Multi-layered Network Exploration via Random Walks: From Offline Optimization to Online Learning

Xutong Liu [1] **Jinhang Zuo** [2] **Xiaowei Chen** [3] **Wei Chen** [4] **John C.S. Lui** [1]

Abstract

Multi-layered network exploration (MuLaNE) problem is an important problem abstracted from many applications. In MuLaNE, there are multiple network layers where each node has an importance weight and each layer is explored by a random walk. The MuLaNE task is to allocate total random walk budget B into each network layer so that the total weights of the unique nodes visited by random walks are maximized. We systematically study this problem from offline optimization to online learning. For the offline optimization setting where the network structure and node weights are known, we provide greedy based constant-ratio approximation algorithms for overlapping networks, and greedy or dynamic-programming based optimal solutions for non-overlapping networks. For the online learning setting, neither the network structure nor the node weights are known initially. We adapt the combinatorial multi-armed bandit framework and design algorithms to learn random walk related parameters and node weights while optimizing the budget allocation in multiple rounds, and prove that they achieve logarithmic regret bounds. Finally, we conduct experiments on a real-world social network dataset to validate our theoretical results.

1. Introduction

Network exploration is a fundamental paradigm of searching/exploring in order to discover information and resources available at nodes in a network, and random walk is often used as an effective tool for network exploration (Lv et al., 2002; Gleich, 2015; Wilder et al., 2018). In this paper, we study the multi-layered network exploration via random walks problem, which can model many real-world applications, including resource searching in peer-to-peer (P2P) networks and web surfing in online social networks (OSNs).

In P2P networks, a user wants to find resources that are scattered on nodes (i.e. peers) in different P2P networks via some platform-specified strategies. A commonly used search strategy is based on multiple random walks (Lv et al., 2002). Since different resources have different importance, the search quality of different random walkers varies. Moreover, the resource-search process typically has a life span, i.e., a total time-limit or hop-limit for random walks. So the user's goal is to decide how to allocate limited budgets to different random walkers to find as many important resources as possible. Another application is the web surfing, where users want to find information in different OSNs, by looking at posts via others' home-pages (Lerman & Jones, 2006). Once the user arrives at one of his friends' home-pages in a particular OSN, he could find some information and continues to browse the home-pages of his friends' friends, which can be regarded as a random walk process. Since the user only has a finite duration in web surfing, the similar question is how to allocate his time in exploring different OSNs so to get the maximum amount of useful information.

We abstract the above application scenarios as the **Mu**lti-**La**yered **N**etwork **E**xploration (MuLaNE) problem. In MuLaNE, we model the overall network to be explored (e.g., combining different OSNs) as a multi-layered network $\mathcal{G}$ that consists of m layers $L_1, ..., L_m$. Each layer L_i (e.g., a single OSN) is represented by a weighted directed graph $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i, w_i)$, where $\mathcal{V}_i$ is the set of nodes to be explored (e.g., users' home-pages with importance weight σ_u for $u \in \mathcal{V}_i$), $\mathcal{E}_i \subseteq \mathcal{V}_i \times \mathcal{V}_i$ is the set of directed edges (e.g., social links), and w_i is the edge weight function on edges $\mathcal{E}_i$. Each layer L_i is associated with an explorer (or random walker) W_i and a fixed starting distribution $\boldsymbol{\alpha}_i$ on nodes $\mathcal{V}_i$. Explorer W_i starts on a node in $\mathcal{G}_i$ following the distribution $\boldsymbol{\alpha}_i$, and then walks on network $\mathcal{G}_i$ following outgoing edges with probability proportional to edge weights. We assume that each random walk step will cost one unit of the bud-

[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China [2]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA [3]Bytedance, Mountain View, CA, USA [4]Microsoft Research, Beijing, China. Correspondence to: Xutong Liu <liuxt@cse.cuhk.edu.hk>, Wei Chen <weic@microsoft.com>, John C.S. Lui <cslui@cse.cuhk.edu.hk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

get[1]. Given a total budget constraint B and individual layer budget constraint c_i, the MuLaNE task is to find the optimal budget allocation $\boldsymbol{k}=(k_1,...,k_m)$ with $\sum_{i=1}^{m} k_i \leq B$, $0\leq k_i \leq c_i$ such that the expected total weights of *unique* nodes visited are maximized.

In real applications, the network structure $\mathcal{G}$, the node weights $\boldsymbol{\sigma}$ and the starting distributions $\boldsymbol{\alpha}_i$'s may not be known in advance, so we consider both the *offline optimization* cases when $\mathcal{G}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\alpha}_i$'s are known and the *online learning* case when $\mathcal{G}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\alpha}_i$'s are unknown. Moreover, different layers may have overlapping vertices (e.g., homepage of the same user may appear in different OSNs), and the starting distributions $\boldsymbol{\alpha}_i$'s may or may not be stationary distributions. In this paper, we provide a systematic study of all these case combinations.

For the offline optimization, we first consider the general overlapping setting and provide constant approximation algorithms based on the lattice submodularity property for non-stationary $\boldsymbol{\alpha}_i$'s and the diminishing-return (DR) submodularity property for stationary $\boldsymbol{\alpha}_i$'s. For the special non-overlapping setting, we design a dynamic programming algorithm that finds the exact optimal solution for MuLaNE.

In the online learning, we conduct multiple rounds of exploration, each of which has the same budget constraints B and c_i's. After each round of exploration, the total weights of unique nodes visited in this round are the reward for this round, and the trajectory of every explorer in every layer and the importance weights of visited nodes are observed as the feedback, which could be used to learn information about the network for the benefit of future explorations.

We adapt the combinatorial multi-armed bandit (CMAB) framework and the CUCB algorithm (Chen et al., 2016) to our setting, and design online learning algorithms to minimize the regret, which is the difference between the cumulative reward achieved by the exact or approximate offline algorithm and that achieved by our learning algorithm, over T rounds. We show that directly learning the graph structure $\mathcal{G}$ is inefficient. Instead, we define intermediate random variables in MuLaNE as the base arms in CMAB and learn these intermediate parameters, which can sufficiently determine the rewards to guide our budget allocation. Moreover, the node weight is not revealed until this node is first visited, which further complicates the design of online exploration algorithms. For the overlapping case, we adapt the CUCB algorithm to address the unrevealed node weights and the extra constraint of monotonicity for the intermediate parameters. We further improve the analysis by leveraging on special properties in our setting and show logarithmic regret bounds in this case. For the non-overlapping case, we define more efficient intermediate parameters and use the exact offline algorithm, and thus achieve a better regret bound. Finally, we conduct experiments on a real-world multi-layered social network dataset to validate the effectiveness of both our offline and online algorithms.

[1]Our model can be extended to move multiple steps with one unit of budget.

Our contributions can be summarized as follows: (1) We are the first to model the multi-layered network exploration via random walks problem (MuLaNE) as an abstraction for many real-world applications (2) We provide a systematic study of the MuLaNE problem via theoretical analysis and empirical validation by considering offline and online settings for both overlapping and non-overlapping multi-layered networks. Due to the space limit, proofs are included in the supplementary material.

1.1. Related Work

Network exploration via random walks has been studied in various application contexts such as community detection (Pons & Latapy, 2005), centrality measuring (Gleich, 2015), large-scale network sampling (Li et al., 2019), and influence maximization (Wilder et al., 2018). Multiple random walks has also been used, e.g. in (Lv et al., 2002) for query resolution in peer-to-peer networks. However, none of these studies address the budgeted MuLaNE problem. MuLaNE is also related to the influence maximization (IM) problem (Kempe et al., 2003; Chen et al., 2009) and can be viewed as a special variant of IM. Different from the standard Independent Cascade (IC) model (Wang et al., 2012) in IM, which randomly broadcasts to all its neighbors, the propagation process of MuLaNE is a random walk that selects one neighbor. Moreover, each unit of budget in MuLaNE is used to propagate one random-walk step, not to select one seed node as in IC.

