Gaussian Process Bandits for Online Influence Maximization*

1st Anonymous
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—We study the online influence maximization problem, which aims to select the seed set with constant cardinality that influences the most participants on a social network while learning the unknown diffusion parameters simultaneously. To optimize the selection strategy, a learner will interact with the network and optimize its strategy according to the influence feedback. As the learner has no information about the underlying diffusion probabilities between nodes on the graph, most previous works model the influence probability based on a specific linear function and minimize the estimation error between the estimations and the ground-truth. In this paper, we propose to utilize the Gaussian Process to estimate the underlying diffusion parameters based on the node's feature vectors and optimize the model using historical statistics. This novel online influence maximization method circumvents the artificial linear influence probability assumption and is applicable for more general influence probability patterns. The proposed method estimates influence probabilities based on the correlation between nodes. We adopt the bandit framework to devise an online influence maximization algorithm, which is a natural combination of Gaussian Process estimation and upper confidence bound algorithm. Experiments conducted on public datasets reveal that the proposed method is superior to State-of-the-art baselines and is robust w.r.t its hyperparameters.**

***Index Terms*—Online influence maximization, Gaussian process, Bandit algorithm, Reinforcement learning,**

I. Introduction

The growth of social networks greatly facilitates information propagation and brings abundant applications, e.g., social network advertising [1] and viral marketing [2]. These applications aim to enlarge the spread of information or ideas on social networks, through the *world-of-mouth* effect. Generally, the marketers or companies select a small number of users to produce free products or give a discount, expecting their product's information can spread from these initial users to influence the maximum potential customers on a social network. However, most of the time, marketers are limited to budgets. Hence, finding the most influential users on a social network to maximize the influence spread under limited cost becomes a critical problem. Formally, we aim to find a set of nodes with a constant cardinality, from which influence spreads to the maximum number of nodes on a specific network. This is referred to as the influence maximization problem [3].

For the general influence maximization formulation, information diffuses on a directed graph, on which each edge's weight represents the strength with which one node can influence the other. The diffusion follows a predefined diffusion model. The elegant Independent Cascade (IC) [3] model is the most common one, in which the influence spreads from the initially selected nodes and then propagates in discrete time-steps according to the graph topology and edge activation probabilities. Kempe et al. [3] proved that the influence maximization problem is NP-hard. However, they utilized the submodularity of spread function and proposed a greedy algorithm achieving the $1 - 1/e - \epsilon$ fraction of the theoretical optimal influence spread.

In some scenarios, we have no information about the influence parameters in diffusion models and have to inference these parameters from interactions with the network [4], [5], which is referred to as Online Influence Maximization (OIM) [6]. A line of work for OIM assumes the diffusion probability is a function of a global hyperparameter and features of nodes or edges of the network [7]. The probability of node u influences node v is not arbitrary but a function that maps node's features to $[0, 1]$, i.e., $p_{u,v} = f(\theta, x_{u,v})$, where $x_{u,v}$ is the features of nodes and θ is the hyperparameter. Generally, the function f is determined by hand, and the hyperparameters are inferred from interactions. This paradigm mainly pertains to the solution of an optimization problem. Most works for online influence maximization adopt bandit models [8]–[11] and assume the mapping f is linear, i.e., $f(\theta, x_{u,v}) = \theta^T x_{u,v}$, or generalized linear [12]. However, the pre-defined linear model leads to restrictions. Firstly, the information of probability distribution and mapping pattern in large-scale networks are generally unknown [12] and the manually-defined linear or generalized linear function is adopted for the reason of tractability [12]. Nevertheless, the hyperparametric models mentioned above restrict the model into the specific function space. e.g., linear function space. Secondly, the hyperparametric assumption assumes the influence probability matrix is low-rank, and the rank is smaller than the dimensionality of the hyperparameter. This assumption ignores the correlations between nodes' features. The correlation information of nodes assists the influence maximization and parameter estimation. For example, the nodes in social networks are commonly correlated and form communities [13], [14], and these structures assist the influence maximization [15], [16].

Motivated by previous linear influence probability restric-

tion and the correlation between node features, we propose to utilize the Gaussian Process ($\mathcal{GP}$) to estimate the influence probabilities based on the node features' correlations. The proposed model has two advantages. Firstly, the Gaussian Process-based estimation is non-parametric and circumvents the prevalent pre-defined linear function assumption. Hence the proposed method does not require the influence probability matrix to be low-rank. Secondly, the Gaussian Process model adopts a kernel function to measure the correlation between node features and estimates influence probabilities based on the covariance computed by the kernel function. The correlation between nodes is the foundation for parameter inference. The proposed method extends the online IM model's utility for more general influence probability patterns.

More precisely, We use the online historical interaction statistics to optimize the $\mathcal{GP}$ model, based on the features of the network entities. Note that even though we state the Gaussian Process-based model is non-parametric, there is indeed one parameter in the adopted radial basis function (RBF) kernel. Nevertheless, this parameter can be learned using the L-BFGS algorithm automatically [17]. On the other side, we absorb the idea in [11] to factorize the feature vector of each node into the influence vector α and the susceptibility vector β, corresponding to the influence ability and the susceptibility. This idea takes the network assortativity [18] into consideration. The $\mathcal{GP}$ model inferences influence probabilities based on these vectors and imposes no assumption about the relation between probabilities and feature vectors. Hence the method can be regarded as a generalized method for different probability patterns. Furthermore, the estimation uncertainty is utilized by the **bandit framework**, which is commonly used in online learning algorithms. We propose the IMGUCB algorithm based on the $\mathcal{GP}$ model and upper confidence bound (UCB) bandit framework [4], [19]. To verify the algorithm's utility for different networks and different probability patterns, We conduct experiments based on two public datasets with two influence probability patterns. Experimental results reveal that the proposed algorithm achieves superior online influence spread and smaller estimation errors on both datasets compared with state-of-the-art baselines. Furthermore, the hyperparameter study reveals that the proposed algorithm is robust for a relatively wide range of hyperparameters. The implementation of the proposed algorithm is publicly available[1].

