Theoretical and Learning-based Approaches for Influence Maximization in Multilayer Social Networks

Xueqin Chang
Zhejiang University
changxq@zju.edu.cn

Ruize Liu
Zhejiang University
lrz@zju.edu.cn

Qing Liu
Zhejiang University
qingliucs@zju.edu.cn

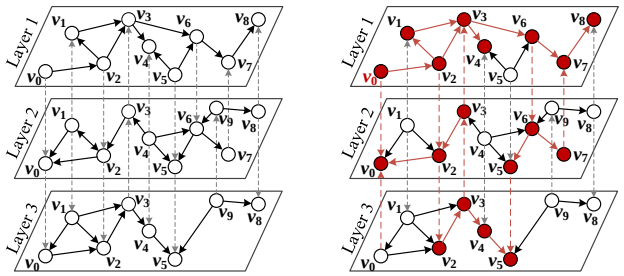
***Abstract*—Motivated by the fact that users in the real world often engage across multiple social networks simultaneously, we study the problem of influence maximization in multilayer social networks (MLIM), aiming to select a small set of nodes that maximizes the total influence spread across all layers. To this end, we introduce a hybrid propagation model that captures both the distinct diffusion patterns within each layer and the cross-layer influence diffusion. Based on this model, we formally define the MLIM problem and establish its NP-hardness, monotonicity, and submodularity. To address the MLIM problem, we first propose a greedy baseline Mlim-Greedy, achieving $(1-1/e)$ approximation. As exact influence computation in Mlim-Greedy is #P-hard, we propose STARIM, a scalable algorithm with multilayer-aware influence sampling that achieves a $(1-1/e-\epsilon)$ approximation. Despite its theoretical guarantees, STARIM remains prohibitive for large-scale graphs. Thus, we propose LGQIM, a learning-based framework that achieves superior efficiency through multilayer-aware representation learning and deep reinforcement learning for seed selection. Extensive experiments on nine real-world datasets demonstrate that (1) STARIM is up to 2 orders of magnitude faster than the baselines while yielding 10%-30% improvement in influence spread, and (2) LGQIM further achieves an average 10× speedup compared with STARIM while maintaining comparable influence spread.**

***Index Terms*—Influence maximization, Multilayer social networks, Theoretical analysis, Graph embedding, Deep reinforcement learning**

I. Introduction

Influence maximization (IM) [1] is a crucial task in social network analysis, with wide-ranging applications, such as viral marketing [2], diffusion control [3], social recommendation [4], among others. Given a graph G and a budget k, the objective of IM is to identify a set of k seed nodes that serve as the sources of information diffusion such that the expected number of influenced nodes is maximized under a specified diffusion model.

Most existing studies have focused on IM in single-layer networks [1], [5]. In practice, however, users typically participate in multiple social networks simultaneously, and information can flow across different layers of social interaction [6]. For example, GWI reports that the average social media user engages with 6.84 different platforms each month [7]. Recent research [8] indicates that companies engaging in product marketing on multilayer social networks achieve 2–5% higher total web sales compared to those restricted to a single layer. These highlight the necessity of studying IM in multilayer social networks [9], [10], which capture real-world diffusion dynamics and support applications such as product marketing and advertising on multilayer social networks [11]–[15].



(a) A multilayer social network G (b) Influence spread in G with S={$v_{0,1}$}

Fig. 1. A motivating example.

Existing diffusion models for IM in multilayer social networks [12], [13], [16]–[21] rely on the simplifying assumption of *uniform* propagation patterns across all layers. However, influence propagation rules varies significantly across different networks in reality. For example, information often diffuses through friendship networks on Facebook [22], while it mainly spreads via broadcasting mechanisms on Twitter [23]. Additionally, prior studies [12]–[14], [19] assume that each influenced user will automatically post information to all their friends on the other social networks. However, this assumption is often unrealistic in practice [21]. For example, users sharing entertainment content on TikTok rarely spread the same content to professional networks like LinkedIn. To address these limitations, we propose a *hybrid propagation model* to capture influence diffusion in multilayer social networks. In this hybrid model, each layer follows a platform-specific information propagation mechanism that accounts for the distinct interaction patterns within each network. Additionally, cross-layer propagation occurs probabilistically, where activation on one layer triggers influence attempts on other layers with certain probabilities rather than deterministically.

Multilayer Influence Maximization Problem. Based on the hybrid propagation model, we revisit the MultiLayer Influence Maximization (MLIM) problem, which is defined as follows. Given a multilayer graph $G(V, E, L)$ and a budget k, the MLIM problem aims to identify a seed node set $S \subset V$ with $|S| = k$ that maximizes the *expected influence spread*, i.e., the expected number of nodes influenced by S across all layers under the hybrid propagation model. Figure 1 illustrates an example of influence spread in a multilayer social network G under the hybrid propagation model. In this example, with budget $k = 1$, the optimal seed node set is $S = \{v_{0,1}\}$, where $v_{0,1}$ denotes node v_0 on layer 1, achieving a maximum influence spread of 18, which represents the total cross-platform exposure [24].

Challenges and Solutions. Existing works [12]–[14], [17] have proposed methods for IM in multilayer networks. However, these approaches face significant limitations in addressing the MLIM problem effectively and efficiently. Specifically, existing methods can be classified into three categories. First, greedy-based algorithms [14], [25] provide theoretical guarantees but rely on Monte Carlo simulations [1] to estimate influence spread, which is computationally expensive and impractical for large-scale networks. Second, heuristic-based methods [9], [12], [17], [26]–[29] are scalable but lack theoretical guarantees. Third, learning-based approaches [16], [18]–[20] primarily learn structural embeddings and rely on ranking-based selection, which capture only static feature rather than real propagation dynamics, leading to suboptimal solutions. Thus, there is a clear need to develop *new theoretical and learning-based algorithms* that effectively and efficiently address the MLIM problem.

We prove the **NP**-hardness, *monotonicity*, and *submodularity* of the MLIM problem under the hybrid propagation model. Given the inherent complexity of MLIM, we first propose theoretical-based algorithms. Specifically, we extend the classical greedy algorithm [1] and propose a baseline Mlim-Greedy, which greedily selects seed node to maximize the total expected influence spread across all layers. By leveraging the monotonicity and submodularity of MLIM, Mlim-Greedy provides a $(1-1/e)$-approximation. However, computing the exact influence spread over all layers for any node set in Mlim-Greedy is $\#\mathbf{P}$-hard, making it impractical for large networks.

To address this limitation, we propose a scalable trial-and-error algorithm STARIM. To this end, we first introduce a novel *Multilayer Reverse Influence Sampling* (*MRIS*) technique. Unlike the classical *RIS*, which is designed for single-layer networks and generates reverse reachable (RR) sets by treating all nodes equally, *MRIS* explicitly accounts for multilayer diffusion dynamics by introducing *multilayer reverse reachable* (M-RR) sets. Specifically, the M-RR set generation process in *MRIS* consists of three steps: (i) select a layer with probability proportional to the number of nodes in that layer, (ii) uniformly sample a node v from the chosen layer, and (iii) perform reverse propagation from v to include all nodes across all layers that can potentially activate it. Based on the M-RR sets generated by *MRIS*, we develop a novel unbiased estimator for influence spread in multilayer networks. Unlike traditional estimators that are designed for single-layer networks, our estimator employs a layer-aware approach that unbiasedly evaluates influence spread within each layer and then aggregate the total influence across all layers. Leveraging *MRIS* and our unbiased estimator, we develop STARIM that selects seed nodes with a $(1-1/e-\epsilon)$-approximation

While STARIM provides theoretical guarantees for the MLIM problem, its reliance on M-RR sets generation through multiple graph traversals incurs prohibitive time complexity on large-scale multilayer graphs, limiting its applicability to *real-time scenarios*. To overcome this limitation, we propose LGQIM, a learning-based framework that achieves superior efficiency by avoiding M-RR set sampling at test time through a two-stage learning approach: multilayer-aware influence prediction and reinforcement learning for seed selection.

First, LGQIM leverages supervised learning to predict node influence scores based on multilayer-aware structural features. Specifically, it learns specialized embeddings that capture both intra-layer and inter-layer structures while encoding the complex cross-layer propagation dynamics. This enables efficient influence prediction without repeated graph traversals. Second, recognizing that optimal seed selection is fundamentally a combinatorial problem where the marginal gain of each node depends on previously selected seeds, LGQIM formulates the process as a Markov Decision Process and leverages deep reinforcement learning to learn an optimal selection policy. In this formulation, the state represents the current seed node set, an action corresponds to adding a new seed node, and the reward reflects the marginal influence gain. By integrating the predicted scores as part of the state representation, the framework learns a policy to sequentially construct high-quality seed node sets. At test time, given a new multilayer graph and budget k, LGQIM first predicts influence scores for all nodes through the learned multilayer-aware embeddings, then sequentially constructs the seed node set according to the learned policy. This design enables LGQIM to achieve superior efficiency for real-time and large-scale applications while maintaining solution quality.

Contributions. Our contributions are summarized as follows.

- We introduce a hybrid propagation model for multilayer networks, formally define the MLIM problem, and prove it is **NP**-hard, monotone, and submodular.
- We propose a greedy algorithm Mlim-Greedy and its scalable version STARIM, achieving $(1-1/e)$ and $(1-1/e-\epsilon)$ approximations, respectively.
- We design LGQIM, a two-stage learning framework: multilayer-aware representation learning for influence prediction and reinforcement learning for seed selection.
- We conduct extensive experiments on nine real multilayer networks to demonstrate the effectiveness, efficiency, and scalability of the proposed algorithms.

Roadmap. Section II reviews related work. Section III defines the hybrid propagation model and MLIM problem. Section IV introduces the theoretical-based algorithms. Section V presents the learning-based framework. Section VI reports experimental results and Section VII concludes the paper.

II. RELATED WORK

Influence Maximization in single-layer networks. The Influence Maximization (IM) problem was first formulated as a combinatorial optimization task by Kempe et al. [1], focusing on the independent cascade (IC) and linear threshold (LT) models. They established the **NP**-hardness of the problem under both models and proposed a greedy algorithm with a $(1-1/e)$-approximation. Subsequent research has focused on developing efficient and scalable IM solutions [30]–[36] and exploring various variants of IM [3], [37]–[43].

Recent research has increasingly explored learning-based approaches to the single-layer IM problem. [44], [45] first

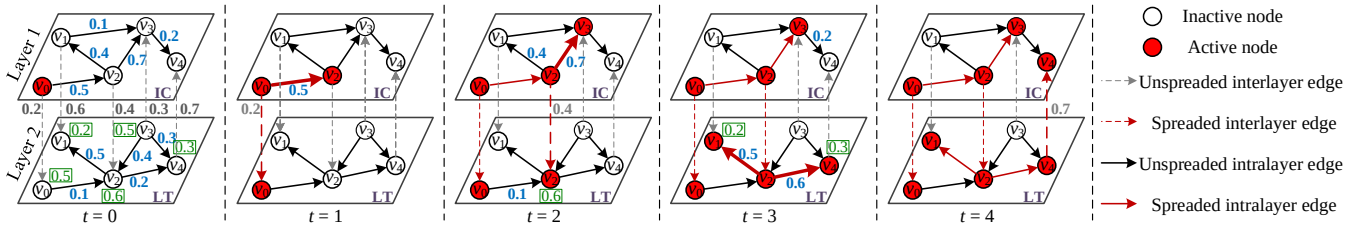


Fig. 2. Illustrating influence propagation under *hybrid propagation model*; green numbers besides nodes are activation thresholds θ, blue numbers on the edges are influence weights w

introduced reinforcement learning (RL) to IM, inspiring extensive follow-up research. Subsequent methods [46]–[49] leverage RL to learn optimal seed selection policies by maximizing cumulative influence spreads. Besides, several methods [50]–[53] utilize graph neural networks to encode individual influence into node embeddings for guiding seed selection. Moreover, GCOMB [54] addresses combinatorial optimization over large graphs by integrating supervised node embeddings with reinforcement learning for seed selection, whereas DeepIM [55] captures diverse information diffusion patterns in a data-driven and end-to-end manner for IM.

Influence Maximization in multilayer social networks. Recent research has extended the IM problem to multilayer networks. Existing approaches can be broadly categorized into three categories [10]: greedy-based methods, heuristic-based methods, and learning-based methods. **(1)** The greedy-based methods [14], [25] typically adapt traditional greedy algorithms to multilayer settings. For example, Kuhnle et al. [14] designed the KSN framework based on the ISF greedy strategy, performing seed selection via a two-phase process of layer-wise activation estimation and knapsack optimization. **(2)** The heuristic-based methods [9], [12], [17], [26]–[29] design heuristics based on network structural properties. For instance, Chowdary [17] proposed the CBIM method, which first detects communities in each layer, and then selects seed nodes from these communities using a quota-based strategy guided by the edge weight sum measure. **(3)** Recent work has also explored machine learning techniques [16], [18]–[20] for seed selection. For example, Keikha et al. [16] selected seed nodes using node feature vectors learned through network embedding. However, the greedy-based methods rely on Monte Carlo simulations, leading to high computational cost on large multilayer networks; the heuristic-based methods lack theoretical guarantees; the learning-based methods often rely on embedding-based ranking strategies that capture only static structural similarity rather than real propagation dynamics, leading to suboptimal seed set. *In contrast, we propose a scalable algorithm achieving $(1 - 1/e - \epsilon)$ approximation guarantee through a novel MRIS technique without expensive simulations, and a learning-based framework that captures real cross-layer propagation dynamics and optimizes seed selection through reinforcement learning.*

III. Preliminaries

In this section, we first introduce the hybrid propagation model in multilayer social networks. Based on it, we formally define and analyze the problem of Multilayer Influence Maximization (MLIM).