The offline budget allocation problem has been studied by Alon et al. (2012); Soma et al. (2014), the latter of which propose the lattice submodularity and a constant approximation algorithm based on this property. Our offline overlapping MuLaNE setting is based on a similar approach, but we provide a better approximation ratio compared to the original analysis in (Alon et al., 2012). Moreover, we further show that the stationary setting enjoys DR-submodularity, leading to an efficient algorithm with a better approximation ratio.

The multi-armed bandit (MAB) problem is first studied by Robbins (1952) and then extended by many studies (cf. (Bubeck & Cesa-Bianchi, 2012)). Our online MuLaNE setting fits into the general Combinatorial MAB (CMAB) framework of (Gai et al., 2012; Chen et al., 2016). CUCB is proposed as a general algorithm for CMAB (Chen et al., 2016). Our study includes several adaptations, such as handling unknown node weights, defining intermediate random variables as base arms and so on.

Chen et al. (2018) study the community exploration problem,

which is essentially a special case of MuLaNE with non-overlapping complete graphs. As a result, their technique is different and much simpler than ours.

The rest of the paper is organized as follows. Section 2 states the settings of MuLaNE; Section 3 states the equivalent bipartite coverage model to derive the explicit form for our reward function; Section 4 states offline algorithms for four offline settings with provable approximation guarantee and running time analysis; Section 5 states two online learning algorithms with regret analysis; Empirical results are shown in Section 6; and Section 7 concludes the paper.

2. Problem Settings

Basic Notations. In this paper, we use $\mathbb{R}$ and $\mathbb{Z}$ to denote the sets of real numbers and integers, respectively, and the associated subscript ≥ 0 and > 0 denote their non-negative and positive subsets, respectively. Suppose we want to explore a multi-layered network $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{\sigma})$, consisting of m layers $L_1, ..., L_m$ and N unique nodes $\mathcal{V}$ with fixed node weights $\boldsymbol{\sigma} \in [0,1]^{\mathcal{V}}$. Let $[m]$ denote the set $\{1, 2, ..., m\}$. Each layer $L_i, i \in [m]$, represents a subgraph of $\mathcal{G}$ we could explore and is modeled as a weighted digraph $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i; w_i)$, where $\mathcal{V}_i \subseteq \mathcal{V}$ is the set of vertices in layer L_i, $\mathcal{E}_i \subseteq \mathcal{V}_i \times \mathcal{V}_i$ is the set of edges in $\mathcal{G}_i$, and $w_i : \mathcal{E}_i \to \mathbb{R}_{>0}$ is the edge weight function associating each edge of $\mathcal{G}_i$. For any two layers L_i and L_j, we say they are *non-overlapping* if $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$. $\mathcal{G}$ is called a *non-overlapping* multi-layered network if any two layers are non-overlapping; otherwise $\mathcal{G}$ is an *overlapping* multi-layered network. Let $n_i = |\mathcal{V}_i|$ denote the size of layer L_i, and the adjacency matrix of $\mathcal{G}_i$ is defined as $\boldsymbol{A_i} \in \mathbb{R}_{\geq 0}^{n_i \times n_i}$, where $\boldsymbol{A_i}[u, v] = w_i((u, v))$ if $(u, v) \in \mathcal{E}_i$ and 0 otherwise.

Exploration Rule. Each layer L_i is associated with an explorer W_i, a budget $k_i \in \mathbb{Z}_{\geq 0}$ and an initial starting distribution $\boldsymbol{\alpha}_i = (\alpha_{i,u})_{u \in \mathcal{V}_i} \in \mathbb{R}_{\geq 0}^{n_i}$ with $\sum_{u \in \mathcal{V}_i} \alpha_{i,u} = 1$. The exploration process of W_i is identical to applying *weighted random walks* within layer L_i. Specifically, the explorer W_i starts the exploration from a random node $v_1 \in \mathcal{V}_i$ with probability α_{i,v_1}; it consumes one unit of budget and continues the exploration by walking from v_1 to one of its out-neighbors, say v_2, with probability $\boldsymbol{P}_i[v_1, v_2]$, where $\boldsymbol{P}_i \in \mathbb{R}_{\geq 0}^{n_i \times n_i}$ is the transition probability matrix, and $\boldsymbol{P}_i[u, v] := \boldsymbol{A_i}[u, v] / (\sum_{w:(u,w) \in \mathcal{E}_i} \boldsymbol{A_i}[u, w])$. Consider visiting the initial node v_1 as the first step, the process is repeated until the explorer W_i walks k_i steps and visits k_i nodes (with possibly duplicated nodes). Note that one can easily generalize our results by moving $\lambda_i \in \mathbb{Z}_{>0}$ steps with one unit of budget for W_i, and for simplicity, we set $\lambda_i = 1$.

Reward Function. We define the exploration trajectory for explorer W_i after exploring k_i nodes as $\Phi(i, k_i) := (X_{i,1}, \ldots, X_{i,k_i})$, where $X_{i,j} \in \mathcal{V}_i$ denotes the node visited at j-th step, and $\Pr(X_{i,1} = u) = \alpha_{i,u}, u \in \mathcal{V}_i$. The reward for $\Phi(i, k_i)$ is defined as the total weights of *unique* nodes visited by W_i, i.e., $\sum_{v \in \bigcup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v$. Considering trajectories of all W_i's, the total reward is the total weights of *unique* nodes visited by all random walkers, i.e., $\sum_{v \in \bigcup_{i=1}^{m} \bigcup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v$. For notational simplicity, let $\mathcal{G} := (\mathcal{G}_1, \ldots, \mathcal{G}_m)$, $\boldsymbol{\alpha} := (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m)$ and $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_{|\mathcal{V}|})$ be the parameters of a problem instance and $\boldsymbol{k} := (k_1, \ldots, k_m)$ be the allocation vector. For overlapping multi-layered network $\mathcal{G}$, the total expected reward is

$$r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k}) := \mathbb{E}_{\Phi(1,k_1),\ldots,\Phi(m,k_m)} \left[\sum_{v \in \bigcup_{i=1}^{m} \bigcup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v \right], \tag{1}$$

where its explicit formula will be discussed later in Sec. 3. For non-overlapping $\mathcal{G}$, the reward function can be simplified and written as the summation over separated layers,

$$r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k}) := \sum_{i=1}^{m} \mathbb{E}_{\Phi(i,k_i)} \left[\sum_{v \in \bigcup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v \right]. \tag{2}$$

Problem Formulation. We are interested in the *budget allocation problem*: how to allocate the total budget $B \in \mathbb{Z}_{\geq 0}$ to the m explorers so as to maximize the total weights of unique nodes visited. Also, we assume there is a budget constraint $\boldsymbol{c} := (c_1, ..., c_m)$ such that the allocated budget k_i should not exceed c_i, i.e., $k_i \leq c_i$, for $i \in [m]$.

Definition 1. *Given graph structures $\mathcal{G} := (\mathcal{G}_1, \ldots, \mathcal{G}_m)$, starting distributions $\boldsymbol{\alpha} := (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m)$, total budget B and budget constraints $\boldsymbol{c} := (c_1, ..., c_m)$, the **Mu**lti-**La**yered **N**etwork **E**xploration problem, denoted as **MuLaNE**, is formulated as the following optimization problem,*

$$\max_{\boldsymbol{k}} \; r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k}) \; s.t. \; \boldsymbol{k} \in \mathbb{Z}_{\geq 0}^{m} \leq \boldsymbol{c}, \; \sum_{i=1}^{m} k_i \leq B, \tag{3}$$

where $r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k})$ is given by Eq. (1) or Eq. (2). We also need to consider the following settings:

Offline Setting. For the offline setting, all problem instance parameters $(\mathcal{G}, \boldsymbol{\alpha}, \boldsymbol{\sigma}, \boldsymbol{c}, B)$ are given, and we aim to find the optimal budget allocation $\boldsymbol{k}^*$ determined by Eq. (3).

