7-2021

CLAIM: Curriculum learning policy for influence maximization in unknown social networks

Dexun LI
Singapore Management University, dexunli.2019@phdcs.smu.edu.sg

MEGHNA LOWALEKAR
Singapore Management University, meghnal.2015@phdis.smu.edu.sg

Pradeep VARAKANTHAM
Singapore Management University, pradeepv@smu.edu.sg

CLAIM: Curriculum Learning Policy for Influence Maximization in Unknown Social Networks

Dexun Li[1] **Meghna Lowalekar**[1] **Pradeep Varakantham**[1]

[1]School of Computing and Information Systems, Singapore Management University, Singapore

Abstract

Influence maximization is the problem of finding a small subset of nodes in a network that can maximize the diffusion of information. Recently, it has also found application in HIV prevention, substance abuse prevention, micro-finance adoption, etc., where the goal is to identify the set of peer leaders in a real-world physical social network who can disseminate information to a large group of people. Unlike online social networks, real-world networks are not completely known, and collecting information about the network is costly as it involves surveying multiple people. In this paper, we focus on this problem of network discovery for influence maximization. The existing work in this direction proposes a reinforcement learning framework. As the environment interactions in real-world settings are costly, so it is important for the reinforcement learning algorithms to have minimum possible environment interactions, i.e, to be sample efficient. In this work, we propose CLAIM - **C**urriculum **L**e**A**rning **P**olicy for **I**nfluence **M**aximization to improve the sample efficiency of RL methods. We conduct experiments on real-world datasets and show that our approach can outperform the current best approach.

1 INTRODUCTION

Social interactions between people play an important role in spreading information and behavioral changes. The problem of identifying a small set of influential nodes in a social network that can help in spreading information to a large group is termed as influence maximization (IM) [Kempe et al., 2003]. It was widely used in applications such as viral marketing [Kempe et al., 2003], rumor control [Budak et al., 2011], etc, which use the online social network. In addition to these, IM has also found useful applications in domains involving real world physical social networks. Some of these applications include identifying peer leaders in homeless youth network to spread awareness about HIV [Wilder et al., 2018b, Yadav et al., 2016], identifying student leaders in school network to disseminate information on substance abuse [Valente and Pumpuang, 2007], identifying users who can increase participation in micro-finance [Banerjee et al., 2013], etc. In the case of real world social networks, the network information is not readily available and it is generally gathered by individually surveying different people who are part of the network. As conducting such surveys is a time intensive process requiring substantial efforts from a dedicated team of social work researchers, it is not practically possible to have access to a complete network structure. Therefore, the influence maximization problem in the real world is coupled with the uncertain problem of discovering network using a limited survey budget (i.e., the number of people who can be queried).

Most of the existing work [Wilder et al., 2018b,a, Yadav et al., 2016] which addresses real-world influence maximization problems perform network discovery by surveying nodes while exploiting a specific network property such as community structure. CHANGE algorithm [Wilder et al., 2018b] is based on the principle of *friendship paradox* and performs network discovery by surveying a random node and one of its neighbour. Each node reveals the information about its neighbors upon querying. The subgraph obtained after querying a limited set of nodes is used to pick a set of influential nodes using an influence maximization algorithm. A recent work by Kamarthi et al. [2019] provides a reinforcement learning based approach to automatically train an agent for network discovery. They developed an extension to DQN referred to as Geometric-DQN to learn policies for network discovery by extracting relevant graph properties, which achieves better performance than the existing approaches. As any other reinforcement learning approach, the work by Kamarthi et al. [2019] needs to perform multiple interactions with the environment to perform exploration.

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:1455–1465.

As in the real world settings, the environment interactions are costly, the approach can be improved by reducing the environment interactions, i.e., by increasing the sample efficiency. This approach employs a myopic heuristic (new nodes discovered) to guide exploration and we employ goal directed learning to provide a forward looking (non-myopic) heuristic.

In this work, we propose to model the network discovery problem as a goal-directed reinforcement learning problem. We take the advantage of the *Hindsight Experience Replay* [Andrychowicz et al., 2017] framework which suggests learning from failed trajectories of agent by replaying each episode with a different goal (e.g. the state visited by agent at the end of its failed trajectory) than the one agent was trying to achieve. This helps in increasing sample efficiency as agent can get multiple experiences for learning in a single environment interaction. To further improve the performance, we use the curriculum guided selection scheme proposed by Fang et al. [2019] to select the set of episodes for experience replay. While there have been some other works which focus on improving the sample-efficiency [Sukhbaatar et al., 2017, Burda et al., 2018, Colas et al., 2019], most of them are designed for domain specific applications and unlike our curriculum guided selection scheme which adaptively controls the exploration-exploitation trade-off by gradually changing the preference on goal-proximity and diversity-based curiosity, they only perform curiosity-driven learning.

