Decentralized Online Influence Maximization

Yigit E. Bayiz
The University of Texas at Austin
Austin, USA
egebayiz@utexas.edu

Ufuk Topcu
The University of Texas at Austin
Austin, USA
utopcu@utexas.edu

***Abstract*—We consider the problem of finding the maximally influential node in random networks where each node influences every other node with constant yet unknown probability. We develop an online algorithm that learns the relative influences of the nodes. It relaxes the assumption in the existing literature that a central observer can monitor the influence spread globally. The proposed algorithm delegates the online updates to the nodes on the network; hence requires only local observations at the nodes. We show that using an explore-then-commit learning strategy, the cumulative regret accumulated by the algorithm over horizon T approaches $O(T^{2/3})$ for a network with a large number of nodes. Additionally, we show that, for fixed T, the worst case-regret grows linearly with the number n of nodes in the graph. Numerical experiments illustrate this linear dependence for Chung-Lu models. The experiments also demonstrate that ε-greedy learning strategies can achieve similar performance to the explore-then-commit strategy on Chung-Lu models.**

***Index Terms*—influence maximization, online learning, social networks, algorithm design**

I. Introduction

Influence maximization is the problem of selecting the most influential subset of nodes in a network, given a budget on the size of the selected subset. With the rise of marketing on social networks and the recent realization of the importance of tracking epidemics, the influence maximization problem has gained a renewed interest in the last few years [1]. The notion of influence in such problems is a measure of how well the selected subset spreads a dispersive effect of interest, such as an advertisement [2] or a disease [3], to the other nodes in the network. For consistency with the existing influence maximization literature, we call this dispersive effect *influence*.

We consider influence maximization problems on networks modeled as random graphs, where we label the nodes that have received the influence as *active* and all other nodes as *inactive*. All nodes are initially inactive. A decision-maker then activates a set of nodes called the *seed nodes* and the influence spreads from these seed nodes based on an influence spread model called the *independent cascade process* [4]. In this model, the influence spreads iteratively, where in each iteration, active nodes in the network can activate their neighboring inactive nodes with some time-independent probability, called the *influence probability*. A more detailed description of the independent cascade process is provided in Section 3.1. Under this model, influence maximization can be characterized as the problem of choosing the node or the set of nodes that, when activated initially, maximizes the number of active nodes in the network after the independent cascade process terminates.

The influence maximization problem we consider is NP-hard with respect to the number of nodes in the network [4]. Thus, only approximate solutions can be found for large networks. We can classify these approximate solution algorithms for influence maximization as offline and online [1]. Both of these approaches have their limitations. The offline approaches provide scalable solutions for large graphs. However, they only work if all influence probabilities of the underlying graph are known beforehand.

Online algorithms have been introduced to relax the constraint that the influence probabilities of the graph are known [5]. These algorithms learn the influence probabilities of the graph at run-time and gradually improve their seed node selections. However, the existing online algorithms require a central decision-maker to observe which nodes the influence spreads to. This assumption may not be realizable in real-world networks. For example, in an epidemic tracking application, observing how much an individual would spread disease to the rest of the community requires the full contact history of everyone within the community. Even if we ignore the privacy concerns arising from having access to such data, the complete contact history is difficult to obtain and computationally expensive to process.

In this work, we introduce two decentralized online influence maximization algorithms that delegate the online updates to the nodes within the network and eliminate the requirement of a central decision-maker to observe the influence spread. To our knowledge, this paper is the first to use such a decentralized online approach to solve the influence maximization problem. In addition to reducing the computational burden on the central decision-maker, one of the algorithms introduced in this paper provides guaranteed asymptotic upper bounds on the expected total regret for networks with symmetric influence probabilities. We also demonstrate the performance of both of the algorithms experimentally on synthetically generated social networks.

The rest of this paper is organized as follows. In section 2, we provide a review of current literature relevant to the work presented in this paper. Section 3 presents some preliminary information and states the decentralized online influence maximization problem. Section 4 introduces algorithms that can solve this problem under a symmetry assumption on influence probabilities, and provides convergence guarantees. Section 5 shows empirical results that verify the guarantees given in section 4. We conclude this paper and give future research directions in section 6.

II. Related Work

Influence maximization on independent cascade networks was introduced by [4]. This paper also showed that the influence maximization problem is NP-hard and provided the first approximate offline solution, greedy forward selection, to the problem. The following decade saw incremental improvements to this algorithm with shortest-path-based algorithms [6], CELF [3], CELF++ [7], and TIM [8]. More recently, Tang et. al. have proposed several algorithms with near-linear time complexity guarantees, which makes these algorithms highly scalable for large networks [9]. However, all of these algorithms are offline and they require the graph structure to be known beforehand. Our algorithm differs from all of these approaches since it does not require the influence probabilities between nodes to be known and can learn the optimal seed node choice incrementally using online updates.

Online influence maximization problem was introduced by [5], to investigate how the influence maximization problem can be adapted to cases where a decision-maker tries to maximize the sum of total influences across multiple iterations. In this problem, the decision-maker repeatedly chooses seed nodes and observes the influence spread over the network until some known iteration number, called *horizon* is reached. The decision-maker needs to learn the best node choices at run-time by refining its beliefs on relative values of nodes or the influence probabilities between nodes. In these algorithms, there is always a trade-off between exploration and exploitation, where the decision-maker either chooses a potentially sub-optimal seed to learn more information about the network or chooses the seed it believes to be optimal at the cost of less information. Approaches to online influence maximization include learning the influence probabilities in the network by observing influence spread over the network and using the learned influence probabilities to solve the problem using offline methods [10], [11] or casting the problem as a bandit problem [12], [13]. Unfortunately, all of these methods require the observation of the activation state of all nodes on the network. So far, attempts to restrict the required information to include only local influence spread, such as the one given by [14], have only provided convergence guarantees for specific network models. The algorithm we present is a decentralized alternative to the approaches given above, and it does not require global observation of node activations. It also yields general worst-case asymptotic convergence bounds on networks with symmetric influence probabilities.