Online Setting. For the online setting, we consider T-round explorations. Before the exploration, we only know an upper bound of the total number of nodes in $\mathcal{G}$ and the number of layers m,[2] but we do not know about the network structure $\mathcal{G}$, the starting distributions $\boldsymbol{\alpha}$ or node weights $\boldsymbol{\sigma}$. In round $t \in [T]$, we choose the budget allocation $\boldsymbol{k}_t := (k_{t,1}, \ldots, k_{t,m})$ only based on observations from previous rounds, where $k_{t,i} \in \mathbb{Z}_{\geq 0}$ is the budget allocated to the i-th layer and $\sum_{i=1}^{m} k_{t,i} \leq B, \boldsymbol{k}_t \leq \boldsymbol{c}$. Define $\boldsymbol{k}_t$ as the *action* taken in round t. By taking the action $\boldsymbol{k}_t$ we mean that we interact with the environment and the random explorer W_i, which is part of the environment, would

[2] More precisely we only need to know the number of independent explorers. If two explorers explore on the same layer, it is equivalent as two layers with identical graph structures.

explore $k_{t,i}$ steps and generate the exploration trajectory $\Phi(i,k_{t,i})=(X_{i,1},...,X_{i,k_{t,i}})$. After we take the action $\boldsymbol{k}_t$, the exploration trajectory $\Phi(i,k_{t,i})$ for each layer L_i as well as the fixed importance weight σ_u of $u\in\Phi(i,k_{t,i})$ is revealed as the *feedback*[3], which we leverage on to learn parameters related to the graph structures $\mathcal{G}$, the starting distribution $\boldsymbol{\alpha}$ and the node weights $\boldsymbol{\sigma}$, so that we can select better actions in future rounds. The reward we gain in round t is the total weights of unique nodes visited by all random explorers in round t. Our goal is to design an efficient online learning algorithm A to give us guidance on taking actions and gain as much cumulative reward as possible in T rounds.

In general, the online algorithm has to deal with the exploration-exploitation tradeoff. The cumulative regret is a commonly used metric to evaluate the performance of an online learning algorithm A. Formally, the T-round ((ξ,β)-approximation) regret of A is:

$$Reg_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(T)=\xi\beta T\cdot r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k}^*)-\mathbb{E}\big[\textstyle\sum_{t=1}^{T} r_{\mathcal{G},\boldsymbol{\alpha},\sigma}(\boldsymbol{k}_t^A)\big], \tag{4}$$

where $r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k}^*)$ is the reward value for the optimal budget allocation $\boldsymbol{k}^*$, $\boldsymbol{k}_t^A$ is budget allocation selected by the learning algorithm A in round t, the expectation is taken over the randomness of the algorithm and the exploration trajectories in all T rounds, and (ξ,β) is the approximation guarantee of the offline oracle as explained below. Similar to other online learning frameworks (Chen et al., 2016; Wang & Chen, 2017), we assume that the online learning algorithm has access to an offline (ξ,β)-approximation oracle, which for problem instance $(\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma},\boldsymbol{c},B)$ outputs an action $\boldsymbol{k}$ such that $\Pr\left(r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k})\geq\xi\cdot r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k}^*)\right)\geq\beta$. We also remark that the actual oracle we use takes certain intermediate parameters as inputs instead of $\mathcal{G}$ and $\boldsymbol{\alpha}$.

Submodularity and DR-Submodularity Over Integer lattices. To solve the MuLaNE problem, we leverage on the submodular and DR-submodular properties of the reward function. For any $\boldsymbol{x},\boldsymbol{y}\in\mathbb{Z}_{\geq 0}^m$, we denote $\boldsymbol{x}\vee\boldsymbol{y},\boldsymbol{x}\wedge\boldsymbol{y}\in\mathbb{Z}_{\geq 0}^m$ as the coordinate-wise maximum and minimum of these two vectors, i.e., $(\boldsymbol{x}\vee\boldsymbol{y})_i=\max\{x_i,y_i\}$, $(\boldsymbol{x}\wedge\boldsymbol{y})_i=\min\{x_i,y_i\}$. We define a function $f:\mathbb{Z}_{\geq 0}^m\to\mathbb{R}$ over the integer lattice $\mathbb{Z}_{\geq 0}^m$ as a *submodular* function if the following inequality holds for any $\boldsymbol{x},\boldsymbol{y}\in\mathbb{Z}_{\geq 0}^m$:

$$f(\boldsymbol{x}\vee\boldsymbol{y})+f(\boldsymbol{x}\wedge\boldsymbol{y})\leq f(\boldsymbol{x})+f(\boldsymbol{y}). \tag{5}$$

Let $\text{supp}^+(\boldsymbol{x}-\boldsymbol{y})$ denote the set $I=\{i\in[m]:x_i>y_i\}$, $\boldsymbol{\chi}_i=(0,...,1,...,0)$ be the one-hot vector whose i-th element is 1 and 0 otherwise, and $\boldsymbol{x}\geq\boldsymbol{y}$ means $x_i\geq y_i$ for all $i\in[m]$. We define a function $f:\mathbb{Z}_{\geq 0}^m\to\mathbb{R}$ as a *DR-submodular* (diminishing return submodular) function if the following inequality holds for any $\boldsymbol{x}\leq\boldsymbol{y}$ and $i\in[m]$,

$$f(\boldsymbol{y}+\boldsymbol{\chi}_i)-f(\boldsymbol{y})\leq f(\boldsymbol{x}+\boldsymbol{\chi}_i)-f(\boldsymbol{x}). \tag{6}$$

We say a function f is *monotone* if for any $\boldsymbol{x}\leq\boldsymbol{y}$, $f(\boldsymbol{x})\leq f(\boldsymbol{y})$. Note that for a function $f:\mathbb{Z}_{\geq 0}^m\to\mathbb{R}$, submodularity does not imply DR-submodularity over integer lattices. In fact, the former is weaker than the latter, that is, a DR-submodular function is always a submodular function, but not vice versa. However, for a typical submodular function $f:\{0,1\}^m\to\mathbb{R}$ defined on a set, they are equivalent.

[3]We further consider random node weights with unknown mean vector $\boldsymbol{\sigma}$, see the supplementary material for details.

3. Equivalent Bipartite Coverage Model

In order to derive the explicit formulation of the reward function $r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k})$ in Eq. (1), we construct an undirected bipartite coverage graph $\mathcal{B}(\mathcal{W},\mathcal{V},\mathcal{E}')$, where $\mathcal{W}=\{W_1,...,W_m\}$ denotes m random explorers, $\mathcal{V}=\bigcup_{i\in[m]}\mathcal{V}_i$ denotes all possible distinct nodes to be explored in $\mathcal{G}$, and the edge set $\mathcal{E}'=\{(W_i,u)|u\in\mathcal{V}_i,i\in[m]\}$ indicating whether node u could be visited by W_i. For each edge $(W_i,u)\in\mathcal{E}'$, we associate it with c_i+1 visiting probabilities denoted as $P_{i,u}(k_i)$ for $k_i\in\{0\}\cup[c_i]$. $P_{i,u}(k_i)$ represents the probability that the node u is visited by the random walker W_i given the budget k_i, i.e., $P_{i,u}(k_i)=\Pr(u\in\Phi(i,k_i))$. Since W_i can never visit the node outside the i-th layer (i.e., $u\notin\mathcal{G}_i$), we set $P_{i,u}(k_i)=0$ if $(W_i,u)\notin\mathcal{E}'$. Then, given budget allocation $\boldsymbol{k}$, the probability of a node u visited by at least one random explorer is $\Pr(u,\boldsymbol{k})=1-\prod_{i\in[m]}(1-P_{i,u}(k_i))$ because each W_i has the independent probability $P_{i,u}(k_i)$ to visit u.