II. Related works

Kempe et al. [3] firstly formulated the offline influence maximization as a discrete combinatorial optimization problem, and then the IM problem was extensively studied [20]–[24]. These works consider a directed graph representing nodes' connections and assume the awareness of diffusion probabilities of the graph. [20] leverages the submodularity of propagation function to dramatically reduce the Monte-Carlo simulations that estimate the influence spread of a seed set and achieves an impressive performance improvement. Some work proposes to learn the influence probabilities from historical action logs if the influence probabilities are unknown [25]–[27]. However, the historical logs or diffusion cascades are itself inaccessible in some scenarios, e.g., considering that a marketer aims to propagate a specific product in a new market. For this reason, a more natural way is to learn these parameters during interactions, i.e., online influence maximization [4], [6], [28]. Chen et al. [4] firstly proposed to solve the online influence maximization based on the combinatorial multi-armed bandit framework. However, they just revealed the feasibility of their algorithmic framework for this interesting problem, without devising a precise algorithm and conducting any experiments. [6] and [29] studies a related but different problem. Their objective is to maximize the number of uniquely activated nodes on a network under constant influence campaigns and seed set cardinality. In [8], a general combinatorial multi-armed bandit (CMAB) framework for OIM was proposed, and the authors utilized both node-level and edge-level semi-bandit feedback to learn the value of each arm. Nevertheless, in their framework, each edge is regarded as an arm and learned independently, which is infeasible and computationally expensive in large networks. [5] proposed to optimize the regret bound of CMAB with probabilistically triggered arms and semi-bandit feedback and considered the OIM problem under their framework, without any practical experiments as well. [10] firstly proposed to construct the latent representation of network edges, and then modeled the edge probability as a linear function of edge representations. The parameters of this linear function are learned under the UCB framework. [9] modeled the edge probabilities as linear functions on this edge's end node vector instead of the edge vector, and devised a similar but *diffusion-model-independent* algorithm to learn parameters of each node. [11] endowed each node on the network with two latent vector representations, i.e., the influence vector and the susceptibility vector. The influence strength is represented as the inner product between giving node's influence and receiving node's susceptibility. [21] improves the algorithmic performance of [20]by reducing Monte-Carlo simulations further. We assume that the probability is related to some intrinsic characteristics of the edge, but the mapping from the characteristics to probability is arbitrary, as stated in section I. Under this assumption, the prior similarity between nodes/edges can be absorbed to boost the overall performance.

III. Method

In this section, we discuss the proposed $\mathcal{GP}$ and UCB based bandit algorithm. We first give the problem formulation and then discuss the $\mathcal{GP}$ model for diffusion probabilities. Finally, we give the algorithmic pseudocodes and analyze the algorithm.

A. Problem Formulation

a) Influence maximization: Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = \{v_1, v_2, v_3, \cdots, v_N\}$ the node set, $\mathcal{E} = \{e_1, e_2, e_3, \cdots, e_L\}$ the edge set, and $P =$

[1]https://github.com/Anonymous-Projects-Maker/IMGUCB

$\{p_1, p_2, p_3, \cdots, p_L\}$ is the edge activation probability vector. $p_i \in [0, 1]$ represents the activation probability of edge e_i, i.e. the probability that receiving node of edge e_i will be activated by e_i's giving node, if the giving node is activated. Another important component is the diffusion model D, which formulates how influence spreads on this network. We adopt the elegant Independent Cascade (IC) model [3]. In each diffusion process, nodes in a subset of V are initially activated, i.e., seeds. At each propagation step, a node activated in the last step has one chance to activate its inactive neighbors following an influence probability. The activated nodes in this step act in the same manner in subsequent steps. The diffusion process terminates until no nodes can be activated. The goal of influence maximization is to select a seed set with a fixed cardinality K to maximize the expectation of finally influenced nodes (or influence spread), i.e., $\max_{S,|S|=K} f_D(S, P)$,where $f_D(S, P)$ represents the expected number of activated nodes choosing S as the seed set, under diffusion model D and influence probability matrix P. In the following, we omit the subscript D since we focus on the IC model.

b) Online influence maximization with semi-bandit feedback: For online influence maximization, the influence between nodes is unknown and the algorithm (or agent) learns these probabilities using feedback in online interactions with the network. For each interaction, the algorithm chooses a node set S with cardinality K to start a propagation process (exploitation), as well as updates its estimation of unknown parameters (exploration). The balance of exploitation and exploration is a fundamental problem. Generally, the multi-armed bandit (MAB) algorithm framework is applied for online influence maximization, especially the combinatorial multi-armed bandit (CMAB) framework. In the CMAB framework, the algorithm invokes an ORACLE subroutine. The ORACLE subroutine is a traditional offline IM algorithm which takes the currently estimated influence probabilities as input and outputs a seed set solution, i.e., $S = \text{ORACLE}(\mathcal{G}, K, \hat{P})$, where $\hat{P}$ denotes the agent's probability estimation. For example, algorithms proposed in [3], [22], [24] can be regarded as such an ORACLE. The influence spreads from S following the underlying diffusion model D. Finally, the agent receives the reward $f(S, P)$, and the observed edges (semi-bandit feedback), where P denotes the ground-truth. Note that the agent will only observe an edge $e_i = (u, v) \in \mathcal{E}$ if node u is activated. The semi-bandit feedback will be used to update the agent's estimation. The objective is to maximize the cumulative reward (influence spread) over n interactions.

B. Gaussian Process based Bandit Solution

In this section, we describe the influence probability estimation based on the Gaussian Process regression model to circumvent the restrictions of previous methods stated in section I. Further, we compute the upper confidence bound of estimations. Note that the Gaussian process provides a posterior Gaussian distribution for each estimation, hence we can construct the estimation upper bound conveniently. The upper confidence bound (UCB) is generally utilized as a substitution of the direct estimation, especially in the UCB bandit algorithm, which we will discuss later. The inference is based on the edge's feature representations, and we We need to learn a general function $g(\cdot)$, i.e.,

$$p_e = g(x_e) \tag{1}$$

where p_e represents the activation probability of edge e. Note that function $g(\cdot)$ takes edge's feature vector x_e as input. We construct the x_e as in [11], i.e., each node's vector is factorized with an influence vector α and a susceptibility vector β. For this setting, the edge vector x_e can be represented as

$$x_e^T = [\alpha_{g_e}^T, \beta_{r_e}^T] \tag{2}$$

where g_e denotes the giving node of edge e and r_e the receiving node.

The $\mathcal{GP}$ model requires a training set to construct its kernel matrix, and the training set is constructed using feature matrix and historical statistics. Recall the notations above, x_e represents the vector of edge e. Let X denotes the matrix of edge vectors with l rows (l edges), i.e.,

$$X = [x_{e_1}, x_{e_2}, x_{e_3}, \cdots, x_{e_l}]^T \tag{3}$$

And let $\hat{Y}$ denotes the historical probabilities of these l edges:

$$\hat{Y} = [\hat{y}_{e_1}, \hat{y}_{e_2}, \hat{y}_{e_3}, \cdots, \hat{y}_{e_l}]^T \tag{4}$$

The covariance function $k(\cdot, \cdot)$ (also called kernel function) of $\mathcal{GP}$ defines how the covariance of any two samples x and x' should be computed, and is referred to as prior information. We adopt the commonly used *radial basis kernel* (RBF):

$$k_{\text{se}}(x, x') = \exp(-\frac{1}{2\tau^2})||x - x'||^2 \tag{5}$$

And $K(X, X)$ denotes the kernel matrix, with its i-row, j-column element $K(X, X)[i, j] = k(x_{e_i}, x_{e_j})$.

