$, it can be activated if $\sum_{u \in S_{i-1} \cap N^-(v)} b_{uv} \geq \theta_v$. We add v into S_i if v is activated successfully at t_i. The influence diffusion terminates when no more inactive nodes can be activated. In this article, we consider the triggering mode, where IC-model and LT-model are its special cases.

Definition 1 (Triggering Model [3]): Each node v selects a triggering set T_v randomly and independently according to a distribution $\mathcal{D}_v$ over the subsets of $N^-(v)$. We set $S_i := S_{i-1}$ at time step t_i $(t \geq 1)$ first; then, for each inactive node $v \in V \backslash S_{i-1}$, it can be activated if there is at least one node in T_v activated in t_{i-1}. We add v into S_i if v is activated successfully at t_i. The influence diffusion terminates when no more inactive nodes can be activated.

From above, a triggering model can be defined as $\Omega = (G, \mathcal{D})$, where $\mathcal{D} = \{\mathcal{D}_{v_1}, \mathcal{D}_{v_2}, \cdots \mathcal{D}_{v_n}\}$ is a set of distribution over the subsets of each $N^-(v_i)$.

B. Realization

For each node $v \in V$, under the IC-model, each node $u \in N^-(v)$ appears in v's random triggering set T_v with probability p_{uv} independently. Under the LT-model, at most one node can appear in T_v; thus, for each node $u \in N^-(v)$, $T_v = \{u\}$ with probability b_{uv} exclusively and $T_v = \emptyset$ with probability $1 - \sum_{u \in N^-(v)} b_{uv}$. Now, we can define the realization (possible world) g of graph G under the triggering model $\Omega = (G, \mathcal{D})$, that is, the following holds.

Definition 2 (Realization): Given a directed graph $G = (V, E)$ and triggering model $\Omega = (G, \mathcal{D})$, a realization $g = \{T_{v_1}, T_{v_2}, \ldots, T_{v_n}\}$ of G is a set of triggering set sampled from distribution $\mathcal{D}$, denoted by $g \sim \Omega$. For each node $v \in V$, we have $T_v \sim \mathcal{D}_v$.

If a node u appears in v's triggering set, $u \in T_v$, we say that edge (u, v) is live or else edge (u, v) is blocked. Thus, realization g can be regarded as a subgraph of G, which is the remaining graph by removing these blocked edges. Let $\Pr[g|g \sim \Omega]$ be the probability of realization g of G sampled from distribution $\mathcal{D}$, that is

$$\Pr[g|g \sim \Omega] = \prod_{i=1}^{n} \Pr\left[T_{v_i} \middle| T_{v_i} \sim \mathcal{D}_{v_i}\right] \tag{1}$$

where $\Pr[T_{v_i}|T_{v_i} \sim \mathcal{D}_{v_i}]$ is the probability of T_{v_i} sampled from $\mathcal{D}_{v_i}$. Under the IC-model, $\Pr[T_v|T_v \sim \mathcal{D}_v] = \prod_{u \in T_v} p_{uv} \prod_{u \in N^-(v) \backslash T_v}(1 - p_{uv})$, and under the LT-model, $\Pr[T_v = \{u\}|T_v \sim \mathcal{D}_v] = b_{uv}$ for each $u \in N^-(v)$ and $\Pr[T_v = \emptyset|T_v \sim \mathcal{D}_v] = 1 - \sum_{u \in N^-(v)} b_{uv}$ deterministically.

Given a seed set $S \subseteq V$, we consider $I_\Omega(S)$ as a random variable that denotes the number of active nodes (influence spread) when the influence diffusion of S terminates under the triggering model $\Omega = (G, \mathcal{D})$. Then, the number of nodes that are reachable from at least one node in S under a realization g, $g \sim \Omega$, is denoted by $I_g(S)$. Thus, the expected influence spread $\sigma_\Omega(S)$, that is

$$\sigma_\Omega(S) = \mathbb{E}_{g \sim \Omega}\left[I_g(S)\right] = \sum_{g \sim \Omega} \Pr[g] \cdot I_g(S) \tag{2}$$

where it is the weighted average of influence spread under all possible graph realizations. The IM problem aims to find a seed set S, such that $|S| \leq k$, to maximize the expected influence spread $\sigma_\Omega(S)$.

Theorem 1 [3]: Under a triggering model $\Omega = (G, \mathcal{D})$, the expected influence spread $\sigma_\Omega(S)$ is monotone and submodular with respect to seed set S.

C. Problem Definition

Under the general marketing strategies, the definition of IM problem will be different from above [3]. Let $\mathbb{Z}_+^d$ be the collection of nonnegative integer vector. A marketing strategy can be denoted by a d-dimensional vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{Z}_+^d$, and we call it "marketing vector." Each component $\boldsymbol{x}(i) \in \mathbb{Z}_+$, $i \in [d] = \{1, 2, \ldots, d\}$, means the number of investment units assigned to marketing action M_i. For example, $\boldsymbol{x}(i) = b$ tells us that marketing strategy $\boldsymbol{x}$ assigns b investment units to marketing action M_i. Given a marketing vector $\boldsymbol{x}$, the probability that node $u \in V$ is activated as a seed is denoted by the strategy function $h_u(\boldsymbol{x})$, where $h_u(\boldsymbol{x}) \in [0, 1]$. Thus, unlike the standard IM problem, the selection of seed set is not deterministic but stochastic. Given a marketing vector $\boldsymbol{x}$, the probability of seed set S sampled from $\boldsymbol{x}$, that is

$$\Pr[S|S \sim \boldsymbol{x}] = \prod_{u \in S} h_u(\boldsymbol{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\boldsymbol{x})) \tag{3}$$

where $\Pr[S|S \sim \boldsymbol{x}]$ is the probability that exactly nodes in S are selected as seeds, but not in S are not selected as seeds under the marketing strategy $\boldsymbol{x}$, because each node is select as a seed independently. Thus, the expected influence spread $\mu_\Omega(\boldsymbol{x})$ of marketing vector $\boldsymbol{x}$ under the triggering model $\Omega(G, \mathcal{D})$ can be formulated, that is

$$\mu_\Omega(\boldsymbol{x}) = \sum_{S \subseteq V} \Pr[S|S \sim \boldsymbol{x}] \cdot \sigma_\Omega(S) \tag{4}$$

$$= \sum_{S \subseteq V} \sigma_\Omega(S) \cdot \prod_{u \in S} h_u(\boldsymbol{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\boldsymbol{x})). \tag{5}$$

As we know, benefit is the gain obtained from influence spread and cost is the price required to pay for marketing strategy. Here, we assume that each unit of marketing action M_i, $i \in [d]$, is associated with a cost $c_i \in \mathbb{R}_+$. Then, the total cost function $c : \mathbb{Z}_+^d \to \mathbb{R}_+$ can be defined as $c(\boldsymbol{x}) = \sum_{i \in [d]} c_i \cdot \boldsymbol{x}(i)$. For simplicity, we consider the expected influence spread as our benefit. Thus, the expected profit $f_\Omega(\boldsymbol{x})$ we can obtain from marketing strategy $\boldsymbol{x}$ is the expected influence spread of $\boldsymbol{x}$ minus the cost of $\boldsymbol{x}$, that is

$$f_\Omega(\boldsymbol{x}) = \mu_\Omega(\boldsymbol{x}) - c(\boldsymbol{x}) \tag{6}$$

where $c(\boldsymbol{x}) = \sum_{i \in [d]} c_i \cdot \boldsymbol{x}(i)$. Therefore, the continuous PM under the CPM-MS problem is formulated as follows.

Problem 1 (CPM-MS): Given a triggering model $\Omega = (G, \mathcal{D})$, a constraint vector $\boldsymbol{b} \in \mathbb{Z}_+^d$, a strategy function $h_u : \mathbb{Z}_+^d \to [0, 1]$ for each user $u \in V$, and a cost function $c : \mathbb{Z}_+^d \to \mathbb{R}_+$, the CPM-MS problem aims to find an optimal marketing vector $\boldsymbol{x}^* \preceq \boldsymbol{b}$ that maximizes its expected profit $f_\Omega(\boldsymbol{x})$, that is, $\boldsymbol{x}^* \in \arg\max_{\boldsymbol{x} \preceq \boldsymbol{b}} f_\Omega(\boldsymbol{x})$.

IV. Properties of CPM-MS

In this section, we introduce the submodularity on integer lattice and then analyze the submodularity and computability of our CPM-MS problem.

A. Submodularity on Integer Lattice

Generally, defined on set, a set function $\alpha : 2^V \to \mathbb{R}$ is monotone if $\alpha(S) \leq \alpha(T)$ for any $S \subseteq T \subseteq V$ and submodular if $\alpha(S) + \alpha(T) \geq \alpha(S \cup T) + \alpha(S \cap T)$. The submodularity of set function implies a diminishing return property, and thus, $\alpha(S \cup \{u\}) - \alpha(S) \geq \alpha(T \cup \{u\}) - \alpha(T)$ for any $S \subseteq T \subseteq V$ and $u \notin T$. These two definitions of submodularity on set function are equivalent. Defined on integer lattice, a vector function $\beta : \mathbb{Z}_+^d \to \mathbb{R}$ is monotone if $\beta(\boldsymbol{s}) \leq \beta(\boldsymbol{t})$ for any $\boldsymbol{s} \preceq \boldsymbol{t} \in \mathbb{Z}_+^d$ and submodular if $\beta(\boldsymbol{s}) + \beta(\boldsymbol{t}) \geq \beta(\boldsymbol{s} \vee \boldsymbol{t}) + \beta(\boldsymbol{s} \wedge \boldsymbol{t})$ for any $s, t \in \mathbb{Z}_+^d$, where $(\boldsymbol{s} \vee \boldsymbol{t})(i) = \max\{\boldsymbol{s}(i), \boldsymbol{t}(i)\}$ and $(\boldsymbol{s} \wedge \boldsymbol{t})(i) = \min\{\boldsymbol{s}(i), \boldsymbol{t}(i)\}$. Here, $\boldsymbol{s} \preceq \boldsymbol{t}$ implies $\boldsymbol{s}(i) \leq \boldsymbol{t}(i)$ for each component $i \in [d]$. Besides, we consider that a vector function is diminishing return submodular (dr-submodular) if $\beta(\boldsymbol{s} + \boldsymbol{e}_i) - \beta(\boldsymbol{s}) \geq \beta(\boldsymbol{t} + \boldsymbol{e}_i) - \beta(\boldsymbol{t})$ for any $\boldsymbol{s} \preceq \boldsymbol{t}$ and $i \in [d]$, where $\boldsymbol{e}_i \in \mathbb{Z}_+^d$ is the ith unit vector with the ith component being 1 and others being 0. Different from the submodularity for a set function, for a vector function, β is submodular, which does not mean that it is dr-submodular, but the opposite is true. Thus, dr-submodularity is stronger than submodularity generally.

