Efficient and Effective Algorithms for A Family of Influence Maximization Problems with A Matroid Constraint

Yiqian Huang
National University of Singapore
yiqian@comp.nus.edu.sg

Shiqi Zhang
National University of Singapore
PyroWis AI
shiqi@pyrowis.ai

Laks V.S. Lakshmanan
University of British Columbia
laks@cs.ubc.ca

Wenqing Lin
Tencent
edwlin@tencent.com

Xiaokui Xiao
National University of Singapore
xkxiao@nus.edu.sg

Bo Tang
Southern University of Science and Technology
tangb3@sustech.edu.cn

ABSTRACT

Influence maximization (IM) is a classic problem that aims to identify a small group of critical individuals, known as seeds, who can influence the largest number of users in a social network through word-of-mouth. This problem finds important applications including viral marketing, infection detection, and misinformation containment. The conventional IM problem is typically studied with the oversimplified goal of selecting a *single* seed set. Many real-world scenarios call for *multiple* sets of seeds, particularly on social media platforms where various viral marketing campaigns need different sets of seeds to propagate effectively. To this end, previous works have formulated various IM variants, central to which is the requirement of multiple seed sets, naturally modeled as a matroid constraint. However, the current best-known solutions for these variants either offer a weak $(1/2 - \epsilon)$-approximation, or offer a $(1 - 1/e - \epsilon)$-approximation algorithm that is very expensive. We propose an efficient seed selection method called AMP, an algorithm with a $(1 - 1/e - \epsilon)$-approximation guarantee for this family of IM variants. To further improve efficiency, we also devise a fast implementation, called RAMP. We extensively evaluate the performance of our proposal against 6 competitors across 4 IM variants and on 7 real-world networks, demonstrating that our proposal outperforms all competitors in terms of result quality, running time, and memory usage. We have also deployed RAMP in a real industry strength application involving online gaming, where we show that our deployed solution significantly improves upon the baselines.

PVLDB Reference Format:
Yiqian Huang, Shiqi Zhang, Laks V.S. Lakshmanan, Wenqing Lin, Xiaokui Xiao, and Bo Tang. Efficient and Effective Algorithms for A Family of Influence Maximization Problems with A Matroid Constraint. PVLDB, 18(2): 117 - 129, 2024.
doi:10.14778/3705829.3705833

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/waetr/IM_GM.

Proceedings of the VLDB Endowment, Vol. 18, No. 2 ISSN 2150-8097.
doi:10.14778/3705829.3705833

1 INTRODUCTION

Given a social network, the problem of influence maximization (IM) seeks to find a small group of users who can (directly or indirectly) influence as many users in the network as possible. It has been applied in various domains, including viral marketing [16, 35], efficient infection detection [37], and misinformation containment [9, 45, 53], among others. Viral marketing, for instance, typically involves a company setting up a campaign to promote a product by incentivizing a set of influential users (called "seeds"), aiming to maximize the number of influenced users (referred to as "spread") by triggering a cascade of adoptions through word-of-mouth. Existing work on IM has mostly focused on this setting of *single seed set*.

Real-world scenarios of viral marketing are often more complex, requiring *multiple* sets of seeds, to optimize the overall spread produced by these sets. This complexity is clearly evident in the rise of influencer marketing platforms, including the official creator marketplaces of social media platforms such as Instagram [33], TikTok [52], and Douyin [22], as well as over 80 third-party platforms [32]. These platforms act as intermediaries, aligning advertisers with key influencers for social advertising campaigns and aiding influencers in monetizing their online presence. E.g., the Douyin Xingtu platform reported a substantial user base of over 1.9 million advertisers and 3.2 million creators as of June 2022 [21]. A critical task for them is coordinating diverse advertiser campaigns and determining optimal seed sets such that the content created by the seeds can reach a broader audience through word-of-mouth.

The above viral marketing scenarios that seek multiple seed sets can be modeled as an IM problem subject to a *general matroid* constraint (IM-GM). A general matroid (formal definition in § 2.2) represents constraints that an IM solution must adhere to. It is defined as a pair $(U, \mathcal{I})$, where U is a ground set of elements from which seeds are selected, and $\mathcal{I}$ is a collection of subsets of U (called independent sets), representing feasible solutions to the problem. For example, we show an instance of IM-GM in the context of *revenue maximization*.

Example 1.1 (Revenue Maximization). Consider the scenario of incentivized social advertising [2, 29] where we need to coordinate T viral marketing campaigns, such that seed sets S_i, S_j selected for different campaigns $i \neq j$ do not overlap, and the sum of spread

produced by all campaigns is maximized. This scenario cannot be trivially treated as independently running IM T times, as a seed can contribute to more than one campaign, motivating the introduction of a matroid. In this case, the ground set U comprises all possible combinations of users and campaigns, while $\mathcal{I}$ corresponds to all feasible solutions $(S_1, ..., S_T)$ which satisfy $S_i \cap S_j = \emptyset, i \neq j$. □

The IM-GM formulation also captures numerous other real-world applications [15, 41, 46, 47, 57]. We illustrate two such applications: *multi-round* IM and *adversarial attacks on* IM.

Example 1.2 (Multi-Round IM*).* In the recruitment-based campaign of the Xingtu mobile app [10], service providers select seeds from a candidate pool in multiple rounds, such that each round's seed set has a size $\leq k$ and the spread across all rounds is maximized; this cardinality constraint can be captured using a matroid. □

Example 1.3 (Adversarial Attacks on IM*).* In combating the spread of illicit activities such as pyramid scheme scams [23], money laundering [34], and phishing scams [56], authorities identify not only *malicious users* but also *suspicious relationships* for blocking in order to control further spread of such activities. Owing to the sensitivity of blocking, in practice there are limits on the number of users and relationships that can be blocked. This problem can be modeled as an instance of IM-GM, with a matroid constraining the maximum number of identified users and relationships for blocking. □

There exist a number of solutions to IM-GM [2, 12, 15, 17, 19, 29, 41, 46, 47, 49, 57], most of which utilize the greedy approach introduced by Fisher et al. [24]. The greedy approach iteratively selects the element with the largest improvement of the objective function under the matroid constraint, resulting in a $(1/2 - \epsilon)$-approximation for any given $\epsilon > 0$. Subsequent work [11] improves upon this result and presents a $(1 - 1/e - \epsilon)$-approximation algorithm for submodular maximization under a matroid constraint. Unlike the greedy approach, it first iteratively searches for a fractional solution which contains a *fraction* for each element in U, with each fraction representing the probability that the element is selected. It then converts the fractional solution into a discrete one, by randomly rounding each fraction to either 0 or 1. However, this algorithm incurs significant computational cost and fails to handle even datasets with a few hundred users (see § 4.1), because it requires sampling a large number of random subsets of U in the evaluation of candidate fractional solutions.

To address this issue, we propose AMP, an algorithm that ensures $(1-1/e-\epsilon)$-approximation for IM-GM while offering superior efficiency. AMP shares a similar framework as mentioned above but performs the search and rounding steps *deterministically*. The linchpin lies in the fact that AMP avoids expensive random sampling in the assessment of fractional solutions by utilizing a new estimation method, which reduces computation costs by exploiting the characteristics of fractional solutions in IM-GM. We also present RAMP, an algorithm that further improves AMP's efficiency by adapting the doubling strategy from Tang et al. [48], but with careful redesign to ensure the correctness for IM-GM. We experimentally evaluate the performance of AMP and RAMP against 6 competitors across 4 IM-GM instances, on 7 real-world networks ranging in size from thousands of nodes and edges to millions of nodes and billions of edges. Notably, AMP outperforms all competitors by up to 871% in terms of result quality. Besides, RAMP surpasses the state-of-the-art by up to 1,000× in running time and 10× in memory usage, respectively. In addition, we have deployed our solution RAMP on a Tencent online gaming platform, and observed that it attracted 160,000 more user engagements compared to the control group using the current state-of-the-art solution.

To summarize, we make the following contributions in this work:

- We propose an efficient $(1 - 1/e - \epsilon)$-approximation solution AMP for IM-GM (§ 4).
- We provide a scalable implementation RAMP for further improving the efficiency of AMP (§ 5).
- Our extensive experiments show that AMP and RAMP can efficiently scale to datasets with billions of edges (§ 6).
- We deploy our solution to a Tencent online game and significantly improve on the state of the art in user engagement (§ 7).

2 PRELIMINARIES

We provide basic terminology and notations in § 2.1, followed by formulating influence maximization subject to a general matroid constraint (IM-GM) in § 2.2. We review typical instances of IM-GM and their solutions in § 2.3 and § 2.4, respectively.

2.1 Terminology and Notations

Social networks. A social network is a directed graph $G = (V, E)$, where V is a set of nodes representing users and E is a set of edges representing relationships. Each directed edge $e_{i,j} \in E$ is associated with $p_{i,j} \in [0, 1]$, and indicates that v_j can be influenced by v_i with probability $p_{i,j}$. We say v_i (resp. v_j) the in-neighbor (resp. out-neighbor) of v_j (resp. v_i), and denote the set of in-neighbors (resp. out-neighbors) of v_i as N_i^{in} (resp. N_i^{out}).

Monotone and submodular functions. Given a set U, a non-negative set function f is *monotone* if $\forall X \subseteq Y \subseteq U, f(X) \leq f(Y)$. f is *submodular* if $\forall X \subseteq Y \subseteq U$ and $u_i \in U \backslash Y, f(u_i|X) \geq f(u_i|Y)$, where $f(u_i|X) = f(X \cup \{u_i\}) - f(X)$ denotes the *marginal gain* of f after adding u_i to X. As an example, given a collection $\mathcal{R}$ of subsets of U, the *coverage* function of a set $S \subseteq U$, defined as

$$\Lambda_{\mathcal{R}}(S) = \sum_{R \in \mathcal{R}} \mathbb{I}(S \cap R \neq \emptyset), \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function, is monotone and submodular.

Notations. Throughout this paper, an upper-case letter A (resp. a calligraphic upper-case letter $\mathcal{A}$) denotes a set (resp. a collection of sets). For a ground set U, a boldface lower-case letter $\mathbf{x} \in [0, 1]^U$ represents a vector corresponding to n elements of U, where $\mathbf{x}[i]$ denotes the value associated with the element $u_i \in U$. As a special case, $\mathbf{1}_i$ (resp. $\mathbf{1}_S$) signifies the indicator vector with value 1 for u_i (resp. for each element $u_i \in S$) and value 0 otherwise. Table 1 lists the frequently used notations.

2.2 Problem Formulation

Given a graph $G = (V, E)$ and a set $S \subseteq V$ of chosen nodes (called seeds), a *diffusion model* describes the process by which information spreads from S to other nodes via social connections in G. In this work, we consider the well-known Independent Cascade (IC) [25] and Linear Threshold (LT) [26] models, where a diffusion instance captures the stochastic diffusion process from S and unfolds in discrete steps. At the beginning of a diffusion instance L of IC

Table 1: Frequently used notations.

Notation	Description
$\mathsf{G} = (V, E)$	A graph G with node set V and edge set E.
$\mathsf{M} = (U, \mathcal{I})$	A matroid M with ground set U and collection $\mathcal{I}$ of independent sets.
n	The number of elements in U.
$r, \mathcal{B}$	The rank of M and the collection of all bases in M.
h, k_i	The number of partitions, and the capacity of the i-th partition in M.
$R, \mathcal{R}$	A random RR set and a collection of random RR sets.
$\Lambda_{\mathcal{R}}, \sigma$	The coverage function in Eq.(1) and the objective in Definition 2.2.
$\mathbf{x}$	The fractional solution of element selection in § 4.

or LT, all seeds in S are set to be active at step 0, leaving other nodes inactive. In IC, at step $t > 0$, each node v_i activated at step $t-1$ has one chance to independently activate its inactive out-neighbor v_j with probability $p_{i,j}$. LT requires that for each node v_j, (i) $\sum_{v_i \in N_j^{in}} p_{i,j} \leq 1$, and (ii) a threshold $\lambda_j \in [0, 1]$ is sampled uniformly at step 0. At step $t > 0$, an inactive node v_j is activated iff the sum of $p_{i,j}$ w.r.t. v_j's activated in-neighbors v_i exceeds λ_j. During a diffusion instance L of IC or LT, once a node is activated, it remains active in all subsequent steps. L terminates if no more nodes can be activated, and the set of eventually activated nodes is denoted as $\Gamma_{\mathsf{L}}(\mathsf{G}, S)$. Accordingly, the *spread* of S in G is defined as

$$\rho_{\mathsf{G}}(S) = \mathbb{E}[|\Gamma_{\mathsf{L}}(\mathsf{G}, S)|]. \tag{2}$$

Based on these concepts, Kempe et al. [36] define the influence maximization (IM) problem as follows.

Definition 2.1 (IM). Given a graph G, a diffusion model and a cardinality k, the IM problem is to select a seed set $S \subseteq V$ with $|S| \leq k$ such that the spread $\rho_{\mathsf{G}}(S)$ defined in Eq.(2) is maximized.

We next review concepts of matroids and then propose a generalized IM problem, viz., IM subject to a general matroid (IM-GM).