By summing over all possible nodes, the reward function is

$$r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k})=\textstyle\sum_{u\in\mathcal{V}}\sigma_u\left(1-\prod_{i\in[m]}(1-P_{i,u}(k_i))\right) \tag{7}$$

According to Eq. (2), for non-overlapping multi-layered network, we can rewrite the reward function as:

$$r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k})=\textstyle\sum_{i\in[m]}\sum_{u\in\mathcal{V}_i}\sigma_u P_{i,u}(k_i). \tag{8}$$

Remark. The bipartite coverage model is needed to integrate the graph structure and the random walk exploration mechanisms into $P_{i,u}(k_i)$ to determine the reward function. It also handles the scenario where each layer is explored by multiple random walkers (see supplementary material).

3.1. Properties of the visiting probability $P_{i,u}(k_i)$

The quantities $P_{i,u}(k_i)$'s in Eq. (7) and (8) are crucial for both our offline and online algorithms, and thus we provide their analytical formulas and properties here. In order to analyze the property of the reward function, we apply the absorbing Markov Chain technique to calculate $P_{i,u}(k_i)$. For $u\notin\mathcal{G}_i$, $P_{i,u}(k_i)=0$; For $u\in\mathcal{G}_i$, we create an absorbing Markov Chain $\boldsymbol{P_i(u)}\in\mathbb{R}^{n_i\times n_i}$ by setting the target

node u as the absorbing node. We derive the corresponding transition matrix $\boldsymbol{P_i(u)}$ as $\boldsymbol{P_i(u)}[v,\cdot] = \boldsymbol{\chi}_u^\top$ if $v = u$ and $\boldsymbol{P_i(u)}[v,\cdot] = \boldsymbol{P_i}[v,\cdot]$ otherwise, where $\boldsymbol{P_i}[v,\cdot]$ denotes the row vector corresponding to the node v of $\boldsymbol{P_i}$, and $\boldsymbol{\chi}_u = (0,...,0,1,0,...,0)^\top$ denotes the one-hot vector with 1 at the u-th entry and 0 elsewhere.[4] Intuitively, $\boldsymbol{P_i(u)}$ corresponds to the transition probability matrix of $\mathcal{G}_i$ after removing all out-edges of u and adding a self loop to itself in the original graph $\mathcal{G}_i$. The random walker W_i will be trapped in u if it ever visits u. We observe that $P_{i,u}(k_i)$ equals to the probability the random walker stays in the absorbing node u at step $k_i \in \mathbb{Z}_{>0}$ (trivially, $P_{i,u}(0) = 0$),

$$P_{i,u}(k_i) = \boldsymbol{\alpha_i}^\top \boldsymbol{P_i(u)}^{k_i-1} \boldsymbol{\chi}_u. \tag{9}$$

Then define the marginal gain of $P_{i,u}(\cdot)$ at step $k_i \geq 1$ as,

$$g_{i,u}(k_i) = P_{i,u}(k_i) - P_{i,u}(k_i - 1). \tag{10}$$

The physical meaning of $g_{i,u}(k_i)$ is the probability that node u is visited exactly at the k_i-th step and not visited before k_i. Now, we can show $g_{i,u}(k_i)$ is non-negative,

Lemma 1. *$g_{i,u}(k_i) \geq 0$ for any $i \in [m]$, $u \in \mathcal{V}_i$, $k_i \in \mathbb{Z}_{>0}$.*

This means $P_{i,u}(k_i)$ is non-decreasing with respect to step k_i. In other words, the more budgets we allocate to W_i, the higher probability W_i can visit u in k_i steps.

Starting from arbitrary distributions: Although $P_{i,u}(k_i)$ is monotone (non-decreasing) w.r.t k_i, $g_{i,u}(k_i)$ may not be monotonic non-increasing under arbitrary staring distributions. Intuitively, there may exist one critical step k_i such that $P_{i,u}(k_i)$ suddenly increases by a large value (see supplementary material). This means that $P_{i,u}(k_i)$ lacks the diminishing return (or "discretely concave") property, which many problems rely on to provide good optimization (Kapralov et al., 2013; Soma & Yoshida, 2015). In other words, we are dealing with a more challenging non-concave optimization problem for the discrete budget allocation.

Starting from the stationary distribution: If our walkers start with the stationary distributions, the following property holds: $g_{i,u}(k_i)$ is non-increasing w.r.t k_i.

Lemma 2. *$g_{i,u}(k_i + 1) - g_{i,u}(k_i) \leq 0$ for any $i \in [m]$, $u \in \mathcal{V}_i$, $k_i \in \mathbb{Z}_{>0}$, if $\boldsymbol{\alpha}_i = \boldsymbol{\pi_i}$, where $\boldsymbol{\pi_i}^\top \boldsymbol{P_i} = \boldsymbol{\pi_i}^\top$.*

4. Offline Optimization for MuLaNE

In this section, we first consider the general case, where layers are overlapping and starting distributions are arbitrary. Next, we consider the starting distribution is the stationary distribution, and give solutions with better solution quality and time complexity. Then we analyze special cases where layers are non-overlapping and give optimal solutions for arbitrary distributions. The summary for offline models and algorithmic results are presented in Table 1.

[4]When related to matrix operations, we treat all vectors as column vectors by default.

4.1. Overlapping MuLaNE

Starting from arbitrary distributions. Based on the equivalent bipartite coverage model, one can observe that our problem formulation is a generalization of the Probabilistic Maximum Coverage (PMC) (Chen et al., 2016) problem, which is NP-hard and has a $(1 - 1/e)$ approximation based on submodular set function maximization. However, our problem is more general in that we want to select multi-sets from $\mathcal{W}$ with budget constraints, and the reward function in general does not have the DR-submodular property for an arbitrary starting distribution. Nevertheless, we have the following lemma to solve our problem.

Lemma 3. *For any network $\mathcal{G}$, distribution $\boldsymbol{\alpha}$ and weights $\boldsymbol{\sigma}$, $r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\cdot) : \mathbb{Z}_{\geq 0}^m \to \mathbb{R}$ is monotone and submodular.*

Leveraging on the monotone submodular property, we design a Budget Effective Greedy algorithm (Alg. 1). The core of Alg. 1 is the BEG procedure. Let $\delta(i, b, \boldsymbol{k})$ be the *per-unit marginal gain* $(r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k} + b\boldsymbol{\chi}_i) - r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k}))/b$ for allocating b more budgets to layer i, which equals to

$$\textstyle\sum_{u\in\mathcal{V}} \sigma_u \prod_{j\neq i}(1 - P_{j,u}(k_j))(P_{i,u}(k_i + b) - P_{i,u}(k_i))/b. \tag{11}$$

BEG procedure consists of two parts and maintains a queue $\mathcal{Q}$, where any pair $(i, b) \in \mathcal{Q}$ represents a tentative plan of allocating b more budgets to layer i. The first part is built around the while loop (line 6-10), where each iteration greedily selects the (i, b) pair in $\mathcal{Q}$ such that the *per-unit marginal gain* $\delta(i, b, \boldsymbol{k})$ is maximized. The second part is a for loop (line 12-13), where in the i-th round we attempt to allocate all c_i budgets to layer i and replace the current best budget allocation if we have a larger reward.

Theorem 1. *Algorithm 1 obtains a $(1 - e^{-\eta}) \approx 0.357$-approximate solution, where η is the solution of equation $e^\eta = 2 - \eta$, to the overlapping MuLaNE problem.*

Line 1 uses $O(m \|\boldsymbol{c}\|_\infty n_{max}^3)$ time to pre-calculate visiting probabilities based on Eq. (9), where $n_{max} = \max_i |\mathcal{V}_i|$. In the BEG procedure, the while loop contains B iterations, the size of queue $\mathcal{Q}$ is $O(m \|\boldsymbol{c}\|_\infty)$, and line 7 uses $O(n_{max})$ to calculate $\delta(i, b, \boldsymbol{k})$ by proper pre-computation and update, thus the time complexity of Algorithm 1 is $O(B \|\boldsymbol{c}\|_\infty mn_{max} + m \|\boldsymbol{c}\|_\infty n_{max}^3)$.