Lemma 1: Given a set function $\alpha : 2^V \to \mathbb{R}$ and a vector function $\beta : \mathbb{Z}_+^d \to \mathbb{R}$, they satisfy

$$\beta(\boldsymbol{x}) = \sum_{S \subseteq V} \alpha(S) \cdot \prod_{u \in S} h_u(\boldsymbol{x}) \cdot \prod_{v \in V \setminus S} (1 - h_v(\boldsymbol{x})). \tag{7}$$

If $\alpha(\cdot)$ is monotone and submodular and $h_u(\cdot)$ is monotone and dr-submodular for each $u \in V$, then $\beta(\cdot)$ is monotone and dr-submodular.

Proof: It is an indirect corollary that has been implied by the proof process in [12, Sec. 7] and [26]. □

Theorem 2: Given a triggering model $\Omega = (G, \mathcal{D})$, the profit function $f_\Omega(\cdot)$ is dr-submodular, but not monotone.

Proof: From Lemma 1, Theorem 1, and (5), we have known that the expected influence spread $\mu_\Omega(\cdot)$ is monotone and dr-submodular because $\sigma_\Omega(\cdot)$ is monotone and submodular. Thus, we have $f_\Omega(\boldsymbol{x} + \boldsymbol{e}_i) - f_\Omega(\boldsymbol{x}) = \mu_\Omega(\boldsymbol{x} + \boldsymbol{e}_i) - \mu_\Omega(\boldsymbol{x}) - c_i \geq \mu_\Omega(\boldsymbol{y} + \boldsymbol{e}_i) - \mu_\Omega(\boldsymbol{y}) - c_i = f_\Omega(\boldsymbol{y} + \boldsymbol{e}_i) - f_\Omega(\boldsymbol{y})$ iff $\boldsymbol{x} \preceq \boldsymbol{y} \in \mathbb{Z}_+^d$. Thus, $f_\Omega(\cdot)$ is dr-submodular. □

B. Computability

Given a seed set $S \subseteq V$, it is #P-hard to compute the expected influence spread $\sigma_\Omega(S)$ under the IC-model [18] and the LT-model [19]. Assume that a marketing vector $\boldsymbol{x} \in \{0, 1\}^n$ and $h_u(\boldsymbol{x}) = \boldsymbol{x}(u)$ for $u \in V$ where user u is a seed if and only if $\boldsymbol{x}(u) = 1$. According to (4), the expected influence spread $\mu_\Omega(\boldsymbol{x})$ is equivalent to $\sigma_\Omega(S)$ in which $S = \{u \in V : \boldsymbol{x}(u) = 1\}$. Thereby, given a marketing vector $\boldsymbol{x}$, computing the expected influence spread $\mu_\Omega(\boldsymbol{x})$ is #P-hard as well under the IC-model and LT-model. Subsequently, a natural question is how to estimate the value of $\mu_\Omega(\boldsymbol{x})$ given $\boldsymbol{x}$ effectively. To estimate $\mu_\Omega(\boldsymbol{x})$, we usually adopt MC simulations. However, it is inconvenient for us to use such a method here because the randomness comes from two parts: one is from the seed selection, and the other is from the process of influence diffusion. Therefore, we require to design a more simple and efficient method.

First, we are able to establish an equivalent relationship between $\sigma_\Omega(\cdot)$ and $\mu_\Omega(\cdot)$. Given a social network $G = (V, E)$ and a marketing vector $\boldsymbol{x} \in \mathbb{Z}_+^d$, we create a constructed graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ by adding a new node $\widetilde{u}$ and a new directed edge $(\widetilde{u}, u)$ for each node $u \in V$ to G. Take IC-model for instance, and the diffusion probability for this new edge $(\widetilde{u}, u)$ can be set as $p_{\widetilde{u}u} = h_u(\boldsymbol{x})$. Then, we have

$$\mu(\boldsymbol{x}|G) = \sigma\left(\widetilde{V} - V|\widetilde{G}\right) - |V| \tag{8}$$

where $\mu(\cdot|G)$ and $\sigma(\cdot|\widetilde{G})$ imply that we compute them under the graph G and the constructed graph $\widetilde{G}$.

Theorem 3: Given a social network $G = (V, E)$ and a marketing vector $\boldsymbol{x} \in \mathbb{Z}_+^d$, the expected influence spread $\mu_\Omega(\boldsymbol{x})$ can be estimated with (γ, δ)-approximation by MC simulations in $O(((m+3n)n^2 \ln(2/\delta))/(2(\gamma \sum_{u\in V} h_u(\boldsymbol{x}))^2))$ running time.

Proof: As mentioned above, we can compute $\sigma(\widetilde{V}-V|\widetilde{G})$ on the constructed graph instead of $\mu(\boldsymbol{x}|G)$ on the original graph. Let $S=\widetilde{V}-V$, and the value of $\sigma_\Omega(S)$ can be estimated by MC simulations according to (2). Based on Hoeffding's inequality, we can note that

$$\Pr\left[|\hat{\sigma}_\Omega(S)-\sigma_\Omega(S)| \geq \gamma\left(\sigma_\Omega(S)-n\right)\right] \leq 2e^{-2r\left(\frac{\gamma(\mu_\Omega(S)-n)}{n}\right)^2}$$

where r is the number of MC simulations and $\sigma_\Omega(S)-n \leq n$. We have $\sigma_\Omega(S)-n \geq \sum_{u\in V} h_u(\boldsymbol{x})$, and to achieve a (γ,δ)-estimation, the number of MC simulations $r \geq (n^2 \ln(2/\delta))/(2(\gamma \sum_{u\in V} h_u(\boldsymbol{x}))^2)$. For each iteration of simulations in the constructed graph, it takes $O(m+3n)$ running time. Thus, we can obtain a (γ,δ)-approximation of $\mu_\Omega(\boldsymbol{x})$ in $O(((m+3n)n^2 \ln(2/\delta)/(2(\gamma \sum_{u\in V} h_u(\boldsymbol{x}))^2))$ running time. □

Based on Theorem 3, we can get an accurate estimation for the objective function $f_\Omega(\boldsymbol{x})$, shown as (6), of CPM-MS problem by adjusting the parameter γ and δ definitely.

V. Algorithms Design

From Section IV, we have known that the objective function of CPM-MS is dr-submodular, but not monotone. In this section, we develop our new methods based on the double greedy algorithm [13] for our CPM-MS and obtain an optimal approximation ratio.

A. Lattice-Based Double Greedy

For nonnegative submodular functions, Buchbinder *et al.* [13] designed a double greedy algorithm to get a solution for the USM problem with a tight theoretical guarantee. Under the deterministic setting, the double greedy algorithm has a (1/3)-approximation, while it has a (1/2)-approximation under the randomized setting. Extending from set to integer lattice, we derive a revised double greedy algorithm that is suitable for dr-submodular functions, namely UDSM problem. We adopt the randomized setting, and the lattice-based double greedy algorithm is shown in Algorithm 1. We omit the subscript of $f_\Omega(\cdot)$, denote it by $f(\cdot)$ from now on.

Here, we denote by $f(\boldsymbol{e}_i|\boldsymbol{x}) = f(\boldsymbol{x}+\boldsymbol{e}_i) - f(\boldsymbol{x})$, the marginal gain of adding component $i \in [d]$ by 1. Generally, this algorithm is initialized by $[\boldsymbol{0},\boldsymbol{b}]$, and for each component $i \in [d]$, we increase $\boldsymbol{x}(i)$ by 1 or decrease $\boldsymbol{y}(i)$ by 1 until they are equal in each inner (while) iteration. The result returned by Algorithm 1 has $\boldsymbol{x}=\boldsymbol{y}$. Then, we have the following conclusion, which can be inferred directly from the double greedy algorithm in [13].

Theorem 4: For our CPM-MS problem, if we initialize Algorithm 1 by $[\boldsymbol{0},\boldsymbol{b}]$ and $f(\boldsymbol{0})+f(\boldsymbol{b}) \geq 0$ is satisfied, the marketing vector $\boldsymbol{x}^\circ$ returned by Algorithm 1 is a (1/2)-approximate solution such that

$$\mathbb{E}\left[f(\boldsymbol{x}^\circ)\right] \geq (1/2)\cdot \max_{\boldsymbol{x}\preceq \boldsymbol{b}} f(\boldsymbol{x}). \quad (9)$$

Here, $f(\boldsymbol{x}) \geq 0$ for any $\boldsymbol{x} \preceq \boldsymbol{b}$ is equivalent to say $f(\boldsymbol{0})+f(\boldsymbol{b}) \geq 0$, namely $f(\boldsymbol{b}) \geq 0$ because of $f(\boldsymbol{0})=0$, which is a natural inference from the dr-submodularity.

Algorithm 1 Lattice-BasedDoubleGreedy

Input: $f:\mathbb{Z}_+^d \to \mathbb{R}$, $[\boldsymbol{s},\boldsymbol{t}]$ where $\boldsymbol{s} \preceq \boldsymbol{t} \in \mathbb{Z}_+^d$
Output: $\boldsymbol{x} \in \mathbb{Z}_+^d$
Initialize: $\boldsymbol{x} \leftarrow \boldsymbol{s}$, $\boldsymbol{y} \leftarrow \boldsymbol{t}$
for $i \in [d]$ **do**
  while $\boldsymbol{x}(i) < \boldsymbol{y}(i)$ **do**
    $a \leftarrow f(\boldsymbol{e}_i|\boldsymbol{x})$ and $b \leftarrow f(-\boldsymbol{e}_i|\boldsymbol{y})$
    $a' \leftarrow \max\{a,0\}$ and $b' \leftarrow \max\{b,0\}$
    $r \leftarrow$ Uniform(0, 1)
    (Note: we set $a'/(a'+b')=1$ if $a'=b'=0$)
    if $r \leq a'/(a'+b')$ **then**
      $\boldsymbol{x} \leftarrow \boldsymbol{x}+\boldsymbol{e}_i$ and $\boldsymbol{y} \leftarrow \boldsymbol{y}$
    else
      $\boldsymbol{y} \leftarrow \boldsymbol{y}-\boldsymbol{e}_i$ and $\boldsymbol{x} \leftarrow \boldsymbol{x}$
    end if
  end while
end for
return $\boldsymbol{x} (=\boldsymbol{y})$

B. Lattice-Based Iterative Pruning

According to Theorem 4, the approximation is based on an assumption that $f(\boldsymbol{0})+f(\boldsymbol{b}) \geq 0$. This is almost impossible in many real applications. It means that we are able to gain profit if giving all marketing actions full investments, which is ridiculous for viral marketing. However, a valid approximation ratio cannot be obtained by using Algorithm 1 when $f(\boldsymbol{b}) < 0$ exists. To address this problem, Tang *et al.* [9] proposed a groundbreaking techniques, called iterative pruning, to reduce the search space such that the objective is nonnegative in this space and without losing approximation guarantee. However, their techniques are designed for set functions, and they cannot be applied to vector functions directly. A set can be regarded as a $\{0,1\}$-vector, which is more convenient to handle. For the integer lattice domain, we have to consider each component repeatedly and handle boundary conditions carefully. Thus, we develop an iterative pruning technique suitable for dr-submodular functions in this section, which is a nontrivial transformation from set to integer lattice.