The matrix X and column vector $\hat{Y}$ are the components necessary for inference using the $\mathcal{GP}$ model. Specifically, the probability estimation $g(x_{e_i})$ for edge e_i is a random variable following a posterior Gaussian distribution, i.e.,

$$g(x_{e_i})|x_{e_i}, X, \hat{Y} \sim \mathcal{N}(m(x_{e_i}), var(x_{e_i})) \tag{6}$$

The mean $m(x_{e_i})$ and variance $var(x_{e_i})$ can be computed as follows:

$$\begin{aligned} \mathbf{A} &= K(X, X) + \sigma^2 \mathbf{I} \\ m(x_{e_i}) &= K(x_{e_i}, X)\mathbf{A}^{-1}\hat{Y} \\ var(x_{e_i}) &= K(x_{e_i}, x_{e_i}) + \sigma^2 \mathbf{I} - K(x_{e_i}, X)\mathbf{A}^{-1}K(X, x_{e_i}) \end{aligned} \tag{7}$$

where σ and τ are hyperparameters. $K(x_{e_i}, X)$ is a row vector representing the kernel value of x_{e_i} and each row of X, and $K(X, x_{e_i})$ is a column vector representing the kernel value of each row of X and x_{e_i}, i.e.,

$$\begin{aligned} K(x_{e_i}, X) &= [k(x_{e_i}, X_1), k(x_{e_i}, X_2), \cdots, k(x_{e_i}, X_l)] \\ K(X, x_{e_i}) &= [k(X_1, x_{e_i}), k(X_2, x_{e_i}), \cdots, k(X_l, x_{e_i})]^T \end{aligned} \tag{8}$$

Next, we describe the construction of estimation upper bound and the UCB bandit framework. Following Eq.7, an

upper bound of $g(x_{e_i})$ can be easily constructed as follows. Let Z be a random variable following the standard normal distribution: $Z \sim \mathcal{N}(0,1)$. We denote the cumulative distribution function (CDF) of the standard Gaussian distribution as Φ and let $z_\alpha = \Phi^{-1}(1-\alpha)$, i.e., $P(Z < z_\alpha) = 1-\alpha$. The $1-\alpha$ confidence upper bound of $g(x_{e_i})$ is $U(x_{e_i}) = m(x_{e_i}) + z_\alpha \sqrt{var(x_{e_i})}$, i.e.,

$$P\left(g(x_{e_i}) \leq m(x_{e_i}) + z_\alpha \sqrt{var(x_{e_i})}\right) = 1-\alpha \quad (9)$$

For example, usually we can let $z_\alpha = 2$ to get an approximate 95% upper confidence bound: $m(x_{e_i}) + 2\sqrt{var(x_{e_i})}$, and in practice the z_α can be optimized to adjust the optimism.

The easily constructed upper confidence bound makes the upper confidence bound based bandit framework feasible. Leveraging the estimation upper confidence bound is a proper way to balance the exploration and exploitation problem in online bandit algorithms. Take the elegant combinatorial UCB bandit framework as an example [4], the ORACLE utilizes the estimation upper bound $U(x_{e_i})$, instead of $m(x_{e_i})$, as the influence probability to compute a seed set. An edge (or arm) with large upper bound is either with large estimation uncertainty or with large influence probability. In both scenarios, we tend to choose that arm. This is how the UCB bandit framework works. The proposed algorithm is a combination of the Gaussian Process and UCB bandit framework.

C. IMGUCB Algorithm

We propose the **IMGUCB** algorithm, which is an acronym for **I**nfluence **M**aximization using **G**aussian Process and **UCB** framework. We describe the overall algorithmic pipeline as follows. Firstly, the algorithm maintains the historical mean and observed number of observed edges. The features and historical means of observed edges are utilized to construct the kernel matrix $K(X,X)$ and $\hat{Y}$. The algorithm computes UCBs of all edges (both observed edges and unobserved edges) following Eq. 7 and Eq. 9. Secondly, the adopted ORACLE subroutine (an offline IM algorithm) computes a seed set based on the UCBs. Finally, the seed set initiates an influence spread campaign, and the algorithm receives the feedback and updates the historical statistics of observed edges.

However, the above scheme results in expensive computational cost, since the kernel matrix scales exponentially w.r.t the number of edges l. Computing the whole kernel matrix and its inverse is impractical. Hence for practical efficiency, we only sample a subset of $\mathcal{E}$ with size C to construct the kernel matrix, where C is a hyperparameter. We devise the following **sampling strategy**: 1. We split $\mathcal{E}$ into 10 sets, accoding to the statistic $\hat{p}_e$, i.e., $\tilde{\mathcal{E}}_i = \{e_j | \hat{p}_{e_j} \in [0.1i, 0.1i+0.1)\}, i = 0, 1, \cdots, 9$. 2. For each group $\tilde{\mathcal{E}}_i$, we sort edges according to their observed number and select the top $\lceil \frac{|\tilde{\mathcal{E}}_i|}{|\mathcal{E}|} C \rceil$ edges. The total C (at most $C+10$) edges are selected to constrct the matrix X, $\hat{Y}$, and $K(X,X)$. The intuition of the sampling strategy is based on how $\mathcal{GP}$ works. Since the estimations lie in $[0,1]$ and we expect to estimate accurately in the whole range of $[0,1]$. However, the influence probabilities are unbalanced and concentrate on small values (generally [0, 0.2]) in practical datasets [10], [11]. Therefore we select edges whose historical statistics cover the whole target range, to predict accurately for most edges. Furthermore, we choose edges with more observed times in each $\tilde{\mathcal{E}}_i$, since these edges produce smaller uncertainty, compared with ones with less observed times. The sampling strategy is illustrated in Algorithm 2.

Algorithm 1 IMGUCB: Influence Maximization Gaussian Process UCB algorithm

1: **Input:** Graph $\mathcal{G}$, seed set cardinality K, ORACLE, node features $\{x_{e_i}\}_{i=1,2,\cdots,L}$, and algorithm parameter σ, C, z_α, $period$.
2: **Initialization:** Choose C edge vectors to construct the matrix X_0, Y_0, and compute $\mathbf{A}_0^{-1}$. Each edge's historical mean $\hat{P} = \{\hat{p}_i \leftarrow 0\}_{i=1,2,\cdots,L}$, and observed numbers $O = \{o_{e_i} \leftarrow 0\}_{i=1,2,\cdots,L}$.
3: **for** $t = 1, 2, \cdots, n$ **do**
4: Compute the mean estimate, i.e. $m_t(e_i) \leftarrow K(x_{e_i}, X_{t-1})\mathbf{A}_{t-1}^{-1} Y_{t-1}$, for each edge $e_i \in \mathcal{E}$, and the UCBs:
$U_t(e_i) \leftarrow Proj_{[0,1]}\{m_t(e_i) + z_\alpha \sqrt{K(x_{e_i}, x_{e_i}) + \sigma^2 \mathbf{I} - K(x_{e_i}, X_{t-1})\mathbf{A}_{t-1}^{-1} K(X_{t-1}, x_{e_i})}\}$
5: Choose $\mathcal{S} = \text{ORACLE}(\mathcal{G}, K, U_t)$, run one influence spread campaign and observe the edge-level semi-bandit feedback.
6: *Update statistics:*
7: For all observed edges e_i, increase its observed count o_{e_i} by 1, and update its historical mean $\hat{p}_{e_i}$.
8: **if** t mod $period$ = 0 **then**
9: Sample C edges according to Algorithm 2, compute $K(X_t, X_t)$ and Y_t; update $\mathbf{A}_t = K(X_t, X_t) + \sigma^2 \mathbf{I}$, and compute $\mathbf{A}_t^{-1}$.
10: **else**
11: $\mathbf{A}_t = A_{t-1}$ and $\mathbf{A}_t^{-1} = A_{t-1}^{-1}$; $Y_t = Y_{t-1}$
12: **end if**
13: **end for**