Matroids. A *general matroid* is a pair $\mathsf{M} = (U, \mathcal{I})$, where U is a finite set (called the ground set) with $|U| = n$ and $\mathcal{I}$ is a collection of subsets of U (called independent sets) satisfying the following axioms: (i) $\mathcal{I} \neq \emptyset$; (ii) if $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$; (iii) if $I, J \in \mathcal{I}$ and $|I| < |J|$, then there exists $u_i \in J \backslash I$ such that $I \cup \{u_i\} \in \mathcal{I}$. Of special interest is the *partition matroid*, whose ground set U is divided into $h \geq 1$ disjoint partitions $U_1, \ldots, U_h$, each U_i associated with a positive integer k_i. The set of independent sets $\mathcal{I}$ in the partition matroid is defined as $\mathcal{I} = \{S \subseteq U : |S \cap U_i| \leq k_i, \forall i \in [h]\}$. In the special case where $h = 1$, the partition matroid reduces to a *uniform matroid*, characterized by $\mathcal{I} = \{S \subseteq U : |S| \leq k\}$.

Given a matroid $\mathsf{M} = (U, \mathcal{I})$, any independent set $B \in \mathcal{I}$ with the largest cardinality, i.e., $|B| = \max\{|I| : I \in \mathcal{I}\}$, is a *base* and its cardinality r is the *rank* of M. E.g., $r = \sum_{i=1}^{h} k_i$ in the partition matroid and $r = k$ in the uniform matroid. A matroid can have multiple bases. We denote the collection of all bases of M as $\mathcal{B}$.

Definition 2.2 (IM-GM). Given a graph G, a diffusion model, and a matroid $\mathsf{M} = (U, \mathcal{I})$ where U is built on G, IM-GM aims to find a set $S \in \mathcal{I}$ that maximizes a given objective function $\sigma : 2^U \to \mathbb{R}_+$, which is associated with the spread function in Eq.(2).

Observe in Definitions 2.1-2.2 that IM is an instance of IM-GM, where M is a uniform matroid and $\sigma(S) = \rho_{\mathsf{G}}(S)$. Besides IM, a plethora of IM variants [2, 12, 15, 17, 29, 41, 46, 47, 57] can also be modeled using IM-GM. The next section will discuss three instances, obtained from different instantiations of M and σ.

2.3 Representative IM-GM Instances

Revenue maximization (RM) [2, 17, 29]. Given a graph $G = (V, E)$, a diffusion model, and T campaigns associated with T unit revenues $\alpha_1, \ldots, \alpha_T$, the RM problem aims to select seed sets $S_1, S_2, \ldots, S_T \subseteq V$ for the T campaigns such that the total revenue $\sigma(S_1, S_2, \ldots, S_T) = \sum_{t=1}^{T} \alpha_t \cdot \rho_G(S_t)$ is maximized. This problem is subject to a user constraint that each node $v_i \in V$ can be included in at most k_i seed sets, where $k_i \leq T$ is a number associated with node v_i. This constraint corresponds to a partition matroid $\mathsf{M} = (U, \mathcal{I})$, where the ground set $U = V \times [T]$ contains all pairs of nodes and campaigns. U is partitioned into $|V|$ disjoint subsets $U_1, U_2, \ldots, U_{|V|}$, where $U_i = \{(v_i, t) : t \in [T]\}$. Accordingly, $\mathcal{I} = \{S \subseteq U : \forall v_i \in V, |S \cap U_i| \leq k_i\}$. As a special case, we have $k_i = 1$ for all $v_i \in V$ in the scenario of incentivized social advertising [2, 29].

Non-adaptive multi-round IM (MRIM) [46]. Given a graph $G = (V, E)$, a diffusion model, a cardinality k and the number of campaign rounds T, the non-adaptive MRIM problem aims to select seed set $S_t \subseteq V$ in each round t, with $|S_t| \leq k$, such that the expected number of activated nodes over all rounds $\sigma(S_1, S_2, \ldots, S_T) = \mathbb{E}\left[\left|\bigcup_{t=1}^{T} \Gamma_{\mathsf{L}_t}(G, S_t)\right|\right]$ is maximized, where each diffusion instance L_t is obtained independently. This constraint can also be viewed as a partition matroid with the ground set $U = V \times [T]$. However, in contrast with RM, this ground set is partitioned w.r.t. rounds, i.e., $U_t = \{(v_i, t) : v_i \in V\}$, and $\mathcal{I} = \{S \subseteq U : \forall t \in [T], |S_t| \leq k\}$.

Adversarial attacks on IM (AdvIM) [47]. Given a graph $G = (V, E)$, a diffusion model, a set of contagious seeds $A \subset V$ and two cardinalities k_v, k_e, the AdvIM problem aims to select a blocking set S consisting of a blocking node set $S_v \subseteq V \backslash A$ with $|S_v| \leq k_v$ and a blocking edge set $S_e \subseteq E$ with $|S_e| \leq k_e$, i.e., $S = S_v \cup S_e$, such that the influence reduction $\sigma(S_v, S_e) = \rho_G(A) - \rho_{G \backslash S}(A)$ is maximized, where $G \backslash S = (V \backslash S_v, E \backslash S_e)$. The constraint in this problem corresponds to a partition matroid with ground set $U = (V \backslash A) \cup E$ and independent sets $\mathcal{I} = \{S \subseteq U : |S_v| \leq k_v, |S_e| \leq k_e\}$.

2.4 Approximation Solutions for IM-GMs

Due to the NP-hardness of each IM-GM problem and the inefficiency of Monte-Carlo simulations, most previous works borrow the idea of *reverse reachable (RR) set* sampling [7], originally designed for vanilla IM. Specifically, an RR set consists of nodes identified by simulating a reverse diffusion process of a given diffusion model from a node v_i selected uniformly at random. For example, the reverse diffusion process of the IC model is a stochastic breadth-first search starting from v_i. For each visited v_a, it explores each in-neighbor $v_b \in N_a^{in}$ with probability $p_{b,a}$. Borgs et al. [7] show that for any seed set $S \subseteq V$, the probability that an RR set overlaps S equals $\frac{\rho_{\mathsf{G}}(S)}{|V|}$. That is, if we have a collection $\mathcal{R}$ of RR sets, we can use $\frac{|V|}{|\mathcal{R}|} \cdot \Lambda_{\mathcal{R}}(S)$ as an unbiased estimator of $\rho_{\mathsf{G}}(S)$. Accordingly, Borgs et al. [7] propose an approximation solution for IM, which runs in two main steps: (i) *RR set generation*: sample a set $\mathcal{R}$ of RR sets, with the size $|\mathcal{R}|$ sufficient to achieve the desired approximation; (ii) *element selection*: use a greedy algorithm for solving maximum coverage which returns a set S that maximizes the coverage $\Lambda_{\mathcal{R}}(S)$. Borgs et al.'s framework is also leveraged in various IM-GM instances [2, 15, 29, 41, 46, 47, 57]. A central role in determining the approximation is played by various greedy element selection algorithms used in these works, briefly reviewed below.

Greedy [24]. This algorithm greedily selects elements with the maximum marginal gain subject to the matroid constraint. Specifically, it starts with a set $S = \emptyset$, and then iteratively adds r elements to S. In each iteration, it adds u_i to S if $\Lambda_{\mathcal{R}}(u_i|S)$ is the largest among the elements of $U \backslash S$ and the inclusion maintains the matroid constraint. This algorithm achieves a 1/2-approximation subject to a general matroid constraint [24] and is the most commonly adopted algorithm in prior works for IM-GM problems [2, 15, 29, 46, 47, 57].
Local-Greedy [46]. Designed for the partition matroid constraint, Local-Greedy selects elements partition by partition. Specifically, it greedily selects elements in the first partition with capacity k_1. Once k_1 elements are selected, it proceeds to the second partition, and so on. This improves the efficiency by pruning the search space from U to each partition. Sun et al. [46] prove that Local-Greedy achieves a $1 - e^{-(1-\frac{1}{e})} \approx 0.46$ approximation.
Threshold-Greedy [8]. Unlike Greedy, Threshold-Greedy adds an element u_i to S if $\Lambda_{\mathcal{R}}(u_i|S)$ exceeds a threshold, which starts at a predefined value and decreases by $(1 - \xi)$ in each iteration. Compared to Greedy, this algorithm boosts efficiency by reducing the number of iterations from r to $O\left(\xi^{-1} \cdot \ln(r/\xi)\right)$, while incurring a loss term ξ, resulting in an approximation ratio of $1/2 - \xi$ [8].

Due to the variety of diffusion models and objectives, IM-GM solutions require two additional steps before RR set generation: (i) select a strategy of RR set construction for an IM-GM instance, where an RR set R consists of *elements* in U instead of nodes in V; (ii) rewire the coverage $\Lambda_{\mathcal{R}}$ and the objective σ such that the scaled $\Lambda_{\mathcal{R}}(S)$ is an unbiased estimator of $\sigma(S)$, i.e., $\sigma(S) = \frac{\kappa}{|\mathcal{R}|} \cdot \mathbb{E}[\Lambda_{\mathcal{R}}(S)]$, where the value of constant κ depends on the given IM-GM instance. E.g., MRIM [46] first selects a random node $v_i \in V$ uniformly, then independently simulates T reverse diffusion processes starting from v_i. Denoting the set of examined nodes in each process i as R'_i, the RR set in MRIM is $R'_1 \times \{1\} \cup \cdots \cup R'_T \times \{T\}$, and we have $\kappa = |V|$. For the remaining instances, we refer interested readers to [31].

To improve the efficiency of Borgs et al.'s framework for the vanilla IM, subsequent solutions, e.g., TIM [51], IMM [50], DSSA [43], and OPIM-C [48], focus on reducing the number of sampled RR sets while ensuring the same approximation ratio. As of now, OPIM-C is the state of the art and it has been applied to subsequent solutions [6, 27–29, 57]. Most scalable solutions of other IM-GM instances also extend the aforementioned ones used in IM. E.g., in RM, [29] utilizes a greedy variant by mixing Greedy and Threshold-Greedy, and then follows OPIM-C to achieve an approximation not exceeding $1/2 - \epsilon$. Additionally, [46] and [47] offer a $(1/2 - \epsilon)$-approximation for MRIM and AdvIM, respectively, by employing Greedy and IMM.

3 RELATED WORK

Since its introduction in [36], the IM problem has been extensively researched – see [40] for a comprehensive survey. Here, we mainly focus on IM under a matroid constraint. Various kinds of matroids have been considered in previous IM studies. Besides the three IM-GM instances introduced in § 2.3, partition matroids are also employed in several IM-GM instances, including capacity constrained IM [57], IM with fairness constraint [19], the lattice IM with partitioned budgets [15], and so forth [12, 49]. Furthermore, other general matroids are used in previous IM-GM studies to accommodate more sophisticated scenarios. For instance, the popularity ratio maximization problem [41] follows the multiple round setting in MRIM but aims to significantly surpass competitors' outreach in popularity by initiating influence cascades. This paper studies the intersection of a uniform matroid that restricts the overall cardinality with a partition matroid, resulting in a *general matroid*[1] constraint. As another example, the general version of IM for multiple products [17] requires a *laminar matroid* constraint that models capacity limits within hierarchical community structures, such as the maximum number of selected customers in different states and the cities they contain. While developing different approaches, the core ideas of these solutions lie in either Greedy, Local-Greedy, or Threshold-Greedy, all limited to at most a $(1/2-\epsilon)$-approximation.

In theoretical computer science, another line of previous works aims to maximize a general submodular function under a matroid constraint. [11] proposed continuous greedy, the first polynomial-time algorithm with a $(1 - 1/e - \epsilon)$ approximation guarantee. [4] later refined it, achieving state-of-the-art results for general matroids. However, adopting these algorithms in IM-GM is highly inefficient, as detailed in § 4.1. Subsequent research has focused on specific categories of matroids [8, 20, 30]. [8] introduce some specific optimizations for partition matroids, resulting in an $O(n^{3/2})$-time algorithm. Prior works [20, 30] further develop algorithms with $O(n \cdot \text{poly}(1/\epsilon, \log n))$ time complexity for partition, graphic, transversal, and laminar matroids. The above approaches cannot be directly adopted for IM-GM instances, as they do not apply for general matroids outside the above matroid classes.

In the next two sections, we present our approach for solving IM-GM. Given that our overall approach is based on generating a collection of RR sets and then selecting a set that maximizes coverage, we divide the presentation along the same theme.