Contributions: In summary, following are the main contributions of the paper along different dimensions:

- **Problem:** We convert the whole process of network discovery and influence maximization into a goal directed learning problem. Unlike standard goal directed learning problems where goal state is known, in this problem, the goal state is not given. We provide a novel heuristic to generate goals for our problem setting.
- **Algorithm:** We propose a new approach CLAIM - **C**urriculum **L**e**A**rning Policy for **I**nfluence **M**aximization in unknown social networks which by using Curriculum guided hindsight experience replay and goal directed Geometric-DQN architecture can learn sample efficient policies for discovering network structure.
- **Experiments:** We perform experiments in social networks from three different domains and show that by using our approach, the total number of influenced nodes can be improved by upto 7.51% over existing approach.

2 PROBLEM DESCRIPTION

The problem considered in this work involves discovering a subgraph of the unknown network such that the set of peer leaders chosen from the discovered subgraph maximizes the number of people influenced by peer leaders. We now describe both the components of the problem, i.e., network discovery and influence maximization in detail. The notations used in the problem description are defined in Table 1.

Notation	Description
$G^* = (V^*, E^*)$	Entire Unknown Graph
S	Set of nodes known initially
$G_t = (V_t, E_t)$	Subgraph of G^* discovered after t queries
$N_{G^*}(u)$	Neighbors of vertex u in graph G^*
$E(X, Y)$	All direct edges that connect a node in set X and a node in set Y
$O(G)$	Set of nodes from graph G selected by influence maximization algorithm O
$I_{G^*}(A)$	Expected Number of nodes influenced in graphG^* on choosing A as the set of nodes to activate

Table 1: Notations

- **Network Discovery Problem:** The network discovery problem can be described as a sequential decision making problem where at each step, the agent queries a node from the discovered subgraph. The queried node reveals its neighbors, expanding the discovered subgraph. The process goes on for a fixed number of steps, determined by the budget constraint. Formally, initially we are given a set of nodes S and the agent can observe all the neighbors of nodes in set S. Therefore, $V_0 = S \cup N_{G^*}(S)$. The agent has a budget of T queries to gather additional information. For $(t+1)^{th}$ query, the agent can choose a node u_t from G_t and observe G_{t+1}.

$$G_{t+1} = (V_t \cup N_{G^*}(u_t), E(G_t) \cup E(N_{G^*}(u_t), \{u_t\})).$$

At the end of network discovery process, i.e., after T queries, we get the final discovered subgraph G_T. This graph is provided as an input to an IM algorithm.

- **Influence Maximization (IM) :** IM is the problem of choosing a set of influential nodes in a social network who can propagate information to maximum nodes. In this paper, the information propagation over the network is modeled using Independent Cascade Model (ICM) [Kempe et al., 2003], which is the most commonly used model in the literature. In the ICM, at the start of the process, only the nodes in the set of chosen initial nodes are active. The process unfolds over a series of discrete time steps, where at every step, each newly activated node attempts to activate each of its inactive neighbors and succeeds with some probability p. The process ends when there are no newly activated nodes at the final step. After discovering the subgraph G_T using network discovery process, we can use any standard influence maximization algorithm to find out

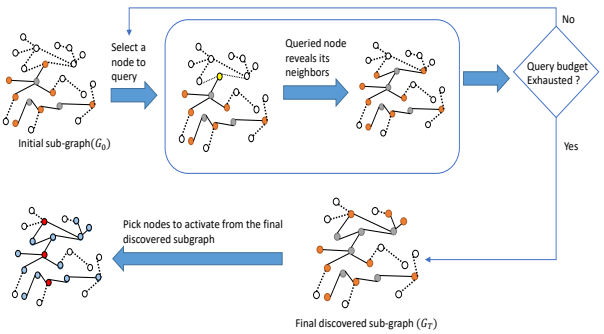

Figure 1: Network discovery and influence maximization. Grey: Set of Queried Nodes; Orange: Set of unqueried nodes (In the initial subgraph G_0, grey nodes will represent the set S and orange nodes will represent the set $N_{G^*}(S)$), Yellow: node picked by agent to query (u_t); Red: nodes selected by influence maximization algorithm in the final discovered subgraph ($O(G_T)$); Blue: other nodes in the final discovered subgraph G_T

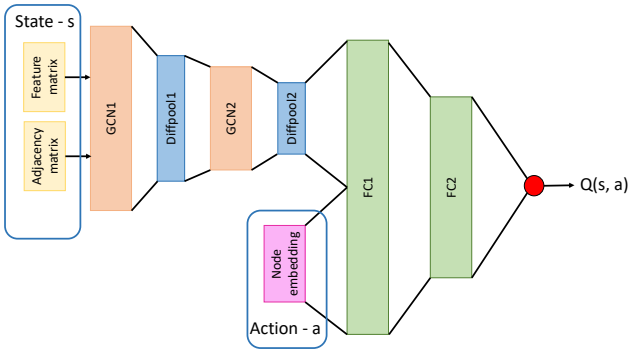

Figure 2: Geometric-DQN Architecture. Figure taken from Kamarthi et al. [2019]. FC1/FC2 - fully connected layers.

the best set of nodes to activate based on the available information. Lowalekar et al. [2016] showed the robustness of the well-known greedy approach [Kempe et al., 2003] on medium scale social network instances, which is also served as the oracle in our paper.

Overall, given a set of initial nodes S and its observed connections $N_{G^*}(S)$, our task is to find sequence of queries $(u_0, u_1, \ldots, u_{T-1})$ such that G_T maximizes $I_{G^*}(O(G_T))$. Figure 1 shows the visual representation of the problem.