Since one trail increases the observed number of edges by one and has minor effects on the historical statistics, we can update the kernel matrix and estimation upper bound periodically, as indicated in Algorithm 1, line 8. The edge sampling strategy and periodical update greatly boost the algorithm's efficiency without sacrificing too much performance in experiments. The proposed online IMGUCB algorithm invokes an ORACLE subroutine. The ORACLE subroutine computes a node set according to the graph and influence UCBs. In this paper, we adopt the DegreeDiscountIC algorithm proposed in [22] as such an ORACLE. DegreeDiscountIC is a heuristic algorithm that selects seeds iteratively. At each iteration, it selects the node with the largest degree and decreases the degree of seed node's neighbors according to a heuristic rule. The pseudocode is described in Algorithm 1.

Algorithm 2 Sampling strategy

1: **Input:** Edges $\mathcal{E}$, selection size C, each edge's historical mean $\hat{P} = \{\hat{p}_i\}_{i=1,2,\cdots,L}$, and observed numbers $O = \{o_{e_i}\}_{i=1,2,\cdots,L}$.
2: **Output:** Selected edges $\bar{\mathcal{E}}$
3: Split $\mathcal{E}$ into 10 sets $\tilde{\mathcal{E}}_0, \tilde{\mathcal{E}}_2, \cdots, \tilde{\mathcal{E}}_9$, $\tilde{\mathcal{E}}_i = \{e_j | \hat{p}_{e_j} \in [0.1i, 0.1i + 0.1)\}, i = 0, 1, \cdots, 9$.
4: For each $\tilde{\mathcal{E}}_i$, sort edges in decending order w.r.t o_e.
5: For each $\tilde{\mathcal{E}}_i$, compute $c_i = \lceil \frac{|\tilde{\mathcal{E}}_i|}{|\mathcal{E}|} C \rceil$, and select top c_i edges $\bar{\mathcal{E}}_i = \{e_1, e_2, \cdots, e_{c_i} | e \in \tilde{\mathcal{E}}_i\}$.
6: **return** $\bar{\mathcal{E}} = \bar{\mathcal{E}}_0 \cup \bar{\mathcal{E}}_1 \cdots \cup \bar{\mathcal{E}}_9$

D. Regret Analysis

In this subsection, we provide the upper (α,γ)-approximation regret bound in IMGUCB. We denote P^* as the ground-truth diffusion probability of graph $\mathcal{G}$, $S^* = ORACLE(\mathcal{G}, P^*)$, and S^{opt} the optimal but unknown solution. For any P, if an ORACLE has the properity: $f(S^*, \mathcal{G}, P) \geq \gamma f(S^{opt}, \mathcal{G}, P)$, with probability α, $\alpha, \gamma \in [0, 1]$, the ORACLE is called an (α,γ)-ORACLE. Note that we ommit the $\mathcal{G}$ in f for conciseness. Since the IM problem is NP-hard [3], and lots of offine IM algorithms act as (α,γ)-ORACLEs [24], [30]. A proper way to define the performance matric is the **(α,γ)-approximation regret** [4], i.e. $R^{\alpha\gamma}(T) = \sum_{t=0}^{T} \mathbb{E}[R_t^{\alpha\gamma}]$, where $R_t^{\alpha\gamma} = f(S^{opt}, P^*) - \frac{1}{\alpha\gamma} f(S_t, P^*)$. Based on the above definition, we provide the (α,γ)-approximation bound with the theorem below.

Theorem 1. *If the ORACLE satisfies the (α,γ)-ORACLE condition, then we provide the upper (α, γ)-approximation regret bound in IMGUCB,*

$$R^{\alpha\gamma}(T) \leq O(\frac{T}{\alpha\gamma}(B|\mathcal{V}|^2) \tag{10}$$

in which B is the bounded smoothness constant [5], and $|\mathcal{V}|$ denotes the number of nodes of graph $\mathcal{G}$.

Proof. We clarify two important mild conditions about the influence spread function $f(S, P)$, as the proof sketch in [5], [11]: 1. Monotonicity [3], $f(S, P^{'}) \geq f(S, P)$, if $p_e^{'} \geq p_e$ for all $e \in \mathcal{E}$; 2. 1-Norm bound smoothness [5], if there exists a bounded constant $B \in R^+$, such that for any two diffusion probability vectors $P^{'}$ and P, $|f(S, P^{'}) - f(S, P)| \leq B \sum_{e \in \tilde{\mathcal{E}}} |p_e^{'} - p_e|$.

According to the definition of $R^{\alpha\gamma}(T)$, we have $R^{\alpha\gamma}(T) = \sum_{t=1}^{T} \mathbb{E}[R_t^{\alpha\gamma}]$, and

$$\begin{aligned}
\mathbb{E}[R_t^{\alpha\gamma}] &= f(S^{opt}, P^*) - \mathbb{E}[\frac{1}{\alpha\gamma} f(S_t, P^*)] \\
&\leq \frac{1}{\alpha\gamma} f(S^*, P^*) - \mathbb{E}[\frac{1}{\alpha\gamma} f(S_t, P^*)] \\
&= \frac{1}{\alpha\gamma} \mathbb{E}[f(S^*, P^*) - f(S_t, P^*)]
\end{aligned} \tag{11}$$

The first inequality of Eq.11 is based on the definition of (α,γ)-ORACLE: $f(S^{opt}, P^*) \leq \frac{1}{\alpha\gamma} f(S^*, P^*)$. If we define ζ_{t-1} as the event $\zeta_{t-1} = \{p_e^* \leq \bar{p}_{e,t}, \forall e \in \mathcal{E}\}$, where $\bar{p}_{e,t}$ is the estimated upper bound for edge e at time t, and $\bar{\zeta}_{t-1}$ is the counterpart event of ζ_{t-1}. We can decompose the $R_t^{\alpha\gamma}$ as follows:

$$\begin{aligned}
\mathbb{E}[R_t^{\alpha\gamma}] &\leq \frac{1}{\alpha\gamma} \mathbb{E}[f(S^*, P^*) - f(S_t, P^*)] \\
&\leq \frac{1}{\alpha\gamma} (\mathbb{P}(\zeta_{t-1}) \mathbb{E}[f(S^*, P^*) - f(S_t, P^*) | \zeta_{t-1}] + |\mathcal{V}|) \\
&\leq \frac{1}{\alpha\gamma} (\mathbb{P}(\zeta_{t-1}) \mathbb{E}[f(S^*, \bar{P}_t) - f(S_t, P^*) | \zeta_{t-1}] + |\mathcal{V}|) \\
&\leq \frac{1}{\alpha\gamma} (\mathbb{P}(\zeta_{t-1}) \mathbb{E}[f(S_t, \bar{P}_t) - f(S_t, P^*) | \zeta_{t-1}] + |\mathcal{V}|) \\
&\leq \frac{1}{\alpha\gamma} (B \sum_{e \in \tilde{\mathcal{E}}_t} |p_e^* - \bar{p}_{e,t}| + |\mathcal{V}|)
\end{aligned} \tag{12}$$