We consider a multilayer social network $G(V, E, L)$, where V, $E \subseteq V \times V \times L$, and $L = \{1, 2, \ldots, l\}$ denote the sets of nodes, edges, and layers, respectively. For each layer $i \in L$, the corresponding intralayer graph $G_i = (V_i, E_i) \subseteq G$ is modeled as a directed graph, where $V_i \subseteq V$ and $E_i \subseteq E$ denote the sets of nodes and edges within layer i. The set of interlayer edges between layers i and j $(i, j \in L, i \neq j)$ is denoted by $E_{i,j}$. Consequently, the overall node and edge sets are given by $V = \bigcup_{i \in L} V_i$ and $E = \bigcup_{i \in L} E_i \cup \bigcup_{i,j \in L} E_{i,j}$. For a directed edge $e = (u, v) \in E$, we refer to u as an in-neighbor of v and v as an out-neighbor of u. Accordingly, we denote the sets of in-neighbors and out-neighbors of v as $N_{in}(v)$ and $N_{out}(v)$, respectively. For clarity, when restricting to a specific layer $i \in L$, we use $N^i_{in}(v)$ and $N^i_{out}(v)$ to denote the sets of in-neighbors and out-neighbors of v within layer i. Each directed edge $e = (u, v) \in E$ is associated with an influence weight $w_{u,v} \in [0, 1]$, quantifying the influence of node u on node v. Note that, for any $i \in L$, both the node set V_i and the edge set E_i may differ across layers.

A. The Hybrid Propagation Model

Building upon the above settings, we introduce the hybrid propagation model. In this paper, we focus on two fundamental and widely adopted diffusion models: the independent cascade (IC) and linear threshold (LT) models [1]. *Note that the hybrid propagation model can incorporate other diffusion models.* Given a multilayer graph $G(V, E)$ and a seed set $S \subseteq V$ being active at timestamp 0, , influence propagation under the hybrid propagation model unfolds at each timestamp t as follows.

Propagation process under the hybrid propagation model. At the outset, each layer is assigned a specific propagation model, e.g., the IC or LT model. Both models simulate influence spread through an iterative process. The overall propagation process across intralayer and interlayer graphs is then described as follows.

- *Intralayer propagation process.* For a layer $i \in L$, (1) in the IC model, when a node u_i becomes active at timestamp $t \geq 0$, it has one chance to activate each inactive intralayer out-neighbor v_i at timestamp $t+1$ with probability w_{u_i,v_i}. (2) In the LT model, each node is assigned an activation threshold θ^i_v uniformly at random from $[0, 1]$. At timestamp $t > 0$, an inactive node v_i is activated iff the total influence weights from its active in-neighbors within the same layer exceeds its threshold, i.e., $\sum_{u_i \in N^i_{in}(v) \cap A_{t-1}} \geq \theta^i_v$, where

A_{t-1} denotes the set of active nodes at timestamp $t-1$. The incoming influence weights are normalized such that $\sum_{u_i \in N^i_{in}(v)} \leq 1$.

- *Interlayer propagation process.* When a node u_i in layer i becomes active at timestamp $t \geq 0$, it has a single chance to activate each inactive interlayer out-neighbor v_j (where $j \neq i$) with probability w_{u_i,v_j}.

Activated nodes remain active throughout the propagation, which terminates when no further nodes can be activated.

Example 1. *Figure 2 illustrates the influence propagation process under the hybrid propagation model. Let $v_{i,l}$ denote node v_i on layer l, and $v_{0,1}$ is the seed node. The influence unfolds as follows:*

- *$t = 0$: node $v_{0,1}$ becomes active.*
- *$t = 1$: nodes $v_{0,2}$ and $v_{2,1}$ are activated by $v_{0,1}$ with influence weights 0.5 and 0.2, respectively.*
- *$t = 2$: on layer 1, node $v_{3,1}$ is activated by $v_{2,1}$. On layer 2, although $v_{2,2}$ receives influence from $v_{0,2}$, it cannot be activated since $w_{v_{0,2},v_{2,2}} < \theta_{v_{2,2}} = 0.6$. However, $v_{2,2}$ becomes active through interlayer influence from $v_{2,1}$.*
- *$t = 3$: nodes $v_{3,2}$ and $v_{4,2}$ are activated by $v_{2,2}$.*
- *$t = 4$: node $v_{4,1}$ is activated by $v_{4,2}$. The propagation process then terminates with the active node sets: $\{v_{0,1}, v_{2,1}, v_{3,1}, v_{4,1}\}$ on layer 1 and $\{v_{0,2}, v_{1,2}, v_{2,2}, v_{4,2}\}$ on layer 2.*

B. Problem Definition

Based on the hybrid propagation model, we present the formal problem statement of MLIM. Let S be a set of seed nodes. For each layer graph G_i of the multilayer graph, the expected influence spread of S on G_i is captured by the set of nodes eventually activated in that layer under the hybrid propagation model, denoted as $I(G_i, S)$. Accordingly, the expected influence spread of S on G_i is defined as:

$$\sigma(S, G_i) = \mathbb{E}[|I(G_i, S)|]. \tag{1}$$

Definition 1. (Expected influence spread). *Given the expected influence spread $\sigma(S, G_i)$ of a seed node set S on each layer $G_i \subseteq G$, the expected influence spread of S on the multilayer graph G is defined as:*

$$\sigma(S, G) = \sum_{i \in L} \sigma(S, G_i), \tag{2}$$

which sums the expected influence spread over all layers to capture the *total cross-platform exposure*. This aligns with practical objectives where multiple user activations across platforms amplify overall effectiveness [24]. Notably, our framework can be extended to alternative influence metrics such as $\sigma(S, G) = \mathbb{E}[|\cup_{i \in L} I(G_i, S)|]$. Then, we formally define the MLIM problem as follows.

Problem 1. (MLIM). *Given a multilayer graph $G(V, E)$ and a budget k, the objective of* MLIM *is to find a seed node set $S \subset V$ of size k that maximizes the total expected influence spread. Formally:*

$$S := \arg \max_{S \subset V, |S| = k} \sigma(S, G). \tag{3}$$

Example 2. *Consider the multilayer graph G with two layers shown in Figure 2. Let $k = 1$. Among all seed sets of size 1, selecting $S = \{v_{0,1}\}$ achieves the maximal total expected influence spread, i.e., the largest total number of activated nodes, with $\sigma(\{v_{0,1}\}, G_1) = 4$ and $\sigma(\{v_{0,1}\}, G_2) = 4$, yielding a total of $\sigma(S, G) = \sum_{i \in \{1,2\}} \sigma(S, G_i) = 8$. Therefore,* MLIM *returns $S = \{v_{0,1}\}$.*

C. Problem Analyses

In this section, we prove that the MLIM problem is **NP**-hard and that computing $\sigma(S, G)$ exactly is $\#\mathbf{P}$-hard. We further show that the MLIM problem is *monotone* and *submodular* under the hybrid propagation model.

Theorem 1. *The* MLIM *problem is* **NP***-hard under the hybrid propagation model.*

Proof. It has been proved that the influence maximization problem in a single-layer graph under the classical IC or LT model is **NP**-hard [1]. We construct a special case of the MLIM problem as follows: (1) consider a multilayer graph that contains only a single layer i, i.e., $G = G_i$; and (2) set the propagation model on G_i to be the specific classical IC or LT model. Under this construction, MLIM reduces to the classical single-layer influence maximization problem. Therefore, MLIM is **NP**-hard. □

Theorem 2. *Given a multilayer graph $G(V, E)$ and a seed node set S, it is $\#\mathbf{P}$-hard to compute the exact value of $\sigma(S, G)$ under the hybrid propagation model.*

Proof. Computing the exact influence spread of a seed node set S on a single-layer graph, i.e., $\sigma(S, G_i)$, under the classical IC or LT model is known to be $\#\mathbf{P}$-hard [1], [56]. Since the hybrid propagation model generalizes the classical IC/LT models by allowing influence to propagate across multiple layers, any instance of IC/LT on a single layer can be regarded as a special case of the hybrid model. Moreover, the total expected influence spread is defined as the sum over all layers, i.e., $\sigma(S, G) = \sum_{i \in L} \sigma(S, G_i)$. Therefore, computing the exact influence spread $\sigma(S, G)$ under the hybrid propagation model is also $\#\mathbf{P}$-hard. □

Theorem 3. *Given a multilayer graph $G(V, E)$, the expected influence spread function $\sigma(S, G)$ is monotone and submodular w.r.t. S under the hybrid propagation model.*

Proof. **(1) Monotonicity:** For any two seed node sets $S_1 \subseteq S_2 \subseteq V$, the expected influence spread on each layer satisfies $\sigma(S_1, G_i) \leq \sigma(S_2, G_i)$, due to the monotonicity property of the expected influence spread under both IC and LT models [1]. Since the total expected influence spread is the sum over all layers, we have $\sigma(S_1, G) = \sum_{G_i \subseteq G} \sigma(S_1, G_i) \leq \sum_{G_i \subseteq G} \sigma(S_2, G_i) = \sigma(S_2, G)$. This establishes the monotonicity of $\sigma(S, G)$.

(2) Submodularity: Let $S_1 \subseteq S_2 \subseteq V$ and $v \in V \setminus S_2$. For each layer $i \in L$, $\sigma(S, G_i)$ is submodular, due to the submodularity property of the expected influence spread under both IC and LT models [1], i.e., $\sigma(S_1 \cup \{v\}, G_i) - \sigma(S_1, G_i) \geq$

Algorithm 1 Mlim-Greedy

Input: $G(V, E)$, k
Output: S
1: $S \leftarrow \emptyset$;
2: $\forall v \in V$, $\Delta(v|S) \leftarrow 0$;
3: **while** $|S| < k$ **do**
4: **for** $\forall v \in V \setminus S$ **do**
5: $\Delta(v|S) \leftarrow \sum_{i \in L}(\sigma(S \cup \{v\}, G_i) - \sigma(S, G_i))$;
6: $v^* \leftarrow \arg\max_{v \in V \setminus S} \Delta(v|S)$;
7: $S \leftarrow S \cup \{v^*\}$;
8: **Return** S.

$\sigma(S_2 \cup \{v\}, G_i) - \sigma(S_2, G_i)$. Since the sum of nonnegative submodular functions is also submodular, the overall influence spread $\sigma(S, G) = \sum_{G_i \subseteq G} \sigma(S, G_i)$ is submodular. Thus, $\sigma(S_1 \cup \{v\}, G) - \sigma(S_1, G) \geq \sigma(S_2 \cup \{v\}, G) - \sigma(S_2, G)$. Therefore, $\sigma(S, G)$ is submodular w.r.t. S. □

IV. Theoretical Algorithms

In this section, we propose Mlim-Greedy for the MLIM problem with a $(1 - 1/e)$ approximation, and its scalable version STARIM achieving a $(1 - 1/e - \epsilon)$ approximation.

A. Mlim-Greedy Algorithm

Since the MLIM problem is *monotone* and *submodular* under the hybrid propagation model, we propose a greedy algorithm called Mlim-Greedy. The algorithm follows the classical greedy framework [1], iteratively selecting the node that maximizes the total influence spread across all layers. Algorithm 1 outlines the pseudo-code of Mlim-Greedy. Initially, it initializes an empty seed node set S, and for each node $v \in V$, it sets the influence marginal gain $\Delta(v|S) = 0$ (Lines 1–2). In each iteration (Lines 3–7), Mlim-Greedy computes the influence marginal loss $\Delta(v|S)$ for all nodes in $V \setminus S$ (Lines 4–5), selects the node with the maximum marginal gain in expected spread over all layer (Line 6), and adds it to B (Line 7). The process continues until k seed nodes are selected. Finally, the algorithm returns the seed node set S (Line 8). The approximation of Mlim-Greedy is guaranteed by Lemma 1.

Lemma 1. *For the* MLIM *problem, let S^* denote the optimal solution to our problem. The solution S returned by Mlim-Greedy satisfies*[1]*:*

$$\sigma(S, G) \geq (1 - 1/e) \cdot \sigma(S^*, G). \tag{4}$$

B. STARIM Algorithm

Mlim-Greedy involves numerous computations of influence spread to determine the seed node set. However, computing the exact influence spread $\sigma(S, G)$ for any node set on a multilayer graph under the hybrid propagation model is $\#\mathbf{P}$-hard, as established in Theorem 2. Therefore, we employ approximation methods to estimate the value of $\sigma(S, G)$.

