

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

¹ The Chinese University of Hong Kong



² Carnegie Mellon University



³ Bytedance



⁴ Microsoft



Overview

Motivation

Problem Formulation

Offline Optimization

 Equivalent Bipartite Coverage Model

 Offline Algorithms

Online Learning

Experiments

Summary

Full paper: <https://arxiv.org/abs/2106.05065>.

Network Exploration via Random Walks

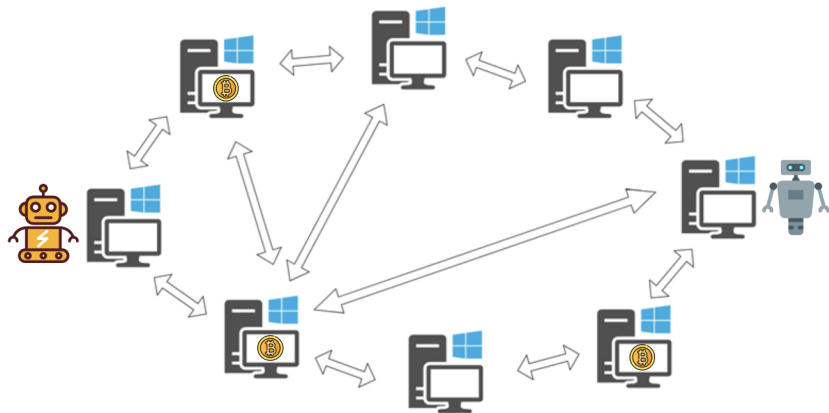
- Network exploration (NE) is a fundamental paradigm of **discovering information or resources** available at nodes in a network.

Network Exploration via Random Walks

- Network exploration (NE) is a fundamental paradigm of **discovering information or resources** available at nodes in a network.
- Random walk is often used as **an effective tool** for NE. [Lv et al. 2002; Gleich 2015; Wilder et al. 2018]
- Representative applications.
 - **Resource searching in peer-to-peer (P2P) networks.**
 - **Web surfing in online social networks (OSNs).**
 - Terrain patrolling to protest forests from poachers.
 - Finding missing youth in suburb/rural areas.

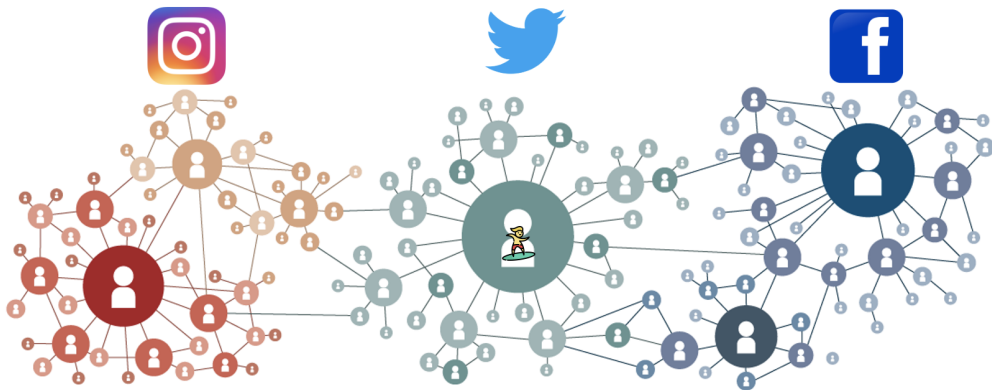
Resource Searching in P2P Networks

Question: how to allocate **limited budgets** to different random walkers, so as to find as many **important** resources as possible?



Web Surfing in OSNs

Question: how to allocate one's **limited time** in exploring different OSNs, so as to get the maximum amount of **useful** information.



MuLaNE Problem

We abstract the above-mentioned applications as the following **Multi-Layered Network Exploration** (MuLaNE) problem.

MuLaNE Problem

We abstract the above-mentioned applications as the following **Multi-Layered Network Exploration** (MuLaNE) problem.

- In MuLaNE, there are **multiple network layers**, where
 - each layer is **explored by a random walk** and

MuLaNE Problem

We abstract the above-mentioned applications as the following **Multi-Layered Network Exploration** (MuLaNE) problem.

- In MuLaNE, there are **multiple network layers**, where
 - each layer is **explored by a random walk** and
 - each node has an **importance weight**.

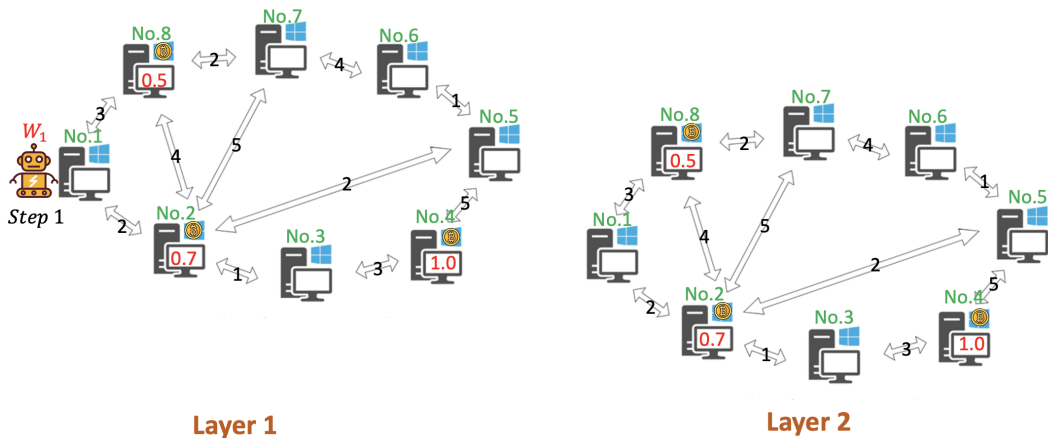
MuLaNE Problem

We abstract the above-mentioned applications as the following **Multi-Layered Network Exploration** (MuLaNE) problem.