The first inequality follows from Eq.11, and the second inequality follows from the value range of f. The third inequality follows from the monotonicity of f, and the fourth inequality is based on the ORACLE solution at time t. The last inequality follows from the 1-Norm smoothness as stated in Condition 2. $\tilde{\mathcal{E}}_t$ is the observed edges at time t. Since we have $|\tilde{\mathcal{E}}_t| \leq |\mathcal{E}|$, we obtain an upper bound of $\mathbb{E}[R_t^{\alpha\gamma}]$ as $O(\frac{1}{\alpha\gamma}(B|\mathcal{V}|^2 + |\mathcal{V}|))$. Finally, we have $R^{\alpha\gamma}(T) = \sum_{t=1}^{T} \mathbb{E}[R_t^{\alpha\gamma}] \leq O(\frac{T}{\alpha\gamma}(B|\mathcal{V}|^2))$. □

IV. Experiments

A. Datasets

We adopte two commomly used network datasets to evaluate our algorithm: Flickr[2] (105938 nodes and 2316948 edges) [31] and NetHEPT[3] (27770 nodes and 352807 edges) [32], [33]. Note that We sample $1/3$ nodes for the Flickr dataset for experiment efficiency.

Since the ground-truth diffusion probabilities are unavailable, previous works generate these parameters artifically [10], [11]. We adopt the setting in [11]. Firstly, we sample the node's influence vector α and susceptibility vector β under a uniform distribution $U[0, 1]$, then normalize them using L2 normalization. To make the *soft degree* (the summation of out edge's activation probability of a node) more balanced, as stated in [11], we scale the node's influence vector according to its hard degree, i.e., we scale the influence vector values inversely proportional to its out-degree. The original edge probabilities are generated using inner product, i.e. $p_{e_i} = \alpha_{e_i}^T \beta_{e_i}$, referred to as *linear probabilitiies*. To compare our algorithm with other competitors under the nonlinear edge probability patterns, we devise a simple nonlinear transformations to map the original *linear probabilities* to *distorted probabilities* (or *nonlinear probabilities*). The distortion function is a simple sin function, i.e. Eq.13, and is adopted on two datasets to generate the distorted versions of two datasets respectively. The information of two datasets is listed in Table I.

$$y = sin(p * \pi) \tag{13}$$

[2]https://snap.stanford.edu/data/web-flickr.html
[3]https://snap.stanford.edu/data/cit-HepTh.html

TABLE I: The statistics of both datasets. Note that we leverage the largest connected components for both graphs.

Dataset	Nodes	Edges	Avg. p	Avg. p(distorted)
Flickr	32851	263484	0.08	0.13
NetHEPT	27770	352807	0.04	0.11

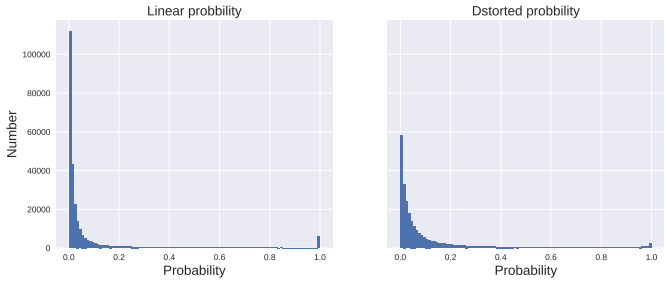


Fig. 1: The diffusion probability histograms of Flickr dataset. The simple artifical nonlinear mapping (Eq.13) distorts the graph's influence probabilities, and the histogram, introducing a *nonlinear pattern*.

We illustrate the probability histograms of the Flickr dataset in Fig. 1, to show the distortion effect intuitively. The distortion mapping acts as a simple proxy for general nonlinear relations between graph features and influence strengths.

B. Baseline Algorithms

We compare our algorithm with the following baselines.

- **Random** It randomly selects K nodes for every interaction, acting as the simplest baseline.
- **ϵ-greedy** It maintains a probability estimation of each edge. For each edge, it returns an random number in $U[0,1]$ with probability ϵ and its current estimation for this edge with probability $1-\epsilon$. The returned value is used by the oracle to compute the seed set. ϵ-greedy explores each edge uniformly without any direction.
- **CUCB** [4] It also maintains a probability estimation for each edge. However, it differs from ϵ-greedy that it computes the upper confidence bound for each edge as the probability estimation. The upper confidence bound is utilized by the oracle to compute the seed set.
- **IMLinUCB** [10] It learns a global hyperparameter of a linear function. Note that the hyperparameter is shared for all edges. We adopt the setting in [11] to construct edge's feature vector, i.e., we use the outer product of each edge's giving node's influence and receiving node's susceptibility, i.e., $\alpha\beta^T$ and flatten it as the edge feature. Since the influence vectors and susceptibility vectors are the ground-truth, the IMLinUCB algorithm should have its potential to perform the best.
- **IMFB** [11] The IMFB algorithm considers the assortativity of networks and factorizes each node's feature vector into influence α and susceptibility β, and represent the diffusion probability by inner product, i.e., Eq2. The algorithm learns each node's influence vector and susceptibility vector alternately using edge-level feedback.

C. Hyperparameter Setting

For the ϵ-greedy algorithm, ϵ is set to 0.1, which is the most common setting in other works [11]. The CUCB algorithm has no hyperparameters. For the IMLinUCB algorithm, α_1 is set to 0.1, and λ is set to 0.4, as indicated in [10]. The edge feature dimension of IMFB and IMGUCB is set to 8, and 16 for the IMLinUCB baseline [11]. Note that the ORACLEs for all algorithms are identical, i.e., we set ORACLE as DegreeDiscountIC for all algorithms. For the IMGUCB, C is commonly set in $[1000, 4000]$, and z_α in $[1, 3]$. We will study the hyperparameter's effect in subsection IV-F. In most experiments, we set C to 2000 and z_α to 2.