Recent studies have adopted simulation-based methods, such as Monte Carlo (MC) simulations [1], and advanced sampling-based methods, such as Reverse Influence Sampling (RIS) [36], for influence spread estimation. Although RIS significantly improves the efficiency of influence spread estimation compared to MC simulations, it faces limitations in our setting. In multilayer graphs, each layer follows a distinct influence propagation pattern, and influence can diffuse across layers. However, existing RIS methods are designed for single-layer graphs under a uniform propagation pattern without considering cross-layer diffusion, making them inapplicable to multilayer graphs. Consequently, classical unbiased estimators developed for single-layer graphs become ineffective. To this end, in this paper, we propose a new multilayer reverse influence sampling method to estimate the influence spread.

Multilayer Reverse Influence Sampling. To effectively estimate the expected influence spread $\sigma(S, G)$ on multilayer graphs for any seed node set S, we propose the *Multilayer Reverse Influence Sampling (MRIS)* method. Unlike classical RIS, which treats all the nodes equally and uniformly samples nodes across the entire graph when generating a reverse reachable (RR) set, multilayer graphs require a different treatment. Because the number of nodes varies significantly across layers, uniform sampling introduces bias: layers with fewer nodes are oversampled, while those with more nodes are undersampled, leading to an imbalanced probability distribution. To overcome this issue, we introduce the concept of a *Multilayer Reverse Reachable* (M-RR) set under the hybrid propagation model. The generation process of M-RR sets is described as follows.

Under the hybrid propagation model, a random M-RR set R on G can be generated in three steps:

1) Select a layer $i \in L$ with probability $p = \frac{n_i}{n}$, where n_i denotes the number of nodes in layer i and $n = \sum_{i \in L} n_i$ represents the total number of nodes in G;
2) Select a node $v \in V_i$ uniformly at random from the chosen layer i;
3) Starting from v, construct a sample set R by performing reverse propagation such that R contains all nodes that can activate v (including v).

The reverse propagation proceeds as follows: (i) for intralayer edges under the IC model, we conduct a reverse stochastic BFS. For each encountered node $u \in V_i$, examine its in-neighbors $a \in N^i_{in}(u)$ within layer $i \in L$. Each edge $e = (a, u)$ is traversed with probability $w_{a,u}$ and ignored with probability $1 - w_{a,u}$, provided that a has not been visited; (ii) for intralayer edges under the LT model, we conduct a reverse random walk. For each encountered node u in layer $i \in L$, the walk stops at u with probability $1 - \sum_{a \in N^i_{in}(u)} w_{a,u}$. Otherwise, select an in-neighbor $b \in N^i_{in}(u)$ with probability $\sum_{a \in N^i_{in}(v)} w_{a,u}$, and continue the walk to b; and (iii) for interlayer edges, when a node u has incoming interlayer edges from nodes a in other layers, each such edge $e = (a, u)$ is traversed with probability $w_{a,u}$ and ignored with $1 - w_{a,u}$. If the traversal succeeds and a has not been visited, the reverse propagation continues from a in its respective layer according to the corresponding intralayer propagation model (IC or LT). The final M-RR set R consists of all nodes visited during this reverse propagation process.

Based on the generated M-RR sets, we need to design unbiased estimation methods to estimate the influence spread of a given seed node set across multilayer graphs. Traditional

[1]Due the limited space, the omitted proof are moved to [57].

Algorithm 2 STARIM

Input: $G(V, E)$, k, ϵ, δ
Output: S
1: $S \leftarrow \emptyset$; $\theta_1 \leftarrow \max\{n_1, n_2, \ldots, n_{|L|}\}$; $i \leftarrow 1$;
2: **while true do**
3: Generate $\mathcal{R}_1$ and $\mathcal{R}_2$ with $|\mathcal{R}_1| = |\mathcal{R}_2| = \theta_i$;
4: $S \leftarrow$ NodeSelection$(\mathcal{R}_1, k)$;
5: Compute $f^{\mathcal{R}_1}(S, G)$ and $f^{\mathcal{R}_2}(S, G)$;
6: $\mu \leftarrow f^{\mathcal{R}_1}(S, G)/f^{\mathcal{R}_2}(S, G)$;
7: $\epsilon_1 \leftarrow \frac{(\epsilon_1+1)(\epsilon_1+2)}{\epsilon_1^2} = f^{\mathcal{R}_2}(S, G)/\ln(\frac{5 \cdot i^2}{\delta}) \cdot \min_{\mathcal{R}_i \subseteq \mathcal{R}_2}\{\frac{|\mathcal{R}_i|}{n_i}\}$;
8: $\epsilon_2 \leftarrow \frac{2\epsilon_1+2}{\epsilon_2^2} = f^{\mathcal{R}_2}(S, G)/\ln(\frac{5 \cdot i^2}{\delta}) \cdot \min_{\mathcal{R}_i \subseteq \mathcal{R}_1}\{\frac{|\mathcal{R}_i|}{n_i}\}$;
9: **if** $0 < \mu \leq (\frac{1-1/e}{1-1/e-\epsilon} \cdot \frac{1-\epsilon_2}{1+\epsilon_1})$, ϵ_1, $\epsilon_2 \in (0, 1)$ **then break**;
10: **if** $\min_{R_i \in \mathcal{R}_1}\{\frac{|R_i|}{n_i}\} \geq (8 + 2\epsilon)(1 + \epsilon_1)\frac{\ln \frac{6}{\delta} + n \ln 2}{\epsilon^2 \cdot f^{\mathcal{R}_2}(S,G)}$ **then**
11: **break**;
12: $i \leftarrow i + 1$; $\theta_i \leftarrow 2\theta_i$;
13: **Return** S.

Algorithm 3 NodeSelection

Input: $\mathcal{R}_1$, k
Output: S
1: $S \leftarrow \emptyset$;
2: Let $\mathcal{R}_1^i$ be the set of M-RR sets with sampled source nodes on layer i;
3: **while** $|S| < k$ **do**
4: **for** each $v \in V \setminus S$ **do**
5: $\Delta_{\mathcal{R}_1}(v) \leftarrow \sum_{i \in L} \frac{n_i}{|\mathcal{R}_1^i|} \times |\{R \in \mathcal{R}_1^i : v \in R\}|$;
6: $v^* \leftarrow \arg\max_{v \in V \setminus S} \Delta_{\mathcal{R}_1}(v)$;
7: $S \leftarrow S \cup \{v^*\}$, $\Delta_{\mathcal{R}_1}(v^*) \leftarrow 0$;
8: **for** each $R \in \mathcal{R}_1$ with $v^* \in R$ **do**
9: **for** each $u \in R$ **do**
10: $\Delta_{\mathcal{R}_1}(u) \leftarrow \Delta_{\mathcal{R}_1}(u) - \frac{n_i}{|\mathcal{R}_1^i|}$;
11: Remove from $\mathcal{R}_1$ all M-RR sets covered by v^*;
12: **Return** S.

unbiased estimation methods are primarily tailored for single-layer graphs and thus cannot capture the cross-layer diffusion dynamics in multilayer networks. Thus, we develop a novel unbiased estimator to estimate the influence spread over the entire multilayer graph.

Given a seed node set S and a random M-RR set R, let $v \in R$ denote the sampled source node of R, which resides on layer $i \in L$. We define a random variable $\Lambda_R^i(S, G)$ such that $\Lambda_R^i(S, G) = 1$ iff S intersects R, i.e., $S \cap R \neq \emptyset$. Otherwise, $\Lambda_R^i(S, G) = 0$. This indicates whether S can activate node v. Since $\sigma(S, G) = \sum_{i \in L} \sigma(S, G_i)$, we can define, given a set $\mathcal{R}$ of random M-RR sets, the following unbiased estimators: $f^{\mathcal{R}}(S, G_i)$ as an unbiased estimator of $\sigma(S, G_i)$ for each layer i, and $f^{\mathcal{R}}(S, G)$ as an unbiased estimator of $\sigma(S, G)$ over the entire multilayer graph. Formally,

$$f^{\mathcal{R}}(S, G) = \sum_{i \in L} f^{\mathcal{R}}(S, G_i) = \sum_{i \in L} \frac{n_i}{|\mathcal{R}_i|} \cdot \sum_{R \in \mathcal{R}_i} \Lambda_R^i(S, G), \quad (5)$$

where $\mathcal{R}_i \subseteq \mathcal{R}$ denotes the set of M-RR sets whose sampled source nodes are on layer i, and n_i is the total number of nodes on layer i.

STARIM. Based on the discussions above, we propose the *Scalable mulTilAyeR Influence Maximization (STARIM)* algorithm. STARIM initializes with an empty seed set S, leverages the MRIS method to estimate the influence spread of nodes, and iteratively selects the node with the maximum marginal gain. The returned S has a theoretical guarantee with sufficient M-RR sets. A key challenge is determining the sample size $|\mathcal{R}|$ required for approximation guarantee without excessive computational cost. To address this, STARIM adopts an adaptive trial-and-error strategy [32]: during M-RR set generation, the sample size is doubled incrementally, and approximation is monitored. STARIM terminates once either the approximation ratio between two independently estimated influence spreads meets a predefined bound, or the required number of M-RR sets reaches the theoretical threshold. Detailed derivations of these conditions are provided in the Appendix [57].

Algorithm 2 presents the pseudo-code of STARIM. Initially, two sets $\mathcal{R}_1$ and $\mathcal{R}_2$ of M-RR sets are generated with $|\mathcal{R}_1| = |\mathcal{R}_2| = \max\{n_1, n_2, \cdots, n_{|L|}\}$ (Line 3), respectively. It then greedily selects the optimal seed set S based on $\mathcal{R}_1$ via the function NodeSelection (Line 4), and verifies its quality using $\mathcal{R}_2$ (Lines 5-10) which is independent of $\mathcal{R}_1$. If the influence spread estimated from $\mathcal{R}_2$ is much smaller than that from $\mathcal{R}_1$, $\mathcal{R}_1$ over-estimates S's influence spread. In this case, STARIM discards S, doubles the sizes of $\mathcal{R}_1$ and $\mathcal{R}_2$ (Line 11), and repeats the process. The iteration stops when either **(1)** $\mu = f^{\mathcal{R}_1}(S, G)/f^{\mathcal{R}_2}(S, G)$ satisfies the condition $0 < \mu \leq (\frac{1-1/e}{1-1/e-\epsilon} \cdot \frac{1-\epsilon_2}{1+\epsilon_1})$ (Line 9), or **(2)** the minimum ratio of sampled M-RR sets per layer meets the criterion: $\min_{R_i \in \mathcal{R}_1}\{\frac{|R_i|}{n_i}\} \geq (8 + 2\epsilon)(1 + \epsilon_1)\frac{\ln \frac{6}{\delta} + n \ln 2}{\epsilon^2 \cdot f^{\mathcal{R}_2}(S,G)}$ (Line 10). STARIM finally returns S (Line 13).

Algorithm 3 presents the pseudo-code of NodeSelection, which greedily selects the seed set S. It initializes S as an empty set (Line 1) and defines $\mathcal{R}_1^i$ as the set of M-RR sets whose sampled source nodes are on layer i (Line 2). In each iteration, it computes the marginal gain of each node v as the sum of $n_i/|\mathcal{R}_1^i|$ over all M-RR sets R in $\mathcal{R}_1$ that contain v (Lines 4–5), selects the node v^* with the maximum marginal gain (Line 6), adds v^* to S, and sets its marginal gain to zero (Line 7). For each node in the M-RR sets covered by v^*, NodeSelection updates their marginal gains accordingly (Lines 8–10), and subsequently removes all M-RR sets covered by v^* from $\mathcal{R}_1$ (Line 11). Finally, NodeSelection returns the seed node set S (Line 12).

In each round of STARIM, the estimations $f^{\mathcal{R}1}(S^*, G)$ and $f^{\mathcal{R}2}(S, G)$ serve as concentration bounds with high probability, as established in Lemma 2.

Lemma 2. *With probability at least $1 - \frac{2\delta}{3}$, for each iteration of Algorithm 2, where $\epsilon_1, \epsilon_2, \mu > 0$, we have*

$$f^{\mathcal{R}_2}(S, G) \leq (1 + \epsilon_1)\,\sigma(S, G), \quad (6)$$
$$f^{\mathcal{R}_1}(S^*, G) \geq (1 - \epsilon_2)\,\sigma(S^*, G) \quad (7)$$

Theorem 4 shows the approximation guarantee provided by Algorithm 2, where S^* denotes the optimal solution and $\delta, \epsilon \in (0, 1)$ are user-defined parameters.