- In MuLaNE, there are **multiple network layers**, where
 - each layer is **explored by a random walk** and
 - each node has an **importance weight**.
- The MuLaNE task is to **allocate total random walk budget B** into different network layers so that the total weights of the **unique** nodes visited are maximized.

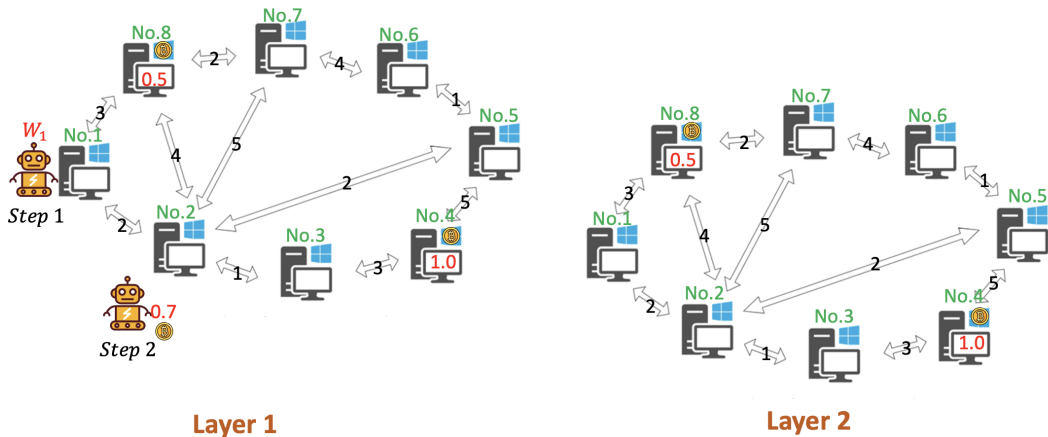
A Concrete Example

- The random walker W_1 in layer 1, which starts from node 1, is allocated $k_1 = 4$ budgets to walk four steps.



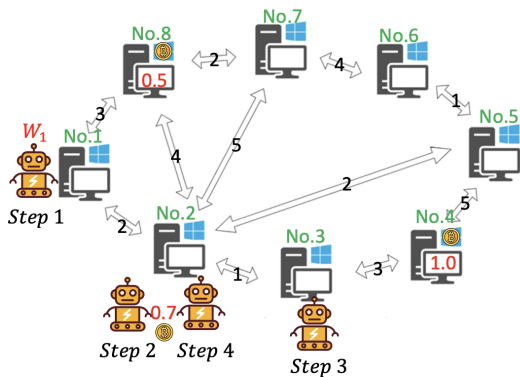
A Concrete Example

- For the second step, it goes from node 1 to node 8 with probability $3/5$ or goes to node 2 with probability $2/5$. After W_1 moves to node 2, it get 0.7 weights.

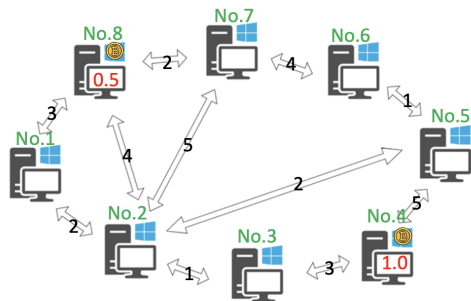


A Concrete Example

- Suppose the trajectory of W_1 is $\Phi(1, 4) = (1, 2, 3, 2)$, the total weights of unique nodes visited for layer 1 are 0.7.



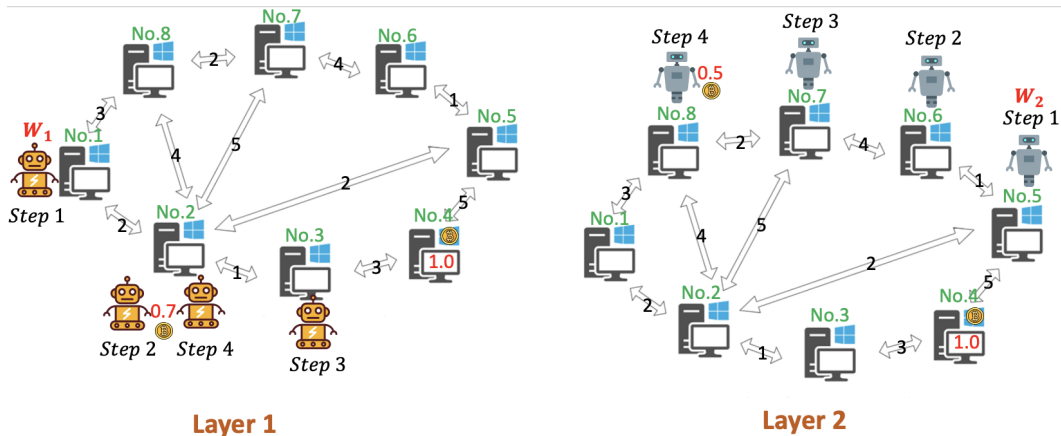
Layer 1



Layer 2

A Concrete Example

- Similarly, W_2 's trajectory is $\Phi(2, 4) = (5, 6, 7, 8)$ when $k_2 = 4$, so the total weights of unique nodes visited for these two random walkers are $0.7 + 0.5 = 1.2$.



MuLaNE Settings

Network Structure

- The overall network to be explored (e.g., combining different P2P networks or OSNs) as a multi-layered network \mathcal{G} that consists of m layers L_1, \dots, L_m .

MuLaNE Settings

Network Structure

- The overall network to be explored (e.g., combining different P2P networks or OSNs) as a multi-layered network \mathcal{G} that consists of m layers L_1, \dots, 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

MuLaNE Settings

Network Structure

- The overall network to be explored (e.g., combining different P2P networks or OSNs) as a multi-layered network \mathcal{G} that consists of m layers L_1, \dots, 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 $u \in \mathcal{V}_i$),

MuLaNE Settings

Network Structure

- The overall network to be explored (e.g., combining different P2P networks or OSNs) as a multi-layered network \mathcal{G} that consists of m layers L_1, \dots, 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)

MuLaNE Settings

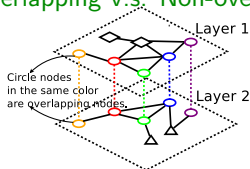
Network Structure

- The overall network to be explored (e.g., combining different P2P networks or OSNs) as a multi-layered network \mathcal{G} that consists of m layers L_1, \dots, 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)
 - w_i is the edge weight function on edges \mathcal{E}_i that navigates the random walk.