3 BACKGROUND

In this section, we describe the relevant research, the MDP formulation and the Geometric-DQN architecture used by Kamarthi et al. [2019] to solve the network discovery and influence maximization problem.

3.1 MDP FORMULATION

The social network discovery and influence maximization problem can be formally modelled as an MDP.

- **State:** The current discovered graph G_t is the state.
- **Actions:** The nodes yet to be queried in network G_t constitute the action space. So, set of possible actions is $V_t \setminus \{S \cup_{i \le t} u_i\}\ \forall t > 0$ and $N_{G^*}(S)$ when $t = 0$.
- **Rewards:** Reward is only obtained at the end of episode, i.e,, after T steps. It is the number of nodes influenced in the entire graph G^* using G_T, i.e., $I_{G^*}(O(G_T))$. The episode reward is denoted by R_T, where T is the length of the episode (budget on the number of queries available to discover the network).

Training: To train the agent in the MDP environment, DQN algorithm is used but the original DQN architecture which takes only the state representation as an input and outputs the action values can not be used as the action set is not constant and depends on the current graph. Therefore, both state and action are provided as an input to DQN and it predicts the state action value. The DQN model can be trained using a single or multiple graphs. If we train simultaneously on multiple graphs, then the MDP problem turn out to be Partially observable MDP, as the next state is determined by both current state and current action as well as the graph we are using. The range of reward values also depends on the size and structure of the graph, therefore, the reward value is normalized when multiple graphs are used for training.

$$R_T = \frac{I_{G^*}(O(G_T))}{OPT(G^*)} \tag{1}$$

3.2 GEOMETRIC-DQN

As described in previous section, the state is the current discovered graph G_t and actions are the unqueried nodes in the current discovered graph. So, a good vector representation of the current discovered graph is required. It is also important to represent nodes such that it encodes the structural information of the node in the context of the current discovered graph. Figure 2 shows the Geometric-DQN architecture which takes the state and action representation as input and outputs the $Q(s, a)$ values. The details about state and action representation are provided below.

- **State representation:** The state is the current graph. and the Geometric-DQN architecture uses Graph Convolutional Networks to generate graph embeddings. The graph G_t is represented with the adjacency matrix $A_t \in \mathbb{R}^{|V_t| \times |V_t|}$ and a node feature matrix $F_t^{(k-1)} \in \mathbb{R}^{|V_t| \times d}$ in layer $k-1$ where d is the number of features. The node features in the input layer of graph convolution network, i.e., F_t^0 are generated by using random-walk based

Deepwalk embeddings[1] [Perozzi et al., 2014].

Now, a Graph Convolutional layer derives node features using a transformation function $F^k = M(A, F^{k-1}; W^k)$, where W^k represent the weights of the k^{th} layer. Using the formulation in Ying et al. [2019], the transformation function is given by

$$F_t^{(k)} = ReLU(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}F_t^{(k-1)}W^{(k)})$$

where $\tilde{A}$ means adjacency matrix A_t with added self-connections, i.e., $\tilde{A} = A_t + I_n$ (I_n is the identity matrix). $D = \sum_j \tilde{A}_{ij}$. To better represent the global representation of graph, differential pooling is used which learns hierarchical representations of the graph in an end-to-end differentiable manner by iteratively coarsening the graph, using graph convolutional layer as a building block. The output of graph convolutional network is provided as an input to a pooling layer.

- **Actions representation:** DeepWalk node embeddings are also utilized for representing actions. We use ϕ to denote the deepwalk embeddings.

Therefore, if G_t is the current graph (state) and u_t is the current node to be queried (action), we represent state as $S_t = (F_t^0, A_t)$ and action as $\phi(u_t)$ which are input to the network as shown in the Figure 2.

4 OUR APPROACH - CLAIM

In this section, we present our approach CLAIM - **C**urriculum **L**e**A**rning Policy for **I**nfluence **M**aximization in unknown social networks. We first explain how the problem can be translated into a Goal directed learning problem. The advantage of translating the problem into goal directed learning problem is that it allows us to increase sample efficiency by using the Curriculum guided Hindsight experience replay (CHER) [Fang et al., 2019]. CHER involves replaying each episode with pseudo goals, so the agent can get multiple experiences in a single environment interaction which results in increasing the sample efficiency.

To use goal directed learning in our setting, we first present our novel heuristic to generate goals and the modifications to the MDP formulation for goal directed learning. After that, we present our algorithm to generate curriculum learning policy using Hindsight experience replay.

4.1 GOAL DIRECTED REINFORCEMENT LEARNING

In the Goal Directed or Goal Conditioned Reinforcement Learning [Andrychowicz et al., 2017, Nair et al., 2018], an agent interacts within an environment to learn an optimal policy for reaching a certain goal state or a goal defined by a function on the state space in an initially unknown or only partially known state space. If the agent reaches the goal, then the reinforcement learning method is terminated, and it solves the goal-directed exploration problem.