Theorem 4. (Theoretical guarantee). *Let $\sigma(S, G)$ denote the influence spread of S on G, and ϵ be the approximation factor for influence estimation by MRIS method. With a*

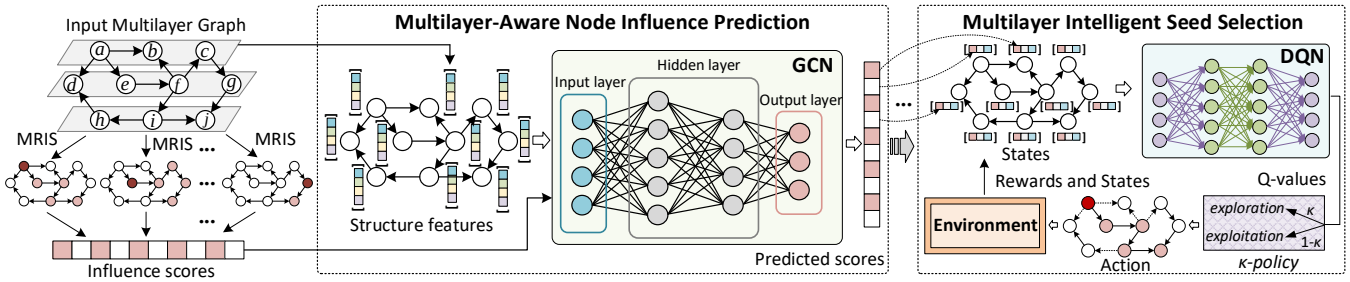


Fig. 3. Workflow of the training phase of LGQIM

probability of at least $(1-\delta)$ *for* $\forall\ \delta \in (0,1)$*, the solution* S *returned by* STARIM *satisfies:*

$$\sigma(S,G) \geq (1-1/e-\epsilon)\cdot\sigma(S^*,G). \tag{8}$$

Theorem 5. (Time complexity). *The expected time complexity of Algorithm 2 is* $O(\frac{(\ln 1/\delta+\ln n)\cdot m\cdot\sigma(\{v^*\})}{\epsilon^2})$*, where* v^* *is the node in* G *with the largest expected spread.*

V. Learning-based Framework

In this section, we propose LGQIM, a two-stage learning framework that combines multilayer-aware node influence prediction with reinforcement learning-based seed selection.

A. Framework Overview

While STARIM effectively tackle the MLIM problem with theoretical guarantees, its time complexity can be prohibitive for real-time or large-scale multilayer graphs due to the need for multiple graph traversals for generating M-RR sets. To address this limitation, we propose LGQIM, a learning-based framework that achieves superior efficiency by avoiding M-RR set sampling. Figure 3 illustrates the training workflow of LGQIM, which consists of three main phases: **(1) Training Data Generation.** We first apply the *MRIS* method to compute the influence score of each node, i.e., the influence spread when each node is selected as the sole seed node. These scores serve as labeled data for supervising the subsequent learning process. **(2) Multilayer-Aware Node Influence Prediction.** Directly computing node influence scores at test time would require expensive online M-RR set sampling. To avoid this computational bottleneck, we employ graph embedding learning to efficiently predict influence scores through supervised learning. The learned embeddings capture both intra-layer and inter-layer structural features that encode the complex propagation dynamics across multiple network layers. The predicted scores are then utilized as input representations for the subsequent reinforcement learning phase. **(3) Multilayer Intelligent Seed Selection.** Since selecting the optimal seed set requires addressing the combinatorial nature of the problem, we formulate this as a sequential decision-making problem solved through deep reinforcement learning. The seed selection process is modeled as a Markov Decision Process, where the state represents the current seed set, an action corresponds to selecting a new node, and the reward reflects the marginal influence gain. This approach learns an optimal policy that sequentially selects seed nodes, effectively balancing exploration of new candidates with exploitation of high-quality nodes to construct the final seed set. **Test Phase.** Given a new multilayer graph and a budget k, we first leverage the learned multilayer-aware embeddings to predict influence scores for all nodes without M-RR set generation. These scores then guide the sequential deep reinforcement learning process to select the final seed set based on the learned policy.

B. Multilayer-Aware Node Influence Prediction

We first train a Graph Convolutional Network (GCN) [58] model in a supervised manner to predict the influence score of each node in a given multilayer graph G. The primary objective is to learn node embeddings that capture both intra-layer and inter-layer structural features, enabling accurate influence prediction directly from the multilayer graph structure without M-RR set sampling. To generate training data, we first apply the *MRIS* method on G to compute the influence score of each node $v \in V$, i.e., the influence spread when v is selected as the sole seed node. This yields a set of influence scores $\mathcal{E} = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_{|V|}\}$ with $\epsilon_v \in \mathbb{R}^+$. The graph structure G together with the scores $\mathcal{E}$ are then used to train the GCN and learn its parameters Θ_S. Once trained, the model can efficiently predict influence scores for nodes in new graphs based on Θ_S, greatly reducing computation cost and enabling online analysis. In what follows, we present a few key components of the GCN training process.

Node feature construction. We embed each node into a feature vector that captures its structural properties in the graph. Specifically, for every node $v \in V$, the initial representation $\boldsymbol{x}_v$ is defined as a 4-dimensional vector:

$$\boldsymbol{x}_v = \left[|N_{out}(v)|,\ |N_{in}(v)|,\ \textstyle\sum_{u\in N_{out}(v)\cap V_i} w_{v,u},\ \sum_{u\in N_{out}(v)\setminus V_i} w_{v,u}\right] \tag{9}$$

Here, V_i denotes the set of nodes on the layer that contains node v. The four features in $\boldsymbol{x}_v$ are interpreted as follows: $|N_{out}(v)|$ and $|N_{in}(v)|$ denote the numbers of out-neighbors and in-neighbors of v in the entire graph, respectively, $\sum_{u\in N_{out}(v)\cap V_i} w_{v,u}$ is the total weight of edges from v to nodes in the same layer, and $\sum_{u\in N_{out}(v)\setminus V_i} w_{v,u}$ is the total weight of edges from v to nodes in other layers. Together, these features provide a comprehensive representation of each node's influence potential, combining connectivity, influence reception, and propagation ability both within and across layers. We denote the feature vectors of all nodes in V collectively as a matrix:

$$\boldsymbol{X} = [\boldsymbol{x}_v]_{v\in V} \in \mathbb{R}^{|V|\times 4}, \tag{10}$$

Algorithm 4 Supervised GCN Training

Input: $G(V, E)$, ground-truth scores $\mathcal{E} = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_{|V|}\}$, node features $\boldsymbol{X}$, depth K, learning rate α, max epochs T
Output: Trained parameters $\Theta_S = \{W^{(l)}\}_{l=1}^{K} \cup \{\boldsymbol{w}\}$
1: Initialize weight matrices $\{W^{(l)}\}_{l=1}^{K}$ and weight vector $\boldsymbol{w}$;
2: **for** epoch $t = 1$ to T **do** //Forward propagation
3: $H_v^{(0)} \leftarrow \boldsymbol{x}_v,\ \forall v \in V$;
4: **for** each layer $l = 1$ to K **do**
5: **for** each node $v \in V$ **do**
6: $H_{N(v)}^{(l)} \leftarrow \textsc{MeanPool}(\{H_u^{(l-1)} : u \in N_{out}(v)\})$;
7: $H_v^{(l)} \leftarrow \textit{ReLU}(W^{(l)} \cdot \textsc{Concat}(H_{N(v)}^{(l)}, H_v^{(l-1)}))$;
8: **for** each node $v \in V$ **do**
9: $\hat{\epsilon}_v \leftarrow \boldsymbol{w}^\top \cdot H_v^{(K)}$;
10: $\mathcal{L}(\Theta_S) \leftarrow \frac{1}{|V|}\sum_{v \in V}(\hat{\epsilon}_v - \epsilon_v)^2$; //Loss computation
11: $\Theta_S \leftarrow \Theta_S - \alpha \nabla_{\Theta_S} \mathcal{L}(\Theta_S)$;
12: **Return** $\Theta_S = \{W^{(l)}\}_{l=1}^{K} \cup \{\boldsymbol{w}\}$.

where each row corresponds to the 4-dimensional feature vector of a node. This feature matrix $\boldsymbol{X}$ serves as the input to the GCN.

GCN architecture. The GCN refines node representations by iteratively aggregating information from their neighbors. At the $(l+1)$-th layer, the node embeddings are updated as:

$$H^{(l+1)} = \phi\Big(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\Big), \tag{11}$$

where $\tilde{A} = A + I$ is the adjacency matrix with self-loops, $\tilde{D}$ is the corresponding degree matrix, $W^{(l)}$ is the learnable weight matrix for layer l, and ϕ is a non-linear activation function. The input to the first layer is initialized as $H^{(0)} = \boldsymbol{X}$.

Supervised GCN training. Given the input node feature matrix $\boldsymbol{X}$ and the GCN architecture described above, the objective is to train the network to predict the influence score of each node in the graph. The ground-truth influence scores are denoted by $\mathcal{E} = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_{|V|}\}$, while the predicted scores produced by the GCN are represented as $\hat{\mathcal{E}} = \{\hat{\epsilon}_1, \hat{\epsilon}_2, \ldots, \hat{\epsilon}_{|V|}\}$. The model parameters Θ_S are learned by minimizing the mean squared error (MSE) between the predicted and true scores, which is formally defined as:

$$\mathcal{L}(\Theta_S) = \frac{1}{|V|}\sum\nolimits_{v \in V}(\hat{\epsilon}_v - \epsilon_v)^2. \tag{12}$$

Building on the supervised training framework described above, Algorithm 4 presents the pseudo-code for the complete GCN training process. The algorithm begins by randomly initializing all learnable parameters, including weight matrices $\{W^{(l)}\}_{l=1}^{K}$ and the final weight vector $\boldsymbol{w}$ (Line 1). The training process then iterates over T epochs. (Line 2). In each epoch, the algorithm performs forward propagation by first initializing each node v with its feature vector, $H_v^{(0)} = \boldsymbol{x}_v$ (Line 3). The forward pass consists of an outer loop that iterates over the depth of the model, from 1 to K (Line 4), and an inner loop that traverses all nodes $v \in V$ (Line 5). For each node v, we first aggregate the current representations of its out-neighbors using a MEANPOOL layer (Line 6). The aggregated vector is then concatenated with v's own representation and passed through a fully connected layer with *ReLU* activation (Line 7). The resulting vector becomes the input for the next iteration of the outer loop. Intuitively, with each iteration of the outer loop, a node progressively incorporates information from neighbors at increasing depths, thereby capturing both local and higher-order structural patterns. During this process, $\{W^{(l)}\}_{l=1}^{K}$ is used in the hidden layers to propagate information across different depths of the model. After the forward propagation, the algorithm generates predicted influence scores by passing the final layer representations through an additional fully connected layer, yielding $\hat{\epsilon}_v$ for each node $v \in V$ (Lines 8-9). The training process then computes the mean squared error loss between predicted and ground-truth scores (Line 10) and updates all parameters using gradient descent (Line 11). This supervised learning process continues until convergence or the maximum number of epochs is reached, ultimately returning the trained parameters $\Theta_S = \{W^{(l)}\}_{l=1}^{K} \cup \{\boldsymbol{w}\}$.

Time Complexity. Algorithm 4 has a time complexity of $O(T \times K \times (|E| + |V| \times d \times h))$, where T is the number of training epochs, K is the network depth, $|E|$ and $|V|$ denote the numbers of edges and nodes, respectively, d is the input feature dimension, and h is the hidden layer dimension.

C. Multilayer Intelligent Seed Selection

After the GCN has predicted the individual influence scores of nodes, we reformulate the MLIM problem as a sequential decision-making task and employ a Q-learning network [59]. The objective is to learn an optimal policy (i.e., a sequence of actions) for selecting seed nodes. The Q-learning network provides an end-to-end framework that connects the agent's perception to its actions, enabling it to automatically extract task-relevant features. In our setting, the agent is the machine that learns to select seed nodes optimally within an uncertain environment, namely the network. The entire decision-making process is modeled as a Markov Decision Process (MDP): at each time step t, the agent observes the state S_t (i.e., the current set of selected nodes), performs an action A_t (i.e., selecting a new node), gets a reward R_{t+1}, and moves to the next state S_{t+1}. This interaction continues until an episode ends. Specifically, for a given set of selected nodes S and a candidate node $v \notin S$, we estimate the n-step reward $Q_n(S, v)$ of adding v to S using the surrogate function $Q'_n(S, v; \Theta_Q)$.

We formulate the Q-learning task in terms of state space, action, reward, and policy with the input consists of the set of nodes along with their predicted influence scores.

State space. At time step t, the state S_t represents the current configuration, where some nodes have already been selected into the seed set and others remain unselected. The process terminates at the final state S_k, where exactly k nodes have been selected. The state space thus characterizes the state of the system at any time step t in terms of the partially constructed solution S_t and the corresponding set of candidate nodes $C_t = V \setminus S_t$. For each candidate node $v \in C_t$, we define its feature representation as a 3-dimensional vector:

$$\boldsymbol{\lambda}_v = [\hat{\epsilon}_v, \rho(v, S_t), \mathbf{1}\big\{|\{i : v \in V_i\}| > 1\big\}], \tag{13}$$

Here, $\hat{\epsilon}_v$ denotes the predicted influence score of v, reflecting its individual quality. Following [54], the term $\rho(v, S_t) = |N_{out}(v) \setminus \cup_{u \in S_t} N_{out}(u)|$ measures the number of out-neighbors of v not yet influenced by the current seed set, thereby capturing v's marginal contribution to influence propagation. The indicator function $\mathbf{1}|i : v \in V_i| > 1$ specifies whether v is an overlap node, i.e., one that belongs to multiple layers (1 if true, 0 otherwise). The representation of the candidate set C_t is then obtained by applying MAXPOOL over $\{\boldsymbol{\lambda}_v \mid v \in C_t\}$, denoted by $\boldsymbol{\lambda}_{C_t}$. $\boldsymbol{\lambda}_{S_t}$ is defined analogously as well. We adopt MAXPOOL as the aggregation function since it highlights the most promising candidate, effectively guiding the agent toward high-quality actions.