D. Online Influence Spread and Running Time

In this subsection, we report the online influence spread achieved for different algorithms and their average running time. For each experiment, we run 300 iterations and repeat 10 times to alleviate randomness and compute the variance. For IMGUCB, we set C to 2000 and z_α to 2 as stated in subsection IV-C.

a) Online Influence Spread: We conduct experiments on Flickr and NetHEPT datasets with both the distorted probability pattern and the linear probability pattern. Fig. 2 illustrates the online influence spread (or reward) of all algorithms on both datasets with distorted probabilities. On the Flickr dataset in Fig. 2a, Random strategy performs the worst as expected. IMLinUCB works a little better than Random, while it's performance decreases at the beginning. This effect may be caused by IMLinUCB's estimation model since it shares an identical parameter for all edges, which mismatches the dataset's underlying pattern. The ϵ-greedy and CUCB algorithm both treat each edge independently. Nevertheless, the CUCB explores according to the estimation upper bound, while ϵ-greedy explores randomly. Hence CUCB outperforms ϵ-greedy under all seed size settings. IMFB utilizes the edge correlations to learn the diffusion parameters and achieves reasonable performance. However, IMGUCB outperforms IMFB in all seed size settings, and this may because IMGUCB could learn the diffusion probabilities under linear or other general mappings, while IMFB merely fits well under the linear probability pattern. On the NetHEPT dataset in Fig. 2b, the results are generally similar to thoses in Fig. 2a. The proposed IMGUCB achieves higher influence spread and learns sharper than IMFB. CUCB and IMLinUCB achieve comparable performance, while the ϵ-greedy performs the worst and learns slowly.

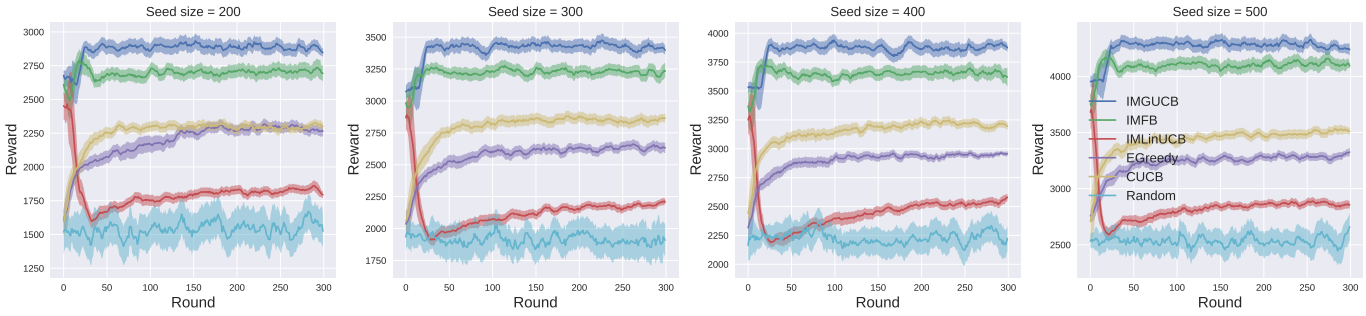
Apart from datasets with distorted probabilities, We also provide results on both datasets with linear probabilities in Tabel II. The linear probability pattern matches the assumption of the IMFB model, and we compare the proposed algorithm's performance with others under this scenario. Table II demonstrates that IMGUCB achieves better influence spread than other algorithms as well. The IMFB also achieves considerable influence spread since it builds the estimation model under the linear probability assumption. IMGUCB's superior influence

TABLE II: (Avg.) Reward comparison on the Flickr and NetHEPT datasets with *linear probabilities*. For IMGUCB, we set C to 2000, z_α to 2 and update *period* to 20. The IMGUCB achieves more average influence spread than other baselines, as marked in bold font.

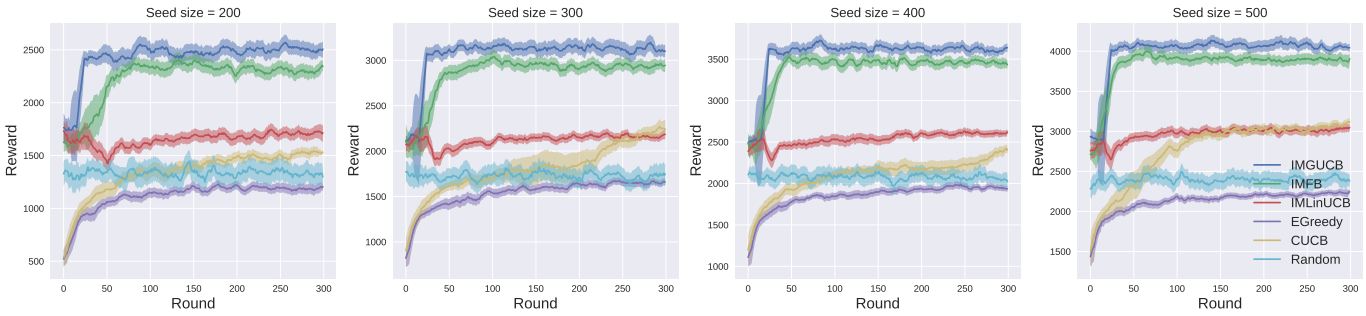
		IMGUCB	IMFB	IMLinUCB	ϵ-Greedy	CUCB	Random
Flickr	$\|S\|$ = 200	**1263.8**	1158.5	861.3	742.6	848.9	495.0
	$\|S\|$ = 300	**1713.9**	1559.7	1199.6	1026.8	1146.3	721.9
	$\|S\|$ = 400	**2102.6**	1921.8	1461.8	1298.4	1442.3	940.8
	$\|S\|$ = 500	**2482.3**	2287.8	1642.3	1515.2	1702.8	1150.5
NetHEPT	$\|S\|$ = 200	**1263.8**	1158.5	861.3	742.6	848.9	495.0
	$\|S\|$ = 300	**1713.9**	1559.7	1199.6	1026.8	1146.3	721.9
	$\|S\|$ = 400	**2102.6**	1921.8	1461.8	1298.4	1442.3	940.8
	$\|S\|$ = 500	**2482.3**	2287.8	1642.3	1515.2	1702.8	1150.5

TABLE III: (Avg.) round running time (seconds) comparison on the Flickr and NetHEPT datasets with *linear probabilities*. For IMGUCB, we set C to 2000, z_α to 2 and update *period* to 20. The IMGUCB's running time is comparable with that of IMFB. IMLinUCB runs faster than IMGUCB and IMFB since it shares a global parameter for all edges, however it achieves inferior online influence spread, as indicated in Table II.

		IMGUCB	IMFB	IMLinUCB	ϵ-Greedy	CUCB	Random
Flickr	$\|S\|$ = 200	4.57	4.47	1.36	6.59	6.73	0.0010
	$\|S\|$ = 300	6.45	6.45	2.07	10.67	10.42	0.0012
	$\|S\|$ = 400	9.10	10.99	2.78	17.89	17.67	0.0015
	$\|S\|$ = 500	9.51	10.29	2.41	19.79	19.23	0.0015
NetHEPT	$\|S\|$ = 200	4.45	3.04	1.40	2.35	3.44	0.0007
	$\|S\|$ = 300	5.24	3.90	2.22	4.87	6.01	0.0008
	$\|S\|$ = 400	6.51	5.13	3.79	7.28	8.87	0.0008
	$\|S\|$ = 500	8.86	6.36	5.87	10.43	12.62	0.0010



(a) Reward comparison with different seed sizes on the Flickr dataset with distorted influence probabilities.



(b) Reward comparison with different seed sizes on the NetHEPT dataset with distorted influence probabilities.