Remark 1. The idea of combining the greedy algorithm with enumerating solutions on one layer is also adopted in previous works (Khuller et al., 1999; Alon et al., 2012), but they only give a $\frac{1}{2}(1 - e^{-1}) \approx 0.316$-approximation analysis. In this paper, we provide a novel analysis with a better $(1 - e^{-\eta}) \approx 0.357$-approximation, .

Remark 2. Another algorithm (Alon et al., 2012) with a

Table 1. Summary of the offline models and algorithms.

Overlapping?	Starting distribution	Algorithm	Apprx ratio	Time complexity
✓	*Arbitrary*	Budget Effective Greedy	$(1-e^{-\eta})$[5]	$O(B\left\|\boldsymbol{c}\right\|_\infty mn_{max} + \left\|\boldsymbol{c}\right\|_\infty mn_{max}^3)$
✓	*Stationary*	Myopic Greedy	$(1-1/e)$	$O(Bmn_{max} + \left\|\boldsymbol{c}\right\|_\infty mn_{max}^3)$
×	*Arbitrary*	Dynamic Programming	1	$O(B\left\|\boldsymbol{c}\right\|_\infty m + \left\|\boldsymbol{c}\right\|_\infty mn_{max}^3)$
×	*Stationary*	Myopic Greedy	1	$O(B\log m + \left\|\boldsymbol{c}\right\|_\infty mn_{max}^3)$

Algorithm 1 Budget Effective Greedy (BEG) Algorithm for the Overlapping MuLaNE

Input: Network $\mathcal{G}$, starting distributions $\boldsymbol{\alpha}$, node weights $\boldsymbol{\sigma}$, budget B, constraints $\boldsymbol{c}$.
Output: Budget allocation $\boldsymbol{k}$.
1: Compute visiting probabilities $(P_{i,u}(b))_{i\in[m],u\in\mathcal{V},b\in[c_i]}$ according to Eq. (9).
2: $\boldsymbol{k} \leftarrow$ BEG($(P_{i,u}(b))_{i\in[m],u\in\mathcal{V},b\in[c_i]}$,$\boldsymbol{\sigma}$, B, $\boldsymbol{c}$).
3: **Procedure** BEG($(P_{i,u}(b))_{i\in[m],u\in\mathcal{V},b\in[c_i]}$, $\boldsymbol{\sigma}$, B, $\boldsymbol{c}$)
4: Let $\boldsymbol{k} := (k_1, ..., k_m) \leftarrow \mathbf{0}$, $K \leftarrow B$.
5: Let $\mathcal{Q} \leftarrow \{(i, b_i) \,|\, i \in [m], 1 \le b_i \le c_i\}$.
6: **while** $K > 0$ and $\mathcal{Q} \neq \emptyset$ **do**
7: $(i^*, b^*) \leftarrow \arg\max_{(i,b)\in\mathcal{Q}} \delta(i, b, \boldsymbol{k})/b$ ▷*Eq. (11)*
8: $k_{i^*} \leftarrow k_{i^*} + b^*$, $K \leftarrow K - b^*$.
9: Modify all pairs $(i, b) \in \mathcal{Q}$ to $(i, b - b^*)$.
10: Remove all paris $(i, b) \in \mathcal{Q}$ such that $b \le 0$.
11: **end while**
12: **for** $i \in [m]$ **do**
13: **if** $r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(c_i\boldsymbol{\chi}_i) > r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k})$, **then** $\boldsymbol{k} \leftarrow c_i\boldsymbol{\chi}_i$.
14: **end for**
15: **return** $\boldsymbol{k} := (k_1, ..., k_m)$.
16: **end Procedure**

better approximation ratio is to use partial enumeration techniques (i.e., BEGE), which can achieve $(1-1/e)$ approximation ratio. This is the best possible solution in polynomial time unless P=NP. But the time complexity is prohibitively high in $O(B^4m^4\left\|\boldsymbol{c}\right\|_\infty n_{max} + m\left\|\boldsymbol{c}\right\|_\infty n_{max}^3)$.

Starting from the stationary distribution. We also consider the special case where each random explorer W_i starts from the stationary distribution $\boldsymbol{\pi_i}$ with $\boldsymbol{\pi_i}^\top \boldsymbol{P} = \boldsymbol{\pi_i}^\top$. In this case, we have the following stronger DR-submodularity.

Lemma 4. *For any network $\mathcal{G}$, stationary distributions $\boldsymbol{\pi}$ and node weights $\boldsymbol{\sigma}$, function $r_{\mathcal{G},\boldsymbol{\pi},\boldsymbol{\sigma}}(\cdot) : \mathbb{Z}_{\ge 0}^m \to \mathbb{R}$ is monotone and DR-submodular.*

Since the reward function is DR-submodular, the BEG procedure can be replaced by the simple MG procedure in Alg. 2 with a better approximation ratio. The time complexity is also improved to $O(Bmn_{max} + m\left\|\boldsymbol{c}\right\|_\infty n_{max}^3)$.

Theorem 2. *Algorithm 2 obtains a $(1-1/e)$-approximate solution to the overlapping MulaNE with the stationary starting distributions.*

Algorithm 2 Myopic Greedy (MG) Algorithm for MuLaNE

1: Same input, output and line 1-2 as in Alg. 1, except replacing BEG with MG procedure below.
2: **Procedure** MG($(P_{i,u}(b))_{i\in[m],u\in\mathcal{V},b\in[c_i]}$, $\boldsymbol{\sigma}$, B, $\boldsymbol{c}$)
3: Let $\boldsymbol{k} := (k_1, ..., k_m) \leftarrow \mathbf{0}$, $K \leftarrow B$.
4: **while** $K > 0$ **do**
5: $i^* \leftarrow \arg\max_{i\in[m],k_i+1\le c_i} \delta(i, 1, \boldsymbol{k})$. ▷*Eq. (11)*
6: $k_{i^*} \leftarrow k_{i^*} + 1$, $K \leftarrow K - 1$.
7: **end while**
8: **return** $\boldsymbol{k} = (k_1, ..., k_m)$.
9: **end Procedure**

4.2. Non-overlapping MuLaNE

For non-overlapping MuLaNE, we are able to achieve the exact optimal solution: for the stationary starting distribution, a slight modification of the greedy algorithm Alg. 2 gives the optimal solution, while for an arbitrary starting distribution, we design a dynamic programming algorithm to compute the optimal solution (see supplementary material).

5. Online Learning for MuLaNE

In the online setting, we continue to study both overlapping and non-overlapping MuLaNE problem. However, the network structure $\mathcal{G}$, the distribution $\boldsymbol{\alpha}$ and node weights $\boldsymbol{\sigma}$ *are not known a priori*. Instead, the only information revealed to the decision maker includes total budget B, the number of layers m, the number of the target nodes $|\mathcal{V}|$ (or an upper bound of it) and the budget constraint $\boldsymbol{c}$.

5.1. Online Algorithm for Overlapping MuLaNE

For the unknown network structure and starting distributions, we bypass the transition matrices $\boldsymbol{P}_i$ and directly estimate the visiting probabilities $P_{i,u}(b) \in [0, 1]$. This avoids the analysis for how the estimated $\boldsymbol{P}_i$ affects the performance of online algorithms, which could be unbounded since we consider general graph structures and the reward function given by Eq. (7) is highly non-linear in $\boldsymbol{P}_i$. Moreover, we can save the matrix calculation by directly using $P_{i,u}(b)$, which is efficient even for large networks.