In these settings, the reward that agent gets from the environment is also dependent on the goal that agent is trying to achieve. A goal-conditioned Q-function $Q(s, a, g)$ [Schaul et al., 2015] learns the expected return for the goal g starting from state s and taking action a. Given a state s, action a, next state s' , goal g and corresponding reward r, one can train an approximate Q-function parameterized by θ by minimizing the following Bellman error:

$$\frac{1}{2}||Q_\theta(s, a, g) - (r + \gamma \cdot \max_{a'} Q_{\theta'}(s', a', g)||^2$$

This loss can be optimized using any standard off-policy reinforcement learning method [Nair et al., 2018].

Generally, in these goal-directed reinforcement learning problems, a set of goal states or goals defined by a function on the state space is given and the agent needs to reach one of the goal states (goals). But in our setting, we do not have an explicit goal state given. To convert the network discovery and influence maximization problem to a goal directed learning problem, we introduce the notion of goals for our problem. We define goal as the expected long term reward, i.e., the expected value of the number of nodes which can be influenced in the network and any state which can achieve this goal value becomes our goal state. As we have limited query budget to discover the network, the goal value will be highly dependent on the initial sub-graph. If we use the same value of goal for each start state, for some start states this common goal value will turn out to be a very loose upper bound (or very loose lower bound). Experimentally, we found that if the goal value is too far from the actual value which can be achieved, it negatively affects the speed of learning. So, we design a heuristic to compute a different goal for each start state.

4.1.1 Goal Generation Heuristic

As we need to generate goal at the start of each episode (i.e., before the agent starts interacting with the environment), we need to compute the goal value without making any queries to the environment. We assume that based on the domain knowledge, agent can get an estimate about the number of nodes ($|\tilde{V}^*|$) and edges ($|\tilde{E}^*|$ in the network and also an estimate about average number of nodes which can be influenced in the network (irrespective of the start state) ($\tilde{I}^*$). We now describe how we use these estimates to design our heuristic to compute the goal value for each start state.

Figure 3 represents the steps for our heuristic. As the network is unknown to the algorithm, we assume a network

[1]Deepwalk learns node representations that are similar to other nodes that lie within a fixed proximity on multiple random walks.

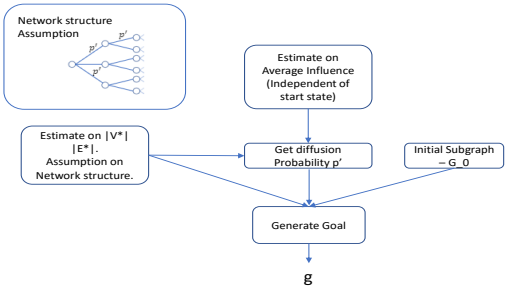

Figure 3: Process to generate the goal for each start state

structure and compute the diffusion probability based on the assumed network structure and estimates about the number of nodes, edges and average influence. By using the computed diffusion probability, given estimates and assumed network structure, we generate a goal value for a given initial subgraph.

We assume that the network is undirected and uniformly distributed, i.e, each node is connected to $\frac{2*|\tilde{E}^*|}{|\tilde{V}^*|}$ nodes. We also assume a local tree structure as shown in Figure 3 [2] to approximate the actual expected influence within the social network [Chen et al., 2010, Wang et al., 2012]. The root of the tree can be any of the $|S|$ nodes (initially given nodes) and each node will be part of only one of such trees. The influence propagation probability is assumed to be p' and is considered same for all edges. We now show how the value of p' can be computed based on the network structure assumption and available information.

1. **Computing p':** We find a value of p' such that the expected influence in our tree-structured network is similar to the estimate on the average value of influence $\tilde{I}^*$. To compute the expected influence or expected number of nodes activated in the network, we need to know the number of layers in the tree structure. Therefore, we first compute the number of layers in our assumed tree structure. Let $K_1 = |S| * 2 * \frac{|\tilde{E}^*|}{|\tilde{V}^*|}$, which is the number of nodes at first layer. For subsequent layers, each node will be connected to $2 * \frac{|\tilde{E}^*|}{|\tilde{V}^*|} - 1$ nodes at the layer below it (one edge will be to the node at the above layer). We use r to denote the quantity $2 * \frac{|\tilde{E}^*|}{|\tilde{V}^*|} - 1$. As the total number of nodes in the graph is $|\tilde{V}^*|$, sum of the number of nodes at all layers should be equal to $|\tilde{V}|^*$. Let L denotes the number of layers.

Then,

$$|\tilde{V}|^* = |S| + K_1 + K_1 * r + K_1 * r^2 + .. + K_1 * r^{L-1} \quad (2)$$

$$\implies \frac{(|\tilde{V}|^* - |S|)}{K_1} = \frac{r^L - 1}{r - 1} \quad (3)$$

Solving for L gives $L = \log_r(1 + \frac{|\tilde{V}|^* - |S|) * (r-1)}{K_1})$. Now, we compute the expected number of nodes activated (influenced) in our assumed network with the propagation probability p'. Let J denotes the expected number of nodes influenced in the network. Then,

$$J = |S| + K_1 * p' + K_1 * r * p'^2 + K_1 * r^2 * p'^3 + .. + K_1 * r^{L-1} * p'^L \quad (4)$$

$$\implies \frac{(J - |S|)}{K_1} = \frac{p' * ((p' * r)^L - 1)}{p' * r - 1} \quad (5)$$

If our assumed network is similar to actual network, the value of J should be close to $\tilde{I}^*$, i.e., the average number of nodes influenced in the network. Therefore, to find the value of p', we perform a search in the probability space and use the value of p' which makes J closest to $|\tilde{I}|^*$.