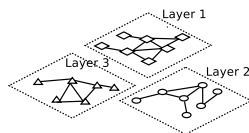
MuLaNE Settings

Network Structure

- The overall network to be explored (e.g., combining different P2P networks or OSNs) as a multi-layered network \mathcal{G} that consists of m layers L_1, \dots, 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)
 - w_i is the edge weight function on edges \mathcal{E}_i that navigates the random walk.
 - **Overlapping v.s. Non-overlapping multi-layered networks.**



(a) Overlapping.



(b) Non-overlapping.

Figure 1: Two types of multi-layered networks.

MuLaNE Settings (Cont.)

Random Walk Exploration

- Each layer L_i is associated with an explorer (or random walker) W_i and a fixed starting distribution α_i on nodes \mathcal{V}_i .

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

MuLaNE Settings (Cont.)

Random Walk Exploration

- Each layer L_i is associated with an explorer (or random walker) W_i and a fixed starting distribution α_i on nodes \mathcal{V}_i .
 - Explorer W_i starts on a node in \mathcal{G}_i following the distribution α_i ,

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

MuLaNE Settings (Cont.)

Random Walk Exploration

- Each layer L_i is associated with an explorer (or random walker) W_i and a fixed starting distribution α_i on nodes \mathcal{V}_i .
 - Explorer W_i starts on a node in \mathcal{G}_i following the distribution α_i ,
 - W_i walks on network \mathcal{G}_i following outgoing edges with probability proportional to edge weights.

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

MuLaNE Settings (Cont.)

Random Walk Exploration

- Each layer L_i is associated with an explorer (or random walker) W_i and a fixed starting distribution α_i on nodes \mathcal{V}_i .
 - Explorer W_i starts on a node in \mathcal{G}_i following the distribution α_i ,
 - W_i walks on network \mathcal{G}_i following outgoing edges with probability proportional to edge weights.
 - Each random walk step will cost one unit of the budget¹.

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

MuLaNE Settings (Cont.)

Rewards

- Let $\Phi(i, k_i) := (X_{i,1}, \dots, X_{i,k_i})$ be the exploration trajectory for explorer W_i after exploring k_i nodes.

MuLaNE Settings (Cont.)

Rewards

- Let $\Phi(i, k_i) := (X_{i,1}, \dots, X_{i,k_i})$ be the exploration trajectory for explorer W_i after exploring k_i nodes.
- The reward for $\Phi(i, k_i)$ is defined as the **total weights of *unique* nodes visited by W_i** , i.e., $\sum_{v \in \cup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v$.

MuLaNE Settings (Cont.)

Rewards

- Let $\Phi(i, k_i) := (X_{i,1}, \dots, X_{i,k_i})$ be the exploration trajectory for explorer W_i after exploring k_i nodes.
- The reward for $\Phi(i, k_i)$ is defined as the **total weights of unique nodes visited by W_i** , i.e., $\sum_{v \in \cup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v$.
- Considering trajectories of all W_i 's, the total reward is the total weights of *unique* nodes visited by all random walkers, i.e., $\sum_{v \in \cup_{i=1}^m \cup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v$.

MuLaNE Settings (Cont.)

Rewards

- Let $\Phi(i, k_i) := (X_{i,1}, \dots, X_{i,k_i})$ be the exploration trajectory for explorer W_i after exploring k_i nodes.
- The reward for $\Phi(i, k_i)$ is defined as the **total weights of unique nodes visited by W_i** , i.e., $\sum_{v \in \cup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v$.
- Considering trajectories of all W_i 's, the total reward is the total weights of *unique* nodes visited by all random walkers, i.e., $\sum_{v \in \cup_{i=1}^m \cup_{j=1}^{k_i} \{X_{i,j}\}} \sigma_v$.

Given network \mathcal{G} , starting distributions α and node weights σ , the expected total reward for budget allocation $\mathbf{k} = (k_1, \dots, k_m)$ is

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

The goal of MuLaNE

We aim to solve the following optimization problem,

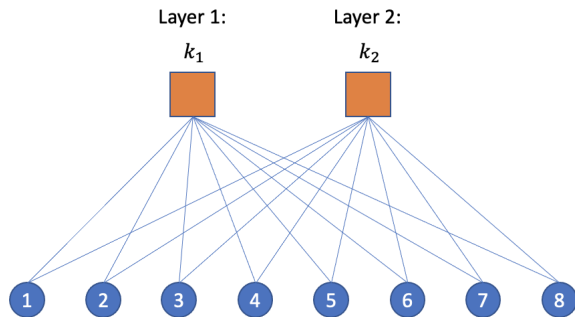
$$\text{Maximize } r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k}) \text{ s.t. } \mathbf{k} \in \mathbb{Z}_{\geq 0}^m \leq \mathbf{c}, \sum_{i=1}^m k_i \leq B, \quad (2)$$

In real applications,

- different layers may have overlapping vertices (e.g., home-page of the same user may appear in different OSNs), and α_i 's may or may not be stationary distributions;
- The network \mathcal{G} , the node weights σ and the starting distributions α_i 's may not be known in advance.

In this paper, we provide a *systematic* study of all these settings.

Equivalent Bipartite Coverage Model (cont.)



Let $P_{i,u}(k_i) = \Pr(u \in \Phi(i, k_i))$ represents the probability that the node u is visited by the random walker W_i given the budget k_i .

By summing over all possible nodes, the reward function is

$$r_{\mathcal{G},\alpha,\sigma}(\mathbf{k}) = \sum_{u \in \mathcal{V}} \sigma_u \left(1 - \prod_{i \in [m]} (1 - P_{i,u}(k_i)) \right) \quad (3)$$

How to Derive the Explicit Form of $P_{i,u}(k_i)$?