Fig. 2: Reward comparison of different algorithms on the Flickr and NetHEPT datasets with distorted influence diffusion probabilities for online influence maximization.

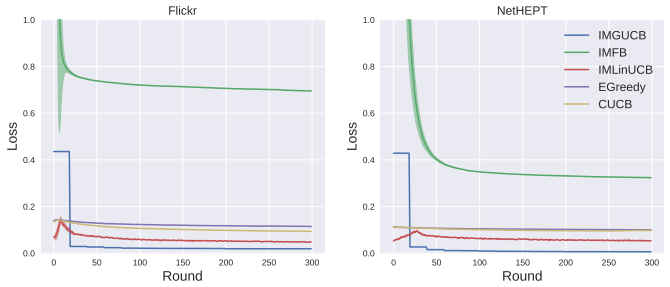


Fig. 3: The influence parameter estimation error comparision on two datasets with *distorted probabilities*. We fix the seed size to 300 for all baseline s. For IMGUCB, we fix z_α to 2, and C to 2000.

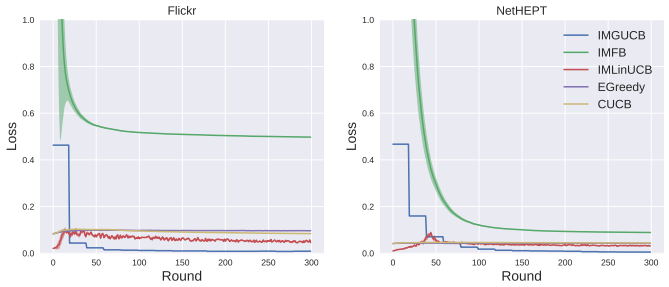


Fig. 4: The influence parameter estimation error comparision on two datasets with *linear probabilities*. We fix the seed size to 300 for all baselines. For IMGUCB, we fix z_α to 2, and C to 2000.

spread may be due to its smaller estimation error, which will be demonstrated in subsection IV-E.

b) Running Time: Besides, we list the average round running time of all algorithms on both datasets with linear probabilities in Table III. Table III shows that IMGUCB requires comparable running time with IMFB, e.g. on the Flickr dataset, IMGUCB is slightly faster than IMFB, while on the NetHEPT dataset, IMGUCB is somewhat slower than IMFB. The difference may be caused by the dataset's topology. IMLinUCB is much faster than IMGUCB and IMFB since it shares a global parameter for all edges, however, it achieves inferior online influence spread as indicated in Table II. CUCB and ϵ-greedy have similar running time since they adopt an identical update strategy, i.e. they both update all the observed edge's estimation independently after each interaction. The Random baseline has the least running time without surprise since it does not optimize estimations.

The results in Fig. 2 and Table II, III indicate that the proposed algorithm achieves superior performance under different probability patterns with comparable running time, not limited to linear diffusion probabilities. Previous linear model based algorithms, including IMFB and IMLinUCB, require the linear generalization assumption to achieve better performance.

E. Influence Parameter Estimation Quality

In this subsection, we report the influence probability estimation performance of different algorithms during the learning process. We compute the mean absolute error (or Loss) between the estimated probabilities and the ground-truth probabilities on all edges, not just on observed edges, i.e., Loss $= \frac{1}{|\mathcal{E}|}\sum_{e_i\in\mathcal{E}}|\hat{p_{e_i}} - p_{e_i}|$. The mean absolute error is a straight metric for influence parameter estimation quality, and the estimation quality affects the online influence spread directly. We omit the Random baseline since it does not estimate parameters.

Fig. 3 demonstrates the loss on both datasets with distorted probabilities. Note that we set seed size to 300 for all algorithms, since we observed that different seed sizes result in generally similar curves. For IMGUCB we set C to 2000 and z_α to 2 as stated in subsection IV-C. Fig. 3 shows that IMGUCB's loss decreases immediately after the first update at round 20 on both datasets, indicating that IMGUCB could estimate parameters effectively. IMFB decreases smoothly on both datasets though it converges to a relatively large value. IMLinUCB increases at the beginning and then decreases, which coincides with its online influence spread performance in Fig. 2. CUCB and ϵ-greedy converge slowly and achieve similar estimation quality on both datasets.

Fig. 4 shows the estimation error of all algorithms on both datasets with linear probabilities. The hyperparameters are identical to those in Fig. 3. The results are analogous to ones in Fig. 3. We can see that the IMGUCB decreases periodically (the period is set to 20 for all experiments), which coincides with the algorithm's periodical update strategy, and it converges to the lowest error level. Compared with Fig. 3, IMFB converges to smaller error levels on both datasets. On NetHEPT, IMFB converges to about 0.32 for distorted probabilities, while it converges to about 0.08 for linear probabilities. This result verifies that an algorithm performs better if the underlying dataset pattern matches its model assumption better. For IMLinUCB, CUCB, and ϵ-greedy, they achieve analogous performance. The CUCB and ϵ-greedy converge smoothly but slowly. The error of IMLinUCB firstly increases then decreases slowly, as in Fig. 3, which may be caused by the exploration strategy. Note that all algorithms have relatively small error variance at the end.

From the results revealed in Fig. 3 and Fig. 4, we conclude that the IMGUCB algorithm can effectively model the probabilities according to feedback, on both datasets with linear or distorted probabilities. The estimation error comparison of IMFB in Fig. 3 and Fig. 4 reveals that if a dataset's parameter pattern matches an algorithm's model assumption well, the algorithm generally performs well. Furthermore, the online influence spread is intimately connected with the influence parameter estimation error, since the estimations are adopted by the ORACLE subroutine to compute a seed set.

F. Hyperparameter Study

In this subsection, we study the hyperparameter robustness of the proposed IMGUCB algorithm. Note that the σ in the

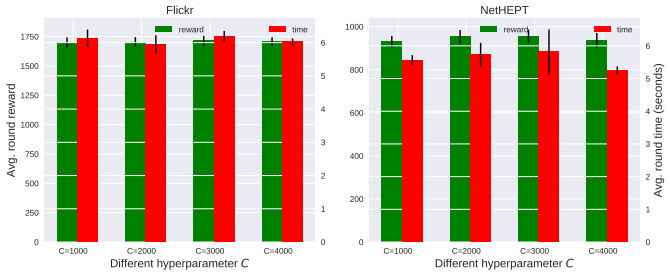


Fig. 5: Avg. reward and time (each ineration) of the proposed IMGUCB algorithm with different C, on both datasets with linear probabilities. The results show IMGUCB achieves similar performance with different $C \in [1000, 4000]$.