2. **Computing goal value g for a given initial subgraph:** Now, to compute the goal value for a given initial subgraph, we use the p' value computed above. The subgraph is known, i.e., the neighbors of nodes in set S ($N_{G^*}(S)$) are known. Therefore, the number of nodes at first layer is equal to $N_{G^*}(S)$, i.e., $K_1 = |N_{G^*}(S)|$. For the next layer onwards, we assume a similar tree structure as before with each node connected to $2 * \frac{|\tilde{E}|}{|\tilde{V}|} - 1$ node at the layer below it. Therefore, to compute the goal value, we substitute K_1 as $|N_{G^*}(S)|$ in equations 3 and 5 to compute the number of layers and influence value. We use the value of p' computed above and solve for J. The J value obtained is the influence value we can achieve for the given subgraph based on the assumptions and available information. We use the value of J as our goal g for the subgraph.

4.1.2 Modifications to the MDP formulation:

The state and action remain the same as before but due to the introduction of goals, the reward function is now parameterized by the goal. Let $R_{t,g}$ denote the reward obtained at timestep t when the goal is g. As we only get episode reward, therefore [3],

$$R_{T-1,g} = \frac{I_{G^*}(O(G_T)) - g}{g} \; and \; R_{t,g} = 0, \forall t \neq T-1 \quad (6)$$

[2] These simplified assumptions work well to approximate the influence propagation. We also observe in our experiments that our heuristic outputs a value which is closer to actual value.

[3] Normalizing the reward using the goal stabilizes the learning.

4.2 ALGORITHM

In this section, we describe the algorithm used to train the reinforcement learning agent. We use the DQN algorithm and use Curriculum Guided Hindsight Experience Replay for improving the sample efficiency. Algorithm 1 describes the detailed steps. Figure 4 provides a visual representation.

Algorithm 1: Train Network

Input: Train graphs $\mathcal{G} = \{G_1, G_2, ..., G_k\}$, number of episodes N. Query budget T

1 Initialize DQN Q_θ and target DQN $Q_{\theta'}$ with $\theta = \theta'$ and the Replay Buffer $\mathcal{B}$;
2 **for** *episode* $= 1$ *to* N **do**
3 $G = sample(\mathcal{G})$, $S = sample(G)$;
4 Initialize the subgraph $G_0 = (S \cup N_{G^*}(S), E(S, N_{G^*}(S)))$ and corresponding desired goal g;
5 $F_0^0 = DeepWalk(G_0)$ and $S_0 = (F_0^0, A_0)$;
6 Get the possible action set $X = N_G(S)$;
7 **for** $t = 0$ *to* $T - 1$ **do**
8 With probability ϵ select a random node u_t from X and with probability 1-ϵ select $u_t \leftarrow \max_{u \in X} Q_\theta(S_t, \phi(u), g)$;
9 Query node u_t and observe new graph G_{t+1};
10 Update the state $F_{t+1}^0 = DeepWalk(G_{t+1})$ and $S_{t+1} = (F_{t+1}^0, A_{t+1})$;
11 Update the possible action set X which is the set of nodes not yet queried in G_{t+1};
12 **for** $t = 0$ *to* $T - 1$ **do**
13 Store the transition $(S_t, \phi(u_t), R_t, g, S_{t+1}, g)$ in $\mathcal{B}$;
14 Sample the additional goals $\mathbb{G}$ for replay ;
15 **for** $g' \in \mathbb{G}$ **do**
16 Add the transition $(S_t, \phi(u_t), R_{t,g'}, S_{t+1}, g')$ to replay buffer $\mathcal{B}$;
17 **for** $t = 0$ *to* $T - 1$ **do**
18 Sample a minibatch A from the replay buffer $\mathcal{B}$(according to the proximity and diversity scores) ;
19 Update the proximity-diversity trade-off parameter $\lambda \leftarrow \gamma \times \lambda$;
20 Update Q_θ using the minibatch A;
21 Update target network $Q_{\theta'}$ with parameters of Q_θ at regular intervals;

We train using multiple training graphs. In each episode, we sample a training graph and then sample initial set of nodes S. We generate the input state by computing the deepwalk embeddings at each timestep and use $\epsilon-$greedy policy to select the action, i.e., the node to be queried. In step 13, we store the transitions according to standard experience replay where we add the goal as well in the experience buffer.

Steps 14-16 are the ***first set of crucial steps*** to improve the sample efficiency, where as per the Hindsight Experience Replay technique proposed by Andrychowicz et al. [2017], we sample pseudo goals and in addition to storing the sample with the actual goal for the episode, we also store each sample by modifying the desired goal (which the agent could not achieve in the failed trajectory) with a pseudo goal g'. The reward with the pseudo goal g' is recomputed as per the Equation 6.