Action and reward. At each time step t, the action A_t corresponds to selecting a node $v \in C_t$ and adding it to the current seed node set S_t. Once the action is taken, the environment transitions to a new state S_{t+1} and returns a reward $R_{t+1} \in \mathbb{R}$. The reward function is defined as the marginal influence gain obtained by adding v, namely,

$$R_{t+1}(S_t, A_t = v) = \sigma(S_{t+1}) - \sigma(S_t) = \sigma(S_t \cup \{v\}) - \sigma(S_t). \quad (14)$$

Optimal policy. The objective of Q-learning is to derive the optimal policy $\pi^* = (A_1, A_2, \ldots, A_k)$ that maximizes the expected cumulative reward, defined as $\mathbb{E}[\sum_{i=1}^{k} R(S_i, A_i)]$. In our setting, this corresponds to identifying the k seed nodes that yield the largest overall influence spread. The trained Q-learning network implicitly encodes this policy by estimating the n-step reward for each candidate action. Formally, the policy selects the node with the highest predicted value:

$$\pi^*(v|S_t) = \arg\max_{v \in C_t} Q'_n(S_t, v; \Theta_Q). \quad (15)$$

Q-learning network training. We adopt the Deep Q-Network (DQN) framework [60], [61] to learn the Q-function $Q(S, v; \Theta_Q)$, which is implemented as a deep neural network over graph embeddings. To enhance stability and efficiency, we employ experience replay: past transitions are stored and randomly sampled for off-policy updates, thereby mitigating correlations between consecutive experiences. Q-learning updates parameters in a single episode via Adam optimizer [62] to minimize the squared loss, which is formally defined as:

$$\mathcal{L}(\Theta_Q) = \mathbb{E}\Big[(\gamma \cdot \max_{v \in V} Q'_n(S_{t+n}, v; \Theta_Q) + \sum_{i=0}^{n-1} R(S_{t+i}, v_{t+i}) - Q'_n(S_t, v; \Theta_Q))^2\Big] \quad (16)$$

where γ is the decaying factor, which balances the importance of immediate reward with the predicted n-step future reward.

The training process is outlined in Algorithm 5. The algorithm begins by randomly initializing the Q-network parameters Θ_Q and the experience replay buffer $\mathcal{J}$ (Lines 1-2). During training, since the Q-network has not yet been sufficiently optimized, we adopt a κ-greedy strategy to balance exploration and exploitation: with probability κ, a random node is chosen from the candidate set C_t; otherwise, the node with the maximum predicted n-step reward is selected

Algorithm 5 Q-learning Network Training

Input: $G(V, E)$, node scores $\mathcal{E}$, budget k, experience replay memory $\mathcal{J}$ with capacity N, number of episodes E_p, n-step parameter n, discount factor γ, batch size B
Output: Learned parameter set Θ_Q

1: Initialize Q-network parameters Θ_Q randomly;
2: Initialize experience replay buffer $\mathcal{J} \leftarrow \emptyset$;
3: **for** episode $e_p \leftarrow 1$ to E_p **do**
4: Initialize state $S_1 = \emptyset$;
5: $\kappa \leftarrow \max\{0.1, 0.8^{e_p}\}$;
6: **for** step $t \leftarrow 1$ to k **do**
7: $C_t \leftarrow V \setminus S_t$;
8: $v_t \leftarrow \begin{cases} \text{random node } v \in C_t & \text{w.p. } \kappa \\ \text{argmax}_{v \in C_t} Q'_n(S_t, v; \Theta_Q) & \text{w.p. } 1-\kappa \end{cases}$;
9: Compute reward $R_t = \sigma(S_t \cup \{v_t\}) - \sigma(S_t)$;
10: $S_{t+1} \leftarrow S_t \cup \{v_t\}$;
11: **if** $t \geq n$ **then**
12: $R_{sum} \leftarrow \sum_{i=t-n+1}^{t} R_i$;
13: Add tuple $\langle S_{t-n+1}, v_{t-n+1}, R_{sum}, S_{t+1} \rangle$ to $\mathcal{J}$;
14: **if** $|\mathcal{J}| \geq B$ **then**
15: Sample random batch $\mathcal{B}$ of size B from $\mathcal{J}$;
16: **for** each transition $(S, v, R_{sum}, S') \in \mathcal{B}$ **do**
17: $y \leftarrow R_{sum} + \gamma \cdot \max_{v' \in V \setminus S'} Q'_n(S', v'; \Theta_Q)$;
18: $\mathcal{L}(\Theta_Q) \leftarrow \frac{1}{B} \sum_{(S,v,R_{sum},S') \in \mathcal{B}} (y - Q'_n(S, v; \Theta_Q))^2$;
19: $\Theta_Q \leftarrow \text{Adam}(\Theta_Q, \nabla \mathcal{L}(\Theta_Q))$;
20: **Return** Θ_Q.

(Line 8). The exploration rate κ decays exponentially with the number of episodes e_p (Line 5), allowing the algorithm to increasingly rely on the model's predictions as training progresses. At each step, the algorithm computes the marginal influence gain as the immediate reward (Line 9) and transitions to the next state (Line 10). To incorporate delayed rewards, we apply n-step Q-learning (Lines 11–20): after observing n steps, the cumulative n-step reward is computed and the corresponding experience tuple is stored in the replay memory $\mathcal{J}$ (Lines 12-13). When sufficient experiences are available, a random batch is sampled from $\mathcal{J}$ to compute target values using the Bellman equation (Lines 14–17), and the network parameters Θ_Q are updated via the Adam optimizer to minimize the squared loss (Lines 18–19). For efficient parameter learning, we follow fitted Q-iteration [63], which leverages mini-batch updates from the replay memory instead of sample-by-sample updates. Combined with a neural network function approximator [61], it accelerates convergence and allows the Q-network to effectively learn the best node-selection policy under different states.

Time Complexity. The time complexity of Algorithm 5 is $O(E_p \times k \times (|V| \times T_Q + T_\sigma))$, where E_p is the number of training episodes, k is the budget, $|V|$ is the number of nodes, T_Q is the cost of a single forward pass of the Q-network (with $T_Q = O(|E| + |V| \times h)$, where h is the hidden dimension), and T_σ is the cost of evaluating the influence function.

D. Test Phase

Given a new multilayer graph G and a budget k, the test phase proceeds in two stages. First, we perform a single forward pass through the trained GCN to predict the influence scores for all nodes in the graph. Node features are propagated through the K-layer GCN using the learned parameters Θ_S

TABLE I
DATASET STATISTICS

Dataset	$\|V\|$	$\|E\|$	$\|L\|$	Type
Venetie (VT) [64]	1,380	19,941	43	Undirected
FF-TW-YT (FTY) [65]	11,863	87,396	3	Directed
Citeseer (CS) [66]	15,533	122,903	3	Undirected
DBLP [66]	41,892	661,883	2	Undirected
Twitter (TW) [64]	47,807	657,456	3	Undirected
MoscowAth (MA) [67]	133,364	309,952	3	Directed
Friendfeed (FF) [65]	1,531,015	20,204,535	3	Undirected
Obama (OB) [67]	3,452,114	6,711,448	3	Directed
StackOverflow (SF) [68]	2,601,977	63,497,050	9	Directed

to generate predicted influence scores. These predicted scores serve as node representations that capture both structural patterns and influence potential within the multilayer network. In the second stage, the predicted influence scores are used to construct node embeddings, which are then fed into the trained Q-learning network. The Q-network operates in a greedy manner, starting from an empty set and iteratively selecting the node that maximizes the predicted n-step reward at each step. This sequential selection process continues until k nodes have been selected, yielding the final seed node set.

VI. EXPERIMENTS

In this section, we empirically evaluate our proposed algorithms. All methods are executed on a Linux server with Intel(R) Xeon(R) 3.70 GHz CPU and 128GB of RAM.

A. Experimental Settings

Dataset. We evaluate the proposed algorithms on nine real multilayer networks, as summarized in Table I, where Citeseer and DBLP are collaboration networks, StackOverflow is temporal network and the others are social networks. For each undirected network, we treat every edge as bidirectional. Moreover, we adopt the *weighted-cascade* model [1] to assign influence weights, where the weight of edge (u, v) is defined as $w_{u,v} = 1/|N_{out}(v)|$.

Algorithms. We test the following algorithms in experiments.

- **STARIM**: Our proposed theoretical-based algorithm with approximation guarantees.
- **LGQIM**: Our proposed learning-based algorithm combining GCN and Q-learning network.
- **KSN** [14]: A greedy method that applies IMM [31] on each layer and then leverages knapsack optimization to select seed nodes.
- **DeepIM** [16]: A deep learning method that selects seed nodes based on learned structural features.
- **HighDegree** (**HD**): A method that selects seed nodes with the highest average out-degrees across all layers.
- **Random** (**RAN**): A method that uniformly selects seed nodes in random.

Parameters. We evaluate the performance of the algorithms under different parameter settings. In each experiment, we vary only one parameter while keeping the others at their default values. Experiments exceeding 24 hours are set to INF.

For the theoretical-based algorithm STARIM, we test the budget k and the sampling error factor ϵ in the approximation achieved by *MRIS* technique. The default values are set as follows: $k = 100$ for the VT, FTY, and CS datasets, $k = 500$ for the remaining datasets, $\epsilon = 0.2$, and $\delta = 0.01$ for all datasets. In all experiments, the influence spread of each algorithm is estimated using $2^5 \times 10^5$ M-RR sets generated independently of the evaluated algorithms.

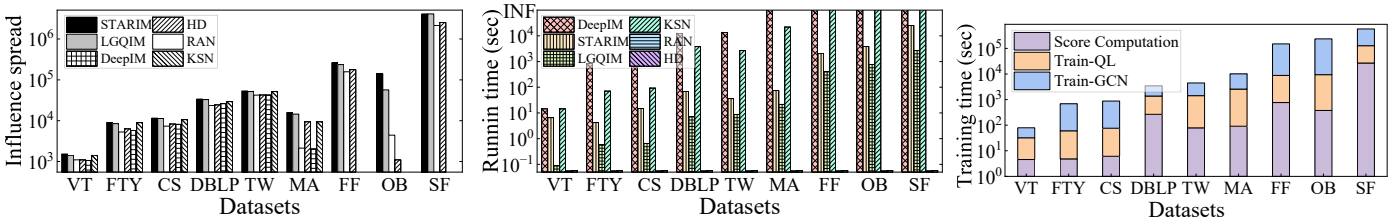
For the learning-based algorithm LGQIM, which consists of a GCN and a Q-learning network. Following [54], the GCN is trained for 500 epochs on the SF dataset and 1000 epochs on the others, with a convolution depth of 2 and an embedding dimension of 60. For the n-step Q-learning network, we set $n = 2$, the decaying factor $\gamma = 0.8$, the hidden dimension of 10, and a learning rate of 0.001. The exploration probability κ is annealed linearly from 1.0 to 0.1. The batch size is set to 20 for the FF, OB, and SF datasets, and 200 for the others. Each dataset is split evenly, with the first half used for training and the second half for testing. In all experiments, the sampled subgraph comprises 10% of the nodes from the entire graph, following [48], [54].

B. Experiment Results

Exp-1: Overall performance. We evaluate the effectiveness of the proposed methods on all datasets under the default settings. Figure 4(a) shows the influence spreads across different methods. The results indicate that STARIM and LGQIM consistently achieve the highest influence spread, surpassing all the other algorithms. This is because STARIM employs our greedy seed selection strategy (Algorithm 3), which provides theoretical guarantees. The experimental results further demonstrate the superiority of LGQIM over the other learning-based solution DeepIM in terms of influence spread. Note that, on large datasets, i.e., FF, OB, and SF, DeepIM and KSN fail to complete within a reasonable time. Therefore, the corresponding influence spread is not shown in the figure.

Furthermore, we evaluate the efficiency of all methods on all the datasets under the default settings. Figure 4(b) presents the running times across different algorithms. Notably, STARIM is consistently faster than KSN, primarily due to the early termination of M-RR set generation in STARIM. In contrast, KSN applies IMM independently on each layer and then performs a knapsack-based optimization, which often requires enumerating a large number of seed set combinations. In addition, LGQIM achieves significantly lower running time than STARIM, highlighting the efficiency advantage of our learning-based framework. Furthermore, we observe that the heuristic algorithms RAN and HD run faster than the other methods, since they do not provide theoretical guarantees. As a result, their influence spreads are substantially lower than those of STARIM and LGQIM, as shown in Figure 4(a).