Given a dr-submodular function $f(\boldsymbol{x})$ defined on $\boldsymbol{x} \preceq \boldsymbol{b}$, we have two vectors $\boldsymbol{g}_1$ and $\boldsymbol{h}_1$ such that: 1) $\boldsymbol{g}_1(i)=0$ if $f(\boldsymbol{e}_i|\boldsymbol{b}-\boldsymbol{b}(i)\boldsymbol{e}_i) \leq 0$ or else $\boldsymbol{g}_1(i)=\max\{k: f(\boldsymbol{e}_i|\boldsymbol{b}-\boldsymbol{b}(i)\boldsymbol{e}_i+(k-1)\boldsymbol{e}_i) > 0\}$ for $k \in \{1,\ldots,\boldsymbol{b}(i)\}$ and 2) $\boldsymbol{h}_1(i)=0$ if $f(\boldsymbol{e}_i|\boldsymbol{0}) < 0$ or else $\boldsymbol{h}_1(i)=\max\{k: f(\boldsymbol{e}_i|\boldsymbol{0}+(k-1)\boldsymbol{e}_i) \geq 0\}$ for $k \in \{1,\ldots,\boldsymbol{b}(i)\}$.

Lemma 2: We have $\boldsymbol{g}_1 \preceq \boldsymbol{h}_1$.

Proof: For any component $i \in [d]$, we have $f(\boldsymbol{e}_i|\boldsymbol{b}-\boldsymbol{b}(i)\boldsymbol{e}_i+\boldsymbol{g}_1(i)\boldsymbol{e}_i) \leq 0$, but $f(\boldsymbol{e}_i|\boldsymbol{b}-\boldsymbol{b}(i)\boldsymbol{e}_i+(\boldsymbol{g}_1(i)-1)\boldsymbol{e}_i) > 0$, and $f(\boldsymbol{e}_i|\boldsymbol{0}+\boldsymbol{h}_1(i)\boldsymbol{e}_i) < 0$, but $f(\boldsymbol{e}_i|\boldsymbol{0}+(\boldsymbol{h}_1(i)-1)\boldsymbol{e}_i) \geq 0$. Because of dr-submodularity, it satisfies $f(\boldsymbol{e}_i|\boldsymbol{b}-\boldsymbol{b}(i)\boldsymbol{e}_i+\boldsymbol{h}_1(i)\boldsymbol{e}_i) \leq f(\boldsymbol{e}_i|\boldsymbol{0}+\boldsymbol{h}_1(i)\boldsymbol{e}_i) < 0$. Thus, $(\boldsymbol{g}_1(i)-1) < \boldsymbol{h}_1(i)$ and $\boldsymbol{g}_1(i) \leq \boldsymbol{h}_1(i)$. Subsequently, $\boldsymbol{g}_1 \preceq \boldsymbol{h}_1$. □

Then, we define a collection denoted by $\pi_1 = [\boldsymbol{g}_1,\boldsymbol{h}_1]$ that contains all the marketing vectors $\boldsymbol{x}$ that satisfies $\boldsymbol{g}_1 \preceq \boldsymbol{h}_1$. Apparently, π_1 is a subcollection of $[\boldsymbol{0},\boldsymbol{b}]$.

Algorithm 2 Lattice-BasedPruning

Input: $f : \mathbb{Z}_+^d \rightarrow \mathbb{R}$, $\boldsymbol{b} \in \mathbb{Z}_+^d$

Output: $\pi_t = [\boldsymbol{g}_t, \boldsymbol{h}_t]$

Initialize: $\boldsymbol{g}_t \leftarrow \boldsymbol{0}$, $\boldsymbol{h}_t \leftarrow \boldsymbol{b}$
Initialize: $t \leftarrow 0$
while $\boldsymbol{g}_t \neq \boldsymbol{g}_{t-1}$ or $\boldsymbol{h}_t \neq \boldsymbol{h}_{t-1}$ **do**
  for $i \in [d]$ **do**
    if $\boldsymbol{g}_t(i) = \boldsymbol{h}_t(i)$ **then**
      $\boldsymbol{g}_{t+1}(i) \leftarrow \boldsymbol{g}_t(i)$
      $\boldsymbol{h}_{t+1}(i) \leftarrow \boldsymbol{h}_t(i)$
      Continue
    end if
    if $f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + \boldsymbol{g}_t(i)\boldsymbol{e}_i) \leq 0$ **then**
      $\boldsymbol{g}_{t+1}(i) \leftarrow \boldsymbol{g}_t(i)$
    else
      $\boldsymbol{g}_{t+1}(i) \leftarrow \boldsymbol{g}_t(i) + \max\{k : f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + \boldsymbol{g}_t(i)\boldsymbol{e}_i + (k-1)\boldsymbol{e}_i) > 0\}$, $k \in \{1, \dots, \boldsymbol{h}_t(i) - \boldsymbol{g}_t(i)\}$
    end if
    if $f(\boldsymbol{e}_i|\boldsymbol{g}_t) < 0$ **then**
      $\boldsymbol{h}_{t+1}(i) \leftarrow \boldsymbol{h}_t(i)$
    else
      $\boldsymbol{h}_{t+1}(i) \leftarrow \boldsymbol{g}_t(i) + \max\{k : f(\boldsymbol{e}_i|\boldsymbol{g}_t + (k-1)\boldsymbol{e}_i) \geq 0\}$, $k \in \{0, \dots, \boldsymbol{h}_t(i) - \boldsymbol{g}_t(i)\}$
    end if
  end for
  $t \leftarrow t + 1$
end while
return $\pi_t = [\boldsymbol{g}_t, \boldsymbol{h}_t]$

Lemma 3: All optimal solutions $\boldsymbol{x}^*$ that satisfy $f(\boldsymbol{x}^*) = \max_{\boldsymbol{x} \preceq \boldsymbol{b}} f(\boldsymbol{x})$ are contained in the collection $\pi_1 = [\boldsymbol{g}_1, \boldsymbol{h}_1]$, i.e., $\boldsymbol{g}_1 \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}_1$ for all $\boldsymbol{x}^*$.

Proof: For any component $i \in [d]$, we consider any vector $\boldsymbol{x}$ with $\boldsymbol{x}(i) < \boldsymbol{g}_1(i)$ and have $f(\boldsymbol{e}_i|\boldsymbol{x}) \geq f(\boldsymbol{e}_i|\boldsymbol{b} - \boldsymbol{b}(i)\boldsymbol{e}_i + \boldsymbol{x}(i)\boldsymbol{e}_i) > 0$ because of dr-submodularity. Thereby, $\boldsymbol{x} + \boldsymbol{e}_i$ has a larger profit than $\boldsymbol{x}$ for sure, so the ith component of the optimal marketing vector $\boldsymbol{x}^*$ at least equals $\boldsymbol{x}(i) + 1$, which indicates that $\boldsymbol{x}^*(i) \geq \boldsymbol{g}_1(i)$. On the other have, consider $\boldsymbol{x}(i) \geq \boldsymbol{h}_1(i)$, and we have $f(\boldsymbol{e}_i|\boldsymbol{x}) \leq f(\boldsymbol{e}_i|\boldsymbol{0} + \boldsymbol{x}(i)\boldsymbol{e}_i) < 0$. Thereby, $\boldsymbol{x} + \boldsymbol{e}_i$ has a less profit than $\boldsymbol{x}$ for sure, so the ith component of the optimal marketing vector $\boldsymbol{x}^*$ at most equals $\boldsymbol{x}(i)$, which indicates that $\boldsymbol{x}^*(i) \leq \boldsymbol{h}_1(i)$. Thus, $\boldsymbol{g}_1 \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}_1$. □

From above, Lemma 3 determines a range for the optimal vector, thus reducing the search space. Then, the collection $\pi_1 = [\boldsymbol{g}_1, \boldsymbol{h}_1]$ can be pruned further in an iterative manner. Now, the upper bound of the optimal vector is $\boldsymbol{h}_1$, i.e., $\boldsymbol{x}^* \preceq \boldsymbol{h}_1$; hereafter, we are able to increase $\boldsymbol{g}_1$ to $\boldsymbol{g}_2$, where $\boldsymbol{g}_2(i) = \boldsymbol{g}_1(i)$ if $f(\boldsymbol{e}_i|\boldsymbol{h}_1 - \boldsymbol{h}_1(i)\boldsymbol{e}_i + \boldsymbol{g}_1(i)\boldsymbol{e}_i) \leq 0$ or else $\boldsymbol{g}_2(i) = \boldsymbol{g}_1(i) + \max\{k : f(\boldsymbol{e}_i|\boldsymbol{h}_1 - \boldsymbol{h}_1(i)\boldsymbol{e}_i + \boldsymbol{g}_1(i)\boldsymbol{e}_i + (k-1)\boldsymbol{e}_i) > 0\}$ for $k \in \{1, \dots, \boldsymbol{h}_1(i) - \boldsymbol{g}_1(i)\}$. The lower bound of the optimal vector is $\boldsymbol{g}_1$, i.e., $\boldsymbol{x}^* \succeq \boldsymbol{g}_1$, and similarly, we are able to decrease $\boldsymbol{h}_1$ to $\boldsymbol{h}_2$, where $\boldsymbol{h}_2(i) = \boldsymbol{h}_1(i)$ if $f(\boldsymbol{e}_i|\boldsymbol{g}_1) < 0$ or else $\boldsymbol{h}_2(i) = \boldsymbol{g}_1(i) + \max\{k : f(\boldsymbol{e}_i|\boldsymbol{g}_1 + (k-1)\boldsymbol{e}_i) \geq 0\}$ for $k \in \{1, \dots, \boldsymbol{h}_1(i) - \boldsymbol{g}_1(i)\}$. In this process, it generates a more compressed collection $\pi_2 = [\boldsymbol{g}_2, \boldsymbol{h}_2]$ than π_1. We repeat this process iteratively until $\boldsymbol{g}_t$ and $\boldsymbol{h}_t$ cannot be increased and decreased further. The Lattice-basedPruning algorithm is shown in Algorithm 2. The collection returned by Algorithm 2 is denoted by $\pi^\circ = [\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$.

Lemma 4: All optimal solutions $\boldsymbol{x}^*$ that satisfy $f(\boldsymbol{x}^*) = \max_{\boldsymbol{x} \preceq \boldsymbol{b}} f(\boldsymbol{x})$ are contained in the collection $\pi^\circ = [\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$ and $\boldsymbol{g}_t \preceq \boldsymbol{g}_{t+1} \preceq \boldsymbol{g}^\circ \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}^\circ \preceq \boldsymbol{h}_{t+1} \preceq \boldsymbol{h}_t$ holds for all $\boldsymbol{x}^*$ and any $t \geq 0$.