III. Preliminaries and Problem Statement

A. Network Model

We use a weighted graph $\mathcal{G} = (V, E, \mathcal{P})$ to model the network on which the influence spreads. Here V denotes the set of vertices in the network and $E \subseteq V \times V$ (or $E \subseteq \{\{a, b\} |\ a, b \in V\}$ for the undirected case) denotes the set of all possible connections in the network. Lastly, $\mathcal{P} : E \to [0, 1]$ is a function which assigns weights to each edge, and these weights represent the influence probabilities on the network. That is $\mathcal{P}((i, j))$ is the probability of i influencing j, given that the influence has already been spread to i.

The *independent cascade process* [4] is a model which describes the diffusion of influence over the network using successive discrete steps called *cascades*. In each cascade, all nodes are labeled as either *active* or *inactive* where active nodes represent the influenced nodes. In the first cascade, the network is initialized with a set of active nodes $S \subseteq V$, called *seeds*, and all other nodes are initialized as inactive. In each cascade after initialization, if a node $i \in V$ becomes active in the current cascade, it activates a node $j \in V$ in the next cascade with probability $\mathcal{P}((i, j))$, given that j is inactive in the current cascade. Once a node becomes active, it remains active for the rest of the cascades. This process terminates if no new nodes are activated in the current cascade. The set of nodes that are active after the independent cascade process terminates is called the *influence set of S on $\mathcal{G}$* and is denoted as $\sigma(S, \mathcal{G})$, or just $\sigma(S)$ if the underlying network structure is clear from the context.

Based on the influence probabilities given by $\mathcal{P}$. $\mathcal{G}$ can occupy two different regimes, called super-critical and sub-critical regimes. We can determine the regime $\mathcal{G}$ occupies the from the largest eigenvalue λ in the *adjacency matrix* $[P((i, j))]_{ij}$ of the graph $\mathcal{G}$. If $\lambda > 1$ then the graph is called super-critical, which means the size of the largest connected component of $\mathcal{G}$ is linear with the number of nodes n in $\mathcal{G}$. For the independent cascade process defined above, the super-criticality translates to the size of $\max_S \sigma(S, \mathcal{G})$ being linear with n. If $\lambda < 1$ then the graph is sub-critical and the size of $\max_S \sigma(S, \mathcal{G})$ grows sub-linearly with n [15].

B. Problem Formulation

Given a network $\mathcal{G} = (V, E, \mathcal{P})$ and a horizon T, we consider a scenario where there exists a central decision-maker, which in each iteration $t \in [T]$ selects a seed $s_t \in V$. Then, s_t is activated by the central decision-maker and influence diffuses over the network based on the independent cascade process. The influence set resulting from this diffusion process is denoted by $\sigma_t(s_t)$. We also let $L_t^{s_t}(i) = \mathbf{1}_{\sigma_t(s_t)}(i)$. That is, $L_t^{s_t}(i)$ is the indicator function of $\sigma_t(s_t)$. Then the *total influence* of s_t at iteration t can be written as

$$\mathcal{I}_t(s_t) = |\sigma_t(s_t)| = \sum_{i \in V} L_t^{s_t}(i). \tag{1}$$

We consider a decentralized scenario where both the central decision-maker and the nodes in $\mathcal{G}$ are capable of executing algorithms, and the central decision-maker cannot directly observe which nodes are influenced, or which edges in E are active at any given iteration $t \in [T]$. This makes the central decision-maker incapable of performing online updates to learn which nodes are the most influential ones. Instead, the nodes in network $\mathcal{G}$ are responsible for keeping and updating their own estimates for how influential they are. This distinction separates the work presented in this paper from the existing influence maximization literature. The nodes are only allowed to make local observations on whether they are influenced or not. That is, each node $i \in V$ can observe $L_t^{s_t}(i)$ and execute local updates based only on this information. Neither the decision-maker nor any node has any information about the influence probabilities $\mathcal{P}$ at iteration $t = 1$.

Ideally, the task of the decision-maker is to choose and activate a seed s_t in each iteration t so that the sum of total influences across all iterations is maximal. That is, the decision-maker wants to maximize

$$\sum_{t=1}^{T} \mathcal{I}_t(s_t). \tag{2}$$

However, due to the lack of knowledge of the influence probabilities at iteration $t = 1$, the decision-maker can never guarantee that the sum of total influences across all iterations are maximized. Instead, the choices for s_t made by the decision-maker will probably be sub-optimal. We measure this sub-optimality using total regret

$$R = \big[\sum_{t=1}^{T} \mathcal{I}_t(s^*) - \mathcal{I}_t(s_t)\big], \tag{3}$$

where s^* is the node on $\mathcal{G}$ that is expected to be the most influential. That is, $s^* = \arg\max_{s \in V} \mathbb{E}(\mathcal{I}_t(s))$.

In the decentralized scenario we have, the central decision-maker and the nodes on the network must work together to ensure that the expectation of the regret $\mathbb{E}[R]$ is kept as low as possible. We call the collection of algorithms executed by the central decision-maker and the nodes a *decentralized algorithm*. With this definition in mind, we present Problem 1, and Assumption 1. In this paper, we provide a solution for Problem 1 under Assumption 1.