Specifically, we maintain a set of base arms $\mathcal{A} = \{(i, u, b) | i \in [m], u \in \mathcal{V}, b \in [c_i]\}$, where the total number $|\mathcal{A}| = \sum_{i\in[m]} c_i|\mathcal{V}|$. For each base arm $(i, u, b) \in \mathcal{A}$,

we denote $\mu_{i,u,b}$ as the true value of each base arm, i.e., $\mu_{i,u,b} = P_{i,u}(b)$. For the unknown node weights, we maintain the optimistic weight $\bar{\sigma}_u = 1$ if $u \in \mathcal{V}$ has not been visited. After u is first visited and its true value σ_u is revealed, we replace $\bar{\sigma}_u$ with σ_u. With a little abuse of the notation, we use $\boldsymbol{\mu}$ to denote the unknown intermediate parameters and $r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k})$ to denote the reward $r_{\mathcal{G},\boldsymbol{\alpha},\boldsymbol{\sigma}}(\boldsymbol{k})$.

We present our algorithm in Alg. 3, which is an adaptation of the CUCB algorithm for the general Combinatorial Multi-arm Bandit (CMAB) framework (Chen et al., 2016) to our setting. Notice that we use $\mathcal{A}$ as defined above in the algorithm, and $\mathcal{A}$ is defined using $\mathcal{V}$, which is the set of node ids and should not be known before the learning process starts. This is not an issue, because at the beginning we can create $|\mathcal{V}|$ (which is known) placeholders for the node ids, and once a new node is visited,we immediately replace one of the placeholders with the new node id. Thus $\mathcal{V}$ appearing in the above definition of $\mathcal{A}$ is just for notational convenience.

In Alg. 3, we maintain an unbiased estimation of the visiting probability $P_{i,u}(b)$, denoted as $\hat{\mu}_{i,u,b}$. Let $T_{i,u,b}$ record the number of times arm (i,u,b) is played so far and $\bar{\sigma}_v$ denote the optimistic importance weight. In each round $t \geq 1$, we compute the confidence radius $\rho_{i,u,b}$ in line 5, which controls the level of exploration. The confidence radius is larger when the arm (i,u,b) is not explored often (i.e. $T_{i,u,b}$ is small), and thus motivates more exploration. Due to the randomness of the exploration process, the upper condifence bound (UCB) value $\tilde{\mu}_{i,u,b}$ could be decreasing w.r.t b, but our offline oracle BEG can only accept non-decreasing UCB values (otherwise the $1-e^{-\eta}$ approximation is not guaranteed). Therefore, in line 8, we increase the UCB value $\tilde{\mu}_{i,u,b}$ and set it to be $\max_{j\in[b]} \tilde{\mu}_{i,u,j}$. This is the adaption of the CUCB algorithm to fix our case, and thus we name our algorithm as CUCB-MAX. After we apply the $1-e^{-\eta}$ approximate solution $\boldsymbol{k}$ given by the BEG oracle (i.e., Alg. 1), we get m trajectories as feedbacks. In line 11, we can update the unknown weights of visited nodes. In line 13, for base arms (i,u,b) with $b \leq k_i$, we update corresponding statistics by the Bernoulli random variable $Y_{i,u,b} \in \{0,1\}$ indicating whether node u is visited by W_i in first b steps.

Regret Analysis. We define the reward gap $\Delta_{\boldsymbol{k}} = \max(0, \xi r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k^*}) - r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k}))$ for all feasible action $\boldsymbol{k}$ satisfying $\sum_{i=1}^{m} k_i = B$, $0 \leq k_i \leq c_i$, where $\boldsymbol{k}^*$ is the optimal solution for parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ and $\xi = 1-e^{-\eta}$ is the approximation ratio of the $(\xi,1)$-approximate offline oracle. For each base arm (i,u,b), we define $\Delta_{\min}^{i,u,b} = \min_{\Delta_{\boldsymbol{k}}>0, k_i=b} \Delta_{\boldsymbol{k}}$ and $\Delta_{\max}^{i,u,b} = \max_{\Delta_{\boldsymbol{k}}>0, k_i=b} \Delta_{\boldsymbol{k}}$. As a convention, if there is no action $\boldsymbol{k}$ with $k_i = b$ such that $\Delta_{\boldsymbol{k}}>0$, we define $\Delta_{\min}^{i,u,b}=\infty$ and $\Delta_{\max}^{i,u,b}=0$. Let $\Delta_{\min} = \min_{(i,u,b)\in\mathcal{A}} \Delta_{\min}^{i,u,b}$ and $\Delta_{\max} = \max_{(i,u,b)\in\mathcal{A}} \Delta_{\max}^{i,u,b}$. The following theorem summarizes the regret bound for Alg. 3.

Algorithm 3 CUCB-MAX Algorithm for the MuLaNE

Input: Budget B, number of layers m, number of nodes $|\mathcal{V}|$, constraints $\boldsymbol{c}$, offline oracle BEG.

1: For each arm $(i,u,b) \in \mathcal{A}$, $T_{i,u,b} \leftarrow 0$, $\hat{\mu}_{i,u,b} \leftarrow 0$.
2: For each node $v \in \mathcal{V}$, $\bar{\sigma}_v \leftarrow 1$.
3: **for** $t = 1, 2, 3, ..., T$ **do**
4: **for** $(i,u,b) \in \mathcal{A}$ **do**
5: $\rho_{i,u,b} \leftarrow \sqrt{3 \ln t / (2T_{i,u,b})}$.
6: $\tilde{\mu}_{i,u,b} \leftarrow \min\{\hat{\mu}_{i,u,b} + \rho_{i,u,b}, 1\}$.
7: **end for**
8: For $(i,u,b) \in \mathcal{A}$, $\bar{\mu}_{i,u,b} \leftarrow \max_{j\in[b]} \tilde{\mu}_{i,u,j}$.
9: $\boldsymbol{k} \leftarrow \text{BEG}((\bar{\mu}_{i,u,b})_{(i,u,b)\in\mathcal{A}}, (\bar{\sigma}_v)_{v\in\mathcal{V}}, B, \boldsymbol{c})$.
10: Apply budget allocation $\boldsymbol{k}$, which gives trajectories $\boldsymbol{X} := (X_{i,1}, ..., X_{i,k_i})_{i\in[m]}$ as feedbacks.
11: For any visited node $v \in \bigcup_{i\in[m]} \{X_{i,1}, ..., X_{i,k_i}\}$, receive its node weight σ_v and set $\bar{\sigma}_v \leftarrow \sigma_v$.
12: For any $(i,u,b) \in \tau := \{(i,u,b) \in \mathcal{A} | \, b \leq k_i\}$, $Y_{i,u,b} \leftarrow 1$ if $u \in \{X_{i,1}, ..., X_{i,b}\}$ and 0 otherwise.
13: For $(i,u,b) \in \tau$, update $T_{i,u,b}$ and $\hat{\mu}_{i,u,b}$: $T_{i,u,b} \leftarrow T_{i,u,b} + 1$, $\hat{\mu}_{i,u,b} \leftarrow \hat{\mu}_{i,u,b} + (Y_{i,u,b} - \hat{\mu}_{i,u,b})/T_{i,u,b}$.
14: **end for**

Theorem 3. *Algorithm 3 has the following distribution-dependent* $(1-e^{-\eta}, 1)$ *approximation regret,*

$$Reg_{\boldsymbol{\mu},\boldsymbol{\sigma}}(T) \leq \sum_{(i,u,b)\in\mathcal{A}} \frac{108m|\mathcal{V}|\ln T}{\Delta_{\min}^{i,u,b}} + 2|\mathcal{A}| + \frac{\pi^2}{3}|\mathcal{A}|\Delta_{\max}.$$

Remark 1. Looking at the above distribution dependent bound, we have the $O(\log T)$ approximation regret, which is asymptotically tight. Coefficient $m|\mathcal{V}|$ in the leading term corresponds to the number of edges in the complete bipartite coverage graph. Notice that we cannot use the true edge number $\sum_{i\in[m]} |\mathcal{V}_i|$, because the learning algorithm does not know which nodes are contained in each layer, and has to explore all visiting possibilities given by the default complete bipartite graph. The set of base arms $\mathcal{A}$ has some redundancy due to the correlation between these base arms, and thus it is unclear if the summation over all base arms in the regret bound is tight. For the non-overlapping case, we further reduce the number of base arms to achieve better regret bounds, but for the overlapping case, how to further reduce base arms to achieve a tighter regret bound is a challenging open question left for the future work.