Proof: First, we show that the collection generated in current iteration is a subcollection of that generated in previous iteration, namely $\boldsymbol{g}_t \preceq \boldsymbol{g}_{t+1} \preceq \boldsymbol{h}_{t+1} \preceq \boldsymbol{h}_t$. We prove it by induction. In Lemma 2, we have shown that $\boldsymbol{g}_0 = \boldsymbol{0} \preceq \boldsymbol{g}_1 \preceq \boldsymbol{h}_1 \preceq \boldsymbol{h}_0 = \boldsymbol{b}$. For any $t > 1$, we assume that $\boldsymbol{g}_{t-1} \preceq \boldsymbol{g}_t \preceq \boldsymbol{h}_t \preceq \boldsymbol{h}_{t-1}$ is satisfied. Given a component $i \in [d]$, for any $q \leq \boldsymbol{g}_t(i)$, we have $f(\boldsymbol{e}_i|\boldsymbol{h}_{t-1} - \boldsymbol{h}_{t-1}(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$. Because of the dr-submodularity, we have $f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \geq f(\boldsymbol{e}_i|\boldsymbol{h}_{t-1} - \boldsymbol{h}_{t-1}(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$, which indicates $\boldsymbol{g}_t \preceq \boldsymbol{g}_{t+1}$. Similarly, for any $q \leq \boldsymbol{h}_{t+1}(i)$, we have $f(\boldsymbol{e}_i|\boldsymbol{g}_t - \boldsymbol{g}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \geq 0$. Because of the dr-submodularity, we have $f(\boldsymbol{e}_i|\boldsymbol{g}_{t-1} - \boldsymbol{g}_{t-1}(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \geq f(\boldsymbol{e}_i|\boldsymbol{g}_t - \boldsymbol{g}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \geq 0$, which indicates $\boldsymbol{h}_{t+1} \preceq \boldsymbol{h}_t$. Moreover, for any $q \leq \boldsymbol{g}_{t+1}(i)$, we have $f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$. Due to $\boldsymbol{g}_t \preceq \boldsymbol{h}_t$ and dr-submodularity, we have $f(\boldsymbol{e}_i|\boldsymbol{g}_t - \boldsymbol{g}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \geq f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$, which indicates $\boldsymbol{g}_{t+1} \preceq \boldsymbol{h}_{t+1}$. Thus, we conclude that $\boldsymbol{g}_t \preceq \boldsymbol{g}_{t+1} \preceq \boldsymbol{h}_{t+1} \preceq \boldsymbol{h}_t$ holds for any $t \geq 0$.

Then, we show that any optimal solutions $\boldsymbol{x}^*$ are contained in the collection $\pi^\circ = [\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$ returned by Algorithm 2, namely $\boldsymbol{g}^\circ \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}^\circ$. We prove it by induction. In Lemma 3, we have shown that $\boldsymbol{g}_1 \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}_1$. For any $t > 1$, we assume that $\boldsymbol{g}_t \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}_t$ is satisfied. Given a component $i \in [d]$, for any $q \leq \boldsymbol{g}_{t+1}(i)$, we have $f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$. Because of the dr-submodularity, we have $f(\boldsymbol{e}_i|\boldsymbol{x}^* - \boldsymbol{x}^*(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \geq f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$, which implies that $\boldsymbol{x}^*(i) \geq q$. Otherwise, if $\boldsymbol{x}^*(i) < q$, we have $f(\boldsymbol{e}_i|\boldsymbol{x}^*) > 0$, which contradicts the optimality of $\boldsymbol{x}^*$, and thus, $\boldsymbol{x}^* \succeq \boldsymbol{g}_{t+1}$. Similarly, for any $q > \boldsymbol{h}_{t+1}(i)$, we have $f(\boldsymbol{e}_i|\boldsymbol{g}_t - \boldsymbol{g}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) < 0$. Because of the dr-submodularity, we have $f(\boldsymbol{e}_i|\boldsymbol{x}^* - \boldsymbol{x}^*(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) < f(\boldsymbol{e}_i|\boldsymbol{g}_t - \boldsymbol{g}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) < 0$, which implies $\boldsymbol{x}^*(i) < q$. Otherwise, if $\boldsymbol{x}^*(i) \geq q$, we have $f(\boldsymbol{e}_i|\boldsymbol{x}^* - \boldsymbol{x}^*(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) < 0$, which contradicts the optimality of $\boldsymbol{x}^*$, thus $\boldsymbol{x}^* \preceq \boldsymbol{h}_{t+1}$. Thus, we conclude that $\boldsymbol{g}_{t+1} \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}_{t+1}$ holds for any $t \geq 0$, and $\boldsymbol{g}^\circ \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}^\circ$. The proof of lemma is completed. □

Lemma 5: For any two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^d$ with $\boldsymbol{x} \preceq \boldsymbol{y}$ and a dr-submodular function $f : \mathbb{Z}_+^d \rightarrow \mathbb{R}$, we have

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + \sum_{i=1}^{d} \sum_{j=1}^{\boldsymbol{z}(i)} f\left(\boldsymbol{e}_i|\boldsymbol{x} + \sum_{k=1}^{i-1} \sum_{l=1}^{\boldsymbol{z}(k)} \boldsymbol{e}_k + \sum_{l=1}^{j-1} \boldsymbol{e}_i\right) \quad (10)$$

where we define $\boldsymbol{z}(i) = \boldsymbol{y}(i) - \boldsymbol{x}(i)$.

To understand Lemma 5, we give a simple example here. Let vector $\boldsymbol{x}$ and $\boldsymbol{y}$ be $\boldsymbol{x} = (1, 1)$, $\boldsymbol{y} = (2, 3)$, and subsequently, we can see $\boldsymbol{x} \preceq \boldsymbol{y}$ and $\boldsymbol{z} = (1, 2)$. From the definition of (10), we have $f(\boldsymbol{x}) + f(\boldsymbol{e}_1|\boldsymbol{x}) + f(\boldsymbol{e}_2|\boldsymbol{x} + \boldsymbol{e}_1) + f(\boldsymbol{e}_2|\boldsymbol{x} + \boldsymbol{e}_1 + \boldsymbol{e}_2) = f(\boldsymbol{x} + \boldsymbol{e}_1 + 2\boldsymbol{e}_2) = f(\boldsymbol{y})$, which reflects the essence and correctness of Lemma 5 definitely.

Lemma 6: $f(\boldsymbol{g}_t)$ and $f(\boldsymbol{h}_t)$ are monotone nondecreasing with the increase of t.

Proof: We prove that $f(\boldsymbol{g}_t) \le f(\boldsymbol{g}_{t+1})$ and $f(\boldsymbol{h}_t) \le f(\boldsymbol{h}_{t+1})$. Given a component $i \in [d]$, for any $q \le \boldsymbol{g}_{t+1}(i)$, we have $f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$. Because of the dr-submodularity, we have $f(\boldsymbol{e}_i|\boldsymbol{g}_{t+1} - \boldsymbol{g}_{t+1}(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \ge f(\boldsymbol{e}_i|\boldsymbol{h}_t - \boldsymbol{h}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) > 0$, where $\boldsymbol{g}_{t+1} \preceq \boldsymbol{h}_t$. According to the Lemma 5, that is

$$\begin{aligned} f(\boldsymbol{g}_{t+1}) &= f(\boldsymbol{g}_t) + \sum_{i=1}^{d}\sum_{j=1}^{z_t(i)} f\left(\boldsymbol{e}_i|\boldsymbol{g}_t + \sum_{k=1}^{i-1}\sum_{l=1}^{z_t(k)} \boldsymbol{e}_k + \sum_{l=1}^{j-1}\boldsymbol{e}_i\right) \\ &\ge f(\boldsymbol{g}_t) + \sum_{i=1}^{d}\sum_{j=1}^{z_t(i)} f\left(\boldsymbol{e}_i|\boldsymbol{g}_{t+1} - z_t(i)\boldsymbol{e}_i + \sum_{l=1}^{j-1}\boldsymbol{e}_i\right) \end{aligned} \tag{11}$$

where $z_t(i) = \boldsymbol{g}_{t+1}(i) - \boldsymbol{g}_t(i)$. The inequality (11) is established since its dr-submodularity, that is, $\boldsymbol{g}_t + \sum_{k=1}^{i-1}\sum_{l=1}^{z_t(k)} \boldsymbol{e}_k \preceq \boldsymbol{g}_{t+1} - z_t(i)\boldsymbol{e}_i$ definitely. Besides, since $f(\boldsymbol{e}_i|\boldsymbol{g}_{t+1} - z_t(i)\boldsymbol{e}_i + \sum_{l=1}^{j-1}\boldsymbol{e}_i) > 0$, we have $f(\boldsymbol{g}_{t+1}) \ge f(\boldsymbol{g}_t)$. Similarly, for any $q > \boldsymbol{h}_{t+1}(i)$, we have $f(\boldsymbol{e}_i|\boldsymbol{g}_t - \boldsymbol{g}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) < 0$. Because of the dr-submodularity, we have $f(\boldsymbol{e}_i|\boldsymbol{h}_{t+1} - \boldsymbol{h}_{t+1}(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) \le f(\boldsymbol{e}_i|\boldsymbol{g}_t - \boldsymbol{g}_t(i)\boldsymbol{e}_i + (q-1)\boldsymbol{e}_i) < 0$, where $\boldsymbol{g}_t \preceq \boldsymbol{h}_{t+1}$. According to Lemma 5, that is, $f(\boldsymbol{h}_t) =$

$$\begin{aligned} &= f(\boldsymbol{h}_{t+1}) + \sum_{i=1}^{d}\sum_{j=1}^{z_t(i)} f\left(\boldsymbol{e}_i|\boldsymbol{h}_{t+1} + \sum_{k=1}^{i-1}\sum_{l=1}^{z_t(k)} \boldsymbol{e}_k + \sum_{l=1}^{j-1}\boldsymbol{e}_i\right) \\ &\le f(\boldsymbol{h}_{t+1}) + \sum_{i=1}^{d}\sum_{j=1}^{z_t(i)} f\left(\boldsymbol{e}_i|\boldsymbol{h}_{t+1} + \sum_{l=1}^{j-1}\boldsymbol{e}_i\right) \end{aligned} \tag{12}$$

where $z_t(i) = \boldsymbol{h}_t(i) - \boldsymbol{h}_{t+1}(i)$. The inequality (12) is established since its dr-submodularity, that is, $\boldsymbol{h}_{t+1} + \sum_{k=1}^{i-1}\sum_{l=1}^{z_t(k)} \boldsymbol{e}_k \succeq \boldsymbol{h}_{t+1}$ definitely. Besides, since $f(\boldsymbol{e}_i|\boldsymbol{h}_{t+1} + \sum_{l=1}^{j-1}\boldsymbol{e}_i) < 0$, we have $f(\boldsymbol{h}_{t+1}) \ge f(\boldsymbol{h}_t)$. □

At this time, we can initialize $\boldsymbol{x}$ and $\boldsymbol{y}$ with $\boldsymbol{x} \leftarrow \boldsymbol{g}^\circ$ and $\boldsymbol{y} \leftarrow \boldsymbol{h}^\circ$ instead of starting with $\boldsymbol{x} \leftarrow \boldsymbol{0}$ and $\boldsymbol{y} \leftarrow \boldsymbol{b}$ in Algorithm 1, where the search space required to be checked is reduced to $[\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$. Then, we are able to build the approximation ratio for our revised lattice-based double greedy algorithm.