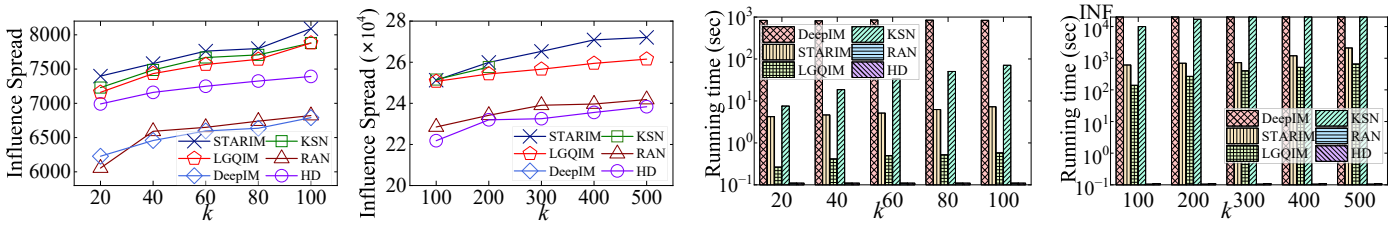
Exp-2: Varying k. We examine the impact of seed size k on the FTY and FF datasets. As shown in Figures 6(a) and 6(b), the influence spread of all methods grows with larger k, since more nodes can be activated under the hybrid propagation model. Among them, STARIM, LGQIM,



(a) Influence spread on all datasets (b) Running time on all datasets

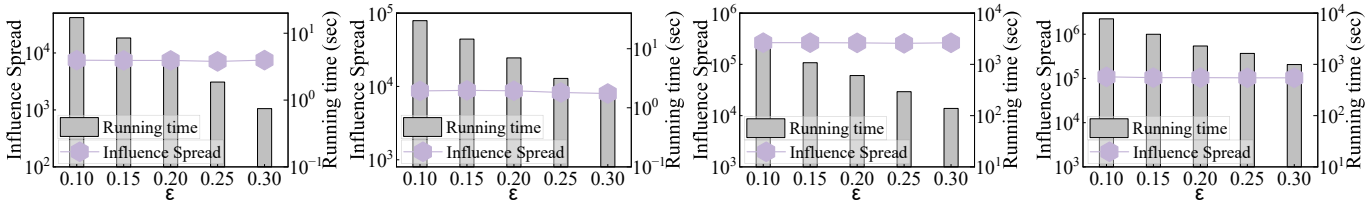
Fig. 4. Overall performance

Fig. 5. Training time of LGQIM



(a) FTY (b) FF (c) FTY (d) FF

Fig. 6. Influence spread and Running time vs. k



(a) FTY ($k = 20$) (b) CS ($k = 20$) (c) FF ($k = 100$) (d) OB ($k = 100$)

Fig. 7. Influence spread and Running time vs. ϵ

and KSN consistently achieve the highest influence spread. Figures 6(c) and 6(d) present the running times. STARIM, LGQIM, DeepIM and KSN all exhibit longer time as k increases, due to the additional iterations in the seed selection process. Among them, LGQIM achieves the best efficiency, as it avoids the costly M-RR set generation entirely. Moreover, STARIM and LGQIM achieve nearly two orders of magnitude speedup over KSN, benefiting from the early termination strategy in M-RR set generation and the elimination of online M-RR set sampling in LGQIM.

Exp-3: Varying ϵ. We investigate the effect of ϵ, the sampling error factor in the approximation guarantee achieved by the *MRIS* technique. Since STARIM is the only method that leverages *MRIS* for seed selection and provides theoretical guarantee, we evaluate its influence spread and running time under different ϵ values. Figure 7 shows that the influence spread remains relatively stable across variations in ϵ. This is because the approximation guarantee of STARIM indicates the worst-case performance, while its empirical performance in real-world scenarios is generally robust. These results highlight the effectiveness of the *MRIS* technique w.r.t. the variations of ϵ. Furthermore, the running time decreases as ϵ increases. This is due to early termination in STARIM (Line 9 in Algorithm 2), which reduces the number of M-RR sets generated. According to Theorem 5, the primary computational cost of STARIM lies in M-RR sets generation, leading to an overall reduction in running time.

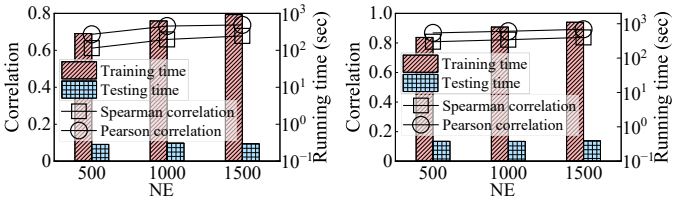
Exp-4: Training time analysis. We analyze the training overhead of LGQIM. Figure 5 shows the distribution of time across different training phases. Score Computation refers to the stage where the *MRIS* technique is applied to obtain influence scores for all nodes, which are used as labels for the GCN. Train-GCN and Train-QL denote the training processes of the GCN and the Q-learning network, respectively. Among these phases, Train-GCN takes the longest time because it involves iterative message passing over the entire graph, where each node updates its embedding by aggregating information from its neighbors. In contrast, Train-QL is much faster, as it directly consumes the GCN-predicted scores and only performs feed-forward updates in a relatively small state space.

TABLE II
ABLATION STUDY

Dataset	Influence spread		Running time (sec)	
	S+Q	G+Q	S+Q	G+Q
VT ($k = 100$)	1222.3	1217.1	4.6362	0.089
FTY ($k = 100$)	7501.8	7479.7	5.0874	0.5753
CS ($k = 100$)	9128.4	9338.8	6.4352	0.6608
DBLP ($k = 500$)	26588.5	26929.2	271.53	7.1983
TW ($k = 500$)	42299.3	42727.4	86.442	8.8831
MA ($k = 500$)	12614.7	13519.9	30.080	21.713
FF ($k = 500$)	240661	239503	1416.1	704.80
OB ($k = 500$)	55291.8	56695.7	1064.0	777.17
SF ($k = 500$)	4139860	4116730	29425	2698.1

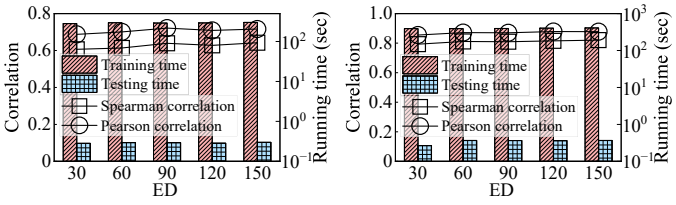
* **S+*Q***: **S**core Computation + ***Q***-learning network; **G+*Q***: **G**CN + ***Q***-learning network.

Exp-5: Ablation study. We evaluate the performance of the GCN component by conducting an ablation study, with results presented in Table II. Specifically, we compare two settings: (i) using the influence scores predicted by the GCN as input to the Q-learning network, and (ii) directly using the



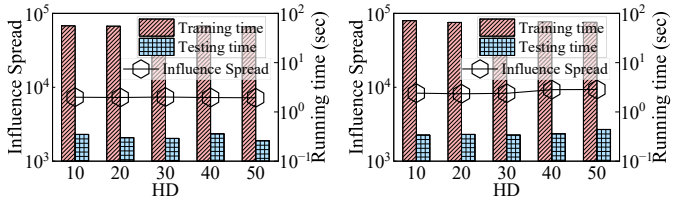
(a) FTY (b) CS

Fig. 8. Hyperparameter analysis vs. NE



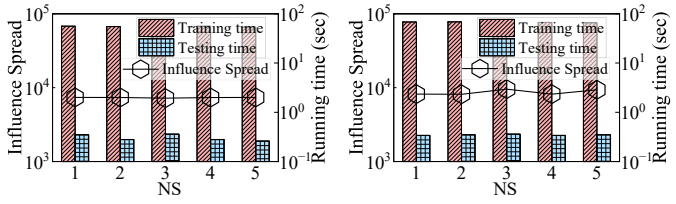
(a) FTY (b) CS

Fig. 9. Hyperparameter analysis vs. ED



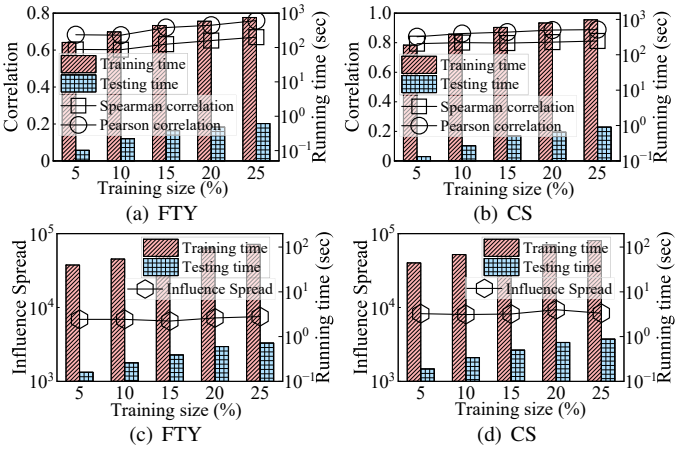
(a) FTY (b) CS

Fig. 10. Hyperparameter analysis vs. HD



(a) FTY (b) CS

Fig. 11. Hyperparameter analysis vs. NS



(a) FTY (b) CS (c) FTY (d) CS

Fig. 12. Training size analysis

influence scores computed by *MRIS* as input. We then measure the influence spread and running time of the final seed sets produced in both cases. The results show that the influence spread remains almost identical under the two settings, while the running time with GCN-predicted scores is, on average, reduced by an order of magnitude. This demonstrates that the GCN effectively serves as a lightweight surrogate for *MRIS*, significantly accelerating the framework without sacrificing influence quality.

Exp-6: Hyperparameter analysis. We analyze the impact of key hyperparameters of LGQIM on the FTY and CS datasets by varying the *Number of Epochs* (*NE*) in $\{500,\ 1000,\ 1500\}$, *Embedding Dimension* (*ED*) in $\{30,\ 60,\ 90,\ 120,\ 150\}$, *Hidden Dimension* (*HD*) in $\{10,\ 20,\ 30,\ 40,\ 50\}$, and *n_step* (*NS*) in $\{1,\ 2,\ 3,\ 4,\ 5\}$. The results are shown in Figures 8–11. We can make the following observations. **(1)** As shown in Figure 8, increasing NE in GCN leads to higher Spearman and Pearson correlations. This is because more epochs enable the model to optimize its parameters and aggregate information from neighbors, thereby capturing higher-order structural patterns more effectively. As a result, the training time increases with larger NE, while the testing time remains almost unchanged since inference only requires a single forward pass. **(2)** In Figure 9, performance of GCN improves as ED increases. Higher dimensional embeddings better represent node features and structural patterns, enabling the model to better estimate nodes' influence scores. Training time shows a slight increase with larger ED, due to the additional parameters and heavier matrix multiplications in both forward and backward propagation, whereas testing time is almost unaffected. **(3)** In Figures 10 and 11, increasing HD and NS in the Q-learning network has little effect on training time, testing time, or influence spread. This is because the Q-learning component mainly performs discrete action selection with relatively low structural complexity, making it less sensitive to these hyperparameters. Such insensitivity indicates that the model is robust and stable across different hyperparameter settings.

Exp-7: Training size analysis. We evaluate the effect of training data size on the performance of GCN and the Q-learning network, with results shown in Figure 12. The training sets are formed by selecting the top-ranked nodes by influence scores, while model training utilizes the full graph structure. We observe that even when trained on 5% of the datasets, the performance remained nearly identical to that trained on a 25% training subgraph. This demonstrates that LGQIM can effectively learn a generalized policy from a relatively small amount of training data. In contrast, both training and testing time increase with dataset size, which was expected.

VII. Conclusion

In this paper, we investigate the MLIM problem, which seeks to select a small set of seed nodes to maximize influence spread across all layers in multilayer networks. We propose a hybrid propagation model that captures the diffusion patterns in multilayer networks. Given the **NP**-hardness of MLIM, we develop a greedy algorithm Mlim-Greedy with $(1 - 1/e)$ approximation, a scalable algorithm STARIM leveraging our *MRIS* technique to achieve $(1 - 1/e - \epsilon)$ approximation, and an efficient learning-based framework LGQIM integrating multilayer-aware representation learning with deep reinforcement learning. Extensive experiments demonstrate that our algorithms are effective, efficient, and scalable.

VIII. AI-Generated Content Acknowledgment

Certain parts of Section VI were linguistically refined using ChatGPT (OpenAI). The authors thoroughly reviewed and revised the content afterward and take full responsibility for the final content of the manuscript.

References

[1] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 137–146.

[2] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 57–66.

[3] J. Wang, Y. Wu, X. Wang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "Efficient influence minimization via node blocking," *Proceedings of the VLDB Endowment*, vol. 17, no. 10, pp. 2501–2513, 2024.

[4] M. Ye, X. Liu, and W.-C. Lee, "Exploring social influence for recommendation: a generative model approach," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 2012, pp. 671–680.

[5] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, "Influence maximization on social graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1852–1872, 2018.

[6] F. Hashemi, A. Behrouz, and L. V. Lakshmanan, "Firmcore decomposition of multilayer networks," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1589–1600.

[7] DATAREPROTAL, "Global social media statistics." 2025. [Online]. Available: https://datareportal.com/social-media-users?utm_source=chatgpt.com

[8] X. Wang, Y. Bart, S. Netessine, and L. Wu, "Impact of multi-platform social media strategy on sales in e-commerce," *arXiv preprint arXiv:2503.09083*, 2025.