4 ELEMENT SELECTION

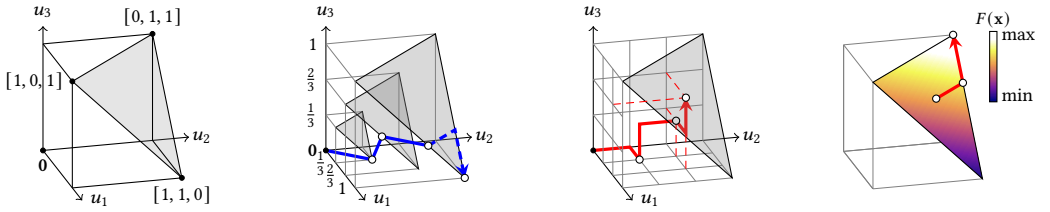
Given a set of RR sets $\mathcal{R}$, this section proposes an element-wise Ascent algorithm over Matroid Polytopes for element selection (AMP), providing a $(1-1/e-\epsilon)$-approximation for maximizing the coverage $\Lambda_{\mathcal{R}}$ subject to the matroid constraint. We first discuss the challenges in adopting prior $(1 - 1/e - \epsilon)$-approximate solutions and the main idea of AMP (§ 4.1), followed by detailed descriptions of two core subroutines AMPSearch/AMPSearch-PM (§ 4.2) and AMPRound (§ 4.3). At last, we provide an analysis of correctness and time complexity (§ 4.4).

4.1 Overview

Main idea of existing solutions. Several continuous greedy algorithms [4, 11] have been proposed to find $(1-1/e-\epsilon)$-approximate solutions for maximizing any monotone submodular function f (e.g., $\Lambda_{\mathcal{R}}$) subject to a general matroid constraint. First, these solutions expand the original function f to its corresponding multilinear extension, defined as follows.

Definition 4.1 (Multilinear Extension). For a set function $f : 2^U \rightarrow \mathbb{R}_+$, its multilinear extension $F : [0,1]^U \rightarrow \mathbb{R}_+$ is defined as $F(\mathbf{x}) = \mathbb{E}\left[f(\Omega(\mathbf{x}))\right]$, where $\Omega(\mathbf{x})$ is a random set that independently includes each element $u_i \in U$ from the fractional result $\mathbf{x}$ with probability $\mathbf{x}[i]$.

[1] It is not a partition matroid, unlike what is claimed in [41]. See [31] for details.



(a) Polytopes **(b) Previous solution** **(c) Searching** **(d) Rounding**

Figure 1: Illustration of continuous greedy solutions and AMP subject to the matroid with $U = \{u_1, u_2, u_3\}$ and $\mathcal{I} = \{I \subseteq U : |I| \leq 2\}$.

Second, these solutions aim to solve the multilinear optimization problem $\max_{\mathbf{x} \in P^{\mathcal{I}}} \{F(\mathbf{x})\}$, where $P^{\mathcal{I}}$ is the *matroid polytope* defined as the convex hull (i.e., all convex combinations of elements) of $\{\mathbf{1}_I : I \in \mathcal{I}\}$. Notice that, due to the monotonicity of f, the fractional solution $\mathbf{x}$ that maximizes $F(\mathbf{x})$ is in the *base polytope* $P^{\mathcal{B}} = \left\{\mathbf{x} \in P^{\mathcal{I}} : \sum_{i \in [n]} \mathbf{x}[i] = r\right\}$ [18]. To find such a solution $\mathbf{x} \in P^{\mathcal{B}}$, given a parameter ϵ, these approaches start from $\mathbf{x} = \mathbf{0}$ and update $\mathbf{x}$ by an iterative *hill-climbing search* process: at each step t, this process examines *basic solution* $B_t \in \mathcal{B}$ and the corresponding fractional solution $\epsilon \cdot \mathbf{1}_{B_t}$ such that $F(\mathbf{x})$ can be improved after adding this fractional solution to $\mathbf{x}$. After $1/\epsilon$ steps, this process returns a fractional solution $\mathbf{x} = \sum_{t=1}^{1/\epsilon} \epsilon \cdot \mathbf{1}_{B_t}$, which is in $P^{\mathcal{B}}$ by definition. Finally, upon obtaining $\mathbf{x} \in P^{\mathcal{B}}$, these solutions employ a *randomized rounding* process that eventually converts $\mathbf{x}$ to a discrete solution $S \in \mathcal{B}$, thus solving the original problem $\max_{S \in \mathcal{I}} \{f(S)\}$.

Example 4.2 (Illustrative Example). Consider a matroid with $U = \{u_1, u_2, u_3\}$ and $\mathcal{I} = \{I \subseteq U : |I| \leq 2\}$. As depicted in Figure 1(a), the origin represents $\mathbf{x} = \mathbf{0}$, and the vertices $[1, 1, 0]$, $[1, 0, 1]$, $[0, 1, 1]$ indicate all bases. Additionally, the polyhedron and the shaded face correspond to the matroid polytope and base polytope, respectively. Assuming $\epsilon = 1/3$, the three shaded faces shown in Figure 1(b) represent the feasible spaces for each search step. The solid and dashed blue lines illustrate the search and rounding process of previous solutions, respectively. Starting from $\mathbf{x} = \mathbf{0}$, the search process of Calinescu et al. [11] selects the direction of B_1 (i.e., $\mathbf{1}_{B_1} = [1, 1, 0]$) and moves $\mathbf{x}$ to $\mathbf{x} + \epsilon \cdot \mathbf{1}_{B_1} = \left[\frac{1}{3}, \frac{1}{3}, 0\right]$ at the first step, reaching the smallest shaded face. Subsequently, it selects $\mathbf{1}_{B_2} = [1, 0, 1]$ and moves $\mathbf{x}$ to $\mathbf{x} + \epsilon \cdot \mathbf{1}_{B_2} = \left[\frac{2}{3}, \frac{1}{3}, \frac{1}{3}\right]$, located in the medium shaded face. At last, it updates $\mathbf{x}$ along the direction of B_3, bringing $\mathbf{x}$ to $\left[1, \frac{2}{3}, \frac{1}{3}\right]$ in the base polytope. Upon completing the search, to round the fractional solution $\mathbf{x}$, the algorithm [13] moves $\mathbf{x}$ on the base polytope in random directions until it reaches a vertex, e.g., $[1, 1, 0]$, indicating the final solution $S = \{u_1, u_2\}$. □

Limitations of existing solutions. Existing algorithms suffer from severe efficiency issues in the hill-climbing search stage, since computing the multilinear extension F of a general monotone submodular function f is expensive. Specifically, the multilinear extension of a set function f is

$$F(\mathbf{x}) = \sum_{S \subseteq U} \left(\prod_{u_i \in S} \mathbf{x}[i] \prod_{u_i \in U \setminus S} (1 - \mathbf{x}[i]) \right) \cdot f(S). \tag{3}$$

Computing $F(\mathbf{x})$ exactly involves enumerating all 2^n subsets of U. Previous works leverage random subset sampling to estimate $F(\mathbf{x})$, however they still have a prohibitively expensive time complexity. E.g., the approach of [11] which selects the t-th basic solution B_t that yields an improvement in F along the direction of the base B_t, requires $O\left(n^7 \log(n)\right)$ samples. The improved algorithm [4] selects each basic solution B_t using Threshold-Greedy, which includes an element u_i into B_t if the partial derivative $\partial F(\mathbf{x})/\partial \mathbf{x}[i]$ exceeds a pre-defined threshold. This compromises the result quality (see § 2.4) yet generates $O(nr\epsilon^{-4} \cdot \log^2(n/\epsilon))$ samples in total.

Additionally, in the rounding stage, existing randomized techniques [1, 13] focus on the efficiency of maintaining $\mathbf{x} \in P^{\mathcal{B}}$ rather than the quality of the rounded result. To elaborate, the state-of-the-art swap rounding [13] iteratively eliminates each distinct item u_i from the difference between the current basic solution B_t and the previous merged one B_{t-1} until all basic solutions are identical. This is done by accepting u_i w.p. $1/t$ if $u_i \in B_t \setminus B_{t-1}$ and with the remaining probability if $u_i \in B_{t-1} \setminus B_t$. As proved by [13], swap rounding returns a discrete set $S \in \mathcal{B}$ satisfying $f(S) \geq (1 - \psi)F(\mathbf{x})$ w.p. at least $1 - e^{-\frac{F(\mathbf{x})\psi^2}{8F^*}}$, where $F^* = \max_{\mathbf{x}} F(\mathbf{x})$. Due to the additional error term ψ, more iterations in the search step or more RR sets are required to ensure the desired total error ϵ, resulting in significant computational overhead. Furthermore, the failure probability is $e^{-\frac{F(\mathbf{x})\psi^2}{8F^*}} \geq e^{-\frac{\psi^2}{8}} > 0.88$, which can severely compromise the quality of the rounded result.

Our proposal. To address the above issues, we consider the multilinear extension of the coverage function $\Lambda_{\mathcal{R}}(\cdot)$ and *show that it can be evaluated efficiently and deterministically*. Specifically, by plugging $\Lambda_{\mathcal{R}}(\cdot)$ defined in Eq.(1) into Eq.(3), we have

$$F(\mathbf{x}) = \sum_{R \in \mathcal{R}} \sum_{S \subseteq U} \left(\prod_{u_i \in S} \mathbf{x}[i] \prod_{u_j \in U \setminus S} (1 - \mathbf{x}[j]) \right) \cdot \mathbb{I}(R \cap S \neq \emptyset),$$

which can be converted to $F(\mathbf{x}) = \sum_{R \in \mathcal{R}} \mathbb{E}\left[\mathbb{I}(R \cap \Omega(\mathbf{x}) \neq \emptyset)\right]$. Since $\mathbb{E}\left[\mathbb{I}(R \cap \Omega(\mathbf{x}) \neq \emptyset)\right] = 1 - \prod_{u_i \in R}(1 - \mathbf{x}[i])$ for a given $R \subseteq U$, $F(\mathbf{x})$ can be further rewritten as

$$F(\mathbf{x}) = \sum_{R \in \mathcal{R}} \left(1 - \prod_{u_i \in R} (1 - \mathbf{x}[i]) \right). \tag{4}$$

Based on this, we can efficiently calculate the exact value of $F(\mathbf{x})$ by maintaining a collection $\mathcal{Q}_{\mathcal{R}}$, consisting of the variable

$$q_R = \prod_{u_i \in R}(1 - \mathbf{x}[i]) \tag{5}$$

for each $R \in \mathcal{R}$ from the first iteration. This enables us to develop new search and rounding algorithms capable of achieving the desired $(1 - 1/e - \epsilon)$-approximation efficiently. It is noteworthy that, besides instances of IM-GM, Eq.(4) has the potential to improve IM scenarios that are solved by leveraging the multilinear extension in

Algorithm 1: AMP $(\mathsf{M}, \mathcal{R}, \epsilon)$

Input: The matroid M, an RR collection $\mathcal{R}$, $0 < \epsilon \leq 1$
Output: A discrete set $S \in \mathcal{B}$ maximizing the coverage

$\mathcal{Q}_{\mathcal{R}} \leftarrow \{q_R : R \in \mathcal{R}\}$; $\forall R \in \mathcal{R}, q_R \leftarrow 1$;
$\forall t = 1, 2, \ldots, 1/\epsilon, B_t \leftarrow \emptyset$; $\mathbf{x} \leftarrow \mathbf{0}$;
for $t = 1, 2, \ldots, 1/\epsilon$ **do**
  if M *is a partition matroid* **then**
    $B_t, \mathcal{Q}_{\mathcal{R}} \leftarrow$ AMPSearch-PM$(\mathcal{R}, \mathbf{x}, \epsilon, \mathcal{Q}_{\mathcal{R}})$;
  else
    $B_t, \mathcal{Q}_{\mathcal{R}} \leftarrow$ AMPSearch$(\mathcal{R}, \mathbf{x}, \epsilon, \mathcal{Q}_{\mathcal{R}})$
  $\mathbf{x} \leftarrow \mathbf{x} + \epsilon \cdot \mathbf{1}_{B_t}$;
$S \leftarrow$ AMPRound$(\mathcal{R}, \epsilon, \mathcal{Q}_{\mathcal{R}}, B_1, B_2, \ldots, B_{1/\epsilon})$;
return S;

Algorithm 2: AMPSearch $(\mathcal{R}, \mathbf{x}, \epsilon, \mathcal{Q}_{\mathcal{R}})$

Input: An RR collection $\mathcal{R}$, a fractional solution $\mathbf{x} \in [0,1]^U$, $0 < \epsilon \leq 1$, $\mathcal{Q}_{\mathcal{R}}$ defined in Algorithm 1
Output: A basic solution $B \in \mathcal{B}$, an updated collection $\mathcal{Q}_{\mathcal{R}}$

$B \leftarrow \emptyset$; $\mathbf{y} \leftarrow \mathbf{x}$;
for $i = 1, 2, \ldots, r$ **do**
  $u_j \leftarrow \underset{u_j:\, u_j \in U \setminus B,\, B \cup \{u_j\} \in \mathcal{I}}{\arg\max} \left\{ \sum_{R \in \mathcal{R}:\, u_j \in R} \frac{q_R}{1 - \mathbf{y}[j]} \right\}$;
  $\mathbf{y}[j] \leftarrow \mathbf{y}[j] + \epsilon$; $B \leftarrow B \cup \{u_j\}$;
  $\forall R \in \mathcal{R} : u_j \in R,\ q_R \leftarrow q_R \cdot \frac{1 - \mathbf{y}[j]}{1 - \mathbf{y}[j] + \epsilon}$;
return $B, \mathcal{Q}_{\mathcal{R}}$;

Eq.(3). This includes IM variants constrained by user groups [55] and group fairness [54].