Remark 2. The $(1-e^{-\eta}, 1)$ approximate regret is determined by the offline oracle BEG that we plug in Line 9 and can be replaced by $(1-1/e, 1)$ regret using BEGE or even by the exact regret if the oracle can obtain the optimal budget allocation. The usage of BEG is a trade-off we make between computational efficiency and learning efficiency and empirically, it performs well as we shall see in Section 6.

Remark 3. The full proof of the above theorem is included

in the supplementary material, where we rely on the following properties of $r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k})$ to bound the regret.

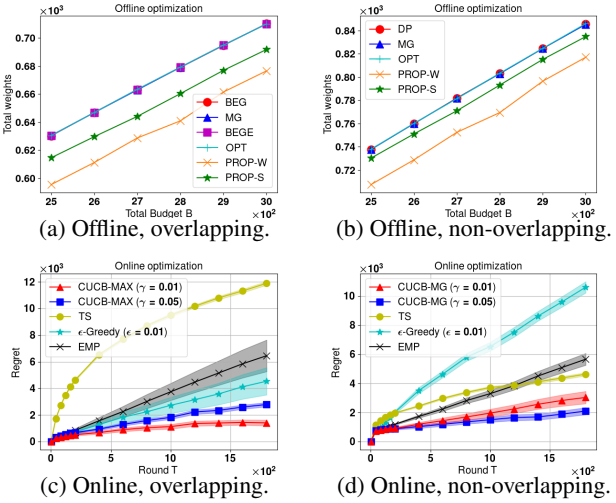
Property 1. *(Monotonicity). The reward $r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k})$ is monotonically increasing, i.e., for any budget allocation $\boldsymbol{k}$, any two vectors $\boldsymbol{\mu}=(\mu_{i,u,b})_{(i,u,b)\in\mathcal{A}}$, $\boldsymbol{\mu'}=(\mu'_{i,u,b})_{(i,u,b)\in\mathcal{A}}$ and any node weights $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}'$, we have $r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k}) \le r_{\boldsymbol{\mu'},\boldsymbol{\sigma'}}(\boldsymbol{k})$, if $\mu_{i,u,b} \le \mu'_{i,u,b}$ and $\sigma_v \le \sigma'_v$, $\forall (i,u,b) \in \mathcal{A}, v \in \mathcal{V}$.*

Property 2. *(1-Norm Bounded Smoothness). The reward function $r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k})$ satisfies the 1-norm bounded smoothness condition, i.e., for any budget allocation $\boldsymbol{k}$, any two vectors $\boldsymbol{\mu}=(\mu_{i,u,b})_{(i,u,b)\in\mathcal{A}}$, $\boldsymbol{\mu'}=(\mu'_{i,u,b})_{(i,u,b)\in\mathcal{A}}$ and any node weights $\boldsymbol{\sigma}$, $\boldsymbol{\sigma}'$, we have $|r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k}) - r_{\boldsymbol{\mu'},\boldsymbol{\sigma'}}(\boldsymbol{k})| \le \sum_{i\in[m],u\in\mathcal{V},b=k_i}(\sigma_u|\mu_{i,u,b}-\mu'_{i,u,b}| + |\sigma_u-\sigma'_u|\,\mu'_{i,u,b})$.*

We emphasize that our algorithm and analysis differ from the original CUCB algorithm (Chen et al., 2016) as follows. First, we have the additional regret caused by the overestimated weights for unvisited nodes, i.e., $|\sigma_u - \sigma'_u|\,\mu'_{i,u,b}$ term in property 2. We carefully bound this term based on the observation that $\mu'_{i,u,b}$ is small and decreasing quickly before u is first visited. Next, we have to take the max (line 8) to guarantee the UCB value is monotone w.r.t b since our BEG oracle can only output $(1-e^{-\eta},1)$-approximation with monotone inputs. Due to the above operation, (i,u,b)'s UCB value depends on the feedback from all arms (i,u,j) for $j \le b$ (set τ in line 12). So we should update all these arms (line 13) to guarantee that the estimates to all these arms are accurate enough. Finally, directly following the standard CMAB result would have a larger regret, because arms in τ are defined as triggered arms, but only arms in $\tau' = \{(i,u,b) \in \mathcal{A} | k_i = b\}$ affect the rewards. So we conceptually view arms in τ' as triggered arms and use a tighter 1-Norm Bounded Smoothness condition as given above to derive a tighter regret bound. This improves the coefficient of the leading $\ln T$ term in the distribution dependent regret by a factor of $|\tau|/|\tau'| = O(B/m)$, and the $1/\Delta^{i,u,b}_{\min}$ term is smaller since the original definition would have $\Delta^{i,u,b}_{\min} = \min_{\Delta_{\boldsymbol{k}}>0, b\le k_i} \Delta_{\boldsymbol{k}}$.

5.2. Online Algorithm for Non-overlapping Case

For the non-overlapping case, we set *layer-wise marginal gains* as our base arms. Concretely, we maintain a set of base arms $\mathcal{A} = \{(i,b) | i \in [m], b \in [c_i]\}$. For each base arm $(i,b) \in \mathcal{A}$., let $\mu_{i,b} = \sum_{u\in\mathcal{V}} \sigma_u(P_{i,u}(b) - P_{i,u}(b-1))$ be the true marginal gain of assigning budget b in layer i. We apply the standard CUCB algorithm to this setting and call the resulting algorithm CUCB-MG (see supplementary material). Note that in the non-overlapping setting we can solve the offline problem exactly, so we can use the exact offline oracle to solve the online problem and achieve an exact regret bound. This is the major advantage over the overlapping setting where we can only achieve an approximate bound. Define $\Delta_{\boldsymbol{k}} = r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k}^*) - r_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\boldsymbol{k})$ for all feasible action $\boldsymbol{k}$, and $\Delta^{i,b}_{\min} = \min_{\Delta_{\boldsymbol{k}}>0, k_i \ge b} \Delta_{\boldsymbol{k}}$, CUCB-MG has a $O(\sum_{(i,b)\in\mathcal{A}} 48B\ln T/\Delta^{i,b}_{\min})$ regret bound.



(a) Offline, overlapping. (b) Offline, non-overlapping. (c) Online, overlapping. (d) Online, non-overlapping.

Figure 1. Above: total weights of unique nodes visited for offline algorithms. Below: regret for online algorithms when $B = 3000$.

Table 2. Statistics for FF-TW-YT network

Layer	FriendFeed	Twitter	YouTube
# of vertices	5,540	5,702	663
# of edges	31,921	42,327	614

6. Experiments

Dataset and settings. We conduct experiments on a real-world multi-layered network FF-TW-YT, which contains $m = 3$ layers representing users' social connections in FriendFeed (FF), Twitter (TW) and YouTube (YT) (Dickison et al., 2016). In total, FF-TW-YT has $6,407$ distinct vertices representing users and $74,836$ directed edges representing connections ("who follows whom") among users. The statistics for each layer is summarized in Table 2. We transform the FF-TW-YT network $\mathcal{G}(\mathcal{V},\mathcal{E})$ into a symmetric directed network (by adding a new edge (v,u) if $(u,v) \in \mathcal{E}$ but $(v,u) \notin \mathcal{E}$) because a user can be visited via her followers or followees. Each edge weight is set to be 1 and the node weights are set to be $\sigma_u \in \{0, 0.5, 1\}$ uniformly at random. Each random walker always starts from the smallest node-id in each layer and we set constraints c_i equal to the total budget B. Note that in order to test the non-overlapping case, we use the same network but relabel node-ids so that they do not overlap between different layers. To handle the randomness, we repeat $10,000$ times and present the averaged total weights of unique nodes visited for offline optimization. We calculate the regret by comparing with the *optimal* solution, which is stronger than comparing with

Table 3. Running time (seconds) for offline and online algorithms.