[9] S. S. Singh, D. Srivastva, M. Verma, and J. Singh, "Influence maximization frameworks, performance, challenges and directions on social network: A theoretical study," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 9, pp. 7570–7603, 2022.

[10] O. Achour and L. B. Romdhane, "A theoretical review on multiplex influence maximization models: Theories, methods, challenges, and future directions," *Expert Systems with Applications*, vol. 266, p. 125990, 2025.

[11] Q. Zhan, J. Zhang, S. Y. Philip, S. Emery, and J. Xie, "Discover tipping users for cross network influencing," in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*. IEEE, 2016, pp. 67–76.

[12] M. Katukuri, M. Jagarapu *et al.*, "Cim: clique-based heuristic for finding influential nodes in multilayer networks," *Applied Intelligence*, vol. 52, no. 5, pp. 5173–5184, 2022.

[13] D. T. Nguyen, H. Zhang, S. Das, M. T. Thai, and T. N. Dinh, "Least cost influence in multiplex social networks: Model representation and analysis," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 567–576.

[14] A. Kuhnle, M. A. Alim, X. Li, H. Zhang, and M. T. Thai, "Multiplex influence maximization in online social networks with heterogeneous diffusion models," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 2, pp. 418–429, 2018.

[15] D. Al Abri and S. Valaee, "Diversified viral marketing: The power of sharing over multiple online social networks," *Knowledge-Based Systems*, vol. 193, p. 105430, 2020.

[16] M. M. Keikha, M. Rahgozar, M. Asadpour, and M. F. Abdollahi, "Influence maximization across heterogeneous interconnected networks based on deep learning," *Expert Systems with Applications*, vol. 140, p. 112905, 2020.

[17] K. V. Rao and C. R. Chowdary, "Cbim: Community-based influence maximization in multilayer networks," *Information Sciences*, vol. 609, pp. 578–594, 2022.

[18] H. Nguyen, H. Dam, N. H. K. Do, C. Tran, and C. Pham, "Rem: A scalable reinforced multi-expert framework for multiplex influence maximization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 25, 2025, pp. 27 099–27 107.

[19] Z. Yuan, M. Shao, and Z. Chen, "Graph bayesian optimization for multiplex influence maximization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 20, 2024, pp. 22 475–22 483.

[20] N. Do, T. Chowdhury, C. Ling, L. Zhao, and M. T. Thai, "Mimreasoner: Learning with theoretical guarantees for multiplex influence maximization," *arXiv preprint arXiv:2402.16898*, 2024.

[21] S. Chen, H. Qian, Y. Wu, C. Chen, and X. Wang, "Efficient adoption maximization in multi-layer social networks," in *2019 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2019, pp. 56–60.

[22] B. Money, "Facebook." 2025. [Online]. Available: https://www.britannica.com/money/Facebook

[23] S. Alhabash and M. Ma, "A tale of four platforms: Motivations and uses of facebook, twitter, instagram, and snapchat among college students?" *Social media+ society*, vol. 3, no. 1, p. 2056305117691544, 2017.

[24] Builtin, "How the mere exposure effect works in marketing and advertising." 2023. [Online]. Available: https://builtin.com/articles/mere-exposure-effect

[25] S. Chen and W. Tan, "Influence diffusion model in multiplex networks." *Computers, Materials & Continua*, vol. 64, no. 1, 2020.

[26] I. Gaye, G. Mendy, S. Ouya, I. Diop, and D. Seck, "Multi-diffusion degree centrality measure to maximize the influence spread in the multilayer social networks," in *International conference on e-infrastructure and e-services for developing countries*. Springer, 2016, pp. 53–65.

[27] G. Li, Y. Chu, J. Feng, and Y. Xu, "Influence maximization on multiple social networks," *Chinese Journal of Computers*, vol. 39, no. 4, pp. 643–656, 2016.

[28] J. Chen, Y. Deng, Z. Su, S. Wang, C. Gao, and X. Li, "Identifying multiple influential users based on the overlapping influence in multiplex networks," *IEEE Access*, vol. 7, pp. 156 150–156 159, 2019.

[29] Q. Zhang, L. Zhong, S. Gao, and X. Li, "Optimizing hiv interventions for multiplex social networks via partition-based random search," *IEEE transactions on cybernetics*, vol. 48, no. 12, pp. 3411–3419, 2018.

[30] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 75–86.

[31] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015, pp. 1539–1554.

[32] J. Tang, X. Tang, X. Xiao, and J. Yuan, "Online processing algorithms for influence maximization," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 991–1005.

[33] Q. Guo, S. Wang, Z. Wei, W. Lin, and J. Tang, "Influence maximization revisited: Efficient sampling with bound tightened," *ACM Transactions on Database Systems (TODS)*, 2022.

[34] Q. Guo, C. Feng, F. Zhang, and S. Wang, "Efficient algorithm for budgeted adaptive influence maximization: An incremental rr-set update approach," *Proceedings of the ACM on Management of Data*, vol. 1, no. 3, pp. 1–26, 2023.

[35] Y. Huang, S. Zhang, L. V. Lakshmanan, W. Lin, X. Xiao, and B. Tang, "Efficient and effective algorithms for a family of influence maximization problems with a matroid constraint," *Proceedings of the VLDB Endowment*, vol. 18, no. 2, pp. 117–129, 2024.

[36] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014, pp. 946–957.

[37] K. Han, B. Wu, J. Tang, S. Cui, C. Aslay, and L. V. Lakshmanan, "Efficient and effective algorithms for revenue maximization in social advertising," in *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*, 2021, pp. 671–684.

[38] P. Banerjee, W. Chen, and L. V. Lakshmanan, "Maximizing welfare in social networks under a utility driven influence diffusion model," in *Proceedings of the 2019 ACM SIGMOD International Conference on Management of Data*, 2019, pp. 1078–1095.

[39] J. Tang, X. Tang, and J. Yuan, "Profit maximization for viral marketing in online social networks: Algorithms and analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 6, pp. 1095–1108, 2017.

[40] X. Chang, X. Ke, L. Chen, C. Ge, Z. Wei, and Y. Gao, "Host profit maximization: Leveraging performance incentives and user flexibility," *Proceedings of the VLDB Endowment*, vol. 17, no. 1, pp. 51–64, 2023.

[41] X. Chang, J. Fu, Q. Liu, Y. Gao, and B. Zheng, "Time-aware influence minimization via blocking social networks," in *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 2025, pp. 557–570.

[42] S. Teng, J. Xie, F. Zhang, C. Lu, J. Fang, and K. Wang, "Optimizing network resilience via vertex anchoring," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 606–617.

[43] S. Cui, K. Han, J. Tang, and H. Huang, "Constrained subset selection from data streams for profit maximization," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 1822–1831.

[44] S.-C. Lin, S.-D. Lin, and M.-S. Chen, "A learning-based framework to handle multi-round multi-party influence maximization on social networks," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 695–704.

[45] K. Ali, C.-Y. Wang, and Y.-S. Chen, "Boosting reinforcement learning in competitive influence maximization with transfer learning," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2018, pp. 395–400.

[46] H. Li, M. Xu, S. S. Bhowmick, C. Sun, Z. Jiang, and J. Cui, "Disco: Influence maximization meets network embedding and deep learning," *arXiv preprint arXiv:1906.07378*, 2019.

[47] S. Tian, S. Mo, L. Wang, and Z. Peng, "Deep reinforcement learning-based approach to tackle topic-aware influence maximization," *Data Science and Engineering*, vol. 5, no. 1, pp. 1–11, 2020.

[48] H. Li, M. Xu, S. S. Bhowmick, J. S. Rayhan, C. Sun, and J. Cui, "Piano: Influence maximization meets deep reinforcement learning," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 3, pp. 1288–1300, 2022.

[49] W. Chen, C. Castillo, and L. V. Lakshmanan, *Information and influence propagation in social networks*. Springer Nature, 2022.

[50] S. Kumar, A. Mallik, A. Khetarpal, and B. S. Panda, "Influence maximization in social networks using graph embedding and graph neural network," *Information Sciences*, vol. 607, pp. 1617–1636, 2022.

[51] H. Kamarthi, P. Vijayan, B. Wilder, B. Ravindran, and M. Tambe, "Influence maximization in unknown social networks: Learning policies for effective graph sampling," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 575–583.

[52] G. Panagopoulos, F. D. Malliaros, and M. Vazirgiannis, "Multi-task learning for influence estimation and maximization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 9, pp. 4398–4409, 2020.

[53] G. Panagopoulos, N. Tziortziotis, M. Vazirgiannis, and F. Malliaros, "Maximizing influence with graph neural networks," in *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, 2023, pp. 237–244.

[54] S. Manchanda, A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, "Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 000–20 011, 2020.

[55] C. Ling, J. Jiang, J. Wang, M. T. Thai, R. Xue, J. Song, M. Qiu, and L. Zhao, "Deep graph representation learning and optimization for influence maximization," in *International conference on machine learning*. PMLR, 2023, pp. 21 350–21 361.

[56] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *2010 IEEE International Conference on Data Mining (ICDM)*, 2010, pp. 88–97.

[57] X. Chang, R. Liu, and Q. Liu, "Theoretical and learning-based approaches for influence maximization in multilayer social networks," 2025. [Online]. Available: https://github.com/ZJU-DAILY/MLIM

[58] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[59] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[60] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[61] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[62] K. D. B. J. Adam *et al.*, "A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, vol. 1412, no. 6, 2014.

[63] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European conference on machine learning*. Springer, 2005, pp. 317–328.

[64] J. A. Baggio, S. B. BurnSilver, A. Arenas, J. S. Magdanz, G. P. Kofinas, and M. De Domenico, "Multiplex social ecological network analysis reveals how social changes affect community robustness more than resource depletion," *Proceedings of the National Academy of Sciences*, vol. 113, no. 48, pp. 13 708–13 713, 2016.

[65] M. E. Dickison, M. Magnani, and L. Rossi, *Multilayer Social Networks*. Cambridge University Press, 2016.

[66] D. Liu and Z. Zou, "Gcore: Exploring cross-layer cohesiveness in multi-layer graphs," *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 3201–3213, 2023.

[67] E. Omodei, M. De Domenico, and A. Arenas, "Characterizing interactions in online social networks during exceptional events," *Frontiers in Physics*, vol. 3, p. 59, 2015.

[68] J. Leskovec and A. Krevl, "Snap datasets: Stanford large network dataset collection," 2014, http://snap.stanford.edu/data.

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

[70] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[71] T. Jin, Y. Yang, R. Yang, J. Shi, K. Huang, and X. Xiao, "Unconstrained submodular maximization with modular costs: Tight approximation and application to profit maximization," *Proceedings of the VLDB Endowment*, vol. 14, no. 10, pp. 1756–1768, 2021.

APPENDIX

A. PROOF OF LEMMA 1

Proof. Given the monotonicity and submodularity of the MLIM problem established in Section III-C, the assurance of approximation follows directly from [69]. □

B. PROOF OF LEMMA 2

Proof. Before proving Lemma 2, we first introduce *Chernoff Inequalities* [70] in Lemma 3 as follows.

Lemma 3. (Chernoff Inequalities [70]). *Let X be the sum of k i.i.d. random variables sampled from a distribution on $[0,1]$ and μ be the mean. Then, for any $\mu > 0$,*

$$\Pr[X - k\mu \geq \mu \cdot k\mu] \leq \exp\left(-\frac{\mu^2}{2+\mu}k\mu\right) \tag{17}$$

$$\Pr[X - k\mu \leq -\mu \cdot k\mu] \leq \exp\left(-\frac{\mu^2}{2}k\mu\right) \tag{18}$$