We create **absorbing Markov chains** $\mathbf{P}_i(\mathbf{u}) \in \mathbb{R}^{n_i \times n_i}$ by setting the target node u as the absorbing node.

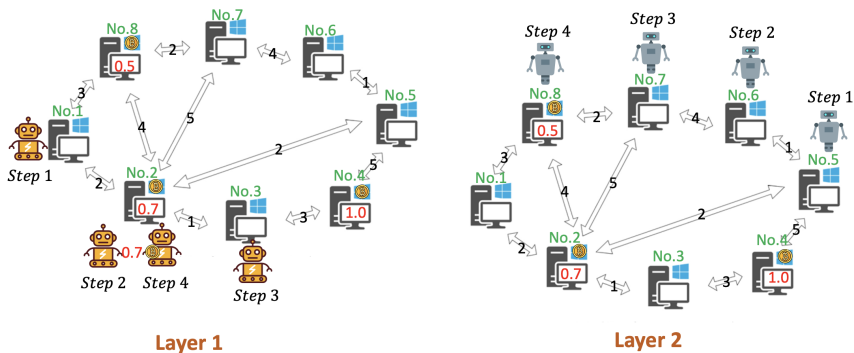
$$P_{i,u}(k_i) = \alpha_i^\top \mathbf{P}_i(\mathbf{u})^{k_i-1} \chi_u. \quad (4)$$

- α_i is the starting distribution of random walker W_i .
- $\chi_u = (0, \dots, 0, 1, 0, \dots, 0)^\top$ denotes the one-hot vector with 1 at the u -th entry and 0 elsewhere.
- Intuitively, $\mathbf{P}_i(\mathbf{u})$ 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 .
- $\mathbf{P}_i(\mathbf{u})[v, \cdot] = \chi_u^\top$ if $v = u$ and $\mathbf{P}_i(\mathbf{u})[v, \cdot] = \mathbf{P}_i[v, \cdot]$ otherwise, where $\mathbf{P}_i[v, \cdot]$ denotes the row vector corresponding to the node v of \mathbf{P}_i .

An Concrete Example

For example, the absorbing transition matrix by setting the node 2 as the target node

for layer 1. $P_1(2) = \begin{bmatrix} 0 & 2/5 & 0 & 0 & 0 & 0 & 0 & 3/5 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & & & & & & \\ 3/9 & 4/9 & 0 & 0 & 0 & 0 & 2/9 & 0 \end{bmatrix}$



Properties of the Visiting Probability $P_{i,u}(k_i)$

Plugging in Eq. (4), our reward function is

$$r_{\mathcal{G},\alpha,\sigma}(\mathbf{k}) = \sum_{u \in \mathcal{V}} \sigma_u \left(1 - \prod_{i \in [m]} (1 - \alpha_i^\top \mathbf{P}_i(\mathbf{u})^{k_i-1} \chi_u) \right).$$

- Still not easy to solve the problem given its explicit form.

Properties of the Visiting Probability $P_{i,u}(k_i)$

Plugging in Eq. (4), our reward function is

$$r_{\mathcal{G},\alpha,\sigma}(\mathbf{k}) = \sum_{u \in \mathcal{V}} \sigma_u \left(1 - \prod_{i \in [m]} (1 - \alpha_i^\top \mathbf{P}_i(\mathbf{u})^{k_i-1} \chi_u) \right).$$

- Still not easy to solve the problem given its explicit form.
- Idea: We can leverage on the **submodular maximization technique** and greedily solve this problem approximately.

Properties of the Visiting Probability $P_{i,u}(k_i)$

Plugging in Eq. (4), our reward function is

$$r_{\mathcal{G},\alpha,\sigma}(\mathbf{k}) = \sum_{u \in \mathcal{V}} \sigma_u \left(1 - \prod_{i \in [m]} (1 - \alpha_i^\top \mathbf{P}_i(\mathbf{u})^{k_i-1} \chi_u) \right).$$

- Still not easy to solve the problem given its explicit form.
- Idea: We can leverage on the **submodular maximization technique** and greedily solve this problem approximately.

Properties of $P_{i,u}(k_i)$.

- $P_{i,u}(k_i)$ is **monotonic increasing**. The more budgets, the higher probability that u is visited.
- Define $g_{i,u}(k_i)$ as the marginal gain of $P_{i,u}(k_i)$,

$$g_{i,u}(k_i) = P_{i,u}(k_i) - P_{i,u}(k_i - 1). \quad (5)$$

Submodular and DR-submodular Property


- To solve the MuLaNE problem, we leverage on the submodular and DR-submodular properties of the reward function.
- Function $f : \mathbb{Z}_{\geq 0}^m \rightarrow \mathbb{R}$ over the integer lattice $\mathbb{Z}_{\geq 0}^m$ is called a *submodular*² function iff for any $\mathbf{x} \in \mathbb{Z}_{\geq 0}^m$ and $i \neq j$:

$$f(\mathbf{x} + \chi_j + \chi_i) - f(\mathbf{x} + \chi_j) \leq f(\mathbf{x} + \chi_i) - f(\mathbf{x}). \quad (6)$$

- Function $f : \mathbb{Z}_{\geq 0}^m \rightarrow \mathbb{R}$ is called a *DR-submodular* (diminishing return submodular) function iff for any $\mathbf{x} \leq \mathbf{y}$ and $i \in [m]$,

$$f(\mathbf{y} + \chi_i) - f(\mathbf{y}) \leq f(\mathbf{x} + \chi_i) - f(\mathbf{x}). \quad (7)$$

- Note that submodularity is *weaker* than DR-submodularity, that is, a DR-submodular function is always a submodular function, but not vice versa.

²This is the equivalent condition proofed by this work, not the original condition $f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$ 

Key Lemma for Arbitrary Distributions

For arbitrary starting distributions, we have the following lemma.

Lemma 1.

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

- The intuition for submodularity: for layer i , the marginal gain is decreasing when more budgets are allocated to other layers $j \in [m] \setminus i$.