As outlined in Algorithm 1, our proposed AMP shares a similar framework with existing continuous greedy methods but introduces two new search strategies, AMPSearch (Algorithm 2) for the general matroid and AMPSearch-PM (Algorithm 3) for the partition matroid, and a deterministic rounding algorithm, AMPRound (Algorithm 4). For ease of presentation, we assume w.l.o.g. that $1/\epsilon$ is an integer, since for any given ϵ', we can use $\frac{1}{\lceil 1/\epsilon' \rceil}$ as ϵ. During each search iteration t, AMPSearch and AMPSearch-PM adopt a more refined approach compared to the coarse-grained search of previous works [4, 11]. Specifically, they aim for the *steepest* improvement in F and include the element u_i with the largest $\partial F(\mathbf{x})/\partial \mathbf{x}[i]$ into B_t, resulting in quicker convergence and fewer search iterations. AMPSearch-PM improves the efficiency of AMPSearch further by pruning the search space U to one partition, reducing the number of evaluations of partial derivatives by a factor of r/k^*, where $k^* = \max\{k_1, k_2, \ldots k_h\}$. As for the rounding phase, unlike the randomized methods, our proposed AMPRound *deterministically* decides the elimination of a distinct element u_i based on the partial derivative w.r.t. $\mathbf{x}_i$. This ensures that $F(\mathbf{x})$ does not decrease throughout rounding until $\mathbf{x}$ becomes integral, yielding the rounded result S satisfying $\Lambda_{\mathcal{R}}(S) \geq F(\mathbf{x})$ *without failure probability.*

For the illustrative example in Figure 1, the red arrowed lines in Figures 1(c) and (d) trace the movement of $\mathbf{x}$ in the search and rounding process of AMP, respectively. Initially, when $\mathbf{x} = \mathbf{0}$ and AMP starts to search for B_1, u_2 is first included as $\partial F(\mathbf{x})/\partial \mathbf{x}[2]$ is the largest, updating $\mathbf{x}$ to $\mathbf{x} = \left[0, \frac{1}{3}, 0\right]$. Next, as $\partial F(\mathbf{x})/\partial \mathbf{x}[1]$ becomes the largest w.r.t. the current $\mathbf{x}$, u_1 is included in B_1, leading to an update of $\mathbf{x}$ to $\left[\frac{1}{3}, \frac{1}{3}, 0\right]$. Analogously, u_3 and u_2 (resp. u_1 and u_3) are selected and the corresponding basic solution is $B_2 = \{u_2, u_3\}$ (resp. $B_3 = \{u_1, u_3\}$), eventually yielding a fractional result $\mathbf{x} = \left[\frac{2}{3}, \frac{2}{3}, \frac{2}{3}\right]$ in the base polytope, as shown in Figure 1(c). After completing the search, as illustrated in Figure 1(d), AMP moves $\mathbf{x}$ to a vertex by ensuring the directions are towards maximizing $F(\mathbf{x})$ (i.e., the brighter area), thereby guaranteeing $F(\mathbf{x})$ is non-decreasing. □

4.2 Element-wise Ascent Search

We first present our element selection algorithm AMPSearch for the general matroid constraint (pseudocode in Algorithm 2). Specifically, Line 1 initializes by setting $B = \emptyset$ and the vector $\mathbf{y} = \mathbf{x}$. It then proceeds to iteratively populate B through r iterations, each time adding an element to construct a basic solution. In Line 3, during each iteration i, AMPSearch examines all elements in the remaining set $U \setminus B$ and selects the element u_j that maximizes the value of $\sum_{R \in \mathcal{R}: u_j \in R} q_R/(1 - \mathbf{y}[j])$, while adhering to the matroid constraint. This value corresponds to the partial derivative $\partial F(\mathbf{y})/\partial \mathbf{y}[j]$ as shown below. *All proofs are available in [31].*

Lemma 4.3. $\forall \mathbf{y} \in [0,1]^U$ *and* $u_i \in U$, $\frac{\partial F(\mathbf{y})}{\partial \mathbf{y}[i]} = \sum_{R \in \mathcal{R}: u_i \in R} \frac{q_R}{1 - \mathbf{y}[i]}$.

In Lines 4 - 5, AMPSearch updates $\mathbf{y}[j]$ to $\mathbf{y}[j] + \epsilon$ and includes u_j in B, followed by maintaining the correctness of each q_R value as per Eq.(5). In particular, for each RR set R containing u_j, AMPSearch updates the corresponding q_R by scaling it with $\frac{1 - \mathbf{y}[j]}{1 - \mathbf{y}[j] + \epsilon}$. This process removes the outdated factor $1 - \mathbf{y}[j] + \epsilon$ and includes the latest value of $1 - \mathbf{y}[j]$.

We next introduce the algorithm AMPSearch-PM tailored for the partition matroid constraint. In contrast to AMPSearch which selects the next element from $U \setminus B$, Algorithm 3 selects an arbitrary partition U_l of U satisfying $|B \cap U_l| < k_l$, and chooses the element u_j from $U_l \setminus B$ during each iteration, thus reducing the search space and running time of AMPSearch by a factor of r/k^*, where $k^* = \max\{k_1, k_2, \ldots k_h\}$. The following lemma offers a guarantee of both algorithms, bounding how much $F(\mathbf{x})$ increases as it approaches the optimal result S^o for maximizing $\Lambda_{\mathcal{R}}(\cdot)$ s.t. the matroid constraint.

Lemma 4.4. *For any input fractional result* $\mathbf{x}$*, the output* B *of both AMPSearch and AMPSearch-PM satisfies* $\Lambda_{\mathcal{R}}(S^o) - F(\mathbf{x} + \epsilon \cdot \mathbf{1}_B) \leq \frac{\Lambda_{\mathcal{R}}(S^o) - F(\mathbf{x})}{1 + \epsilon}$*, where* $S^o = \underset{S \in \mathcal{I}}{\arg\max}\, \Lambda_{\mathcal{R}}(S)$.

4.3 Element-wise Ascent Rounding

We introduce the rounding algorithm AMPRound, outlined in Algorithm 4. Given the basic solutions $B_1, B_2, \ldots, B_{1/\epsilon}$, AMPRound maintains a vector

$$\mathbf{y} = \textstyle\sum_{l=t+1}^{1/\epsilon} \epsilon \cdot \mathbf{1}_{B_l} + t \cdot \epsilon \cdot \mathbf{1}_{B_t}, \tag{6}$$

which represents the value of the fractional solution (i.e., $\mathbf{x}$) after merging the first t fractional solutions $\epsilon\mathbf{1}_{B_1}, \epsilon\mathbf{1}_{B_2}, \ldots, \epsilon\mathbf{1}_{B_t}$ corresponding to the basic solutions $B_1, B_2, ..., B_t$. It is initialized to $\mathbf{y} = \sum_{l=1}^{1/\epsilon} \epsilon\mathbf{1}_{B_l}$. AMPRound then scans each pair of successive basic solutions from (B_1, B_2) to $(B_{1/\epsilon - 1}, B_{1/\epsilon})$. For each pair (B_t, B_{t+1}), it iteratively eliminates distinct elements between the two basic

Algorithm 3: AMPSearch-PM ($\mathcal{R}, \mathbf{x}, \epsilon, Q_{\mathcal{R}}$)

Input: An RR collection $\mathcal{R}$, a fractional solution $\mathbf{x} \in [0,1]^U$, $0 < \epsilon \leq 1$, $Q_{\mathcal{R}}$ defined in Algorithm 1
Output: A basic solution $B \in \mathcal{B}$, an updated collection $Q_{\mathcal{R}}$

$B \leftarrow \emptyset$; $\mathbf{y} \leftarrow \mathbf{x}$;
for $i = 1, 2, \ldots, r$ **do**
  $U_l \leftarrow$ an arbitrary partition such that $|B \cap U_l| < k_l$;
  $u_j \leftarrow \arg\max_{u_j : u_j \in U_l \setminus B} \left\{ \sum_{R \in \mathcal{R}:\, u_j \in R} \frac{q_R}{1-\mathbf{y}[j]} \right\}$;
  $\mathbf{y}[j] \leftarrow \mathbf{y}[j] + \epsilon$; $B \leftarrow B \cup \{u_j\}$;
  $\forall R \in \mathcal{R} : u_j \in R,\ q_R \leftarrow q_R \cdot \frac{1-\mathbf{y}[j]}{1-\mathbf{y}[j]+\epsilon}$;
return $B, Q_{\mathcal{R}}$;

solutions until B_{t+1} is identical to B_t, and thereby merged with all preceding basic solutions $B_{t-1}, \ldots, B_1$. More precisely, AMPRound picks an arbitrary element u_i from $B_t \setminus B_{t+1}$, then identifies an element $u_j \in B_{t+1} \setminus B_t$ such that swapping u_i and u_j leads to $B_t \setminus \{u_i\} \cup \{u_j\}$ and $B_{t+1} \setminus \{u_j\} \cup \{u_i\}$ being bases of the matroid, as outlined in Lines 4-5. Notably, the existence of such an element u_j is assured by the following matroid exchange property.

Lemma 4.5 (Matroid Exchange property [13]). *Let B_1, B_2 be two bases of M. For any $u_i \in B_1 \setminus B_2$, there exists $u_j \in B_2 \setminus B_1$ such that $B_1 \setminus \{u_i\} \cup \{u_j\}$ and $B_2 \setminus \{u_j\} \cup \{u_i\}$ are also bases.*

Upon identifying a conflicting element pair (u_i, u_j), Lines 6-13 retain the one with the larger partial derivative. If the partial derivative w.r.t. u_i is not less than that w.r.t. u_j, AMPRound includes $u_i \in B_t \setminus B_{t+1}$ into B_{t+1} while removing u_j from it. Correspondingly, it updates the fractional result $\mathbf{y}$ by adding $\epsilon(\mathbf{1}_i - \mathbf{1}_j)$ as per Eq.(6), and modifies the values of q_R for all R that include u_i or u_j based on Eq.(5). Conversely, if u_j is to be preserved, AMPRound updates B_t by replacing u_i with u_j, and adds $t\epsilon \cdot (\mathbf{1}_j - \mathbf{1}_i)$ to $\mathbf{y}$ since the $t\epsilon \mathbf{1}_{B_t}$ term in Eq.(6) is outdated. Finally, it updates the corresponding q_R values. In each iteration, $F(\mathbf{y})$ remains non-decreasing and the quality of the rounded result is guaranteed by the following lemma.

Lemma 4.6. *For any $1/\epsilon$ basic solutions $B_1, \ldots, B_{1/\epsilon}$ with $\mathbf{x} = \epsilon \cdot \sum_{t=1}^{1/\epsilon} \mathbf{1}_{B_t}$, AMPRound returns an $S \in \mathcal{B}$ satisfying $\Lambda_{\mathcal{R}}(S) \geq F(\mathbf{x})$.*

In the example shown in Figure 1(d), AMPRound takes the basic solutions $B_1 = \{u_1, u_2\}$, $B_2 = \{u_2, u_3\}$, and $B_3 = \{u_1, u_3\}$ as inputs. Initially, AMPRound sets $\mathbf{y}$ to $\frac{1}{3} \cdot (\mathbf{1}_{B_1} + \mathbf{1}_{B_2} + \mathbf{1}_{B_3}) = \left[\frac{2}{3}, \frac{2}{3}, \frac{2}{3}\right]$. In the first iteration, (u_1, u_3) is the only conflicting pair for merging B_1 and B_2. AMPRound accepts u_3 but removes u_1 from B_1 based on their partial derivatives, yielding the merged $B_1 = B_2 = \{u_2, u_3\}$. AMPRound then updates $\mathbf{y} = \mathbf{y} + \frac{1}{3} \cdot (\mathbf{1}_3 - \mathbf{1}_1) = \left[\frac{1}{3}, \frac{2}{3}, 1\right]$. In the second iteration, (u_2, u_1) is the only conflicting pair for merging (the updated) B_2 and B_3. Following similar operations as in the previous iteration, the merged $B_2 = B_3 = \{u_2, u_3\}$ and the fractional result becomes $\mathbf{y} = \mathbf{y} + \frac{1}{3} \cdot (\mathbf{1}_2 - \mathbf{1}_1) = [0, 1, 1]$. Hence, the final discrete result after rounding is $S = \{u_2, u_3\}$. □

4.4 Putting It Together

Correctness. Recall in Algorithm 1 that AMP iteratively invokes AMPSearch or AMPSearch-PM in each iteration t. We denote the

Algorithm 4: AMPRound ($\mathcal{R}, \epsilon, Q_{\mathcal{R}}, B_1, B_2, \ldots, B_{1/\epsilon}$)

Input: An RR collection $\mathcal{R}$, $0 < \epsilon \leq 1$, $Q_{\mathcal{R}}$ defined in Algorithm 1, basic solutions $B_1, B_2, \ldots, B_{1/\epsilon}$
Output: A discrete set $S \in \mathcal{B}$