Lemma 7: If we initialize Algorithm 1 by $[\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$, the solution $\boldsymbol{x}^\circ$ returned by Algorithm 1 satisfies

$$\mathbb{E}\left[f(\boldsymbol{x}^\circ)\right] \ge \frac{f\left((\boldsymbol{x}^* \vee \boldsymbol{g}^\circ) \wedge \boldsymbol{h}^\circ\right) + \frac{1}{2}\cdot\left(f(\boldsymbol{g}^\circ) + f(\boldsymbol{h}^\circ)\right)}{2}. \tag{13}$$

Proof: It can be extended from the proof in [13, Lemma 3.1]. This procedure is complicated, so we omit here because of space limitation. □

C. Time Complexity

First, we assume that there is a value oracle for computing the marginal gain of increasing or decreasing component $i \in [d]$ by 1. If we initialize $\boldsymbol{x} \leftarrow \boldsymbol{0}$ and $\boldsymbol{y} \leftarrow \boldsymbol{b}$ at the beginning of lattice-based double greedy algorithm (Algorithm 1), we have to take $2\cdot\sum_{i=1}^{d}\boldsymbol{b}(i)$ times together for checking each component whether to increase or decrease it by 1. Consider shrinking collection $[\boldsymbol{0}, \boldsymbol{b}]$ to $[\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$ by applying lattice-based iterative pruning (Algorithm 2) first, and we use it to initialize $\boldsymbol{x}$ and $\boldsymbol{y}$ at the beginning of Algorithm 1 and then running Algorithm 1. For each component $i \in [d]$, we check its marginal gain $\boldsymbol{g}^\circ(i) + (\boldsymbol{b}(i) - \boldsymbol{h}^\circ(i))$ times in the iterative pruning, and thus, totally $\sum_{i=1}^{d}(\boldsymbol{g}^\circ(i) + (\boldsymbol{b}(i) - \boldsymbol{h}^\circ(i)))$ times. Then, we are required to check $2\cdot\sum_{i=1}^{d}(\boldsymbol{h}^\circ(i) - \boldsymbol{g}^\circ(i))$ times in subsequent double greedy initialized by $[\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$. Combining together, we have to check $\sum_{i=1}^{d}(\boldsymbol{b}(i) + \boldsymbol{h}^\circ(i) - \boldsymbol{g}^\circ(i)) \le 2\cdot\sum_{i=1}^{d}\boldsymbol{b}(i)$ times. Hence, the time complexity is $O(\|\boldsymbol{b}\|_1)$.

Theorem 5: For our CPM-MS problem, if we initialize Algorithm 1 by $[\boldsymbol{g}^\circ, \boldsymbol{h}^\circ]$ and $f(\boldsymbol{g}^\circ) + f(\boldsymbol{h}^\circ) \ge 0$ is satisfied, the marketing vector $\boldsymbol{x}^\circ$ returned by Algorithm 1 is a (1/2)-approximate solution in $O(\|\boldsymbol{b}\|_1)$ running time.

Proof: Based on Lemma 4, we have $\boldsymbol{g}^\circ \preceq \boldsymbol{x}^* \preceq \boldsymbol{h}^\circ$, and hence, $(\boldsymbol{x}^* \vee \boldsymbol{g}^\circ) \wedge \boldsymbol{h}^\circ = \boldsymbol{x}^*$. If $f(\boldsymbol{g}^\circ) + f(\boldsymbol{h}^\circ) \ge 0$, we can get that $\mathbb{E}[f(\boldsymbol{x}^\circ)] \ge (1/2)\cdot(f(\boldsymbol{x}^*) + (1/2)(f(\boldsymbol{g}^\circ) + f(\boldsymbol{h}^\circ))) \ge (1/2)\cdot f(\boldsymbol{x}^*) = (1/2)\cdot\max_{\boldsymbol{x}\preceq\boldsymbol{b}} f(\boldsymbol{x})$. □

From Theorem 5, it enables us to obtain the same approximation ratio by applying the lattice-based double greedy algorithm initialized by using iterative pruning if we have $f(\boldsymbol{g}^\circ) + f(\boldsymbol{h}^\circ) \ge 0$. According to Lemma 6, this is $f(\boldsymbol{0}) + f(\boldsymbol{b}) = f(\boldsymbol{g}_0) + f(\boldsymbol{h}_0) \le f(\boldsymbol{g}_1) + f(\boldsymbol{h}_1) \le \cdots \le f(\boldsymbol{g}^\circ) + f(\boldsymbol{h}^\circ)$. To achieve this condition $f(\boldsymbol{g}^\circ) + f(\boldsymbol{h}^\circ) \ge 0$ is much easier than $f(\boldsymbol{0}) + f(\boldsymbol{b}) \ge 0$. Therefore, the applications of Algorithm 1 with a theoretical bound are extended greatly by the technique of lattice-based iterative prunning.

VI. Speedup by Sampling Techniques

However, to compute the marginal gain of profit is a time-consuming process, and the running time is given by Theorem 3, which is not acceptable in a large-scale social graph as well as a large searching space. In this section, we discuss how to reduce their running time by sampling techniques.

A. Sampling Techniques

To overcome the #P-hardness of computing the objective $f(\cdot)$, we borrow from the idea of RIS [14]. In the beginning, consider traditional IM problem, and we need to introduce the concept of reverse reachable set (RR-set) first. Given a triggering model $\Omega = (G, \mathcal{D})$, a random RR-set can be generated by selecting a node $u \in V$ uniformly and sampling a graph realization g from Ω, and then collecting those nodes can reach u in g. RR-sets rooted at u is the collected nodes that are likely to influence u. A larger expected influence spread a seed set S has, the higher the probability that S intersects with a random RR-set is. Given a seed set S and a random RR-set R, we have $\sigma(S) = n\cdot\Pr[R \cap S \neq \emptyset]$.

Extended to the lattice domain, given a marketing vector $\boldsymbol{x}$, its expected influence spread under the triggering model Ω can be denoted by $\mu(\boldsymbol{x}) = n\cdot\mathbb{E}_R[1 - \prod_{u\in R}(1 - h_u(\boldsymbol{x}))]$ [25]. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_\theta\}$ be a collection of random RR-sets generated independently, and we have

$$\hat{f}(\mathcal{R}, \boldsymbol{x}) = \frac{n}{\theta}\cdot\sum_{R\in\mathcal{R}}\left(1 - \prod_{u\in R}(1 - h_u(\boldsymbol{x}))\right) - c(\boldsymbol{x}) \tag{14}$$

Algorithm 3 DG-IP-RIS

Input: $\hat{f}: \mathcal{R} \times \mathbb{Z}_+^d \to \mathbb{R}$, $\boldsymbol{b} \in \mathbb{Z}_+^d$, $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$, δ
Output: $\hat{\boldsymbol{x}} \in \mathbb{Z}_+^d$
1: Initialize: θ_1 define on (16)
2: $\underline{OPT} \leftarrow$ OptEstimation$(\hat{f}, \boldsymbol{b}, \theta_1)$
3: Initialize: θ_2, θ_3 defined on (17) (18)
4: $\theta \leftarrow \max(\theta_1, \theta_2, \theta_3)$
5: Generate a collection of random RR-sets $\mathcal{R}$ with $|\mathcal{R}| = \theta$
6: $[\hat{\boldsymbol{g}}^\circ, \hat{\boldsymbol{h}}^\circ] \leftarrow$ Lattice-basedPruning$(\hat{f}(\mathcal{R}, \cdot), \boldsymbol{b})$
7: $\hat{\boldsymbol{x}} \leftarrow$ Lattice-basedDoubleGreedy$(\hat{f}(\mathcal{R}, \cdot), [\hat{\boldsymbol{g}}^\circ, \hat{\boldsymbol{h}}^\circ])$
8: **return** $\hat{\boldsymbol{x}}$

Algorithm 4 OptEstimation

Input: $\hat{f}: \mathcal{R} \times \mathbb{Z}_+^d \to \mathbb{R}$, $\boldsymbol{b} \in \mathbb{Z}_+^d$, θ_1
Output: $\underline{OPT}$
1: Initialize: $\boldsymbol{x} \leftarrow 0$, $t \leftarrow 0$
2: Generate a collection of random RR-sets $\mathcal{R}$ with $|\mathcal{R}| = \theta_1$
3: **while** $t < \sum_{i=0}^d \boldsymbol{b}(i)$ **do**
4: $\quad i^* \leftarrow \arg\max_{i \in [d], \boldsymbol{x}(i) < \boldsymbol{b}(i)} \hat{f}(\boldsymbol{e}_i | \mathcal{R}, \boldsymbol{x})$
5: $\quad$ **if** $\hat{f}(\boldsymbol{e}_{i^*} | \mathcal{R}, \boldsymbol{x}) \leq 0$ **then**
6: $\quad\quad$ Break
7: $\quad$ **end if**
8: $\quad \boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{e}_{i^*}$, $t \leftarrow t + 1$
9: **end while**
10: $\underline{OPT} \leftarrow \hat{f}(\mathcal{R}, \boldsymbol{x}) - 2\varepsilon_1$
11: **return** $\underline{OPT}$

that is an unbiased estimator of $f(\boldsymbol{x})$. From here, the vector $\boldsymbol{x}$ that maximizes $\hat{f}(\mathcal{R}, \boldsymbol{x})$ will be close to the optimal solution intuitively and more and more close with the increase of $|\mathcal{R}|$. Similar to $f(\boldsymbol{x})$, fix the collection $\mathcal{R}$ and $\hat{f}(\mathcal{R}, \boldsymbol{x})$ is dr-submodular, but not monotone as well. By Theorem 5, Algorithm 1 offers a (1/2)-approximation if $\hat{f}(\mathcal{R}, \boldsymbol{g}^\circ) + \hat{f}(\mathcal{R}, \boldsymbol{h}^\circ) \geq 0$ is satisfied after the process of pruning.

Now, we begin to design our algorithm based on the idea of reverse sampling. First, we need to sample the enough number of random RR-sets so that its estimation to the objective function is accurate. Let ε_1, ε_2, and ε_3 be three adjustable parameters, where they satisfy

$$\varepsilon_2 + (1/2) \cdot \varepsilon_3 = \varepsilon \tag{15}$$

where $\varepsilon_1, \varepsilon_2$, and $\varepsilon_3 > 0$. Then, we can set that

$$\theta_1 = \sqrt{\frac{n^2 \cdot \ln\left(3\delta \cdot \prod_{i=1}^d (\boldsymbol{b}(i) + 1)\right)}{2\varepsilon_1^2}} \tag{16}$$

$$\theta_2 = \frac{n\left(2n + \varepsilon_2^2 \cdot \underline{\mathrm{OPT}}\right) \cdot \ln\left(3\delta \cdot \prod_{i=1}^d (\boldsymbol{b}(i) + 1)\right)}{\varepsilon_2^2 \cdot \underline{\mathrm{OPT}}^2} \tag{17}$$

$$\theta_3 = \frac{2n^2 \cdot \ln(3\delta)}{\varepsilon_3^2 \cdot \underline{\mathrm{OPT}}^2} \tag{18}$$

where the $\underline{\mathrm{OPT}}$ is the lower bound of the optimal objective $f(\boldsymbol{x}^*)$. The algorithm that combining double greedy with reverse sampling and iterative pruning, called DG-IP-RIS algorithm, is shown in Algorithm 3.