While there are multiple possible strategies to generate the set of pseudo goals [Andrychowicz et al., 2017], the most common strategy to generate the pseudo goals is to use the goal achieved at the end of episode. Therefore, in this work, we use g' as $I_{G^*}(O(G_T))$.

Step 18 is the ***second crucial step*** towards improving the sample efficiency where for sampling experiences from the replay buffer, we use a curriculum guided selection process which relies on the goal-proximity and diversity based curiosity [Fang et al., 2019]. Instead of sampling experiences uniformly, we select a subset of experiences based on the trade-off between goal-proximity and diversity based curiosity. This plays an important role in guiding the learning process. A large proximity value enforces the training to proceed towards the desired goals, while a large diversity value ensures exploration of different states and regions in the environment. To sample a subset A of size k for replay from the experience buffer $\mathcal{B}$, the following optimization needs to be solved:

$$\max_{A \subseteq B, |A| \leq k} F(A) = \max_{A \subseteq B, |A| \leq k} (F_{prox}(A) + \lambda F_{div}(A)) \tag{7}$$

where B is the uniformly sampled subset of size mk from the buffer $\mathcal{B}$. Let $m = 3$ as Fang et al. [2019] does. $F_{prox}(A)$ measures the proximity of the achieved goals g' in A to its desired goal g. The second term $F_{div}(A)$ denotes the diversity of states and regions of the environment in A. And the weight λ is used to balance the trade-off between the proximity and the diversity. The trade-off between the two values is balanced such that it enforces a human-like learning strategy, where there is more curiosity in exploration in the earlier stages and later the weight is shifted to the goal-proximity.

In our work, we define the proximity as the similarity between goal values and diversity based on the distance between visited states. This is because even though goal values (influence achieved) can be different, the states visited can still be very similar to each other. Formally, to define proximity, we use the difference between achieved goal g_i' and the desired goal g_i as distance and subtract it from a large constant to get the similarity, i.e.,

$$F_{prox}(A) = \sum_{i \in A} (c - |(g_i' - g_i)|) \tag{8}$$

where c is a large number to guarantee $(c - |(g_i' - g_i)|) \geq$

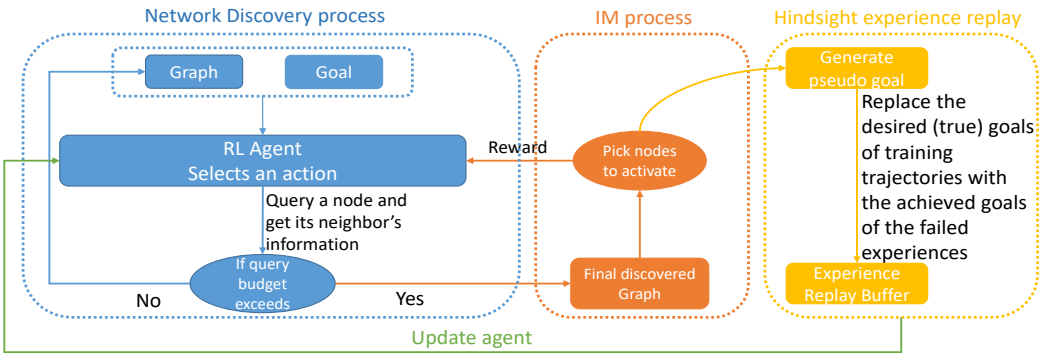

Figure 4: Network discovery framework for influence maximization

0 for all possible g_i, and g_i is the goal corresponding to experience i in set A. For defining diversity, we need to compute similarity between states, and the Geometric DQN architecture allows us to easily compute this value. Diversity is defined as follows

$$F_{div}(A) = \sum_{j \in B} \max_{i \in A} \{0, sim(s_i^{emb}, s_j^{emb})\} \quad (9)$$

where we use s_i^{emb} to denote the embedding vector of the state (representation of the graph in the embedding space) corresponding to the experience i. The embedding vector of the state is the output of the graph convolution and pooling layer (input to FC1) in Figure 2. $sim(s_i^{emb}, s_j^{emb})$ denotes the similarity score between the vector representations and is computed by taking the dot product of the vectors.

This definition of diversity is inspired by the facility location function [Cornuejols et al., 1977, Lin et al., 2009] which was also used by Fang et al. [2019]. Intuitively, this diversity term is measuring how well the selected experiences in set A can represent the experiences from B. A large diversity score $F_{div}(A)$ indicates that every achieved state in B can find a sufficiently similar state in A. A diverse subset is more informative and thus helps in improving the learning.

It has been shown that $F(A)$ defined in equation 7 is a monotone non-decreasing submodular function [4] Therefore, even though exactly solving equation 7 is NP-hard, due to the submodularity property, greedy algorithm can provide a solution with an approximation factor $1 - \frac{1}{e}$ [Nemhauser et al., 1978]. The greedy algorithm picks top k experiences from the buffered experiences B. It will start by taking A as an empty set and at each step, it will add the experience i which maximizes the marginal gain. We denote the marginal gain for experience i by $F(i|A)$ and it is given by

$$F(i|A) = F(i \cup A) - F(A) \quad (10)$$

[4]It is a weighted sum of a non-negative modular function ($F_{prox}(A)$) and a submodular function ($F_{div}(A)$). Please refer to the paper by Fang et al. [2019] for details.