Budget Effective Greedy Algorithm (BEG)

- Leveraging on the monotone submodular property, we design a Budget Effective Greedy algorithm (BEG, Algorithm 1).
- Core: the greedy step and the final for loop.

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

Input: Network \mathcal{G} , starting distributions α , node weights σ , budget B , constraints c .

Output: Budget allocation \mathbf{k} .

- 1: Compute visiting probabilities $(P_{i,u}(b))_{i \in [m], u \in \mathcal{V}, b \in [c_i]}$ according to Eq. (9). Eq. (4) in this slide.
 - 2: $\mathbf{k} \leftarrow \text{BEG}((P_{i,u}(b))_{i \in [m], u \in \mathcal{V}, b \in [c_i]}, \sigma, B, c)$.
 - 3: **Procedure** $\text{BEG}((P_{i,u}(b))_{i \in [m], u \in \mathcal{V}, b \in [c_i]}, \sigma, B, c)$
 - 4: Let $\mathbf{k} := (k_1, \dots, k_m) \leftarrow \mathbf{0}$, $K \leftarrow B$.
 - 5: Let $\mathcal{Q} \leftarrow \{(i, b_i) \mid i \in [m], 1 \leq b_i \leq c_i\}$.
 - 6: **while** $K > 0$ and $\mathcal{Q} \neq \emptyset$ **do** Eq. (8) in this slide.
 - 7: $(i^*, b^*) \leftarrow \arg \max_{(i,b) \in \mathcal{Q}} \delta(i, b, \mathbf{k})/b$ ▷ Eq. (11)
 - 8: $k_{i^*} \leftarrow k_{i^*} + b^*$, $K \leftarrow K - b^*$. Greedy step.
 - 9: Modify all pairs $(i, b) \in \mathcal{Q}$ to $(i, b - b^*)$.
 - 10: Remove all pairs $(i, b) \in \mathcal{Q}$ such that $b \leq 0$.
 - 11: **end while**
 - 12: **for** $i \in [m]$ **do**
 - 13: **if** $r_{\mathcal{G}, \alpha, \sigma}(c_i \chi_i) > r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k})$, **then** $\mathbf{k} \leftarrow c_i \chi_i$.
 - 14: **end for** Final for loop.
 - 15: **return** $\mathbf{k} := (k_1, \dots, k_m)$.
 - 16: **end Procedure**
-

Budget Effective Greedy Algorithm (BEG) (cont.)

- Let $\delta(i, b, \mathbf{k})$ be the *per-unit marginal gain* $(r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k} + b\chi_i) - r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k})) / b$ for allocating b more budgets to layer i , which equals to

$$\sum_{u \in \mathcal{V}} \sigma_u \prod_{j \neq i} (1 - P_{j,u}(k_j)) (P_{i,u}(k_i + b) - P_{i,u}(k_i)) / b. \quad (8)$$

³This part is crucial to guarantee the solution quality in case that initial steps yield bad solutions. It also improves the time-consuming partial enumeration procedure of the existing algorithm BEGE [Alon, Gamzu, and Tennenholtz 2012].

Budget Effective Greedy Algorithm (BEG) (cont.)

- Let $\delta(i, b, \mathbf{k})$ be the *per-unit marginal gain* $(r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k} + b\chi_i) - r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k}))/b$ for allocating b more budgets to layer i , which equals to

$$\sum_{u \in \mathcal{V}} \sigma_u \prod_{j \neq i} (1 - P_{j,u}(k_j)) (P_{i,u}(k_i + b) - P_{i,u}(k_i)) / b. \quad (8)$$

- Each iteration greedily selects the (i, b) pair in \mathcal{Q} such that the *per-unit marginal gain* $\delta(i, b, \mathbf{k})$ is maximized.

³This part is crucial to guarantee the solution quality in case that initial steps yield bad solutions. It also improves the time-consuming partial enumeration procedure of the existing algorithm BEGE [Alon, Gamzu, and Tennenholtz 2012].

Budget Effective Greedy Algorithm (BEG) (cont.)

- Let $\delta(i, b, \mathbf{k})$ be the *per-unit marginal gain* $(r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k} + b\chi_i) - r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k}))/b$ for allocating b more budgets to layer i , which equals to

$$\sum_{u \in \mathcal{V}} \sigma_u \prod_{j \neq i} (1 - P_{j,u}(k_j)) (P_{i,u}(k_i + b) - P_{i,u}(k_i)) / b. \quad (8)$$

- Each iteration greedily selects the (i, b) pair in \mathcal{Q} such that the *per-unit marginal gain* $\delta(i, b, \mathbf{k})$ is maximized.
- In the final for loop, we attempt to allocate all budgets to layer i and replace the current best budget allocation if we have a larger reward³.

³This part is crucial to guarantee the solution quality in case that initial steps yield bad solutions. It also improves the time-consuming partial enumeration procedure of the existing algorithm BEGE [Alon, Gamzu, and Tennenholtz 2012].

Theoretical Guarantee for BEG

Theorem 2.

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.

Theoretical Guarantee for BEG

Theorem 2.

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.

- We provide a novel analysis with 0.357-approximation, which improves the existing works [Khuller, Moss, and Naor 1999; Alon, Gamzu, and Tennenholtz 2012] with only $\frac{1}{2}(1 - e^{-1}) \approx 0.316$ approximate solution.

Theoretical Guarantee for BEG

Theorem 2.

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.

- We provide a novel analysis with 0.357-approximation, which improves the existing works [Khuller, Moss, and Naor 1999; Alon, Gamzu, and Tennenholtz 2012] with only $\frac{1}{2}(1 - e^{-1}) \approx 0.316$ approximate solution.
- One can use partial enumeration [Alon, Gamzu, and Tennenholtz 2012] to get a better $(1 - 1/e)$ approximation, but the time complexity is much higher ($B^3 m^3$ times higher) than ours.

Key Lemma for Stationary Distributions

For stationary distributions, we have another lemma.

Lemma 3.

For any network \mathcal{G} , stationary distributions π and node weights σ , function $r_{\mathcal{G}, \pi, \sigma}(\cdot) : \mathbb{Z}_{\geq 0}^m \rightarrow \mathbb{R}$ is monotone and DR-submodular.