In DG-IP-RIS algorithm, we estimate the number of random RR-sets θ in line 4 and then generate a collection $\mathcal{R}$ of random RR-sets with the size of θ. The objective $\hat{f}(\mathcal{R}, \cdot)$ is computed based on this $\mathcal{R}$, from which we are able to get a solution by iterative pruning and double greedy algorithm. In the first step, we require to compute a lower bound of optimal value $f(\boldsymbol{x}^*)$, which is shown in Algorithm 4. Here, we increase the component by 1 with the largest marginal gain at each iteration until there is no component having positive marginal gain. After the while loop, we can obtain a vector $\boldsymbol{x}$ and set $\underline{\mathrm{OPT}} \leftarrow \hat{f}(\mathcal{R}, \boldsymbol{x}) - 2\varepsilon_1$ because $\Pr[|\hat{f}(\mathcal{R}, \boldsymbol{x}) - f(\boldsymbol{x})| \leq \varepsilon_1]$ is satisfied with a high probability under the setting of θ_1. Like this, we have $\underline{\mathrm{OPT}} > 0$ as well because the largest marginal gain should be greater than 0 due to the dr-submodularity or else the definition of our problem is not valid and meaningless. For convenience, given a random RR-set R, we denote $p(R, \boldsymbol{x}) = 1 - \prod_{u \in R}(1 - h_u(\boldsymbol{x}))$ in subsequent proof.

Lemma 8: The $\underline{\mathrm{OPT}}$ returned by Algorithm 4 satisfies $f(\boldsymbol{x}^*) \geq \underline{\mathrm{OPT}}$ with at least $1 - 1/(3\delta)$ probability.

Proof: For any marketing vector $\boldsymbol{x}$, we want to obtain $\Pr[|\hat{f}(\mathcal{R}, \boldsymbol{x}) - f(\boldsymbol{x})| \geq \varepsilon_1] \leq 1/(3\delta \cdot \prod_{i=1}^d \boldsymbol{b}(i))$. By the additive form of Chernoff–Hoeffding inequality, it is equivalent to compute, that is

$$\Pr\left[\left|\frac{1}{\theta_1} \sum p(R_i, \boldsymbol{x}) - \frac{\mu(\boldsymbol{x})}{n}\right| \geq \frac{\varepsilon_1}{n}\right] \leq \exp\left(-\frac{2\theta_1^2 \varepsilon_1^2}{n^2}\right).$$

When θ_1 is defined as (16), we have $1/(3\delta \cdot \prod_{i=1}^d (\boldsymbol{b}(i) + 1)) = \exp(-2\theta_1^2 \varepsilon_1^2 / n^2)$ definitely. By the union bound, the above relationship holds for the $\boldsymbol{x}'$ generated in line 10 of Algorithm 3 with a probability less than $1/(3\delta)$. □

Remark 1: Given a marketing vector $\boldsymbol{x} \preceq \boldsymbol{b}$, for each component $i \in [d]$, the possible values of $\boldsymbol{x}(i)$ are $\{0, 1, 2, \ldots, \boldsymbol{b}(i)\}$, and thus, the number of possible values for $\boldsymbol{x}(i)$ is $\boldsymbol{b}(i) + 1$. Thereby, the total number of possible combinations for vector $\boldsymbol{x}$ is $\prod_{i=1}^d (\boldsymbol{b}(i) + 1)$, which explains why the union bound in the previous lemma happened.

Lemma 9 (Chernoff Bounds [28]): Given a collection $\mathcal{Z} = \{Z_1, Z_2, \ldots, Z_\theta\}$, each $Z_i \in [0, 1]$ is an independent identically distributed (i.i.d.) random variable with $\mathbb{E}[Z_i] = \nu$, and we have

$$\Pr\left[\sum_{i=1}^{\theta} Z_i \geq (1 + \gamma) \cdot \nu\theta\right] \leq \exp\left(-\frac{\gamma^2 \cdot \nu\theta}{2 + \gamma}\right) \tag{19}$$

$$\Pr\left[\sum_{i=1}^{\theta} Z_i \leq (1 - \gamma) \cdot \nu\theta\right] \leq \exp\left(-\frac{\gamma^2 \cdot \nu\theta}{2}\right) \tag{20}$$

where we assume that $\gamma > 0$.

Lemma 10: Given a collection $\mathcal{R}$ with $|\mathcal{R}| = \theta_2$, for any marketing vector $\boldsymbol{x} \preceq \boldsymbol{b}$, it satisfies $\hat{f}(\mathcal{R}, \boldsymbol{x}) - f(\boldsymbol{x}) < \varepsilon_2 \cdot f(\boldsymbol{x}^*)$ with at least $1 - 1/(3\delta)$ probability.

Proof: For any marketing vector $\boldsymbol{x}$, we want to obtain $\Pr[\hat{f}(\mathcal{R}, \boldsymbol{x}) - f(\boldsymbol{x}) \geq \varepsilon_2 \cdot f(\boldsymbol{x}^*)] \leq 1/(3\delta \cdot \prod_{i=1}^d (\boldsymbol{b}(i) + 1))$. By the Chernoff bound, defined as (19), it is equivalent to compute, that is

$$\Pr\left[\sum p(R_i, \boldsymbol{x}) \geq \left(1 + \frac{\varepsilon_2 f(\boldsymbol{x}^*)}{\mu(\boldsymbol{x})}\right) \cdot \frac{\mu(\boldsymbol{x})\theta_2}{n}\right] \leq \exp\left(-\frac{\left(\frac{\varepsilon_2 f(\boldsymbol{x}^*)}{\mu(\boldsymbol{x})}\right)^2 \cdot \frac{\mu(\boldsymbol{x})\theta_2}{n}}{2 + \frac{\varepsilon_2 f(\boldsymbol{x}^*)}{\mu(\boldsymbol{x})}}\right). \quad (21)$$

From Lemma 9 and $\mu(\boldsymbol{x}) \leq n$, we have

$$(21) \leq \exp\left(-\frac{\theta_2 \cdot \varepsilon_2^2 \cdot \underline{\mathrm{OPT}}^2}{n \cdot (2n + \varepsilon_2 \cdot \underline{\mathrm{OPT}})}\right) \leq \frac{1}{3\delta \cdot \prod_{i=1}^{d}(\boldsymbol{b}(i)+1)}.$$

By the union bound, the above relationship holds for any $\boldsymbol{x} \preceq \boldsymbol{b}$ with at most $1/(3\delta)$ probability. □

Lemma 11: Given a collection $\mathcal{R}$ with $|\mathcal{R}| = \theta_3$, for an optimal solution $\boldsymbol{x}^*$, it satisfies $\hat{f}(\mathcal{R}, \boldsymbol{x}^*) - f(\boldsymbol{x}^*) > -\varepsilon_3 \cdot f(\boldsymbol{x}^*)$ with at least $1 - 1/(3\delta)$ probability.

Proof: For an optimal solution $\boldsymbol{x}^*$, we want to obtain $\Pr[\hat{f}(\mathcal{R}, \boldsymbol{x}^*) - f(\boldsymbol{x}^*) \leq -\varepsilon_3 \cdot f(\boldsymbol{x}^*)] \leq 1/(3\delta)$. By the Chernoff bound, defined as (20), it is equivalent to compute, that is

$$\Pr\left[\sum p(R_i, \boldsymbol{x}^*) \leq \left(1 - \frac{\varepsilon_3 f(\boldsymbol{x}^*)}{\mu(\boldsymbol{x}^*)}\right) \cdot \frac{\mu(\boldsymbol{x}^*)\theta_2}{n}\right] \leq \exp\left(-\frac{\left(\frac{\varepsilon_3 f(\boldsymbol{x}^*)}{\mu(\boldsymbol{x}^*)}\right)^2 \cdot \frac{\mu(\boldsymbol{x}^*)\theta_3}{n}}{2}\right). \quad (22)$$

From Lemma 9 and $\mu(\boldsymbol{x}) \leq n$, we have

$$(22) \leq \exp\left(-\frac{\theta_3 \cdot \varepsilon_3^2 \cdot \underline{\mathrm{OPT}}^2}{2n^2}\right) \leq \frac{1}{3\delta}.$$

The above relationship holds for the optimal solution $\boldsymbol{x}^*$ with at most $\leq 1/(3\delta)$ probability. □

B. Time Complexity

First, we consider the running time of Algorithm 3. Given a collection R with $|R| = \theta = \max\{\theta_1, \theta_2, \theta_3\}$, we have $\theta = O(n^2)$. To compute $\hat{f}(R, \cdot)$, it takes $O(\theta n)$ time, and to generate a random RR-set, it takes $O(m)$ times. Thus, the time complexity of Algorithm 3 is $O(m\theta + \|\boldsymbol{b}\|_1 \cdot n\theta) = O((m+n)n^2)$. Besides, this running time can be reduced further. Look at the forms of (16)–(18), θ_1 is apparently less than θ_2 and θ_3. Therefore, we are able to select the remaining two parameters $(\varepsilon_2, \varepsilon_3)$ such that $(\varepsilon_2, \varepsilon_3) = \arg\min_{\varepsilon_2 + (1/2)\cdot\varepsilon_3 = \varepsilon} \max\{\theta_2, \theta_3\}$.

Let $\hat{\boldsymbol{x}}^\circ$ be the result returned by Algorithm 3. If $\hat{f}(\mathcal{R}, \hat{\boldsymbol{x}}^\circ)$ and $\hat{f}(\mathcal{R}, \boldsymbol{x}^*)$ are accurate estimations to $f(\hat{\boldsymbol{x}}^\circ)$ and $f(\boldsymbol{x}^*)$, we can say that this solution $\hat{\boldsymbol{x}}$ has an effective approximation guarantee, which is shown in Theorem 6.

Theorem 6: For our CPM-MS problem, if it satisfies $\hat{f}(\mathcal{R}, \hat{\boldsymbol{g}}^\circ) + \hat{f}(\mathcal{R}, \hat{\boldsymbol{h}}^\circ) \geq 0$, for any $\varepsilon \in (0, 1/2)$ and $\delta > 0$, the marketing vector $\hat{\boldsymbol{x}}^\circ$ returned by Algorithm 3 is a $(1/2 - \varepsilon)$-approximation solution with at least $1 - 1/\delta$ probability in $O((m+n)n^2)$ running time.