Problem 1. *Find a decentralized algorithm for online influence maximization with convergence guarantees on* $\mathbb{E}[R]$ *such that:*

- *Network nodes* $i \in V$ *keep and update their own estimates for how influential they are based only on the local activation information* $L_t^{s_t}(i)$.
- *Central decision-maker selects and activates a seed node* s_t *in each turn-based only on the estimates made by the nodes in the network.*

Assumption 1. *All influence probabilities on the network are symmetric. That is, given the network* $\mathcal{G} = (V, E, \mathcal{P})$, *for all* $(i, j) \in E$ *we have* $(j, i) \in E$, *and*

$$\mathcal{P}((i,j)) = \mathcal{P}((j,i)). \tag{4}$$

In particular, for all $t \in [T]$ *we have*

$$\mathbb{E}[L_t^{s_t}(i)] = \mathbb{E}[L_t^{i}(s_t)]. \tag{5}$$

Notation	Definition
$\mathcal{G}$	The graph representing the network. $\mathcal{G} = (V, E, \mathcal{P})$.
n	Number of nodes in $\mathcal{G}$, that is $n := \lvert V \rvert$.
m	Exploration horizon.
$X_{t,i}$	Total reward collected by the node i until time t.
$\sigma(s)$	Set of influenced nodes in $\mathcal{G}$ for seed s.
L_t^s	The indicator function of $\sigma(s)$.
$\mathcal{I}_t$	$\mathcal{I}_t(s) := \lvert\sigma_t(s)\rvert = \sum_{i\in V} L_t^s(i)$.
$\Delta_{i,j}$	$\Delta_{i,j} := \mathbb{E}[\mathcal{I}_t(i) - \mathcal{I}_t(j)]$.
Δ_i	$\Delta_i := \max_j \mathbb{E}[\mathcal{I}_t(j)] - \mathbb{E}[\mathcal{I}_t(i)]$.

TABLE I: A summary of notations and their definitions.

IV. Decentralized Influence Maximization

We now propose an online algorithm (Algorithm 1), to solve Problem 1 under Assumption 1. In this algorithm, each node i is responsible for keeping its own value $X_{t,i}$. In each iteration t of Algorithm 1, the decision-maker picks a seed node s_t. Then each node i in the network updates its own value simultaneously based on $L^{s_t}(i)$, which determines whether i is influenced by s_t or not. When selecting seed nodes, the decision-maker must choose between exploration and exploitation. In Algorithm 1 the decision-maker makes this choice based on the explore-then-commit (ETC) rule. Under the ETC rule, given some predetermined *exploration horizon* m, if iteration number t is less than or equal to m, the algorithm chooses seed nodes uniformly at random from V. Otherwise, the algorithm chooses a seed node with a maximal value.

Algorithm 1 ETC with decentralized updates

```
  Input A network $\mathcal{G} = (V, E, \mathcal{P})$,
  horizon $T$.
  exploration horizon $m$.
  Initialize $\forall i \in V,\ X_{0,i} = 0$.
for $t = 1$ to $T$ do
  if $t < m$ then
    Choose a seed $s_t \in V$ uniformly at random.
  else
    Choose a seed $s_t = \arg\max_i X_{t,i}$.
  end if
  The network generates activations $L_t$.
  for Each node $i \in V$ in parallel do
    Observe the local activation $L_t(i)$.
    Update $X_{t+1,i} \leftarrow X_{t,i} + L_t(i)$.
  end for
  Update $t \leftarrow t + 1$.
end for
```

Unlike existing algorithms, Algorithm 1 does not require the decision-maker to make global observations on the activation state $L^{s_t}(i)$ of all nodes $i \in V$ in the network. Instead, Algorithm 1 delegates the online updates of node value $X_{t,i}$ to the nodes themselves. This way, any given node i is only required to observe its own activation $L^{s_t}(i)$. The only global information the decision-maker needs to observe in any given iteration t is the node

$$\bar{i} = \arg\max_i X_{t,i}. \tag{6}$$

That is, the decision-maker must know at least one node with a maximal value.

There are several different methods for the decision-maker to observe $\bar{i}$. A naive approach the decision-maker can use to obtain this information is sequentially polling each node in the network and comparing their values to find the maximal one. However, polling each node defeats the purpose of Algorithm 1, since it requires the decision-maker to globally observe each nodes' value. Instead, depending on the physical or technical limitations of the network, it may be possible to compute $\bar{i}$ without polling each node individually. For example, if the physical realization of the network $\mathcal{G}$ allows the nodes to transfer arbitrary messages between each other over the network edges, e.g. social networks, then methods such as decentralized optimization or loopy-message-passing [16] can be used to efficiently find $\bar{i}$ without the need for polling.

In addition to reducing the computational burden on the central decision-maker, Algorithm 1 enjoys several theoretical convergence guarantees under Assumption 1. Firstly for large enough m, the seed nodes Algorithm 1 chooses converge to node i^* with the highest expectation of being activated by another node chosen uniformly at random from V. That is, for a randomly selected node $J \sim \text{Uniform}(V)$, the seed choices converge to

$$i^* = \arg\max_{i\in[n]} \mathbb{E}[L^J(i)]. \tag{7}$$

For more general networks where Assumption 1 does not hold, node i^* is not necessarily the most influential node in the network. However, due to the symmetry enforced by Assumption 1, i^* is guaranteed to maximize expected influence $\mathbb{E}[\mathcal{I}(i^*)]$ because

$$\mathbb{E}[L^J(i)] = \sum_{j\in[n]} \mathbb{P}(J=j)\mathbb{E}[L^j(i)] = \frac{1}{n}\mathbb{E}[\mathcal{I}(i)]. \tag{8}$$

In other words, under Assumption 1, the node with the highest expectation of being influenced by a randomly chosen node is also the most influential node in the network. Thus, given sufficiently large m, the seed choices of Algorithm 1 converge to the node with the highest expected influence in the network.

We can also exploit the symmetry enforced by Assumption 1 to yield upper bounds on the expected regret of Algorithm 1. To derive this upper bound, we first define the *sub-optimality* Δ_i of a node i.

Definition 1. *The sub-optimality Δ_i of node $i \in V$ is defined as*

$$\Delta_i = \max_j \mathbb{E}[\mathcal{I}_t(j)] - \mathbb{E}[\mathcal{I}_t(i)]. \tag{9}$$

Lemma 1 gives an upper bound on the probability of a suboptimal node having a higher value than the node with the highest expected influence $\mathbb{E}[\mathcal{I}_t(j)]$ in the network.