$\mathbf{y} \leftarrow \epsilon \cdot \sum_{l=1}^{1/\epsilon} \mathbf{1}_{B_l}$;
for $t = 1, 2, \ldots, 1/\epsilon - 1$ **do**
  while $B_t \neq B_{t+1}$ **do**
    $u_i \leftarrow$ an arbitrary element in $B_t \setminus B_{t+1}$;
    $u_j \leftarrow$ an element in $B_{t+1} \setminus B_t$ s.t. $B_t \setminus \{u_i\} \cup \{u_j\} \in \mathcal{B}$ and $B_{t+1} \setminus \{u_j\} \cup \{u_i\} \in \mathcal{B}$;
    if $\sum_{R \in \mathcal{R}:\, u_i \in R} \frac{q_R}{1-\mathbf{y}[i]} \geq \sum_{R \in \mathcal{R}:\, u_j \in R} \frac{q_R}{1-\mathbf{y}[j]}$ **then**
      $B_{t+1} \leftarrow B_{t+1} \setminus \{u_j\} \cup \{u_i\}$; $\mathbf{y} \leftarrow \mathbf{y} + \epsilon(\mathbf{1}_i - \mathbf{1}_j)$;
      $\forall R \in \mathcal{R} : u_i \in R,\ q_R \leftarrow q_R \cdot \frac{1-\mathbf{y}[i]}{1-\mathbf{y}[i]+\epsilon}$;
      $\forall R \in \mathcal{R} : u_j \in R,\ q_R \leftarrow q_R \cdot \frac{1-\mathbf{y}[j]}{1-\mathbf{y}[j]-\epsilon}$;
    else
      $B_t \leftarrow B_t \setminus \{u_i\} \cup \{u_j\}$; $\mathbf{y} \leftarrow \mathbf{y} + t\epsilon(\mathbf{1}_j - \mathbf{1}_i)$;
      $\forall R \in \mathcal{R} : u_j \in R,\ q_R \leftarrow q_R \cdot \frac{1-\mathbf{y}[j]}{1-\mathbf{y}[j]+t\epsilon}$;
      $\forall R \in \mathcal{R} : u_i \in R,\ q_R \leftarrow q_R \cdot \frac{1-\mathbf{y}[i]}{1-\mathbf{y}[i]-t\epsilon}$;
return $B_{1/\epsilon}$;

fractional result at the end of iteration t as $\mathbf{x}_t = \epsilon \cdot \sum_{i=1}^{t} \mathbf{1}_{B_i}$, with $\mathbf{x}_0 \leftarrow \mathbf{0}$. We can then rewrite the improvement of $F(\mathbf{x}_t)$ compared to $F(\mathbf{x}_{t-1})$ (see Lemma 4.4) as $\Lambda_{\mathcal{R}}(S^o) - F(\mathbf{x}_t) \leq \frac{\Lambda_{\mathcal{R}}(S^o) - F(\mathbf{x}_{t-1})}{1+\epsilon}$. By induction, this leads to $\Lambda_{\mathcal{R}}(S^o) - F(\mathbf{x}_t) \leq \frac{\Lambda_{\mathcal{R}}(S^o)}{(1+\epsilon)^t}$. Upon completion of $1/\epsilon$ search iterations, this inequation becomes $F(\mathbf{x}_{1/\epsilon}) \geq \left(1 - (1+\epsilon)^{-1/\epsilon}\right) \cdot \Lambda_{\mathcal{R}}(S^o)$. Given this inequation and Lemma 4.6, the result S returned by AMP satisfies the following theorem.

Theorem 4.7. *The result S of the AMP algorithm satisfies*

$$\Lambda_{\mathcal{R}}(S) \geq \left(1 - \frac{1}{(1+\epsilon)^{\frac{1}{\epsilon}}}\right) \Lambda_{\mathcal{R}}(S^o).$$

As per Theorem 4.7, by taking as input $\epsilon_s = \max\left\{\epsilon_s : \frac{1}{\epsilon_s} \in \mathbb{N},\ 1 - (1+\epsilon_s)^{-\frac{1}{\epsilon_s}} \geq 1 - \frac{1}{e} - \epsilon\right\}$ where $\mathbb{N} = \{0, 1, 2, \ldots\}$, the AMP algorithm achieves a $(1-1/e-\epsilon)$-approximation for maximizing the coverage $\Lambda_{\mathcal{R}}(S)$. This input ϵ_s can be obtained by binary search.

Running time for the partition matroid. In each iteration, AMPSearch-PM computes the partial derivative w.r.t. each u_j from a given partition U_l, incurring a cost of $O\left(\sum_{u_j \in U_l} \sum_{R \in \mathcal{R}} \mathbb{I}(u_j \in R)\right)$. Upon completion, AMPSearch-PM selects k_l elements from each U_l, leading to a total cost of $O\left(\sum_{l=1}^{h} \left(k_l \cdot \sum_{u_j \in U_l} \sum_{R \in \mathcal{R}} \mathbb{I}(u_j \in R)\right)\right) = O\left(k^* \cdot \sum_{R \in \mathcal{R}} |R|\right)$, where $k^* = max\{k_1, k_2, \ldots k_h\}$. AMPRound operates over $1/\epsilon - 1$ iterations. In each iteration t, it computes partial derivatives for every conflicting pair (u_i, u_j), with $u_i \in B_t \setminus B_{t+1}$ and $u_j \in B_{t+1} \setminus B_t$. Hence, the total running time of AMPRound is $O\left(\sum_{t=1}^{1/\epsilon - 1} \sum_{R \in \mathcal{R}} \sum_{u_i \in (B_t \setminus B_{t+1}) \cup (B_{t+1} \setminus B_t)} \mathbb{I}(u_i \in R)\right)$, which equals $O\left(\epsilon^{-1} \sum_{R \in \mathcal{R}} |R|\right)$. As shown in Algorithm 1, AMP runs AMPSearch-PM for $1/\epsilon$ iterations and invokes AMPRound once. Therefore, the running time of AMP subject to the partition matroid is

$$O\left(k^* \cdot \epsilon^{-1} \cdot \sum_{R \in \mathcal{R}} |R|\right).$$

As discussed in § 5.2, $\sum_{R\in\mathcal{R}}|R|$ is nearly linear to the size of the graph and seed sets. For instance, $\sum_{R\in\mathcal{R}}|R| = O(k\epsilon^{-2}|V|\ln|V|)$ in vanilla IM, and this running time becomes $O\left(k^2\epsilon^{-3}|V|\ln|V|\right)$.

Running time for general matroid. Unlike the partition matroid case, the general matroid scenario incurs additional cost to confirm if $B \in \mathcal{I}$. Specifically, Line 3 of Algorithm 2 checks whether $B \cup \{u_i\} \in \mathcal{I}$ given that $B \in \mathcal{I}$, and we assume that this operation can be finished by an *incremental independence call* in ϕ_i time. Additionally, Line 5 of Algorithm 4 identifies an element $u_j \in B_{t+1}\backslash B_t$ satisfying the matroid exchange property for given $B_t, B_{t+1} \in \mathcal{B}$. We assume that each identification can be completed by an *exchange call* in ϕ_e time. Following [30], we implement AMPSearch (see Algorithm 2) by examining the elements in U in decreasing order of their partial derivatives and selecting the first u_j with $B \cup \{u_j\} \in \mathcal{I}$. Whenever an element is found to be incompatible, i.e., adding it breaks independence, we can prune it from the candidate list for further iterations. With the above process, AMPSearch incurs $O(n)$ independence calls across all iterations. Notice that AMPSearch runs in r iterations, and each examines the derivatives of all n elements. Hence, the time complexity of AMPSearch is $O\left(n\phi_i + r\left(\sum_{u_i\in U}\sum_{R\in\mathcal{R}}\mathbb{I}(u_i \in R)\right)\right) = O\left(n\phi_i + r\cdot\sum_{R\in\mathcal{R}}|R|\right)$. Regarding AMPRound, as $|B_{t+1}\backslash B_t| \le r$, $O(r)$ exchange calls are required for finding a feasible u_j for each of the elements in $B_{t+1}\backslash B_t$. Hence, the entire algorithm requires $O(r/\epsilon)$ exchange calls. Combining this with the cost analyzed above, AMPRound costs a time of $O\left(\epsilon^{-1}\left(r\phi_e + \sum_{R\in\mathcal{R}}|R|\right)\right)$. Therefore, the running time of AMP subject to the general matroid is $O\left(\epsilon^{-1}\left(r\cdot\sum_{R\in\mathcal{R}}|R| + n\phi_i + r\phi_e\right)\right)$.

As an example, in the general version of IM for multiple products [17] subject to the laminar matroid constraint, the cost of each incremental independence call and exchange call is $\phi_i = O(\log n)$ and $\phi_e = O(\log n)$ [30]. Here, the running time of AMP is $O\left(\epsilon^{-1}\left(r\sum_{R\in\mathcal{R}}|R| + (n+r)\log n\right)\right)$.

5 RR SET GENERATION SCHEME

Recall in Theorem 4.7 that the proposed AMP offers the desired approximation guarantee w.r.t. the coverage function $\Lambda_{\mathcal{R}}(\cdot)$. In this section we present a Rapid version of AMP, called RAMP, which helps AMP to rapidly determine the required number of RR sets, ensuring the approximation guarantee for the original objective function $\sigma(\cdot)$. RAMP follows the state-of-the-art OPIM-C approach [48] but is generalized to IM-GM. In what follows, we briefly introduce the RAMP algorithm in § 5.1 and its time complexity in § 5.2.

5.1 RAMP

Main idea. RAMP runs iteratively to mitigate the generation of excessive RR sets. Initially, it generates two distinct collections of RR sets, denoted as $\mathcal{R}_1$ and $\mathcal{R}_2$, each with a size of θ_1. At each iteration i, the element selection algorithm AMP is performed on $\mathcal{R}_1$. Upon selecting a set S, $\mathcal{R}_2$ is leveraged to verify whether S satisfies $\sigma(S) \ge (1-1/e-\epsilon)\cdot\sigma(S^*)$, where $S^* = \arg\max_{S\in\mathcal{I}}\sigma(S)$. If it does, S is returned as the solution. If not, RAMP first increases the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ for the next iteration $i+1$ by including new RR sets, such that each size is equal to $\theta_{i+1} = 2\cdot\theta_i$, and then repeats the above process. RAMP runs in at most i_{max} iterations, and the result S in iteration i_{max} is obtained based on $\mathcal{R}_1$ with $|\mathcal{R}_1| = \theta_{max}$,

Algorithm 5: RAMP $(\mathsf{G}, \kappa, \mathsf{M}, \epsilon, \delta)$

Input: The graph $\mathcal{R}$, the constant κ, the matroid M, an error tolerance ϵ, a failure probability δ
Output: An IM-GM solution $S \in \mathcal{I}$ that provides a $1-1/e-\epsilon$ approximation w.p. at least $1-\delta$

1 $\epsilon_s \leftarrow$ Eq.(7);
2 $\theta_{max} \leftarrow$ Eq.(8); $i_{max} \leftarrow \ln\kappa$; $\theta_1 \leftarrow \left\lceil \frac{\theta_{max}}{2^{i_{\max}}} \right\rceil$;
3 Generate $\mathcal{R}_1$ and $\mathcal{R}_2$ with $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_1$;
4 **for** $i = 1, 2, \ldots, i_{max}$ **do**
5 $\quad S \leftarrow$ AMP $(\mathcal{R}_1, \mathsf{M}, \epsilon_s)$;
6 $\quad$ Compute $\sigma^u(\mathcal{S}^*)$ by Eq.(9) on $\mathcal{R}_1$ with $p_f = \frac{\delta}{3\cdot i_{max}}$;
7 $\quad$ Compute $\sigma^l(\mathcal{S})$ by Eq.(10) on $\mathcal{R}_2$ with $p_f = \frac{\delta}{3\cdot i_{max}}$;
8 $\quad$ **if** $\frac{\sigma^l(S)}{\sigma^u(S^*)} \ge 1-1/e-\epsilon$ ***or*** $i = i_{max}$ **then return** S;
9 $\quad \theta_{i+1} \leftarrow 2\cdot\theta_i$;
10 $\quad$ Increase the sizes of $\mathcal{R}_1, \mathcal{R}_2$ such that $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_{i+1}$;

where θ_{max} is a carefully-designed constant and ensures that S still provide the desired approximation in this worst case.