Proof: Based on Lemma 10, $\hat{f}(\mathcal{R}, \hat{\boldsymbol{x}}^\circ) - f(\hat{\boldsymbol{x}}^\circ) < \varepsilon_2 \cdot f(\boldsymbol{x}^*)$ holds with at least $1 - 1/(3\delta)$ probability, and on Theorem 5, we have $\hat{f}(\mathcal{R}, \hat{\boldsymbol{x}}^\circ) \geq (1/2) \cdot \hat{f}(\mathcal{R}, \boldsymbol{x}^*)$. Thus, $f(\hat{\boldsymbol{x}}^\circ) \geq \hat{f}(\mathcal{R}, \hat{\boldsymbol{x}}^\circ) - \varepsilon_2 \cdot f(\boldsymbol{x}^*) \geq (1/2) \cdot \hat{f}(\mathcal{R}, \boldsymbol{x}^*) - \varepsilon_2 \cdot f(\boldsymbol{x}^*)$. By Lemma 11, $\hat{f}(\mathcal{R}, \boldsymbol{x}^*) - f(\boldsymbol{x}^*) > -\varepsilon_3 \cdot f(\boldsymbol{x}^*)$ holds with at least $1 - 1/(3\delta)$ probability, and thus, we have $f(\hat{\boldsymbol{x}}^\circ) \geq (1/2 - (\varepsilon_2 + 1/2 \cdot \varepsilon_3)) \cdot f(\boldsymbol{x}^*) = (1/2 - \varepsilon) \cdot f(\boldsymbol{x}^*)$. Combined with that $f(\boldsymbol{x}^*) \geq \underline{\mathrm{OPT}}$ holds with $1 - 1/(3\delta)$, by the union bound, (23) holds with at least $1 - 1/\delta$ probability. □

TABLE I
DATA SETS STATISTICS ($K = 10^3$)

Dataset	n	m	Type	Avg.Degree
NetScience	0.4K	1.01K	undirected	5.00
Wiki	1.0K	3.15K	directed	6.20
HetHEPT	12.0K	118.5K	undirected	19.8
Epinions	75.9K	508.8K	directed	13.4

VII. Experiments

In this section, we carry out several experiments on different data sets to test the efficiency of DG-IP-RIS algorithm (Algorithm 3) and its effectiveness compared to other heuristic algorithms. All of our experiments are programmed by python and run on Windows machine. There are four data sets used in our experiments.

1) *NetScience [29]:* A coauthorship network, coauthorship among scientists to publish papers about network science.
2) *Wiki [29]:* A who-votes-on-whom network, which come from the collection Wikipedia voting.
3) *HetHEPT [30]:* An academic collaboration relationship on high-energy physics area.
4) *Epinions [30]:* A who-trust-whom OSN on Epinions.com, a general consumer review site. The statistics information of these four data sets is represented in Table I. For undirected graph, each undirected edge is replaced with two reversed directed edges.

A. Experimental Settings

We test different algorithms based on the IC-/LT-model. For the IC-model, the diffusion probability p_{uv} for each $(u, v) \in E$ is set to the inverse of v's in-degree, i.e., $p_{uv} = 1/|N^-(v)|$, and for the LT-model, the weight $b_{uv} = 1/|N^-(v)|$ for each $(u, v) \in E$ is set as well, which are adopted by previous studies of IM widely [3], [14]–[16], [20]. Then, we need to consider our strategy function, that is

$$h_u(\boldsymbol{x}) = 1 - \prod_{i\in[d]} \prod_{j=1}^{\boldsymbol{x}(i)} \left(1 - \eta^{j-1} \cdot r_{ui}\right) \quad (23)$$

where $\eta \in (0, 1)$ is an attenuation coefficient and $r_{ui} \in [0, 1]$ for $u \in V$ and $i \in [d]$, where a unit of investment to marketing action M_i activates user u to be a seed with the probability r_{ui}, and each activation is independent. Here, we define vector $\boldsymbol{r}_u = (r_{u1}, r_{u2}, \ldots, r_{ud})$.

We assume that there are five marketing action totally, namely $\boldsymbol{x} = (x_1, x_2, \ldots, x_5)$ and $d = 5$, and $\boldsymbol{b} = \{5\}^d$. Thus, $\boldsymbol{x}(i) \leq 5$ for each $i \in [5]$. Besides, we set

TABLE II

PARAMETERS SETTING FOR ALGORITHMS THAT ADOPT SPEEDUP BY SAMPLING TECHNIQUES

Dataset	ε_1	ε_2	ε_3	δ
NetScience	0.10	0.10	0.10	10.00
Wiki	0.10	0.10	0.10	10.00
HetHEPT	0.15	0.10	0.10	10.00
Epinions	1.00	0.10	0.10	10.00

$\eta = 0.8$, $\{r_{u1}, r_{u3}\}$ is sampled from $[0, 0.1]$, and $\{r_{u2}, r_{u4}, r_{u5}\}$ is sampled from $[0, 0.05]$ uniformly. Apparently, $h_u(\boldsymbol{x})$ is monotone and dr-submodular with respect to $\boldsymbol{x}$. For example, consider a marketing vector $\boldsymbol{x} = (1, 3, 0, 0, 2)$ and a node u with $\boldsymbol{r}_u = (0.1, 0.04, 0.08, 0, 0.05)$, and we have $h_u(\boldsymbol{x}) = 1-[(1-0.1)][(1-0.04)(1-0.8\times0.04)(1-0.8^2\times0.04)][(1-0.05)(1-0.8\times0.05)] = 0.257$ definitely. For the cost function c, we adopt a uniform cost distribution. The cost c_i for a unit of marketing action M_i, $i \in [d]$, is set as $c_i = \lambda\cdot n/\|\boldsymbol{b}\|_1$, where $\lambda \geq 0$ is a cost coefficient. The cost coefficient λ defined above is used to regulate the effect of cost on objective function. For example, $f(\cdot)$ is monotone dr-submodular if $\lambda = 0$. When we set $\lambda = 1$, it implies $f(\boldsymbol{b}) = 0$ if all users in a given social network can be influenced by full marketing vector $\boldsymbol{b}$ or else this profit is negative. If $\lambda > 1$, we have $f(\boldsymbol{b}) < 0$ definitely.

In addition, the number of MC simulations for each estimation to profit function is 2000. For those algorithms that adopt speedup by sampling techniques, the parameters setting of four data sets is shown in Table II. Next, we denote that "XXX" is achieved by MC simulations, but "XXXS" is achieved with speedup by sampling techniques. The algorithms we compare in this experiment are shown as follows.

1) *DG(S):* Lattice-based double greedy feed with $[\mathbf{0}, \boldsymbol{b}]$.
2) *DGIT(S):* Lattice-based double greedy feed with the collection returned by lattice-based iterative pruning.
3) *Greedy(S):* Select the component with maximum marginal gain until no one has positive gain.
4) *Random:* Select the component randomly until reaching negative marginal gain.

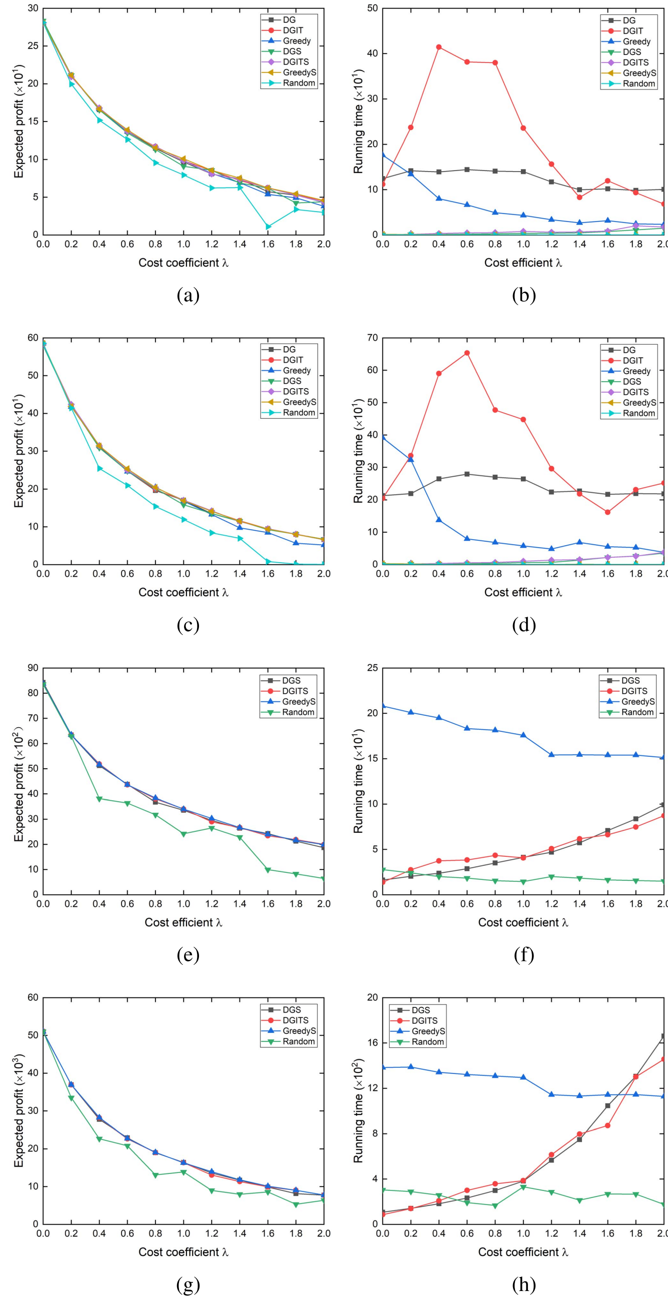

Fig. 1. Performance and running time comparisons among different algorithms under the IC-model. (a) NetScience, Performance. (b) NetScience, Time (s). (c) Wiki, Performance. (d) Wiki, Time (s). (e) HetHEPT, Performance. (f) HetHEPT, Time (s). (g) Epinions, Performance. (h) Epinions, Time (s).

B. Experimental Results

Figs. 1 and 2 show the expected profit and running time produced by different algorithms under the IC-model and LT-model, respectively. From the left columns of Figs. 1 and 2, the expected profits decrease with the increase of cost coefficient, which is obvious because a larger cost coefficient implies a larger cost for a unit of investment. Its trend is close to the inverse proportional relationship, namely $f \propto (1/\lambda)$. Then, the expected profits achieved by DG, DGIT, DGS, and DGITS(DG-IP-RIS) only have very slight even negligible gaps. By comparing the performance between DG and DGS (between DGIT and DGITS), it can show that speedup by sampling techniques is completely effective, which can estimate the objective function accurately. By comparing the performance between DG and DGIT (between DGS and DGITS), it can prove that the optimal solution lies in the shrinked collection returned by iterative pruning because DGIT does not make the performance of original DG worse. It means that the expected profit will not be reduced at least if we initial double greedy with the shrinked collection returned by iterative pruning. However, doing such a thing can provide a theoretical bound, so as to avoid some extreme situations. In addition, even if Greedy(S) gives a satisfactory solution in our experiment, there are still some exceptions, for example, in Figs. 1(c) and 2(g). It happens in some positions with larger cost coefficients.