Lemma 1. *Suppose Assumption 1 holds. Then if $n = |V|$, s_t is selected uniformly from V for all $t \in [m-1]$, and $X_{t,i} = \sum_{t=1}^{t}(L_t^{s_t}(i))$, then any node pair $i, j \in V$ satisfies*

$$\mathbb{P}(X_{m,j} \geq X_{m,i}) \leq \exp\left(-\frac{m\Delta_{i,j}^2}{2n^2}\right), \tag{10}$$

where $\Delta_{i,j} = \mathbb{E}[\mathcal{I}_t(i) - \mathcal{I}_t(j)]$. In particular if node i^* is optimal, then

$$\mathbb{P}(X_{m,i} \geq X_{m,i^*}) \leq \exp\left(-\frac{m\Delta_i^2}{2n^2}\right). \tag{11}$$

Proof. For all $t \in [T]$ we have

$$\mathbb{E}[L_t^{s_t}(i)] = \mathbb{E}[L_t^i(s_t)] = \mathbb{E}[\mathcal{I}_t(i)]/n. \tag{12}$$

Then the proof is given by a chain of inequalities,

$$\mathbb{P}(X_{m,i} \geq X_{m,j}) = \mathbb{P}\left(\frac{1}{m}(X_{m,j} - X_{m,i}) \geq 0\right) \tag{13a}$$

$$= \mathbb{P}\left(\frac{1}{m}\sum_{t=1}^{m}(L_t^{s_t}(j) - L_t^{s_t}(i)) \geq 0\right) \tag{13b}$$

$$= \mathbb{P}\left(\frac{1}{m}\sum_{t=1}^{m}(L_t^{s_t}(j) - L_t^{s_t}(i)) - \frac{\Delta_{j,i}}{n} \geq \frac{\Delta_{i,j}}{n}\right) \tag{13c}$$

$$\leq \exp\left(-\frac{m\Delta_{i,j}^2}{2n^2}\right). \tag{13d}$$

Here, (13d) follows from Hoeffding's inequality. □

Theorem 2 and Corollary 3 provides means for choosing the exploration parameter m such that asymptotic regret bounds exist under Assumption 1. These results show that under Assumption 1, regret is bounded by an $O(T^{2/3})$ function for fixed n. The sub-linearity of regret with respect to T shows that Algorithm 1 is indeed capable of successfully learning the most influential node in the network for large enough T.

Theorem 2. *Suppose Assumption 1 holds and let $\mathcal{G} = (V, E, \mathcal{P})$ be a network and define $c(n) = \max_s \sigma(S, \mathcal{G})/n$. Then, if the exploration horizon is chosen as $m = \alpha T^{2/3}$, the expected regret $\mathbb{E}[R]$ of Algorithm 1 satisfies*

$$\mathbb{E}[R] < c(n)(1-c(n))\alpha n T^{2/3} + e^{-1/2}\alpha^{-1/2}n^2T^{2/3}. \tag{14}$$

Proof. (Sketch) The regret R can be decomposed as $R = R_{(0,m]} + R_{(n,T]}$ where $R_{(0,m]} = \sum_{t=1}^{m}\Delta_i$ and $R_{(m,T]} = \sum_{t=m+1}^{T}\Delta_i$. From the definitions of Δ_i and $c(n)$, it is simple to show that $\mathbb{E}[R_{(0,m]}] \leq mnc(n)(1-c(n))$. On the other hand, using Lemma 1, we can give an upper bound on $\mathbb{E}[R_{(m,T]}]$ as

$$\mathbb{E}[R_{(m,T]}] \leq (T-m)\sum_{i=1}^{n}\exp\left(-\frac{m\Delta_i^2}{2n^2}\right)\Delta_i. \tag{15}$$

Notice that $\exp(-\frac{m\Delta_i^2}{2n^2})\Delta_i$ is maximized for $\Delta_i = n/\sqrt{m}$, And its value for this Δ_i is $e^{-1/2}n/\sqrt{m}$. This maximal value can be used to simplify the bound on $\mathbb{E}[R_{(m,T]}]$ as

$$\mathbb{E}[R_{(m,T]}] \leq \left(\frac{T}{\sqrt{m}} - \sqrt{m}\right)n^2e^{-1/2}. \tag{16}$$

Adding up the upper bounds for $\mathbb{E}[R_{(0,m]}]$ and $\mathbb{E}[R_{(m,T]}]$ and simplifying the resulting algebraic expression completes the proof. □

Corollary 3. *Suppose that Assumption 1 holds. Then for any arbitrary positive constant α there are non-negative constants C_1, and C_2 such that for all horizons T and number $n = |V|$ of nodes, if the learning horizon m is set as $m = \alpha T^{2/3}$, we have*

$$\mathbb{E}[R] \le \min(C_1 + C_2 n^2 T^{2/3},\ nT). \tag{17}$$

Proof. From Theorem 2, we can immediately deduce that $\mathbb{E}[R] \le C_1 + C_2 n^2 T^{2/3}$ for some constants C_1, C_2. Also, since

$$\mathbb{E}[R] = \sum_{t=1}^{T} \Delta_t \le \sum_{t=1}^{T} n = nT, \tag{18}$$

we have $\mathbb{E}[R] \le \min(C_1 + C_2 n^2 T^{2/3},\ nT)$. □

Algorithm 2 ε-greedy with decentralized updates

```
    Input A network $\mathcal{G} = (V, E, \mathcal{P})$,
    horizon $T$.
    an array of exploration parameters $[\varepsilon_1 \dots \varepsilon_T]$.
    Initialize $\forall i \in V,\ X_{0,i} = 0$.
 1: for $t = 1$ to $T$ do
 2:     Pick $r_t \sim \text{Uniform}([0,1])$
 3:     if $r_t < 1 - \varepsilon_t$ then
 4:         Choose a seed $s_t \in V$ uniformly at random.
 5:     else
 6:         Choose a seed $s_t = \arg\max_i X_{t,i}$.
 7:     end if
 8:     The network generates activations $L_t$.
 9:     for Each node $i \in V$ in parallel do
10:         Observe the local activation $L_t(i)$.
11:         if $s_t \ne i$ then
12:             Update $X_{t+1,i} \leftarrow X_{t,i} + L_t(i)$.
13:         end if
14:     end for
15:     Update $t \leftarrow t + 1$.
16: end for
```