Detailed implementation. As illustrated in Algorithm 5, RAMP takes as input a graph G, an IM-GM instance associated with the matroid M and the constant κ, an error tolerance ϵ, and a failure probability δ. Recall in § 2.4 that the constant κ is the factor to ensure that $\Lambda_{\mathcal{R}}(\cdot)$ is an unbiased estimator of $\sigma(\cdot)$ for given IM-GM instance. As shown in Table 2, $\kappa = |V|$ in vanilla IM and MRIM, $\kappa = \sum_{t=1}^{T}\alpha_t\cdot|V|$ in RM, and $\kappa = \rho_G(A) - |A|$ in AdvIM. Akin to the input ϵ_s discussed in § 4.4, Line 1 of Algorithm 5 introduces

$$\epsilon_s = \max\left\{\epsilon_s : \tfrac{1}{\epsilon_s} \in \mathbb{N},\ 1-(1+\epsilon_s)^{-\frac{1}{\epsilon_s}} \ge 1-\tfrac{1}{e}-\tfrac{\epsilon}{2}\right\} \tag{7}$$

for the AMP algorithm, such that its result S satisfies $\Lambda_{\mathcal{R}_1}(S) \ge (1-1/e-\epsilon/2)\cdot\Lambda_{\mathcal{R}_1}(S_1^o)$, where $S_1^o = \arg\max_{S\in\mathcal{I}}\Lambda_{\mathcal{R}_1}(S)$. In Line 2, RAMP initializes three constants: (i) the maximum number θ_{max} of required RR sets, (ii) the maximum number of iterations i_{max}, and (iii) the number of RR sets θ_1 in the first iteration. To ensure that the result S returned in iteration i_{max} achieves $\sigma(S) \ge (1-1/e-\epsilon)\cdot\sigma(S^*)$ w.p. at least $1-\delta/3$, where $S^* = \arg\max_{S\in\mathcal{I}}\sigma(S)$, we set θ_{max} to

$$\theta_{max} = \frac{8\kappa\left((1-\frac{1}{e}-\frac{\epsilon}{2})\sqrt{\ln\frac{6}{\delta}} + \sqrt{(1-\frac{1}{e}-\frac{\epsilon}{2})(\ln|\mathcal{B}| + \ln\frac{6}{\delta})}\right)^2}{\epsilon^2\cdot\sigma^l(S^*)}. \tag{8}$$

In Eq.(8), $\sigma^l(S^*)$ is a lower bound of $\sigma(S^*)$. For IM, RM, and MRIM, $\sigma^l(S^*)$ can be set as k, $\max_{i\le i\le n}\{\alpha_i\}\cdot|V|$, and Tk, respectively. In AdvIM, $\sigma(S^*)$ can be arbitrarily close to 0 on degenerate instances. Since we are only interested in problem instances where the influence reduction is at least 1, i.e., $\sigma(S^*) \ge 1$, w.l.o.g. we can set $\sigma^l(S^*) \leftarrow 1$.

At each iteration i, upon obtaining $\mathcal{R}_1$ and $\mathcal{R}_2$ with $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_i$, RAMP employs AMP to select a set S to maximize the coverage on $\mathcal{R}_1$ (Lines 4-5). To verify whether current S achieves the desired approximation ratio (Lines 6-8), RAMP next computes an upper bound of the optimal solution $\sigma^u(S^*)$, and a lower bound of the current solution $\sigma^l(S)$ by leveraging $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively. The intuition behind this verification is that if $\frac{\sigma^l(S)}{\sigma^u(S^*)}$ reaches $1-1/e-\epsilon$,

then $\frac{\sigma(S)}{\sigma(S^*)} \geq \frac{\sigma^l(S)}{\sigma^u(S^*)}$ also satisfies this approximation. By setting

$$\sigma^u(S^*) = \left(\sqrt{\Lambda^u_{\mathcal{R}_1}(S^*) - \frac{\ln p_f}{2}} + \sqrt{\frac{-\ln p_f}{2}}\right)^2 \cdot \frac{\kappa}{|\mathcal{R}_1|}, \text{ and} \quad (9)$$

$$\sigma^l(S) = \left(\left(\sqrt{\Lambda_{\mathcal{R}_2}(S) - \frac{2\ln p_f}{9}} - \sqrt{\frac{-\ln p_f}{2}}\right)^2 + \frac{\ln p_f}{18}\right) \cdot \frac{\kappa}{|\mathcal{R}_2|} \quad (10)$$

with $p_f = \frac{\delta}{3 \cdot i_{max}}$, we can obtain that $\sigma^u(S^*)$ is an upper bound of $\sigma(S^*)$ w.p. at least $1 - \frac{\delta}{3 \cdot i_{max}}$ and $\sigma^l(S)$ is a lower bound of $\sigma(S)$ w.p. at least $1 - \frac{\delta}{3 \cdot i_{max}}$. In Eq.(9), $\Lambda^u_{\mathcal{R}_1}(S^*)$ is an upper bound of $\Lambda_{\mathcal{R}_1}(S^*)$, which can be set to $\frac{\Lambda_{\mathcal{R}_1}(S)}{1-\frac{1}{e}-\frac{\epsilon}{2}}$. To tighten $\Lambda^u_{\mathcal{R}_1}(S^*)$, we follow the idea of [48] that computes it based on the current fractional solution $\mathbf{x}$ during the iterations of AMP. See [31] for a detailed explanation.

To summarize, based on the above settings of θ_{max}, $\sigma^u(S^*)$, and $\sigma^l(S)$, by the union bound, the correctness of RAMP follows.

Theorem 5.1. *Given a graph* G, *an* IM-GM *instance associated with the matroid* M *and the constant* κ, *an error tolerance* ϵ, *and a failure probability* δ, *RAMP returns a set* S *satisfying* $\sigma(S) \geq (1 - 1/e - \epsilon) \cdot \sigma(S^*)$ *with probability at least* $1 - \delta$, *where* $S^* = \arg\max_{S \in \mathcal{I}} \sigma(S)$.

5.2 Time Complexity Analysis

Denote the expected time of generating an RR set as EPT, and the expected size of an RR set as EPS. In each iteration i, the computational overhead of RAMP comes from (i) generating RR sets in $O(\mathbb{E}[\theta_i \cdot EPT])$, and (ii) an invocation of AMP in $O\left(k^* \cdot \epsilon_s^{-1} \cdot \sum_{R \in \mathcal{R}_1} |R|\right) = O\left(k^* \cdot \epsilon_s^{-1} \cdot \mathbb{E}[\theta_i \cdot EPS]\right)$ for the partition matroid, as analyzed in § 4.4. Let i' be the iteration where RAMP stops. As proved in [50], $\mathbb{E}[\theta_{i'} \cdot EPT] = \mathbb{E}[\theta_{i'}] \cdot EPT$ and $\mathbb{E}[\theta_{i'} \cdot EPS] = \mathbb{E}[\theta_{i'}] \cdot EPS$. Thus, the time complexity of RAMP for the partition matroid is $O\left(\mathbb{E}\left[\sum_{i=1}^{i'} \theta_i\right]\left(EPT + \frac{k^*}{\epsilon_s} \cdot EPS\right)\right)$, where $k^* = \max\{k_1, \ldots k_h\}$, and $\mathbb{E}\left[\sum_{i=1}^{i'} \theta_i\right]$ is bounded as follows.

Lemma 5.2. *When* $\delta \leq 1/2$, *RAMP totally generates* $O((\ln|\mathcal{B}| + \ln(1/\delta))\kappa\epsilon^{-2}/\sigma(S^*))$ *RR sets in expectation.*

Given this lemma and $\epsilon_s = O(\epsilon)$, when $\delta \leq 1/2$, the expected time complexity of RAMP subject to the partition matroid is

$$O\left(\frac{(\ln|\mathcal{B}| + \ln(1/\delta))\kappa\epsilon^{-2}}{\sigma(S^*)} \cdot \left(EPT + \epsilon^{-1} \cdot k^* \cdot EPS\right)\right).$$

Akin to the above analysis, when $\delta \leq 1/2$, the time complexity of RAMP for the general matroid is

$$O\left(\frac{(\ln|\mathcal{B}| + \ln(1/\delta))\kappa\epsilon^{-2}}{\sigma(S^*)} \cdot \left(EPT + \epsilon^{-1} r \cdot EPS\right) + \frac{(r \cdot \phi_e + n \cdot \phi_i) \cdot \ln\kappa}{\epsilon}\right).$$

For IM, [50] shows that $EPT = O\left(\frac{|E|}{|V|}\rho^*\right)$ and $EPS = O(\rho^*)$ under IC and LT models, where $\rho^* = \max_{v_i \in V} \rho_G(\{v_i\})$. Besides IM, the original works of RM, MRIM, and AdvIM [29, 46, 47] also provide analyses for EPT and EPS, which are summarized in Table 2. Since they are related to ρ^* and $\rho^* \leq \sigma(S^*)$, the above complexities can be further simplified. E.g., for MRIM with $\kappa = |V|$ and $|\mathcal{B}| = \binom{|V|}{k}^T$, the time complexity is $O\left(T^2 k \epsilon_t^{-2}(\ln|V| + \ln(1/\delta)) \cdot (|E| + k\epsilon^{-1}|V|)\right)$.

Table 2: κ, EPT, and EPS in instances ($\rho^* = \max_{v_i \in V} \rho_G(\{v_i\})$).

Name	κ	EPT	EPS
Vanilla IM	$\|V\|$	$O\left(\frac{\|E\|}{\|V\|}\rho^*\right)$	$O(\rho^*)$
RM	$\sum_{t=1}^{T} \alpha_t \cdot \|V\|$	$O\left(\frac{\|E\|}{\|V\|}\rho^*\right)$	$O(\rho^*)$
MRIM	$\|V\|$	$O\left(\frac{T \cdot \|E\|}{\|V\|}\rho^*\right)$	$O(T \cdot \rho^*)$
AdvIM	$\rho_G(A) - \|A\|$	$O\left(\frac{\|V \setminus A\|}{\rho_G(A) - \|A\|} \cdot \frac{\|E\|}{\|V\|}\rho^*\right)$	$O(\rho^*)$

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

Name	#nodes	#edges	Type	Avg. degree
Facebook	$4.0K$	$176.4K$	friendship	43.6
Wiki	$7.1K$	$103.6K$	who-votes-on-whom	14.5
Twitter	$81.3K$	$1.7M$	who-follows-whom	21.7
Google+	$107.6K$	$13.7M$	who-follows-whom	283.3
Pokec	$1.6M$	$30.6M$	friendship	18.7
LiveJournal	$4.8M$	$69.0M$	friendship	14.2
Twitter-large	$41.7M$	$1.5B$	who-follows-whom	35.2

6 EXPERIMENTS

We first introduce the experimental settings in § 6.1, and then evaluate the performance of our proposal in § 6.2. All experiments are conducted on a Linux machine with Intel Xeon(R) Gold 6240@2.60GHz CPU and 377GB RAM in single-thread mode.

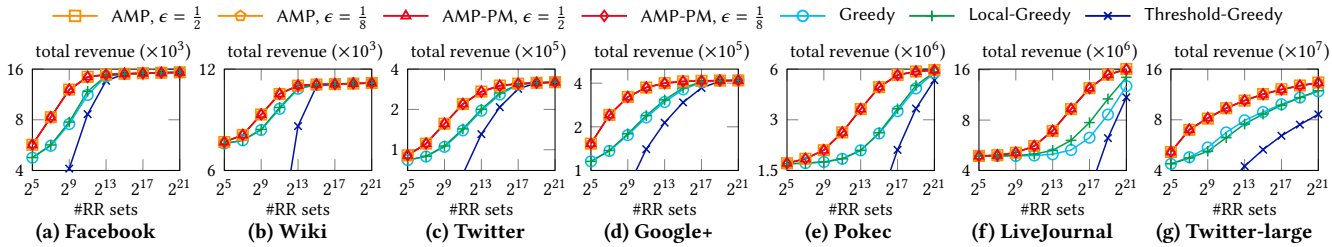
6.1 Experimental Setup

Datasets. We experiment with 7 real-world datasets that are widely adopted in previous works and publicly available at SNAP [38]: Facebook [5, 39, 58], Wiki [39], Twitter [5], Google+ [58], Pokec [48], LiveJournal [48, 58], and Twitter-large [48]. The Twitter-large dataset is the largest dataset ever used in instances RM, MRIM, and AdvIM. The dataset statistics are listed in Table 3.

IM-GM instances and configurations. We focus on the instances IM, RM, MRIM, and AdvIM, as illustrated in § 2.2-2.3. Following the default configurations in related works [27, 29, 46, 47], we use the IC model in RM, MRIM, IM and the LT model in AdvIM. We set $p_{i,j} = 1/|N^{in}_j|$ for all $e_{i,j} \in E$. We configure $T = 10, \alpha_i = 1, k_j = 1$ for all $i \in [T], v_j \in V$ for RM, set $T = 20, k = 100$ for MRIM, and $k = 2000$ for IM. In the absence of a ground-truth set A for AdvIM, where $|A|$ represents the number of contagious seeds as detected by the authority, we generate a synthetic set A. This is accomplished by initially adding the node v_i with the largest out-degree to set A. Following this, each out-neighbor v_j of v_i is independently added to A with a probability of $p_{i,j}$. Furthermore, we set $k_v = 500$ and $k_e = 1000$, which are comparable in magnitude to $|A|$.

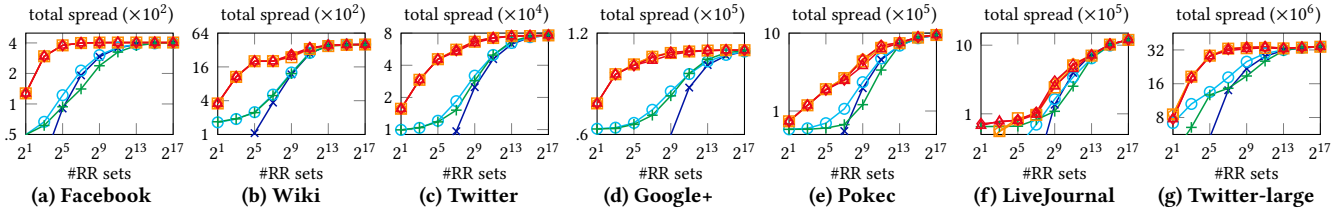
Algorithms and constant settings. We evaluate the performance of 9 algorithms in total, which are divided into two parts: 5 algorithms for element selection, and 4 algorithms for their scalable implementation. These algorithms are as follows.