Therefore, by using equations 7-9, we get

$$F(i|A) = (c - (|g_i' - g_i|)) + \lambda \sum_{j \in B} \max\{0, (sim(s_i^{emb}, s_j^{emb}) - \max_{l \in A}(sim(s_l^{emb}, s_j^{emb})))\}$$

At the end of each episode, the trade-off coefficient λ is multiplied by a discount rate γ, which produces the continuous shifting of weights from diversity to proximity score. Then effect of $F_{div}(A)$ will go to zero when $\lambda \to 0$.

5 EXPERIMENTS

The goal of the experiment section is to evaluate the performance of our approach CLAIM in comparison to following state-of-the-art approaches:

- **Random** - At each step, it randomly queries a node from available unqueried nodes.
- **CHANGE** Algorithm by Wilder et al. [2018b]
- **Geometric-DQN (Baseline)** Algorithm by Kamarthi et al. [2019]

Table 2: Train and test networks

Category	Train networks	Test networks
Retweet	copen, occupy	israel, damascus, obama, assad
Animal	plj, rob	bhp, kcs
FSW	zone 1	zone 2, zone 3

Dataset: The first network is the Retweet Network from twitter [Rossi and Ahmed, 2015]. The second network is Animal Interaction networks which are a set of contact networks of field voles (Microtus agrestis) inferred from mark–recapture data collected over 7 years and from four sites [Davis et al., 2015]. The third network is a real-world physical network between Female Sex Workers (FSW) in a large Asian city divided into multiple zones. This is a confidential dataset physically collected by a non-profit by

Table 3: Comparison of influence score of our proposed approach and existing approaches for each test network. For each network, a paired t-test is performed and $*$ indicates statistical significance of better performance at $\alpha = 0.05$ level, $**$ at $\alpha = 0.01$ level, and $***$ at $\alpha = 0.001$ level.

Network category	Retweet networks				Animals networks		FSW networks	
Test networks	israel	damascus$**$	obama$**$	assad$*$	bhp$***$	kcs$**$	zone 2$*$	zone 3
OPT value	113.9	195.8	154.7	134.2	111.9	113.4	20.98	16.40
Random value	31.17	84.71	40.81	69.44	36.80	54.39	13.26	12.31
CHANGE value	32.42	92.41	48.61	69.77	35.87	54.52	12.60	10.51
Geometric DQN	37.33	105.2	52.01	75.12	40.12	60.81	13.65	12.35
CLAIM approach	**38.55**	**113.1**	**54.67**	**77.49**	**42.25**	**64.58**	**13.94**	**12.48**
Improve percent	3.27%	7.51%	5.11%	3.15%	5.31%	6.20%	2.12%	1.05%

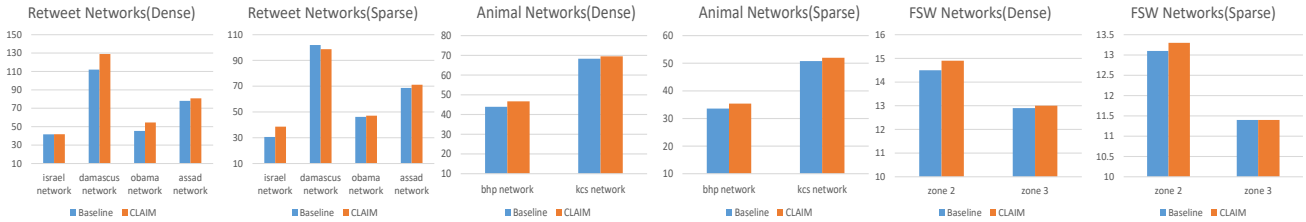

Figure 5: Comparison of performance of our approach and baseline approach in dense and sparse network environment.

surveying different female sex workers recently. The goal in FSW networks is to discover the network and select a subset of FSW from the discovered network to be enrolled in the HIV prevention programs. The enrolled FSWs should be such that they can pass on the information (influence) maximum FSWs in the complete network. For each family of network, we divide them into train and test data as shown in Table 2.

Experimental Settings: Our experimental settings are similar to the settings used in Kamarthi et al. [2019]. There are 5 nodes in the set S. All nodes in S and their neighbors are known. We have further budget of $T = 5$ queries to discover the network. After getting the final subgraph G_T, we pick 10 nodes to activate using greedy influence maximization algorithm. We use $p = 0.1$ as the diffusion probability for all the edges.