- The intuition for DR-submodularity: **the marginal gain is decreasing** when more budgets are allocated to **any layers** $j \in [m]$ (including layer i itself).

Myopic Greedy Algorithm (MG)

Since the reward function is DR-submodular, the BEG procedure **can be replaced by the simple MG procedure** in Algorithm 2 with a better approximation ratio and better time complexity.

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** $\text{MG}((P_{i,u}(b))_{i \in [m], u \in \mathcal{V}, b \in [c_i]}, \sigma, B, c)$

3: Let $\mathbf{k} := (k_1, \dots, k_m) \leftarrow \mathbf{0}$, $K \leftarrow B$.

4: **while** $K > 0$ **do**

5: $i^* \leftarrow \arg \max_{i \in [m], k_i + 1 \leq c_i} \delta(i, 1, \mathbf{k})$. $\triangleright \text{Eq. (11)}$

6: $k_{i^*} \leftarrow k_{i^*} + 1$, $K \leftarrow K - 1$. Greedy with one more budget to layer i .

7: **end while**

8: **return** $\mathbf{k} = (k_1, \dots, k_m)$.

9: **end Procedure**

Theoretical guarantee for MG

Theorem 4.

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

Non-overlapping Multi-layered Networks

For non-overlapping networks, since nodes between different layers do not overlap, we can rewrite the reward function in Eq. (3) by

$$r_{\mathcal{G},\alpha,\sigma}(\mathbf{k}) = \sum_{i \in [m]} \sum_{u \in \mathcal{V}_i} \sigma_u P_{i,u}(k_i). \quad (9)$$

We design a dynamic programming algorithm for arbitrary starting distributions and a greedy algorithm for stationary distributions, each of which gives **optimal** solutions.

Summary for Offline Optimization

The summary for offline models and algorithmic results are presented in the following Table for further reference.

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

Overlapping?	Starting distribution	Algorithm	Aprpx ratio	Time complexity
✓	<i>Arbitrary</i>	Budget Effective Greedy	$(1 - e^{-\eta})^4$	$O(B \ \mathbf{c}\ _{\infty} mn_{max} + \ \mathbf{c}\ _{\infty} mn_{max}^3)$
✓	<i>Stationary</i>	Myopic Greedy	$(1 - 1/e)$	$O(Bmn_{max} + \ \mathbf{c}\ _{\infty} mn_{max}^3)$
×	<i>Arbitrary</i>	Dynamic Programming	1	$O(B \ \mathbf{c}\ _{\infty} m + \ \mathbf{c}\ _{\infty} mn_{max}^3)$
×	<i>Stationary</i>	Myopic Greedy	1	$O(B \log m + \ \mathbf{c}\ _{\infty} mn_{max}^3)$

Table 1: Summary of the offline models and algorithms.

⁴ $1 - e^{-\eta} \approx 0.357$, where η is the solution of $e^{\eta} = 2 - \eta$.

Settings for Online Learning

- We consider T -round explorations.

⁵More 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.

Settings for Online Learning

- 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 .⁵

⁵More 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.

Settings for Online Learning

- 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 .⁵
- We **do not know** about the network structure \mathcal{G} , the starting distributions α or node weights σ .

⁵More 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.

Settings for Online Learning (cont.)

- In round $t \in [T]$, we choose the budget allocation $\mathbf{k}_t := (k_{t,1}, \dots, k_{t,m})$ only based on observations from previous rounds.

⁶We further consider random node weights with unknown mean vector σ in this paper.

Settings for Online Learning (cont.)

- In round $t \in [T]$, we choose the budget allocation $\mathbf{k}_t := (k_{t,1}, \dots, k_{t,m})$ only based on observations from previous rounds.
- After taking the action \mathbf{k}_t , the random explorer W_i , which is part of the environment, would explore $k_{t,i}$ steps and generate the exploration trajectory $\Phi(i, k_{t,i}) = (X_{i,1}, \dots, X_{i,k_{t,i}})$.

⁶We further consider random node weights with unknown mean vector σ in this paper.

Settings for Online Learning (cont.)

- In round $t \in [T]$, we choose the budget allocation $\mathbf{k}_t := (k_{t,1}, \dots, k_{t,m})$ only based on observations from previous rounds.
- After taking the action \mathbf{k}_t , the random explorer W_i , which is part of the environment, would explore $k_{t,i}$ steps and generate the exploration trajectory $\Phi(i, k_{t,i}) = (X_{i,1}, \dots, X_{i,k_{t,i}})$.
- Reward: the total weights of unique nodes visited by all random explorers in round t .

⁶We further consider random node weights with unknown mean vector σ in this paper.

Settings for Online Learning (cont.)

- In round $t \in [T]$, we choose the budget allocation $\mathbf{k}_t := (k_{t,1}, \dots, k_{t,m})$ only based on observations from previous rounds.
- After taking the action \mathbf{k}_t , the random explorer W_i , which is part of the environment, would explore $k_{t,i}$ steps and generate the exploration trajectory $\Phi(i, k_{t,i}) = (X_{i,1}, \dots, X_{i,k_{t,i}})$.
- Reward: the total weights of unique nodes visited by all random explorers in round t .
- Feedbacks:
 - The exploration trajectory $\Phi(i, k_{t,i})$ for each layer L_j .
 - The fixed importance weight σ_u of $u \in \Phi(i, k_{t,i})$ ⁶.

⁶We further consider random node weights with unknown mean vector σ in this paper.

The Goal of Online MuLaNE

- Our goal: design an efficient online learning algorithm A to **gain as much cumulative reward as possible in T rounds**.
- Exploration-exploitation trade-off.
- The **cumulative regret** as our evaluation metric.