- *Element selection*: Greedy, Local-Greedy, and Threshold-Greedy (baselines in § 2.4); AMP and AMP-PM for the general matroid and partition matroid, respectively (Algorithm 1).
- *Scalable solutions*: RM-A for RM [29], CR-NAIMM for MRIM [46], and AAIMM for AdvIM [47] (baselines); RAMP (Algorithm 5).



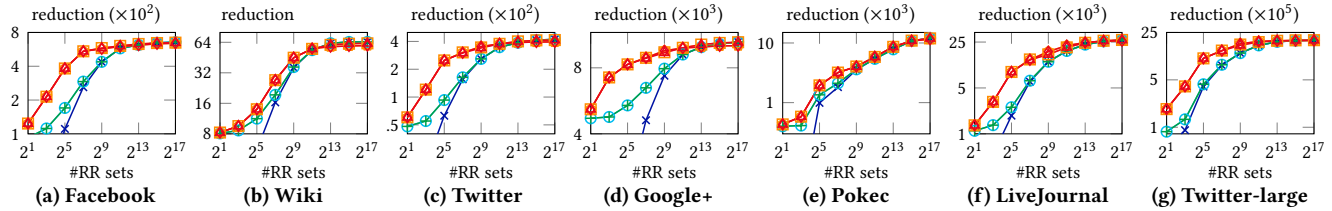
(a) Facebook (b) Wiki (c) Twitter (d) Google+ (e) Pokec (f) LiveJournal (g) Twitter-large

Figure 2: Total revenue in RM on various graphs.



(a) Facebook (b) Wiki (c) Twitter (d) Google+ (e) Pokec (f) LiveJournal (g) Twitter-large

Figure 3: Total spread in MRIM on various graphs.



(a) Facebook (b) Wiki (c) Twitter (d) Google+ (e) Pokec (f) LiveJournal (g) Twitter-large

Figure 4: Influence reduction in AdvIM on various graphs.

For all element selection algorithms, we apply the lazy evaluation technique in [37, 42] to improve their empirical efficiency. For AMP and AMP-PM, ϵ is set to 1/2 and 1/8. Based on Theorem 4.7, these settings provide an approximation of 0.56 and 0.61 for maximizing $\Lambda_{\mathcal{R}}$, respectively, indicating a focus on efficiency but at the cost of result quality, and vice versa. For Threshold-Greedy, ξ is set to the default value 0.05 [6]. For all scalable implementations, we set $\delta = 1/|V|$ by default and vary ϵ from 0.5 to 0.1 [48, 50]. For a fair comparison, we have also made other necessary modifications to scalable competitors and refer interested readers to [31] for details. In addition, for each algorithm, we estimate the objective function value σ by averaging over 10,000 Monte Carlo simulations. We repeat each algorithm 5 times and report the average when evaluating the running time and the objective function value. All algorithms are implemented in C++ with -O3 optimization.

6.2 Performance Evaluations

Performance of element selection in RM, MRIM, and AdvIM. In the first set of experiments, we evaluate the result quality and running time of element selection algorithms on the IM-GM instances RM, MRIM, and AdvIM. For a fair comparison, we provide all algorithms with the same collection of RR sets $\mathcal{R}$. Furthermore, we vary $|\mathcal{R}|$ from 2^5 to 2^{21} in RM; and from 2^1 to 2^{17} in MRIM and AdvIM. The choices of the smallest and largest values for $|\mathcal{R}|$ roughly correspond to θ_1 and θ_{max} in Algorithm 5 when $\epsilon = 0.1$.

Result quality. As shown in Figures 2-4, both AMP and AMP-PM return better solutions compared to their competitors in terms of related objective functions across all datasets. For instance, AMP or AMP-PM outperforms all competitors by up to 86% on Pokec in RM, 708% on Wiki in MRIM, and 205.8% on Twitter-large in AdvIM. To explain, AMPSearch and AMPSearch-PM select each element based on the largest partial derivative and subsequently update the value of q_R for each R containing the selected element. In contrast to the competitors Greedy, Local-Greedy, and Threshold-Greedy, our update strategy circumvents the direct exclusion of each related R and offers a more fine-grained marginal coverage for each element. As a result, it allows for a more refined and potentially more effective selection process. Furthermore, we observe that the result quality of Threshold-Greedy is much worse than other approaches when the number of RR sets is small. This is because it disregards all elements whose marginal coverage is below the minimum threshold, rendering the number of selected elements fewer than expected.

Running time. In the second set of experiments, we use RM as the instance and report the running times of element selection algorithms in Figure 5. Similar trends were observed in the running time results for other instances. Specifically, Local-Greedy and AMP-PM (with $\epsilon = 1/2$) are the fastest and second-fastest algorithms respectively, across all datasets, each being $4 - 10\times$ faster than Greedy. Recall from the first set of experiments that Local-Greedy, despite being the fastest, is the second least effective algorithm, quality wise. Furthermore, we compare AMP and AMP-PM with the same ϵ values across these problems using partition matroids. In particular, AMP-PM improves the running time by an order of magnitude. These observations demonstrate the efficiency of AMP-PM, as proved in § 4.4. Note that, in Figure 5(e), the running time of AMP decreases when $|\mathcal{R}| \geq 2^{15}$, due to the lazy evaluation technique. We also

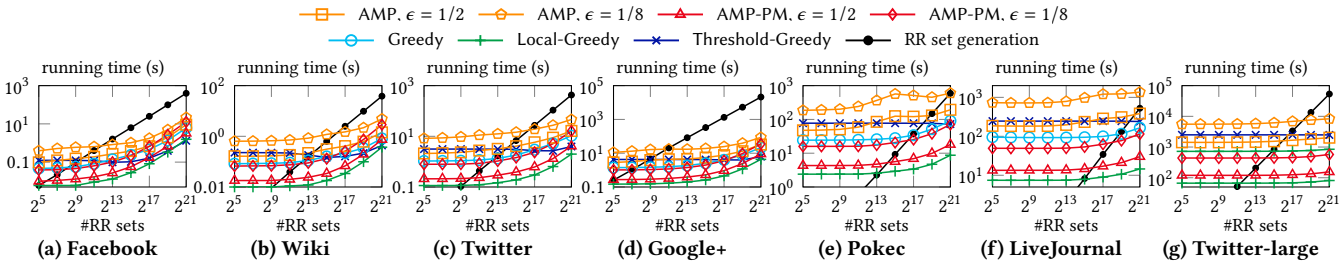


Figure 5: Running time of element selection in RM.

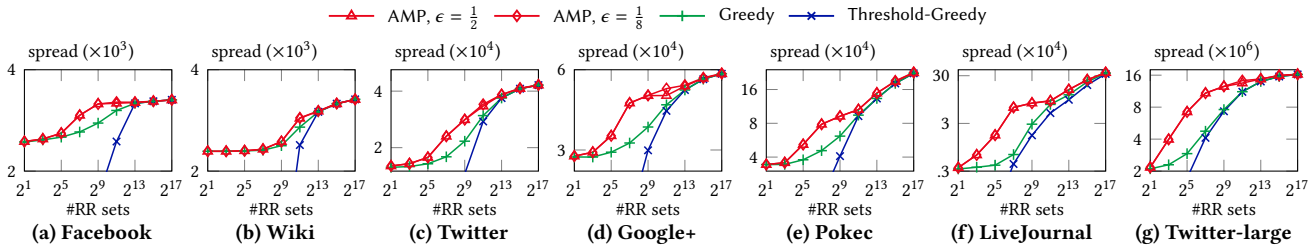


Figure 6: The spread in IM on various graphs.

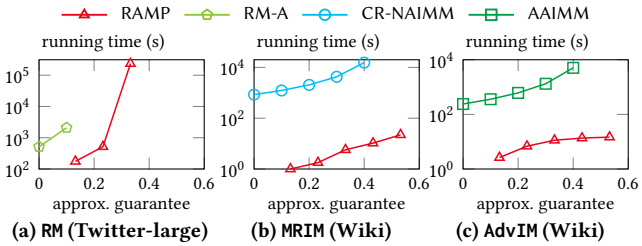


Figure 7: Total running time of scalable algorithms.

report the time taken to generate $\mathcal{R}$. RR set generation becomes more expensive than all methods as the size of $\mathcal{R}$ increases. For instance, when $|\mathcal{R}| > 2^9$ on Google+, the time taken to generate $\mathcal{R}$ exceeds that of AMP-PM by more than an order of magnitude. This motivates the design of an efficient scheme for RR set generation, as proposed in § 5.

Performance of element selection in IM. In the third set of experiments, we evaluate the performance of the proposed element selection methods on the conventional IM problem. Here, we only compare AMP with Greedy and Threshold-Greedy as AMP and AMP-PM (resp. Greedy and Local-Greedy) are essentially equivalent. Figure 6 reports the spread of each method on different datasets as $|\mathcal{R}|$ is varied from 2^1 to 2^{17}. Notably, AMP outperforms competitors by up to 871% on LiveJournal. This is attributed to the more fine-grained marginal coverage, as discussed earlier.

Effect of ϵ in AMP. Next, we examine the impact of the ϵ parameter in the proposed AMP algorithm. Here, we focus on the result qualities of AMP and Greedy, as shown in Figures 2-6. Note that AMP is equivalent to Greedy when $\epsilon = 1$. Based on observations in the first and third sets of experiments, the result quality of AMP improves significantly by decreasing ϵ from 1 to 1/2 (i.e., increasing the number of iterations in Line 4 of Algorithm 1 from 1 to 2). This improvement becomes less pronounced with further increases in the number of iterations. For example, in MRIM, the solution quality of AMP improves by 84% on Pokec when $|\mathcal{R}| = 2^9$ as ϵ decreases from 1 to 1/2. In contrast, the improvement is only 7% when comparing $\epsilon = 1/2$ with $\epsilon = 1/8$.

Performance of scalable implementations. Finally, we evaluate the running times of RAMP and all scalable competitors by varying ϵ from 0.5 to 0.1. In Figure 7, the approximation ratio of each solution is used as the x-axis, with RAMP having $1 - 1/e - \epsilon$ and all competitors having $1/2 - \epsilon$; in Figure 7(a), we exclude approaches that run more than 72 hours. Due to space constraints, we only report results for the largest dataset on which the baseline algorithm with $\epsilon = 0.5$ can terminate within 72 hours. These datasets are Twitter-large in RM and Wiki in MRIM and AdvIM. Interested readers can find more experimental results, including the memory usage of scalable algorithms, in [31].

Specifically, RAMP consistently outperforms the baseline regarding running time while ensuring the same approximation guarantee across all cases. Notably, RAMP is at least 1,000× faster than CR-NAIMM in MRIM and at least 100× faster than AAIMM in AdvIM. In addition, RAMP completes within 66 hours for RM on Twitter-large when $\epsilon = 0.3$, whereas RM-A only returns results for $\epsilon \geq 0.4$ and fails to terminate even after 7 days when $\epsilon = 0.3$. The reason for the efficiency of RAMP is twofold. First, RAMP introduces the early termination and the tightened bound $\Lambda^u_{\mathcal{R}_1}(S^*)$, which allows us to generate an appropriate number of RR sets as soon as a $(1-1/e-\epsilon)$-approximation is reached. Furthermore, AMP provides superior result qualities during iterations, enabling RAMP to reach the desired approximation faster. In contrast, RM-A, CR-NAIMM, and AAIMM generate a large number of RR sets and rely on Greedy or Local-Greedy for element selection, leading to inferior result qualities. To summarize, the proposed RAMP is significantly faster than all competitors while providing the same result quality.

7 DEPLOYMENT

We have deployed our proposed RAMP in a real industry setting for GameX of Tencent. The details are described below.

Table 4: Numbers of engaged pairs in GameX ($M = 10^6$).

Algorithm	Most-Click	RM-A	RAMP
#engaged pairs	7.34M	7.36M	7.50M

Application scenario. GameX is a Tencent multiplayer online game with hundreds of millions of users and a massive number of user-generated maps. A map in the game is a vibrant and dynamic environment featuring several obstacles and puzzle-solving settings, allowing multiple users to compete with each other simultaneously. Besides, the social network in GameX can be constructed based on friendships formed in the game, with an average of ~16 friends per user, making it sufficiently dense [3]. As such, users can interact with their friends on the social network, such as playing or inviting to play the game in some maps, which contributes to cascades of maps in the game. In other words, a user might play the game in a map if their friends have played or shared it. To further promote the maps, the game environment can generate k maps as recommendations for each user, who can receive them during the game and click the maps they are interested in playing in. The performance of GameX is measured by the number of engaged user-map pairs, each of which denotes that a user plays a user-generated map, reflecting the engagement of users in the game.