Instead of basing the exploration-exploitation decision on an explore-then-commit rule, we can modify Algorithm 1 such that it uses ε-greedy decision rule. In this decision rule, the central decision makes is provided with a sequence $(\varepsilon_t)_T \in [0,1]^T$ of learning parameters, and at each time t the central decision-maker chooses to explore with probability ε_t and chooses to exploit with probability $1 - \varepsilon_t$. The modified algorithm which uses this decision rule is shown in Algorithm 2.

The ε-greedy decision rule makes Algorithm 2 a generalization of Algorithm 1. Specifically, Algorithm 2 reduces to Algorithm 1 given that

$$\varepsilon_t = \begin{cases} 1, & t < m \\ 0, & t \ge m \end{cases} \qquad \forall t \in [T]. \tag{19}$$

The advantage of using the ε-greedy decision rule as compared to the explore-then-commit rule is that the former allows finer control over the exploration-exploitation trade-off. That is, depending on the specific problem requirements, ε_t can also be tailored in such a way that the convergence of Algorithm 2 satisfies these requirements. For example, if the network in which the algorithm is to operate has influence probabilities that are not stationary over time, then the explore-then-commit based method given in Algorithm 1 can fail to converge since the algorithm cannot adapt to changing influence probabilities after the exploration period is over. In contrast, Algorithm 2 does not have a definite exploration period, and choosing the sequence $(\varepsilon_t)_T$ as a constant series $\varepsilon_t := \varepsilon$ allows the algorithm to adapt to changes in the influence probabilites indefinitely.

The optimal choice for the sequence ε_t depends on the influence probabilities of the network $\mathcal{G}$. This optimal choice can be difficult to compute exactly. In the following section we test two variants of Algorithm 2 based on different choices for the sequence ε_t. For convenience, we label these variants EGreedy1 and EGreedy2. These variants have exploration parameters defined as

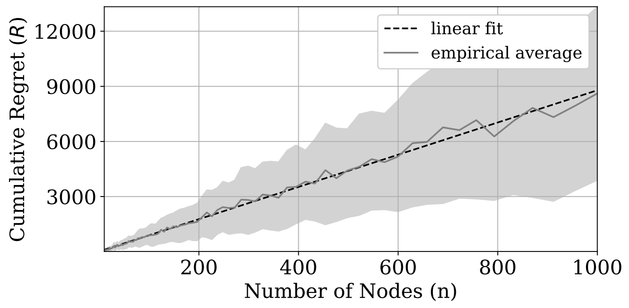
$$\begin{aligned} &\text{EGreedy1:} \quad \varepsilon_t := \beta^{t-1}, \quad \text{where} \quad \beta \in [0,1] \\ &\text{EGreedy2:} \quad \varepsilon_t := \max(1,\ C/t), \quad \text{where} \quad C > 0, \end{aligned} \tag{20}$$

where β and C are parameters that determine the trade-off between exploration and exploitation. EGreedy1 and EGreedy2 are not necessarily optimal. However, they are common choices for ε_t in other online learning algorithms based on ε-greedy decision rule, and they are known to perform well in many applications.

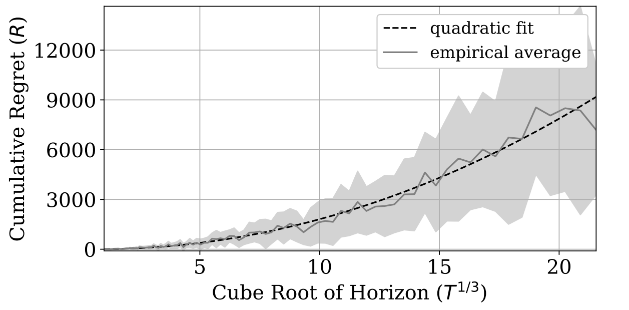
V. Experiments

In this section, we test the dependence of Algorithms 1 and 2 to their respective algorithm parameters on randomly generated networks. The networks in the numerical experiments are based on Chung-Lu models [17], simple yet commonly used approximations for real social networks. These models are specified using a weight vector $[w_1, w_2 \dots w_n]$ that assigns a weight to each node on the network, and the influence probabilities of the network is defined as $P((i,j)) = w_i w_j / \sum_k w_k$. We choose $w_i \sim$ Uniform([0, W]) where W is a constant which denotes the maximal allowed expected degree of the nodes on the graph. We now report observations from test where $W = 2$. Though not included in the paper, it is worth emphasizing that similar observations are valid for other choices for W as well.

To test the dependence of cumulative regret R of Algorithm 1, on the number n of nodes in the network, we first set the horizon $T = 100$ and $m = T^{2/3}$. Then, we sample 100 different values for n logarithmically from the interval $[10, 1000]$. For each n, we simulate the cumulative regret obtained by Algorithm 1 after T iterations on a Chung-Lu model with n nodes. We then repeat this simulation 200 times and compute the empirical average and one standard deviation confidence bounds of the cumulative regret across



(a) Number n of nodes in the network.



(b) Horizon T.

Fig. 1: Dependence of cumulative regret against: (a) Number n of nodes in the network, (b) horizon T. The light gray area indicates one standard deviation confidence bounds.

all simulations. Figure 1 (top) shows these empirical averages and confidence bounds as functions of n. The coefficient of determination for the best linear fit to the empirical averages has a $R^2 \geq 1 - 2.2 \times 10^{-11}$. The high value of R^2 means that for fixed T, the cumulative regret is approximated well by a linear function of n, which suggests that, for fixed T, the regret $\mathbb{E}[R]$ is $O(n)$, as predicted by Corollary 3.