Formally, the T -round $((\xi, \beta)$ -approximation) regret of A is:

$$\text{Reg}_{\mathcal{G}, \alpha, \sigma}(T) = \xi \beta T \cdot r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k}^*) - \mathbb{E} \left[\sum_{t=1}^T r_{\mathcal{G}, \alpha, \sigma}(\mathbf{k}_t^A) \right], \quad (11)$$

where \mathbf{k}^* is the optimal budget allocation, \mathbf{k}_t^A is the budget allocation selected by A at t , the expectation is taken over the algorithm and the random trajectories.

Our Idea to Handle the Unknown Networks

- For unknown networks and starting distributions, our idea is to **bypass the transition matrices P_i** and directly estimate the visiting probabilities $P_{i,u}(b) \in [0, 1]$.

Our Idea to Handle the Unknown Networks

- For unknown networks and starting distributions, our idea is to **bypass the transition matrices P_i** and directly estimate the visiting probabilities $P_{i,u}(b) \in [0, 1]$.
 - Avoids the analysis for how the estimated P_i affects the algorithm's performance, which could be unbounded because of the general graph structure and the reward function that is highly non-linear in P_i .

Our Idea to Handle the Unknown Networks

- For unknown networks and starting distributions, our idea is to **bypass the transition matrices \mathbf{P}_i** and directly estimate the visiting probabilities $P_{i,u}(b) \in [0, 1]$.
 - Avoids the analysis for how the estimated \mathbf{P}_i affects the algorithm's performance, which could be unbounded because of the general graph structure and the reward function that is highly non-linear in \mathbf{P}_i .
 - Save the matrix calculation by directly using the estimated $P_{i,u}(b)$.

Our Idea to Handle the Unknown Networks

- For unknown networks and starting distributions, our idea is to **bypass the transition matrices P_i** and directly estimate the visiting probabilities $P_{i,u}(b) \in [0, 1]$.
 - Avoids the analysis for how the estimated P_i affects the algorithm's performance, which could be unbounded because of the general graph structure and the reward function that is highly non-linear in P_i .
 - Save the matrix calculation by directly using the estimated $P_{i,u}(b)$.
- For the unknown weights, we maintain the **optimistic weight** $\bar{\sigma}_u = 1$ if $u \in \mathcal{V}$ has not been visited, and replace $\bar{\sigma}_u$ with the revealed σ_u after its first visit.

CUCB-MAX Algorithm

- An adaptation of the combinatorial upper confidence bound algorithm (CUCB) [Chen et al. 2016] to our setting.
- Base arms $\mathcal{A} = \{(i, u, b) | i \in [m], u \in \mathcal{V}, b \in [c_i]\}$, each of which maintains the visiting probability $\mu_{i,u,b} = P_{i,u}(b)$.
- Offline oracle: BEG algorithm.

CUCB-MAX Algorithm (cont.)

Algorithm 3 CUCB-MAX Algorithm for the MuLaNE

Input: Budget B , number of layers m , number of nodes $|\mathcal{V}|$, constraints \mathbf{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** $t = 1, 2, 3, \dots, T$ **do**
 - 4: **for** $(i, u, b) \in \mathcal{A}$ **do**
 - 5: $\rho_{i,u,b} \leftarrow \sqrt{3 \ln t / (2T_{i,u,b})}$. Calculate UCB.
 - 6: $\tilde{\mu}_{i,u,b} \leftarrow \min\{\hat{\mu}_{i,u,b} + \rho_{i,u,b}, 1\}$.
 - 7: **end for**
 - 8: For $(i, u, b) \in \mathcal{A}$, $\bar{\mu}_{i,u,b} \leftarrow \max_{j \in [b]} \tilde{\mu}_{i,u,j}$. Make sure UCB is monotone.
 - 9: $\mathbf{k} \leftarrow \text{BEG}((\bar{\mu}_{i,u,b})_{(i,u,b) \in \mathcal{A}}, (\bar{\sigma}_v)_{v \in \mathcal{V}}, B, \mathbf{c})$. BEG oracle.
 - 10: Apply budget allocation \mathbf{k} , which gives trajectories $\mathbf{X} := (X_{i,1}, \dots, X_{i,k_i})_{i \in [m]}$ as feedbacks.
 - 11: For any visited node $v \in \bigcup_{i \in [m]} \{X_{i,1}, \dots, X_{i,k_i}\}$, receive its node weight σ_v and set $\bar{\sigma}_v \leftarrow \sigma_v$. Update node weights.
 - 12: For any $(i, u, b) \in \tau := \{(i, u, b) \in \mathcal{A} \mid b \leq k_i\}$, $Y_{i,u,b} \leftarrow 1$ if $u \in \{X_{i,1}, \dots, X_{i,b}\}$ and 0 otherwise.
 - 13: For $(i, u, b) \in \tau$, update $T_{i,u,b}$ and $\hat{\mu}_{i,u,b}$:
 $T_{i,u,b} \leftarrow T_{i,u,b} + 1$, $\hat{\mu}_{i,u,b} \leftarrow \hat{\mu}_{i,u,b} + (Y_{i,u,b} - \hat{\mu}_{i,u,b}) / T_{i,u,b}$. Update base arms.
 - 14: **end for**
-

Challenges for the Proof of Regrets

Two challenges.

- The BEG oracle only works when the inputs are **monotonic increasing** w.r.t b , but random UCB values may not.
- We need to handle the additional regret caused by the optimistic weights $\bar{\sigma}_u$ for **unrevealed node weights**, while CUCB only considers regrets caused by the unknown parameters of base arms.

How to Guarantee the UCB Values are Monotonic Increasing?

- Since BEG can only output $(1 - e^{-\eta}, 1)$ -approximation with monotone inputs, we **take the max** in Line 8 (if UCB $\bar{\mu}_{i,u,b} \leq \bar{\mu}_{i,u,b-1}$, then raise $\bar{\mu}_{i,u,b} = \bar{\mu}_{i,u,b-1}$) to guarantee the UCB value is monotonic increasing w.r.t b .
- Combining the fact that the truth value is indeed increasing, we can bound the regret.
- This technique can apply to other bandit algorithms when UCB values need to be increasing.

How to Handle the Over-estimated Node Weights?

We rely on the following property and a key observation.

Property 1.