	B=2.6k	B=2.8k	B=3.0k
BEG	0.274	0.316	0.363
BEGE	34.37	45.51	59.20
OPT	91.16	98.42	105.63

(a) Running time of offline algorithms for the overlapping case with different budgets B.

	B=2.6k	B=2.8k	B=3.0k
DP	0.038	0.044	0.050
MG	0.008	0.009	0.010
OPT	70.10	75.78	81.11

(b) Running time of offline algorithms for the non-overlapping case with different budgets B.

Overlapping?	✓	×
BEG	1.22	NA
BEGE	60.03	NA
DP	NA	0.86
OPT	107.33	82.01

(c) Per-round running time for CUCB-MAX (or CUCB-MG) with different oracles when B=3.0k.

$(1 - e^{-\eta}, 1)$-approximate solution as defined in Eq. (4). We average over 200 independent experiments to provide the mean regret with 95% confidence interval. To evaluate the computational efficiency, we also present the running time for both offline and online algorithms in Table 3.

Algorithms in comparison. For the offline setting, we present the results for Alg. 1 (denoted as BEG), Alg. 2 (denoted as MG), BEG with partial enumeration (denoted as BEGE) and the dynamic programming algorithm in the supplementary material (denoted as DP). We provide two baselines PROP-S and PROP-W, which allocates the budget proportional to the layer size and proportional to the total weights if we allocate $B/3$ budgets to that layer, respectively. The optimal solution (denoted as OPT) is also provided by enumerating all possible budget allocations. For online settings, we consider CUCB-MAX (Alg. 3) and CUCB-MG (Sec. 5.2) algorithms. We shrink the confidence interval by γ, i.e., $\rho_{i,u,b} \leftarrow \gamma\rho_{i,u,b}$, to speed up the learning, though our theoretical regret bound requires $\gamma = 1$. For baselines, we consider the EMP algorithm which always allocates according to the empirical mean, and the ϵ-Greedy algorithm which allocates budgets according to empirical mean with probability $1-\epsilon$ and allocates all B budgets to the i-th layer with probability ϵ/m. We also compare with the Thompson sampling (TS) method (Wang & Chen, 2018), which uses Beta distribution Beta(α, β) (where $\alpha = \beta = 1$ initially) as prior distribution for each base arm.

Experimental results. We show the results for MuLaNE problems in Figure 1. For the offline overlapping case, both BEG and MG outperform two baselines PROP-W and PROP-S in receiving total weights. Although not guaranteed by the theory, BEG are empirically close to BEGE and the optimal solution (OPT). As for the computational efficiency, in Table 3a, the BEG is at least two orders of magnitude (e.g.,163 times when B=3.0k) faster than BEGE and OPT. Combining that the reward of BEG is empirically close to BEGE and the optimal solution, this shows that BEG is empirically better than BEGE and OPT. For the offline non-overlapping case, the results are similar, but the difference is that we have the theoretical guarantee for the optimality of DP. For the online setting, all CUCB-MAX/CUCB-MG curves outperform the baselines. This demonstrates empirically that CUCB-MAX algorithm can effectively learn the unknown parameters while optimizing the objective. For the computational efficiency of online learning algorithms, since the running time for algorithms with the same oracle is similar, we present the running time for CUCB-MAX with different oracles in Table 3c. CUCB-MAX with BEG is 50 times faster than BEGE, which is consistent with our theoretical analysis. The results for different budgets B are consistent with $B = 3000$, which are included in the supplementary material. Results and analysis for stationary starting distributions are also in thesupplementary material.

7. Conclusions and Future Work

This paper formulates the multi-layered network exploration via random walks (MuLaNE) as a budget allocation problem, requiring that the total weights of distinct nodes visited on the multi-layered network is maximized. For the offline setting, we propose four algorithms for MuLaNE according to the specification of multi-layered network (overlapping or non-overlapping) and starting distributions (arbitrary or stationary), each of which has a provable guarantee on approximation factors and running time. We further study the online setting where network structure and the node weights are not known a priori. We propose the CUCB-MAX algorithm for overlapping MuLaNE and the CUCB-MG algorithm for the non-overlapping case, both of which are bounded by a $O(\log T)$ (approximate) regret. Finally, we conduct experiments on a social network dataset to show the empirical performance of our algorithms.

There are many compelling directions for the future study. For example, it would be interesting to extend our problem where the decision maker can jointly optimize the starting distribution and the budget allocation. One could also study the adaptive MuLaNE by using the feedback from the exploration results of the previous steps to determine the exploration strategy for future steps.

8. Acknowledgement

The work of John C.S. Lui was supported in part by the GRF 14200420.

References

Alon, N., Gamzu, I., and Tennenholtz, M. Optimizing budget allocation among channels and influencers. In *Proceedings of the 21st international conference on World Wide Web*, pp. 381–388. ACM, 2012.

Bubeck, S. and Cesa-Bianchi, N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

Chen, W., Wang, Y., and Yang, S. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 199–208, 2009.

Chen, W., Wang, Y., Yuan, Y., and Wang, Q. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *The Journal of Machine Learning Research*, 17(1):1746–1778, 2016.

Chen, X., Huang, W., Chen, W., and Lui, J. C. Community exploration: From offline optimization to online learning. In *Advances in Neural Information Processing Systems*, pp. 5474–5483, 2018.

Dickison, M. E., Magnani, M., and Rossi, L. *Multilayer social networks*. Cambridge University Press, 2016.

Gai, Y., Krishnamachari, B., and Jain, R. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5): 1466–1478, 2012.

Gleich, D. F. Pagerank beyond the web. *SIAM Review*, 57 (3):321–363, 2015.

Kapralov, M., Post, I., and Vondrák, J. Online submodular welfare maximization: Greedy is optimal. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1216–1225. SIAM, 2013.

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

Khuller, S., Moss, A., and Naor, J. S. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999.

Lerman, K. and Jones, L. Social browsing on flickr. *arXiv preprint cs/0612047*, 2006.

Li, Y., Wu, Z., Lin, S., Xie, H., Lv, M., Xu, Y., and Lui, J. C. Walking with perception: Efficient random walk sampling via common neighbor awareness. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 962–973. IEEE, 2019.

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pp. 84–95, 2002.

Pons, P. and Latapy, M. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pp. 284–293. Springer, 2005.

Robbins, H. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

Soma, T. and Yoshida, Y. A generalization of submodular cover via the diminishing return property on the integer lattice. In *Advances in Neural Information Processing Systems*, pp. 847–855, 2015.

Soma, T., Kakimura, N., Inaba, K., and Kawarabayashi, K.-i. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *International Conference on Machine Learning*, pp. 351–359, 2014.

Wang, C., Chen, W., and Wang, Y. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012.

Wang, Q. and Chen, W. Improving regret bounds for combinatorial semi-bandits with probabilistically triggered arms and its applications. In *Advances in Neural Information Processing Systems*, pp. 1161–1171, 2017.

Wang, S. and Chen, W. Thompson sampling for combinatorial semi-bandits. In *International Conference on Machine Learning*, pp. 5114–5122, 2018.

Wilder, B., Immorlica, N., Rice, E., and Tambe, M. Maximizing influence in an unknown social network. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.