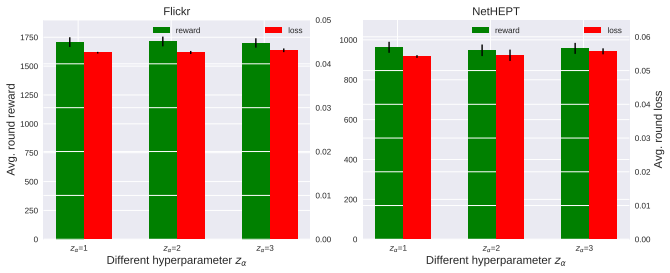


Fig. 6: Avg. reward and estimation error (loss) of the proposed IMGUCB algorithm with diverse z_α, on both datasets with linear probabilities. The results illustrate IMGUCB is insensitive w.r.t $z_\alpha \in [1, 3]$.

proposed algorithm represents our prior knowledge about the noise level, which we regard as a constant, and we set σ to 10^{-4} for all experiments. The periodical update strategy in IMGUCB is proposed for running efficiency, and it has trivial effects for the estimation performance (smaller period certainly results in more updates and more running time). We fix the period to 20 for all experiments. Hence we mainly focus on the effects of C and z_α. C decides the kernel size of the algorithm and affects estimation error on all edges. z_α decides the parameter estimation upper bound, affecting to which extent we explore and exploit. Therefore, the upper bound (or z_α) affects the ORACLE algorithm and the online influence spread straightly.

Fig. 5 and Fig. 6 reveal the algorithm's robustness w.r.t C and z_α, respectively. We set seed size to 300 for all algorithms. In Fig. 5, we compare IMGUCB's performance with different C on both datasets. The red bar represents average round reward and the green bar represents average round time. The results demonstrate that IMGUCB achieves almost identical average reward and running time, e.g., the reward is 1698, 1702, 1712, 1715, w.r.t $C = 1000$, 2000, 3000, 4000, respectively, on the Flickr dataset. The results on the NetHEPT dataset are analogous with those on the Flickr dataset. Fig. 6 demonstrates the average round reward (green bar) and average estimation error (red bar) of the IMGUCB, since z_α affects the estimation error and influence spread directly. We vary z_α from 1 to 3, and the results are almost identical on both datasets. For example, on the Flickr dataset, the average reward is 1707, 1713, 1700, and the average estimation error is 0.0425, 0.0426. 0.0431 for $z_\alpha = 1$, 2, 3, respectively. The results in Fig. 5 and Fig. 6 demonstrate the proposed algorithm's robustness w.r.t its important hyperparameters, hence generally we can set C in $[1000, 4000]$ and z_α in $[1, 3]$, without much sophisticated tunning. In most cases, we set C to 2000 and z_α to 2 as stated in subsection IV-C.

V. Conclusions and Future Work

In this paper, we propose to model the influence diffusion parameters using Gaussian Process according to the historical feedback. The Gaussian Process model generalizes previous common linear models to represent the influence probabilities reasonably, to capture the general relations between the influence parameters and graph feature representations. We propose the IMGUCB algorithm based on the Gaussian Process to utilize the bandit and UCB framework for exploration-exploitation balance. We conduct experiments on two practical datasets to verify the effectiveness of our algorithm. However, in our work, we utilize edge-level feedback for updating and rely on node features. In the future, we hope to combine online feature embedding subroutines with the proposed IMGUCB algorithm to learn node embeddings and optimize models better. Furthermore, how to utilize node-level feedback instead of edge-level ones is another promising direction to promote the algorithm's utility.

Acknowledgment

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

[1] D. Easley, J. Kleinberg *et al.*, *Networks, crowds, and markets*. Cambridge university press Cambridge, 2010, vol. 8.

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

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

[4] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *International Conference on Machine Learning*, 2013, pp. 151–159.

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

[6] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart, "Online influence maximization," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 645–654.

[7] D. Kalimeris, Y. Singer, K. Subbian, and U. Weinsberg, "Learning diffusion using hyperparameters," in *International Conference on Machine Learning*, 2018, pp. 2420–2428.

[8] S. Vaswani, L. Lakshmanan, M. Schmidt *et al.*, "Influence maximization with bandits," *arXiv preprint arXiv:1503.00024*, 2015.

[9] S. Vaswani, B. Kveton, Z. Wen, M. Ghavamzadeh, L. V. Lakshmanan, and M. Schmidt, "Model-independent online learning for influence maximization," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3530–3539.

[10] Z. Wen, B. Kveton, M. Valko, and S. Vaswani, "Online influence maximization under independent cascade model with semi-bandit feedback," in *Advances in neural information processing systems*, 2017, pp. 3022–3032.

[11] Q. Wu, Z. Li, H. Wang, W. Chen, and H. Wang, "Factorization bandits for online influence maximization," *arXiv preprint arXiv:1906.03737*, 2019.

[12] D. Kalimeris, G. Kaplun, and Y. Singer, "Robust influence maximization for hyperparametric models," *arXiv preprint arXiv:1903.03746*, 2019.

[13] Z. Zhao, S. Feng, Q. Wang, J. Z. Huang, G. J. Williams, and J. Fan, "Topic oriented community detection through social objects and link analysis in social networks," *Knowledge-Based Systems*, vol. 26, pp. 164–173, 2012.

[14] P. Bedi and C. Sharma, "Community detection in social networks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 3, pp. 115–135, 2016.

[15] Y.-C. Chen, W.-Y. Zhu, W.-C. Peng, W.-C. Lee, and S.-Y. Lee, "Cim: Community-based influence maximization in social networks," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 2, pp. 1–31, 2014.

[16] A. Bozorgi, S. Samet, J. Kwisthout, and T. Wareham, "Community-based influence maximization in social networks under a competitive linear threshold model," *Knowledge-Based Systems*, vol. 134, pp. 149–158, 2017.

[17] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[18] M. E. Newman, "Assortative mixing in networks," *Physical review letters*, vol. 89, no. 20, p. 208701, 2002.

[19] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 397–422, 2002.

[20] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 420–429.

[21] A. Goyal, W. Lu, and L. V. Lakshmanan, "Celf++: optimizing the greedy algorithm for influence maximization in social networks," in *Proceedings of the 20th international conference companion on World wide web*. ACM, 2011, pp. 47–48.

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

[23] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1029–1038.

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

[25] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 241–250.

[26] K. Saito, R. Nakano, and M. Kimura, "Prediction of information diffusion probabilities for independent cascade model," in *International conference on knowledge-based and intelligent information and engineering systems*. Springer, 2008, pp. 67–75.

[27] S. Bourigault, S. Lamprier, and P. Gallinari, "Representation learning for information diffusion through social networks: an embedded cascade model," in *Proceedings of the Ninth ACM international conference on Web Search and Data Mining*, 2016, pp. 573–582.

[28] W. Chen, Y. Wang, Y. Yuan, and Q. Wang, "Combinatorial multi-armed bandit and its extension to probabilistically triggered arms," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1746–1778, 2016.

[29] P. Lagrée, O. Cappé, B. Cautis, and S. Maniu, "Algorithms for online influencer marketing," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 1, p. 3, 2018.

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

[31] J. McAuley and J. Leskovec, "Image labeling on a network: using social-network metadata for image classification," in *European conference on computer vision*. Springer, 2012, pp. 828–841.

[32] J. Gehrke, P. Ginsparg, and J. Kleinberg, "Overview of the 2003 kdd cup," *Acm SIGKDD Explorations Newsletter*, vol. 5, no. 2, pp. 149–151, 2003.

[33] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 177–187.