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

- We bound $|\sigma_u - \sigma'_u| \mu'_{i,u,b}$ term based on the observation that $\mu'_{i,u,b}$ is **small and decreasing quickly** ($\mu'_{i,u,b} \sim O(\frac{1}{\sqrt{T_{i,u,b}}})$) **before u is first visited**.

Regret Guarantee for CUCB-MAX

Theorem 5.

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

$$\text{Reg}_{\mu, \sigma}(T) \leq \sum_{(i, u, b) \in \mathcal{A}} \frac{108m|\mathcal{V}| \ln T}{\Delta_{\min}^{i, u, b}} + 2|\mathcal{A}| + \frac{\pi^2}{3} |\mathcal{A}| \Delta_{\max}. \quad (12)$$

- Remark 1: $O(\log T)$ regret is **asymptotically tight**. Coefficient $m|\mathcal{V}|$ corresponds to the number of edges in the **(unknown) complete bipartite coverage graph**.
- Remark 2: The $(1 - e^{-\eta}, 1)$ approximate regret is determined by the offline oracle BEG. This regret can be replaced by $(1 - 1/e, 1)$ regret using BEGE or even the exact regret if the oracle can obtain the optimal budget allocation.

Online Learning for Non-overlapping MuLaNE

For the non-overlapping case, we set *layer-wise marginal gains* as our base arms.

- $\mathcal{A} = \{(i, b) | i \in [m], b \in [c_i]\}$.
- $\mu_{i,b} = \sum_{u \in \mathcal{Y}} \sigma_u(P_{i,u}(b) - P_{i,u}(b-1))$ is the marginal gain of assigning budget b in layer i .
- Apply the standard CUCB algorithm.
- Use MG as our oracle.
- We can achieve the exact regret bound $O(\sum_{(i,b) \in \mathcal{A}} 48B \ln T / \Delta_{\min}^{i,b})$.

Dataset and Settings

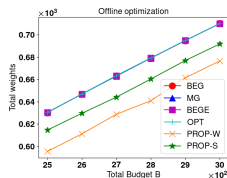
FF-TW-YT (FriendFeed-Twitter-Youtube) dataset.

- $m = 3$ social network layers, 6,407 distinct nodes representing users and 74, 836 directed edges representing connections.
- Edge weight is set to be 1 and node weights are set to be $\sigma_u \in \{0, 0.5, 1\}$ uniformly at random.
- Three random walkers starting from a fixed node in each layer.

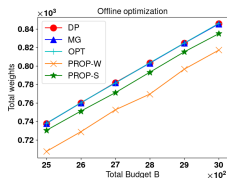
Table 2: Statistics for FF-TW-YT network

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

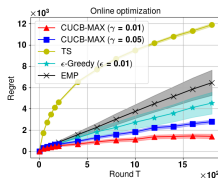
Experimental Results



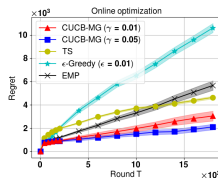
(a) Offline, overlapping.



(b) Offline, non-overlapping.



(c) Online, Handle overlapping.



(d) Online, non-overlapping.

Figure 3: Left: total weights of unique nodes visited for offline algorithms. Right: regret for online algorithms when $B = 3000$.

For offline settings, BEG is empirically close to the optimal solution.
For online settings, CUCB-MAX achieves the smallest regret.

Experimental Results (Cont.)

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

	B=2.6k	B=2.8k	B=3.0k		B=2.6k	B=2.8k	B=3.0k	Overlapping?	✓	×
BEG	0.274	0.316	0.363	DP	0.038	0.044	0.050	BEG	1.22	NA
BEGE	34.37	45.51	59.20	MG	0.008	0.009	0.010	BEGE	60.03	NA
OPT	91.16	98.42	105.63	OPT	70.10	75.78	81.11	DP	NA	0.86
								OPT	107.33	82.01

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

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

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

Figure 4: Computational Efficiency

BEG is computational efficient and is at least two orders magnitude faster than BEGE/OPT.

Conclusions and Future Work

- Formulates the MuLaNE as a **budget allocation** problem, requiring that the total weights of distinct nodes visited are maximized.

Conclusions and Future Work

- Formulates the MuLaNE as a **budget allocation** problem, requiring that the total weights of distinct nodes visited are maximized.
- Propose algorithms for MuLaNE (**offline or online**) according to the multi-layered network structure (**overlapping or non-overlapping**) and starting distributions (**arbitrary or stationary**), each of which has a **provable** approximation factors, running time or regret guarantee and are validated by experiments.

Conclusions and Future Work

- Formulates the MuLaNE as a **budget allocation** problem, requiring that the total weights of distinct nodes visited are maximized.
- Propose algorithms for MuLaNE (**offline or online**) according to the multi-layered network structure (**overlapping or non-overlapping**) and starting distributions (**arbitrary or stationary**), each of which has a **provable** approximation factors, running time or regret guarantee and are validated by experiments.
- Future directions:
 - Jointly optimize starting distributions and budget allocation.
 - Adaptive MuLaNE.

Q&A

Thanks for listening!

References I

-  Lv, Qin et al. (2002). “Search and replication in unstructured peer-to-peer networks”. In: *Proceedings of the 16th international conference on Supercomputing*, pp. 84–95.
-  Gleich, David F (2015). “PageRank beyond the Web”. In: *SIAM Review* 57.3, pp. 321–363.
-  Wilder, Bryan et al. (2018). “Maximizing influence in an unknown social network”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
-  Alon, Noga, Iftah Gamzu, and Moshe Tennenholtz (2012). “Optimizing budget allocation among channels and influencers”. In: *Proceedings of the 21st international conference on World Wide Web*. ACM, pp. 381–388.
-  Khuller, Samir, Anna Moss, and Joseph Seffi Naor (1999). “The budgeted maximum coverage problem”. In: *Information processing letters* 70.1, pp. 39–45.
-  Chen, Wei et al. (2016). “Combinatorial multi-armed bandit and its extension to probabilistically triggered arms”. In: *The Journal of Machine Learning Research* 17.1, pp. 1746–1778.