Control and treatment groups. We have deployed three approaches: (i) Most-Click, (ii) RM-A, and (iii) RAMP. Specifically, for the control group, we have deployed Most-Click, the state-of-the-art solution for this scenario. It operates by first ranking maps for each user v_i based on the descending order of probability $w_{i,t}$ that v_i will click map t and then selecting the top-k maps with the highest click probabilities for each user. We learn the probability $w_{i,t}$ using XGBoost [14], a machine learning model to predict whether a given user would play the game in a given map.

We have deployed RM-A and RAMP for two different treatment groups. For treatment groups, we focus on addressing the RM problem and returning k user-map pairs for each user. Notably, we set each unit revenue to 1 and employ the IC model as the diffusion model. In this model, the influence probability $p_{i,j}$ for each $e_{i,j} \in E$ is calculated as $\frac{c_{i,j}}{\sum_{v_i \in N_j^{in}} c_{i,j}}$, where $c_{i,j}$ denotes the number of historical interactions on $e_{i,j}$, such as co-playing, gifting, etc. To address RM in this context, we first leverage the competing solution RM-A [29], as mentioned in § 6, the core of which is selecting k seeds for each map using Greedy. Next, we implement the proposed RAMP by targeting an enhanced objective function $\sigma^W(S)$ based on $\sigma(S)$. This function is defined as

$$\sigma^W(S) = \sum_{S' \subseteq S} \prod_{(v_i,t) \in S'} w_{i,t} \prod_{(v_i,t) \in S \setminus S'} (1 - w_{i,t}) \cdot \sigma(S'),$$

where W is a given (or learned) set of click probabilities that contains $w_{i,t}$ for every user-map pair (v_i, t). By adjusting the scale of q_R in AMP, RAMP can be adapted to this new objective while maintaining the same approximation guarantee and time complexity. For further details, readers are referred to [31].

Deployment setups. This deployment is conducted on an in-house cluster in Tencent consisting of hundreds of machines, each of which has 16GB memory and 12 Intel Xeon Processor E5-2670 CPU cores. To mitigate the network effect, the phenomenon whereby user behaviors are influenced by others within the same network, we follow [44] and conduct cluster-level experimentation. Specifically, we first partition all users into several communities of almost the same size with high edge connectivity and node feature similarity, and then randomly assign the live traffic in the same community to the treatment or control groups. Furthermore, we set the number of recommended maps k to 2. As a result, each approach is assigned to 0.7 million sampled users with 400 randomly selected maps to recommend, leading to 277.5 million user-map pairs for observation. In this scenario, the constraint is a partition matroid $\mathsf{M} = (U, \mathcal{I})$, where U comprises all pairs of users and maps (277.5 million) and $\mathcal{I}$ consists of all solutions, each containing $k = 2$ maps per user.

Online performance. Table 4 reports the number of engaged pairs for each algorithm. Notably, the treatment groups achieve more engaged pairs compared to the control group Most-Click – 20,000 (0.27%) for RM-A and 160,000 (2.17%) for RAMP, demonstrating the usefulness of involving the word-of-mouth effect in this scenario. In addition, RAMP yields 140,000 (1.9%) more engaged pairs than RM-A. To explain, unlike Most-Click which solely focuses on the probability of a seed adopting the promoted map, RAMP takes into account not only this inclination but also the subsequent word-of-mouth effect, as captured by the IC model with its associated influence probability. The improvement of RAMP over RM-A can be attributed to two factors. First, RAMP achieves a $(1 - 1/e - \epsilon)$-approximation, while the result of RM-A is $(1/2 - \epsilon)$-approximate. Second, the enhanced objective $\sigma^W(S)$ comprehensively considers the adoption inclination of any subset of S. In contrast, even if the revenue unit α_t is leveraged, RM-A can only involve the average inclination for each map t. While the percentage improvement may appear modest, the improved numbers of engaged pairs lead to substantial additional revenue for the industry.

8 CONCLUSIONS

In this paper, we focus on applications requiring multiple seed sets for IM, which are formulated as instances of IM subject to a matroid constraint. For effectiveness, we propose AMP, which achieves a $(1 - 1/e - \epsilon)$-approximation guarantee for this problem. For enhanced efficiency, we also propose a fast implementation called RAMP. Our comprehensive experiments demonstrate that our proposal outperforms state-of-the-art methods in both effectiveness and efficiency. Moreover, we have successfully deployed RAMP in an online gaming propagation scenario in a real industry setting, yielding considerable improvements. In the future, to exploit the infrastructure in the industry further, we will consider solutions in the distributed setting. It is also interesting to explore real-world applications of IM under other types of matroid constraints and develop scalable and effective solutions for them.

ACKNOWLEDGMENTS

This research is supported by the Ministry of Education, Singapore, under its MOE AcRF TIER 3 Grant (MOE-MOET32022-0001). Lakshmanan's research was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada (Grant Number RGPIN-2020-05408).

REFERENCES

[1] Alexander A Ageev and Maxim I Sviridenko. 2004. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization* 8 (2004), 307–328.

[2] Cigdem Aslay, Francesco Bonchi, Laks VS Lakshmanan, and Wei Lu. 2017. Revenue maximization in incentivized social advertising. *Proc. VLDB Endowment* 10, 11 (2017), 1238–1249.

[3] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. 2012. Four degrees of separation. In *Web Science*. 33–42.

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

[5] Ruben Becker, Gianlorenzo d'Angelo, and Hugo Gilbert. 2021. Influence Maximization With Co-Existing Seeds. In *CIKM*. 100–109.

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

[7] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *SODA*. 946–957.

[8] Niv Buchbinder, Moran Feldman, and Roy Schwartz. 2017. Comparing apples and oranges: Query trade-off in submodular maximization. *Mathematics of Operations Research* 42, 2 (2017), 308–329.

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

[10] ByteDance. 2023. Recruitment-based Campaign in the Document of Douyin Xingtu Platform. https://www.xingtu.cn/help-center/demander/126583.

[11] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* 40, 6 (2011), 1740–1766.

[12] T-H Hubert Chan, Li Ning, and Yong Zhang. 2020. Influence maximization under the non-progressive linear threshold model. In *International Workshop on Frontiers in Algorithmics*. 37–48.

[13] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. 2010. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*. 575–584.

[14] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *SIGKDD*. 785–794.

[15] Wei Chen, Ruihan Wu, and Zheng Yu. 2020. Scalable lattice influence maximization. *IEEE Transactions on Computational Social Systems* 7, 4 (2020), 956–970.

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

[17] Nan Du, Yingyu Liang, Maria-Florina Balcan, Manuel Gomez-Rodriguez, Hongyuan Zha, and Le Song. 2017. Scalable Influence Maximization for Multiple Products in Continuous-Time Diffusion Networks. *J. Mach. Learn. Res.* 18, 2 (2017), 1–45.

[18] Jack Edmonds. 1971. Matroids and the Greedy Algorithm. *Math. Program.* 1, 1 (1971), 127–136.

[19] Marwa El Halabi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub M Tarnawski. 2020. Fairness in Streaming Submodular Maximization: Algorithms and Hardness. In *NeurIPS*. 13609–13622.

[20] Alina Ene and Huy L. Nguyen. 2019. Towards Nearly-Linear Time Algorithms for Submodular Maximization with a Matroid Constraint. In *ICALP*. 54:1–54:14.

[21] Ocean Engine. 2022. ByteDance Xingtu Creator Ecosystem Report. https://trendinsight.oceanengine.com/arithmetic-report/detail/773.

[22] Ocean Engine. 2023. China Influencer Marketing with Douyin and Xingtu. https://www.oceanengine.io/resource/china-influencer-marketing-douyin-xingtu-tips-strategies.

[23] Pihu Feng, Xin Lu, Zaiwu Gong, and Duoyong Sun. 2021. A case study of the pyramid scheme in China based on communication network. *Physica A: Statistical Mechanics and its Applications* 565 (2021), 125548.

[24] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. 1978. An analysis of approximations for maximizing submodular set functions—II. In *Polyhedral combinatorics*. 73–87.

[25] Jacob Goldenberg, Barak Libai, and Eitan Muller. 2001. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters* 12, 3 (2001), 211–223.

[26] Mark Granovetter. 1978. Threshold models of collective behavior. *American journal of sociology* 83, 6 (1978), 1420–1443.

[27] Qintian Guo, Sibo Wang, Zhewei Wei, and Ming Chen. 2020. Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *SIGMOD*. 2167–2181.

[28] Qintian Guo, Sibo Wang, Zhewei Wei, Wenqing Lin, and Jing Tang. 2022. Influence Maximization Revisited: Efficient Sampling with Bound Tightened. *ACM Transactions on Database Systems* 47, 3 (2022).

[29] Kai Han, Benwei Wu, Jing Tang, Shuang Cui, Cigdem Aslay, and Laks VS Lakshmanan. 2021. Efficient and effective algorithms for revenue maximization in social advertising. In *SIGMOD*. 671–684.

[30] Monika Henzinger, Paul Liu, Jan Vondrák, and Da Wei Zheng. 2023. Faster Submodular Maximization for Several Classes of Matroids. In *ICALP*. 74:1–74:18.

[31] Yiqian Huang, Shiqi Zhang, Laks V. S. Lakshmanan, Wenqing Lin, Xiaokui Xiao, and Bo Tang. 2024. Efficient and Effective Algorithms for A Family of Influence Maximization Problems with A Matroid Constraint [Technical Report]. arXiv:2410.16603 [cs.SI] https://arxiv.org/abs/2410.16603

[32] Influencer Marketing Hub. 2024. Top Influencer Marketing Platforms for April 2024. https://influencermarketinghub.com/influencer-marketing-platforms.

[33] Instagram. 2022. Introducing Creator Marketplace, Where Brands Can Discover Creators to Collaborate With. https://business.instagram.com/blog/creator-marketplace-discover-partnerships.

[34] Emilia A Isolauri and Irfan Ameer. 2023. Money laundering as a transnational business phenomenon: a systematic review and future agenda. *Critical Perspectives on International Business* 19, 3 (2023), 426–468.

[35] Raghuram Iyengar, Christophe Van den Bulte, and Thomas W Valente. 2011. Opinion leadership and social contagion in new product diffusion. *Marketing science* 30, 2 (2011), 195–212.

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

[37] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *SIGKDD*. 420–429.

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

[39] Weihua Li, Quan Bai, Minjie Zhang, and Tung Doan Nguyen. 2018. Automated influence maintenance in social networks: an agent-based approach. *IEEE Transactions on Knowledge and Data Engineering* 31, 10 (2018), 1884–1897.

[40] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. 2018. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1852–1872.

[41] Hao Liao, Sheng Bi, Jiao Wu, Wei Zhang, Mingyang Zhou, Rui Mao, and Wei Chen. 2023. Popularity Ratio Maximization: Surpassing Competitors through Influence Propagation. *Proc. ACM Manag. Data* 1, 2 (2023).

[42] Michel Minoux. 1978. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques* (1978), 234–243.

[43] Hung T Nguyen, My T Thai, and Thang N Dinh. 2016. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *SIGMOD*. 695–710.

[44] Martin Saveski, Jean Pouget-Abadie, Guillaume Saint-Jacques, Weitao Duan, Souvik Ghosh, Ya Xu, and Edoardo M Airoldi. 2017. Detecting network effects: Randomizing over randomized experiments. In *SIGKDD*. 1027–1035.

[45] Michael Simpson, Farnoosh Hashemi, and Laks VS Lakshmanan. 2022. Misinformation mitigation under differential propagation rates and temporal penalties. *Proc. VLDB Endowment* 15, 10 (2022), 2216–2229.

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

[47] Lichao Sun, Xiaobin Rui, and Wei Chen. 2023. Scalable Adversarial Attack Algorithms on Influence Maximization. In *WSDM*. 760–768.

[48] Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. 2018. Online processing algorithms for influence maximization. In *SIGMOD*. 991–1005.

[49] Shaojie Tang. 2018. When social advertising meets viral marketing: Sequencing social advertisements for influence maximization. In *AAAI*, Vol. 32.

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

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

[52] Tiktok. 2023. 5 ways TikTok Creator Marketplace makes creator marketing easy. https://www.tiktok.com/business/en-US/blog/making-creator-marketing-easy/.

[53] Amo Tong, Ding-Zhu Du, and Weili Wu. 2018. On Misinformation Containment in Online Social Networks. In *NeurIPS*, Vol. 31.

[54] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. 2019. Group-Fairness in Influence Maximization. In *IJCAI*. 5997–6005.

[55] Rajan Udwani. 2018. Multi-Objective Maximization of Monotone Submodular Functions with Cardinality Constraint. In *NeurIPS*. 9513–9524.

[56] Arun Vishwanath. 2015. Diffusion of deception in social media: Social contagion effects and its antecedents. *Information Systems Frontiers* 17 (2015), 1353–1367.

[57] Shiqi Zhang, Yiqian Huang, Jiachen Sun, Wenqing Lin, Xiaokui Xiao, and Bo Tang. 2023. Capacity Constrained Influence Maximization in Social Networks. In *SIGKDD*. 3376–3385.

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