In the following, we denote $\sigma(S, G)$ and $\sigma(S^*, G)$ by $\sigma(S)$ and $\sigma(S^*)$, respectively. Given any solution S to our problem and the optimal solution S^*, and any set $\mathcal{R}$ of M-RR sets, we extend Lemma 3 and introduce the following concentration bounds:

$$\Pr[f_1^{\mathcal{R}_2}(S) - \sigma_1(S) \geq \epsilon_1 \cdot \sigma_1(S)] \leq \exp\left(-\frac{\epsilon_1^2}{2+\epsilon_1}\frac{|R_1|}{n_1}\sigma_1(S)\right)$$

$$\Pr[f_2^{\mathcal{R}_2}(S) - \sigma_2(S) \geq \epsilon_1 \cdot \sigma_2(S)] \leq \exp\left(-\frac{\epsilon_1^2}{2+\epsilon_1}\frac{|R_2|}{n_2}\sigma_1(S)\right)$$

$$\dots$$

$$\Pr[f_l^{\mathcal{R}_2}(S) - \sigma_l(S) \geq \epsilon_1 \cdot \sigma_l(S)] \leq \exp\left(-\frac{\epsilon_1^2}{2+\epsilon_1}\frac{|R_l|}{n_l}\sigma_l(S)\right)$$

where $|R_i|$ denotes the number of M-RR sets belonging to $\mathcal{R}_2$ sampled in layer i. Then, summing thm yields:

$$\begin{aligned}
&\Pr[f^{\mathcal{R}_2}(S) - \sigma(S) \geq \epsilon_1 \cdot \sigma(S)] \\
&\leq \exp\left(-\frac{\epsilon_1^2}{2+\epsilon_1}\left(\frac{|R_1|}{n_1}\sigma_1(S) + \frac{|R_2|}{n_2}\sigma_2(S) + \dots + \frac{|R_l|}{n_l}\sigma_l(S)\right)\right) \\
&= \exp\left(-\frac{\epsilon_1^2}{2+\epsilon_1}\sum_{i=1, R_i \in \mathcal{R}_2}^{l}\frac{|R_i|}{n_i}\sigma_i(S)\right)
\end{aligned} \tag{19}$$

where $f^{\mathcal{R}_2}(S) = f_1^{\mathcal{R}_2}(S) + f_2^{\mathcal{R}_2}(S) + \dots + f_l^{\mathcal{R}_2}(S)$ and $\sigma(S) = \sigma_1(S) + \sigma_2(S) + \dots + \sigma_l(S)$. Similarly, we also have:

$$\begin{aligned}
&\Pr[f^{\mathcal{R}_1}(S^*) - \sigma(S^*) \leq -\epsilon_2 \cdot \sigma(S^*)] \\
&\leq \exp\left(-\frac{\epsilon_2^2}{2}\left(\frac{|R_1|}{n_1}\sigma_1(S^*) + \frac{|R_2|}{n_2}\sigma_2(S^*) + \dots + \frac{|R_l|}{n_l}\sigma_l(S^*)\right)\right) \\
&= \exp\left(-\frac{\epsilon_2^2}{2}\sum_{i=1, R_i \in \mathcal{R}_1}^{l}\frac{|R_i|}{n_i}\sigma_i(S^*)\right)
\end{aligned} \tag{20}$$

where $|R_i|$ represents the number of M-RR sets belonging to $\mathcal{R}_1$ sampled in layer i. Based on this, in the i-th iteration, let Ω_{1i} denote the event that Eq. (6) holds, and Ω_{2i} denote the event that Eq. (7) holds. We set ϵ' and $\bar{\epsilon}$ as the solutions to Eq. (19) and Eq. (20) respectively, thus we have following equations:

$$\exp\left(-\frac{(\epsilon')^2}{2+\epsilon'}\sum_{i=1, R_i \in \mathcal{R}_2}^{l}\frac{|R_i|}{n_i}\sigma_i(S)\right) = \frac{\delta}{5i^2}. \tag{21}$$

$$\exp\left(-\frac{(\bar{\epsilon})^2}{2}\sum_{i=1, R_i \in \mathcal{R}_1}^{l}\frac{|R_i|}{n_i}\sigma_i(S)\right) = \frac{\delta}{5i^2}. \tag{22}$$

Then, we have $\Pr[\Omega_{i1}] \geq 1 - \delta/(5i^2)$ and $\Pr[\Omega_{i2}] \geq 1 - \delta/(5i^2)$.

$$\begin{aligned}
\frac{(\epsilon')^2}{(2+\epsilon')(1+\epsilon')} &= \frac{\ln\left(5i^2/\delta\right)}{(1+\epsilon')\cdot\sum_{i=1, R_i\in\mathcal{R}_2}^{l}\frac{|R_i|}{n_i}\sigma_i(S)} \\
&\leq \frac{\ln\left(5i^2/\delta\right)}{(1+\epsilon')\cdot\min_{R_i\in\mathcal{R}_2}\{\frac{|R_i|}{n_i}\}\cdot\sum_i^l\sigma_i(S)} \\
&= \frac{\ln\left(5i^2/\delta\right)}{(1+\epsilon')\cdot\min_{R_i\in\mathcal{R}_2}\{\frac{|R_i|}{n_i}\}\cdot\sigma(S)} \\
&\leq \frac{\ln\left(5i^2/\delta\right)}{\min_{R_i\in\mathcal{R}_2}\{\frac{|R_i|}{n_i}\}\cdot f^{\mathcal{R}_2}(S)}
\end{aligned} \tag{23}$$

$$\begin{aligned}
\frac{\bar{\epsilon}^2}{2(1+\epsilon_1)} &= \frac{\ln\left(5i^2/\delta\right)}{(1+\epsilon_1)\cdot\sum_{i=1, R_i\in\mathcal{R}_1}^{l}\frac{|R_i|}{n_i}\sigma_i(S^*)} \\
&\leq \frac{\ln\left(5i^2/\delta\right)}{(1+\epsilon_1)\cdot\min_{R_i\in\mathcal{R}_1}\{\frac{|R_i|}{n_i}\}\cdot\sum_i^l\sigma_i(S^*)} \\
&= \frac{\ln\left(5i^2/\delta\right)}{(1+\epsilon_1)\cdot\min_{R_i\in\mathcal{R}_1}\{\frac{|R_i|}{n_i}\}\cdot\sigma(S^*)} \\
&\leq \frac{\ln\left(5i^2/\delta\right)}{(1+\epsilon_1)\cdot\min_{R_i\in\mathcal{R}_1}\{\frac{|R_i|}{n_i}\}\cdot\sigma(S)} \\
&\leq \frac{\ln\left(5i^2/\delta\right)}{\min_{R_i\in\mathcal{R}_1}\{\frac{|R_i|}{n_i}\}\cdot f^{\mathcal{R}_2}(S)}
\end{aligned} \tag{24}$$

We have $\Pr[\Omega_{i1}] \geq 1 - \delta/(5i^2)$, $\Pr[\Omega_{i2}|\Omega_{i1}] \geq 1 - \delta/(5i^2)$. Thus, $\Pr[\Omega_{i2} \cap \Omega_{i1}] = \Pr[\Omega_{i2}|\Omega_{i1}] \cdot \Pr[\Omega_{i1}] \geq 1 - 2\delta/(5i^2)$. For all iterations, we have

$$\begin{aligned}
\Pr\left[\bigcap_{i=1}^{\infty}\Omega_{1i}\bigcap_{i=1}^{\infty}\Omega_{2i}\right] &\geq \prod_{i=1}^{\infty}\Pr\left[\Omega_{1i}\cap\Omega_{2i}\right] \geq \prod_{i=1}^{\infty}\left(1 - \frac{2\delta}{5i^2}\right) \\
&\geq 1 - \sum_{i=1}^{\infty}\frac{2\delta}{5i^2} \geq 1 - \frac{\pi^2\delta}{15} \geq 1 - \frac{2\delta}{3}.
\end{aligned} \tag{25}$$

The details of proof are similar in spirit to those in [71]. □

C. PROOF OF THEOREM 4

Proof. With the above conclusions, we further prove Theorem 4.

We consider two cases that depend on whether Line 9 in Algorithm 2 is satisfied.

Case 1: Line 9 is satisfied. Then in the last iteration, we have

$$\mu \leq \frac{1 - 1/e}{1 - 1/e - \epsilon} \cdot \frac{1 - \epsilon_2}{1 + \epsilon_1} \tag{26}$$

where $\epsilon_1, \epsilon_2 \in (0, 1)$ and $\mu > 0$. Suppose that both Eq. (6) and Eq. (7) hold. Then,

$$\begin{aligned}
f^{\mathcal{R}_1}(S) &\geq (1 - 1/e) f^{\mathcal{R}_1}(S^*) \\
&\geq (1 - 1/e)(1 - \epsilon_2)\sigma(S^*)
\end{aligned} \tag{27}$$

Afterwards, via Eq. (6) we have:

$$f^{\mathcal{R}_1}(S) = \mu f^{\mathcal{R}_2}(S) \leq \mu(1+\epsilon_1)\delta(S) \tag{28}$$

Finally, we have

$$\begin{aligned}\delta(S) &\geq \frac{1}{\mu(1+\epsilon_1)} f^{\mathcal{R}_1}(S) \\ &\geq \frac{(1-1/e)(1-\epsilon_2)}{\mu(1+\epsilon_1)}\delta(S^*) \\ &\geq (1-1/e-\epsilon)\delta(S^*)\end{aligned} \tag{29}$$

where the first inequality is from Eq. (28), the second inequality is from Eq. (27), and the third inequality is from Eq. (26). By Lemma 2, when Line 9 in Algorithm 2 is satisfied, with probability at least $1-\frac{2\delta}{3}$, Theorem 4 holds.

Case 2: Line 9 is not satisfied. Then we have

$$\min_{R_i \in \mathcal{R}_1} \{\frac{|R_i|}{n_i}\} \geq (8+2\epsilon)(1+\epsilon_1)\frac{\ln\frac{6}{\delta} + n\ln 2}{\epsilon^2 \cdot f^{\mathcal{R}_2}(S)}$$

when Algorithm **??** terminates. Note that when $\bigcap_i \Omega_{1i}$ occurs, it implies that $g^{\mathcal{R}_2}(S) \leq (1+\epsilon_1)\sigma(S) \leq (1+\epsilon_1)\sigma(S^*)$. Then

$$\min_{R_i \in \mathcal{R}_1} \{\frac{|R_i|}{n_i}\} \geq (8+2\epsilon)\frac{\ln\frac{6}{\delta} + n\ln 2}{\epsilon^2 \cdot \sigma(S^*)}$$

when Algorithm **??** terminates. Then by Lemma 3, let $x = \frac{\epsilon\sigma(S^*)}{2\sigma(S)}$ for any $S \subseteq V$,

$$\begin{aligned}&\Pr[f^{\mathcal{R}_1}(S) - \sigma(S) \geq \frac{\epsilon}{2}\cdot\sigma(S^*)] \\ &\leq \exp\left(-\frac{x^2}{2+x}\cdot\sum_{i=1, R_i\in\mathcal{R}_1}^{l}\frac{|R_i|}{n_i}\sigma_i(S)\right) \\ &\leq \exp\left(-\frac{x^2}{2+x}\cdot\min_{R_i\in\mathcal{R}_1}\{\frac{|R_i|}{n_i}\}\cdot\sigma(S)\right) \\ &\leq \exp\left(-\frac{\epsilon^2}{8+2\epsilon}\cdot\min_{R_i\in\mathcal{R}_1}\{\frac{|R_i|}{n_i}\}\cdot\sigma(S^*)\right) \\ &\leq \frac{\delta}{6\cdot 2^n},\end{aligned}$$

where the second inequality is due to the fact that if $\sigma(S) = \sigma(S^*)$, the right side of the first inequality achieves its maximum. Similarly, we also have $\Pr[f^{\mathcal{R}_1}(S) - \sigma(S) \leq (-\frac{\epsilon}{2})\cdot\sigma(S^*)] \leq \frac{\delta}{6\cdot 2^n}$. Thus, we have $\Pr[|f^{\mathcal{R}_1}(S) - \sigma(S)| \leq \frac{\epsilon}{2}\cdot\sigma(S^*), \forall S \subseteq V] \geq 1-\frac{\delta}{3}$. When $|f^{\mathcal{R}_1}(S) - \sigma(S)| \leq \frac{\epsilon}{2}\sigma(S^*)$ for all $S \subset V$, we have

$$f^{\mathcal{R}_1}(S^*) \geq (1-\frac{\epsilon}{2})\sigma(S^*)\ , \tag{30}$$

$$f^{\mathcal{R}_1}(S) \leq \sigma(S) + \frac{\epsilon}{2}\sigma(S^*). \tag{31}$$

Based on the above results, when the event $\bigcap_i \Omega_{1i}$ occurs, we have

$$\begin{aligned}\sigma(S) &\geq f^{\mathcal{R}_1}(S) - \frac{\epsilon}{2}\sigma(S^*) \\ &\geq (1-1/e)f^{\mathcal{R}_1}(S^*) - \frac{\epsilon}{2}\sigma(S^*) \\ &\geq (1-1/e)(1-\frac{\epsilon}{2})\sigma(S^*) - \frac{\epsilon}{2}\sigma(S^*) \\ &= (1-1/e-\frac{\epsilon}{2}+\frac{\epsilon}{2e})\sigma(S^*) - \frac{\epsilon}{2}\Sigma(S^*) \\ &= (1-1/e-\epsilon+\frac{\epsilon}{2e})\sigma(S^*) \\ &\geq (1-1/e-\epsilon)\cdot\sigma(S^*)\end{aligned}$$

According to Eq. (25), the event $\bigcap_i \Omega_{1i}$ happens with probability at least $1-\frac{2\delta}{3}$. Hence, when Line 9 is not satisfied, with probability at least $1-\frac{2\delta}{3}-\frac{\delta}{3} \geq 1-\delta$, we have

$$\sigma(S) \geq (1-1/e-\epsilon)\cdot\sigma(S^*)$$

Finally, we combine **Case 1** and **Case 2**, the Theorem 4 is demonstrated. □

D. PROOF OF THEOREM 5

Proof. The time complexity of Algorithm 2 is primarily determined by the cost of M-RR set generation. In Algorithm 2, the total number of M-RR set generated is at most $O(n\cdot(\ln 1/\delta + \ln n)/\epsilon^2)$. The expected time to generate a random M-RR set is bounded by $\frac{m}{n}\cdot\sigma(\{v^*\})$ [30]. Thus, the expected time complexity of Algorithm 2 is $O(\frac{(\ln 1/\delta+\ln n)\cdot m\cdot\sigma(\{v^*\})}{\epsilon^2})$. □