Similarly, we test the dependence of regret of Algorithm 1 on horizon T by sampling 100 values of T logarithmically from $[1, 10000]$ while setting $m = T^{2/3}$ and $n = 100$. Figure 1 (bottom) shows the empirical means and one standard deviation bounds of the regret as functions of $T^{1/3}$. The coefficient of determination of the quadratic best fit to the empirical mean is $R^2 \geq 1 - 1.8 \times 10^{-10}$, which shows that the average regret $\mathbb{E}[R]$ is a quadratic function of $T^{1/3}$ for fixed n. In other words, $\mathbb{E}[R]$ is of order $T^{2/3}$, which is consistent with the theoretical findings presented in Corollary 3.

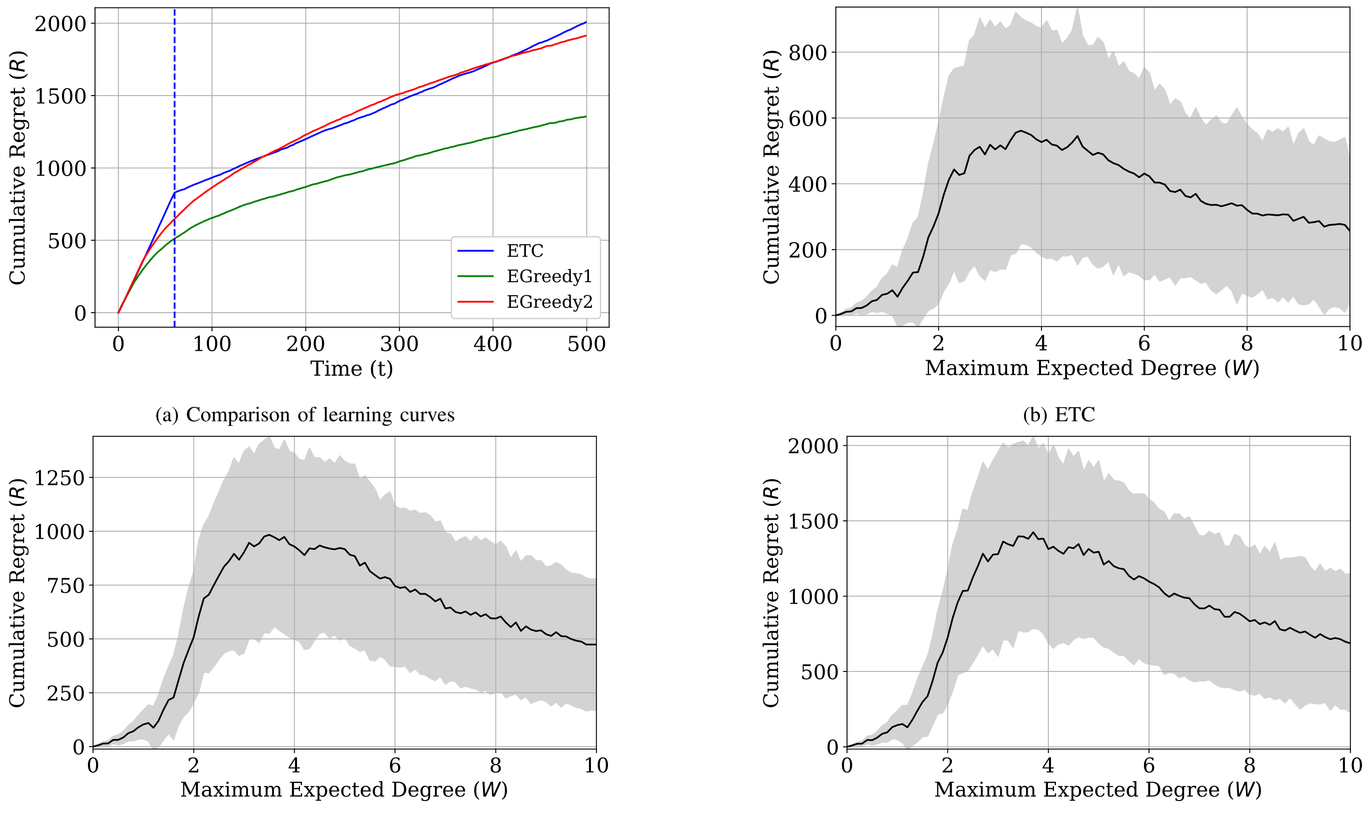
Recall that for $m = \alpha T^{2/3}$, Theorem 3 provides convergence bounds on Algorithm 1, where α is an arbitrary positive constant. Unfortunately, none of the theoretical results provide an optimal choice for α that minimizes regret. However, at least for the randomly generated Chung-Lu models, there is empirical evidence that suggests small deviations from the optimal α do not affect the cumulative regret greatly and $\alpha = 1$ yields nearly optimal results. For fixed $T = 500$ and $n = 100$ we discovered that while the optimal choice for α is around 0.95, choosing $\alpha = 1$ instead of this optimal value only results in a 5% increase in cumulative regret, which is not very significant. Similar results were obtained for different choices for n and W.

As mentioned in the previous section, we test the performance of Algorithm 2 using the two different settings for exploration parameters ε_t. We call the algorithm variants arising from these two settings EGreedy1 and EGreedy2, the exploration parameters of which can be seen in (20). For tests done on Chung-Lu models with $n = 100$ and $W = 2$, and horizon T set to 500, we have empirically found that choosing $\beta = 0.97$ and $C = 20$ provides near-minimal cumulative regret. Hence we use these values for our consequent tests. Figure 2a shows average learning curves of EGreedy1, EGreedy2, and explore-then-commit based Algorithm 1, labeled as ETC, with $m = 60 \approx T^{2/3}$. These learning curves are averaged over 2000 separate simulations. Figure 2a shows that although the average one step regret decreases over time in all algorithms, the EGreedy1 algorithm provides better results than both ETC and EGreedy2. Nevertheless it is important to note that the relative performance of these algorithms in general depends on the problem at hand.