From the right columns of Figs. 1 and 2, the trend of running time with cost coefficient is a little complex, but

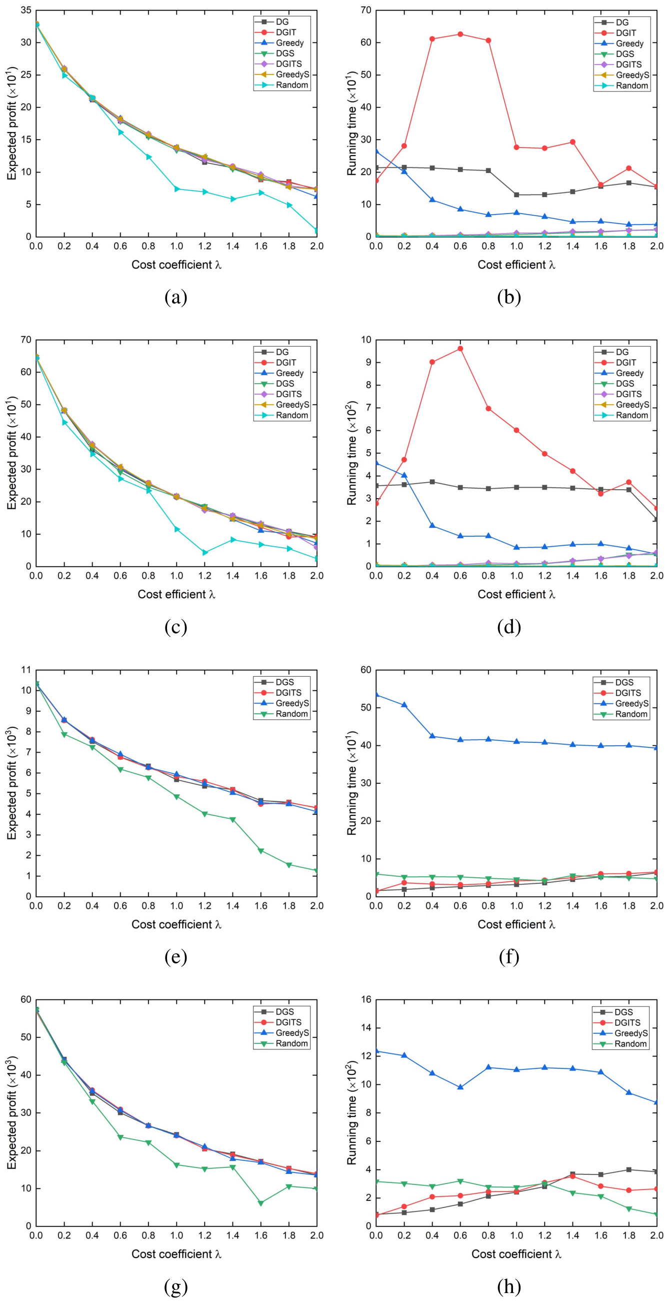

Fig. 2. Performance and running time comparisons among different algorithms under the LT-model. (a) NetScience, Performance. (b) NetScience, Time (s). (c) Wiki, Performance. (d) Wiki, Time (s). (e) HetHEPT, Performance. (f) HetHEPT, Time (s). (g) Epinions, Performance. (h) Epinions, Time (s).

there are two apparent characteristics. First, by comparing between DG and DGS (between DGIT and DGITS or between Greedy and GreedyS), we can see that their running times are reduced significantly by our sampling techniques. Here, in order to test the running time of different algorithms, we do not use parallel acceleration in our implementations. Generally speaking, the running times of algorithms implemented by sampling do not exceed 10% of the corresponding algorithms implemented by MC simulations in average. Second, look at DGS and DGITS, their running times increase with the increase of cost confidence. This is because the lower bound of optimal solution returned by Algorithm 4 will be smaller and

TABLE III
SUM OF INITIALIZED OBJECTIVE VALUE UNDER THE IC-MODEL

	NetScience		Wiki		HetHEPT	
λ	A	B	A	B	A	B
0.8	-22	**219**	-127	**379**	-586	**7050**
1.0	-97	**178**	-303	**256**	-2860	**848**
1.2	-174	**101**	-481	**213**	-5065	**751**
1.4	-250	-2	-658	**147**	-7329	-317
1.6	-325	**82**	-836	-11	-9523	-463
1.8	-401	**55**	-1015	-269	-11795	**500**
2.0	-477	-17	-1192	-792	-14031	-137

TABLE IV
SUM OF INITIALIZED OBJECTIVE VALUE UNDER THE LT-MODEL

	NetScience		Wiki		HetHEPT	
λ	A	B	A	B	A	B
0.8	24	294	-64	**482**	1406	1410
1.0	-51	**229**	-242	**390**	-869	**1186**
1.2	-126	**33**	-420	**177**	-3080	**887**
1.4	-202	**159**	-599	**228**	-5332	**392**
1.6	-278	-6	-776	**166**	-7603	-948
1.8	-353	**11**	-954	**66**	-9807	-106
2.0	-429	**93**	-1132	-778	-12063	**229**

smaller as cost efficient grows, resulting in a larger θ_2 and θ_3. Hence, the number of random RR-sets needed to be generated and searched will increase certainly. Third, by comparing between DGS and DGITS, their running times are roughly equal, as shown in Figs. 1(f) and (h) and 2(f) and (h). It infers that initializing by iterative pruning will not increase the time complexity actually, which is very meaningful.

Tables III and IV show the effect of lattice-based iterative prunning on the sum of initialized objective values under the IC-model and LT-model, where we denote $A = f(\mathbf{0}) + f(\boldsymbol{b})$ and $B = f(\boldsymbol{g}^\circ)+f(\boldsymbol{h}^\circ)$ for convenience. When cost coefficient $\lambda \geq 1$, $A < 0$ in all cases, and thus, there is no approximation guarantee if we run double greedy algorithm feed with $[\mathbf{0}, \boldsymbol{b}]$ directly. However, with the help of iterative prunning, $B \geq 0$ holds for most of cases. Like this, our DGIT(S) algorithm is able to offer a $(1/2 - \varepsilon)$-approximate solution according to Theorems 5 and 6.

VIII. Conclusion

In this article, we propose the continuous PM problem first, and based on it, we study unconstrained dr-submodular problem further. For UDSM problem, lattice-based double greedy is an effective algorithm, but there is not approximation guarantee unless all objective values are nonnegative. To solve it, we propose lattice-based iterative pruning and derive it step-by-step. With the help of this technique, the possibility of satisfying nonnegative is enhanced greatly. Our approach can be used as a flexible framework to address the UDSM problem. Then, back to the CPM-MS problem, we design a speedup strategy by using sampling techniques, which reduces its running time significantly without losing approximation guarantee. Eventually, we evaluate our proposed algorithms on four real networks and the results validate their effectiveness and time efficiency thoroughly.

REFERENCES

[1] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2001, pp. 57–66.

[2] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2002, pp. 61–70.

[3] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2003, pp. 137–146.

[4] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.

[5] J. Guo and W. Wu, "A novel scene of viral marketing for complementary products," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 4, pp. 797–808, Aug. 2019.

[6] J. Guo and W. Wu, "A k-Hop collaborate game model: Adaptive strategy to maximize total revenue," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 4, pp. 1058–1068, Aug. 2020.

[7] W. Lu and L. V. S. Lakshmanan, "Profit maximization over social networks," in *Proc. IEEE 12th Int. Conf. Data Mining*, Dec. 2012, pp. 479–488.

[8] H. Zhang, H. Zhang, A. Kuhnle, and M. T. Thai, "Profit maximization for multiple products in online social networks," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[9] J. Tang, X. Tang, and J. Yuan, "Profit maximization for viral marketing in online social networks," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–10.

[10] G. Tong, W. Wu, and D.-Z. Du, "Coupon advertising in online social systems: Algorithms and sampling techniques," 2018, *arXiv:1802.06946*. [Online]. Available: http://arxiv.org/abs/1802.06946

[11] J. Guo, T. Chen, and W. Wu, "Budgeted coupon advertisement problem: Algorithm and robust analysis," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1966–1976, Jul. 2020.

[12] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," *Theory Comput.*, vol. 11, no. 4, pp. 105–147, 2015.

[13] N. Buchbinder, M. Feldman, J. Seffi, and R. Schwartz, "A tight linear time (1/2)-approximation for unconstrained submodular maximization," *SIAM J. Comput.*, vol. 44, no. 5, pp. 1384–1402, Jan. 2015.

[14] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*. Philadelphia, PA, USA: SIAM, 2014, pp. 946–957.

[15] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 75–86.

[16] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1539–1554.

[17] J. Tang, X. Tang, X. Xiao, and J. Yuan, "Online processing algorithms for influence maximization," in *Proc. Int. Conf. Manage. Data*, May 2018, pp. 991–1005.

[18] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2010, pp. 1029–1038.

[19] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 88–97.

[20] 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. Int. Conf. Manage. Data*, 2016, pp. 695–710.

[21] U. Feige, V. S. Mirrokni, and J. Vondrák, "Maximizing non-monotone submodular functions," *SIAM J. Comput.*, vol. 40, no. 4, pp. 1133–1153, Jan. 2011.

[22] T. Soma and Y. Yoshida, "A generalization of submodular cover via the diminishing return property on the integer lattice," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 847–855.

[23] T. Soma and Y. Yoshida, "Maximizing monotone submodular functions over the integer lattice," *Math. Program.*, vol. 172, nos. 1–2, pp. 539–563, Nov. 2018.

[24] A. Bian, K. Y. Levy, A. Krause, and J. M. Buhmann, "Continuous dr-submodular maximization: Structure and algorithms," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 486–496.

[25] W. Chen, R. Wu, and Z. Yu, "Scalable lattice influence maximization," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 4, pp. 956–970, Aug. 2020.

[26] J. Guo, T. Chen, and W. Wu, "Continuous activity maximization in online social networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2775–2786, Oct. 2020.

[27] T. Soma, "Non-monotone dr-submodular function maximization," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 898–904.

[28] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[29] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI*, 2015, pp. 4292–4293. [Online]. Available: http://networkrepository.com

[30] J. Leskovec and A. Krevl. (Jun. 2014). *SNAP Datasets: Stanford Large Network Dataset Collection*. [Online]. Available: http://snap.stanford.edu/data

Jianxiong Guo received the B.S. degree in energy engineering and automation from the South China University of Technology, Guangzhou, China, in 2015, and the M.S. degree in chemical engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA.

His research interests include social networks, Internet of Things (IoT) application, blockchain, and combinatorial optimization.

Weili Wu (Senior Member, IEEE) received the M.S. and Ph.D. degrees from the Department of Computer Science, University of Minnesota, Minneapolis, MN, USA, in 1998 and 2002, respectively.

She is currently a Full Professor with the Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA. Her research mainly deals in the general research area of data communication and data management. Her research focuses on the design and analysis of algorithms for optimization problems that occur in wireless networking environments and various database systems.