Lastly, we experiment with changing the maximum allowed degree W in the Chung-Lu models we used to see how the cumulative regret changes with the operating regime of the graph. Recall that a network is called super-critical if the size of the largest connected component is linear with n, and it is called sub-critical otherwise. For the Chung-Lu models we are using, the operating regime of the network is directly dependent on W, where $W > 2$ leads to a super-critical network and $W < 2$ leads to a sub-critical network. Figures 2b, 2c, 2d show the cumulative regrets of Algorithm 1, EGreedy1, EGreedy2, against the maximum allowed degree W. The data obtained here were averaged over 1000 runs, where the number of nodes n in the network and horizon T is both fixed to 100. For these tests, have set $m = 22 \approx T^{2/3}$, $\beta = 0.97$, and $C = 20$. One standard deviation confidence bounds are also shown as grey regions around the average regret. From Figure 2, we can observe that, compared to super-critical networks, the cumulative regrets of all three algorithms are generally smaller in sub-critical networks for the range of values of W we have tested. An intuitive explanation for this behavior is that the worst-case regret in a single iteration is bounded above by the size of the largest influence set, $\max_S \sigma(S, \mathcal{G})$, which is $o(n)$ for sub-critical networks as opposed to $O(n)$ for super-critical networks. Thus, for large enough n, sub-critical networks enjoy smaller upper bounds on their one-step regrets. Also, the cumulative regret approaches 0 for sufficiently large values for W since the probability of the entire network being connected approaches 1 as W increases. The maximal regret R is achieved when W is small, yet large enough to cause the network to be super-critical.

VI. Conclusion and Future Work

We consider the online influence maximization on networks, where the nodes on the network are capable of performing decentralized computations. We propose an explore-then-commit algorithm and an ε-greedy algorithm that delegate the online updates to nodes on the network. These algorithms



(a) Comparison of learning curves

(b) ETC

(c) EGreedy1

(d) EGreedy2

Fig. 2: Comparison of three different influence maximization algorithms. (a) The comparison of average learning curves for $W = 2$, vertical dashed blue line indicates exploration horizon m of the ETC algorithm. (b) (c) (d) cumulative final regret R for $T = 100$ against maximum expected degree W in: (b) ETC, (c) EGreedy1, (d) EGreedy2.

remove the requirement of the central decision-maker being able to observe node activations globally. We prove that the explore-then-commit algorithm has convergence guarantees under the assumption that the network has symmetric influence probabilities, Lastly, we show that the empirical findings are consistent with these guarantees.

The theoretical guarantees in this paper require the symmetry given in Assumption 1. This assumption holds for all undirected networks but does not hold for all directed networks. Future work will be devoted to relaxing this assumption.

ACKNOWLEDGEMENT

This work was supported in part by ARL W911NF-17-2-0181, and National Science Foundation grants 1646522 and 1652113. Any opinions, findings, and conclusions expressed in this material do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] S. Banerjee, M. Jenamani, and D. K. Pratihar, "A survey on influence maximization in a social network," *Knowledge and Information Systems*, vol. 62, no. 9, pp. 3417–3455, 9 2020. [Online]. Available: https://doi.org/10.1007/s10115-020-01461-4

[2] M. Richardson and P. Domingos, "Mining the network value of customers," in *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 57–66.

[3] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, 2007, p. 420–429. [Online]. Available: https://doi.org/10.1145/1281192.1281239

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

[5] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart, "Online influence maximization," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2015, p. 645–654. [Online]. Available: https://doi.org/10.1145/2783258.2783271

[6] M. Kimura and K. Saito, "Tractable models for information diffusion in social networks," in *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2006, p. 259–271.

[7] A. Goyal, W. Lu, and L. V. Lakshmanan, "Celf++: Optimizing the greedy algorithm for influence maximization in social networks," in *Proceedings of the International Conference Companion on World Wide Web*, 2011, p. 47–48. [Online]. Available: https://doi.org/10.1145/1963192.1963217

[8] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proceedings of the ACM International Conference on Management of Data*, 2014, p. 75–86. [Online]. Available: https://doi.org/10.1145/2588555.2593670

[9] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proceedings of the ACM International Conference on Management of Data*, 2015, p. 1539–1554. [Online]. Available: https://doi.org/10.1145/2723372.2723734

[10] K. Saito, R. Nakano, and M. Kimura, "Prediction of information diffusion probabilities for independent cascade model," in *Knowledge-*

Based Intelligent Information and Engineering Systems, I. Lovrek, R. J. Howlett, and L. C. Jain, Eds., 2008, pp. 67–75.

[11] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2010, p. 241–250. [Online]. Available: https://doi.org/10.1145/1718487.1718518

[12] S. Vaswani and L. V. S. Lakshmanan, "Influence maximization with bandits," *CoRR*, vol. abs/1503.00024, 2015. [Online]. Available: http://arxiv.org/abs/1503.00024

[13] Q. Wu, Z. Li, H. Wang, W. Chen, and H. Wang, "Factorization bandits for online influence maximization," in *Proceedings of the ACM International Conference on Knowledge Discovery Data Mining*, 2019, p. 636–646. [Online]. Available: https://doi.org/10.1145/3292500.3330874

[14] G. Lugosi, G. Neu, and J. Olkhovskaya, "Online influence maximization with local observations," in *Proceedings of the International Conference on Algorithmic Learning Theory*, A. Garivier and S. Kale, Eds., vol. 98. PMLR, 22–24 Mar 2019, pp. 557–580. [Online]. Available: https://proceedings.mlr.press/v98/lugosi19a.html

[15] B. Bollobás, S. Janson, and O. Riordan, "The phase transition in inhomogeneous random graphs," *Random Struct. Algorithms*, vol. 31, no. 1, p. 3–122, Aug. 2007.

[16] G. T. Cantwell and M. E. J. Newman, "Message passing on networks with loops," *Proceedings of the National Academy of Sciences*, vol. 116, no. 47, pp. 23 398–23 403, 2019. [Online]. Available: https://www.pnas.org/content/116/47/23398

[17] F. Chung and L. Lu, "Connected components in random graphs with given expected degree sequences," *Annals of combinatorics*, vol. 6, no. 2, pp. 125–145, 2002. [Online]. Available: https://doi.org/10.1007/PL00012580

Appendix A
Proof of Theorem 2

Let $\mathcal{G} = ([n], E, \mathcal{P})$ be a network, and let $\mathbb{E}[\max_i \mathcal{I}_t(i)/n] = c(n)$. Also assume, without loss of generality, that node 1 is optimal, That is $1 = \arg\max_{i\in[n]} \mathbb{E}[\mathcal{I}_t(i)]$ for all $t \in [T]$, Then if we define $R_{(a,b]} = \sum_{t=a+1}^{b} \mathcal{I}_t(s^*) - \mathcal{I}_t(s_t)$ we have $R = R_{(0,m]} + R_{(m,T]}$. To prove the theorem, we provide upper bounds on each of these terms separately. For $R_{(0,m]}$ we have the chain of inequalities,

$$\mathbb{E}[R_{(0,m]}] = \sum_{t=1}^{m} \Delta_{s_t} \tag{21a}$$

$$= \sum_{t=1}^{m} \mathbb{E}[\mathcal{I}_t(1)] - \mathbb{E}[\mathcal{I}_t(s_t)] \tag{21b}$$

$$\le \sum_{t=1}^{m} \mathbb{E}[\max_i \mathcal{I}_t(i)] - \mathbb{E}[\mathcal{I}_t(s_t)] \tag{21c}$$

$$= \sum_{t=1}^{m} nc(n) - \mathbb{E}[\mathcal{I}_t(s_t)] \tag{21d}$$

$$\le \sum_{t=1}^{m} nc(n) - \mathbb{P}(s_t = \arg\max_i \mathcal{I}_t(i))\mathbb{E}[\max_i \mathcal{I}_t(i)] \tag{21e}$$

$$= \sum_{t=1}^{m} nc(n) - \mathbb{P}(s_t = \arg\max_i \mathcal{I}_t(i))nc(n) \tag{21f}$$

$$= \sum_{t=1}^{m} nc(n) - \mathbb{P}(s_t \in \sigma(j) | j = \arg\max_i \mathcal{I}_t(i))nc(n) \tag{21g}$$

$$= \sum_{t=1}^{m} nc(n) - \mathbb{E}\left[\frac{|\sigma(j)|}{n} \, | j = \arg\max_i \mathcal{I}_t(i)\right] nc(n) \tag{21h}$$

$$= \sum_{t=1}^{m} nc(n) - \mathbb{E}\left[\frac{\mathcal{I}_t(j)}{n} \, | j = \arg\max_i \mathcal{I}_t(i)\right] nc(n) \tag{21i}$$

$$= \sum_{t=1}^{m} nc(n) - nc(n)^2 \tag{21j}$$

$$= mnc(n)(1 - c(n)), \tag{21k}$$

where (21h) is because Algorithm 1 chooses s_t uniformly at random from $|V|$ for all $t \le m$. We can also give an upper bound on $R_{(m,T]}$ as,

$$\mathbb{E}[R_{(m.T]}] = \sum_{t=m+1}^{T} \Delta_{s_t} \tag{22a}$$

$$= (T - m)\Delta_{s_m} \tag{22b}$$

$$= (T - m) \sum_{i=1}^{n} \mathbb{P}(i = s_m)\Delta_i \tag{22c}$$

$$= (T - m) \sum_{i=1}^{n} \mathbb{P}(X_{m,i} = \max_{j\in[n]} X_{m,j})\Delta_i \tag{22d}$$

$$\le (T - m) \sum_{i=1}^{n} \mathbb{P}(X_{m,i} \ge X_{m,1})\Delta_i \tag{22e}$$

$$\le (T - m) \sum_{i=1}^{n} \exp\left(-\frac{m\Delta_i^2}{2n^2}\right)\Delta_i, \tag{22f}$$

where (22c) follows from the fact that Algorithm 1 chooses $s_t = s_m$ for all $t \ge m$, and (22f) follows from Lemma 1. This expression can be simplified further using the fact that the function $f(x) = x\exp(-mx^2/(2n^2))$ is pseudoconcave on interval $[0, +\infty)$. It is also easy to show that $f'(n/\sqrt{m}) = 0$, hence $x^* = n/\sqrt{m}$ maximizes f on interval $[0, +\infty)$. That is, $f(x) < f(x^*) = e^{-1/2}n/\sqrt{m}$ for all $x \in [0, +\infty)$. Using this fact, we can continue the chain of inequelities given in (22) as,

$$\mathbb{E}[R_{(m.T]}] \le (T - m) \sum_{i=1}^{n} \exp\left(-\frac{m\Delta_i^2}{2n^2}\right)\Delta_i \tag{23a}$$

$$\le (T - m) \sum_{i=1}^{n} e^{-1/2} \frac{n}{\sqrt{m}} \tag{23b}$$

$$= \left(\frac{T}{\sqrt{m}} - \sqrt{m}\right) n^2 e^{-1/2}. \tag{23c}$$

By (21) and (23), we have

$$\mathbb{E}[R] \le mnc(n)(1 - c(n)) + \left(\frac{T}{\sqrt{m}} - \sqrt{m}\right) n^2 e^{-1/2}. \tag{24}$$

We complete the proof by choosing $m = \alpha T^{2/3}$ and simplifying (8) under this choice. This yields

$$\mathbb{E}[R] \le nc(n)(1 - c(n))\alpha T^{2/3} + T^{2/3} n^2 e^{-1/2}. \tag{25}$$