5.1 RESULTS

To demonstrate sample efficiency, we measure the performance of our approach against past approaches by the average number of nodes influenced over 100 runs under a fixed number of queries. Here are the key observations:

- **Average influence value:** Table 3 shows the comparison of number of nodes influenced by different algorithms. Each algorithm selects the set of nodes to activate from the discovered graph. As shown in the table, our approach consistently outperforms all existing approaches across different networks. CLAIM learns a better policy in the same number of episodes and hence more sample efficient. We would like to highlight here that even a small consistent improvement in these settings is very important as it can ensure more life safety (as an example by educating people about HIV prevention).
- **Effect of density of the initial subgraph:** The number of nodes which can be influenced in the graph is highly dependent on the position of initial subgraph in the whole social network. Therefore, we also test the performance of CLAIM against the baseline approach on the *dense* and *sparse* initial subgraphs (we identify the initial subgraph as dense or sparse based on the ratio of $\frac{|S \cup N_{G^*}(S)|}{|S|}$). We compare the average influence values as shown in Figure 5. CLAIM outperforms the baseline in most of the cases, except the sparse case in the damascus network. The reason for this may be that the damascus network is an extremely sparse network, and it has some specific structure property that leads this result.
- **Ablation Study:** We also present the detailed results for our ablation study over all datasets in Table 4. We observe the effect of adding each additional component in CLAIM one by one. First, we add only goal as a feature to the baseline model. Next, we add the Hindsight Experience Replay and finally we add the curriculum guided selection of experiences for replay. These results indicate that a single component can not guarantee a better result for all networks, and we need all three components to improve the performance across multiple datasets.
- **Stability check:** We check the stability of CLAIM by comparing the performance of models trained us-

Table 4: Ablation study for each test network

Network category	Retweet Networks				Animals networks		FSW networks	
Test networks	israel	damascus	obama	assad	bhp	kcs	zone 2	zone 3
Baseline (Geometric DQN)	37.33	105.2	52.01	75.12	40.12	60.81	13.65	12.35
Goal-directed Geometric DQN	36.24	110.5	51.61	73.68	41.59	62.80	13.79	12.32
Hindsight Experience Replay	37.79	109.4	53.51	76.32	42.00	**64.64**	13.81	**12.48**
Our proposed approach (CLAIM)	**38.55**	**113.1**	**54.67**	**77.49**	**42.25**	**64.58**	**13.94**	**12.48**

Table 5: Stability of our approach compared to the baseline on different sets of 100 runs

Networks\Method	Geometric DQN	CLAIM
israel	37.11 ± 0.42	38.32 ± 0.32
damascus	104.2 ± 5.22	112.8 ± 4.01
obama	52.15 ± 1.05	54.78 ± 0.78
assad	74.53 ± 1.71	77.45 ± 1.02
bhp	40.24 ± 1.25	42.37 ± 0.81
kcs	59.67 ± 1.91	63.21 ± 1.43
zone 2	13.62 ± 0.01	13.94 ± 0.00
zone 3	12.23 ± 0.01	12.45 ± 0.00

ing different random seeds. We train three models for both baseline and CLAIM. Table 5 shows the mean and deviation of influence value for different networks. CLAIM not only achieves high mean it also provides a low deviation reflecting the stability of approach.

- **Property insight:** We also explore the properties of the selected nodes to further investigate why CLAIM performs better. We look at *degree centrality measures, closeness centrality measures,* and *betweenness centrality measures* of the nodes queried in the underlying graph. In particular, we conduct experiment using *assad*, a retweet network with sparsely interconnected star-graph. As we can see in Figure 6, compared to the baseline approach, on an average, CLAIM can recognize nodes with higher degree, closeness and betweenness centrality. As a result, CLAIM is able to discover a bigger network. The higher degree centrality, higher closeness centrality and higher betweenness also show that CLAIM can explore nodes which plays an important role in influence maximization problem. Besides, these values are large at the beginning which means that CLAIM tends to explore a bigger graph first, and then leverage the available information with the learned graph to find complex higher-order patterns in the graphs that enable it to find key nodes during the intermediate timesteps, and finally utilise all the information to expand the discovered graph at the end.

6 DISCUSSION

We provide a justification for the choices made in the paper.

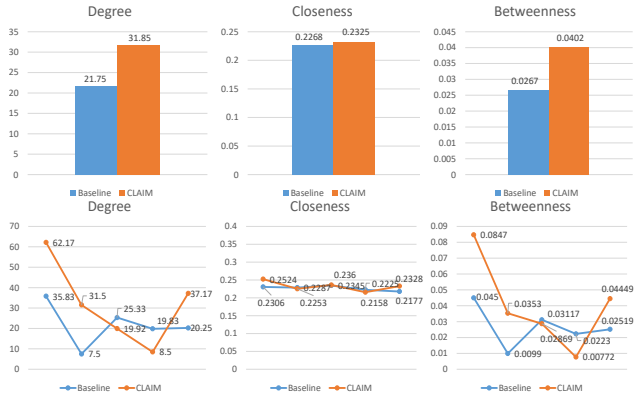

Figure 6: Top Graph - Average degree, closeness and betweenness centrality of nodes queried in the full graph by CLAIM and baseline. Bottom Graph - Variation of these properties across timesteps.

- **Network structure assumption for goal generation:** As we have no prior information about the networks except the initial nodes, we need to make some assumption to compute the goal value. We make the assumption of network being uniformly distributed and use a tree structure to approximate the information propagation as most networks observed for these problems have similar structure or can be converted in these forms with minimal loss of information.
- **Goodness of heuristic used for goal generation:** Experimentally, we observe that the goal value computed by our heuristic is closer to the actual value. For example, for the training network *copen*, the achieved influence value by the model after training is at most within 20% of goal value computed using heuristic. In addition, most of the achieved influence value is much closer and is smaller than the computed goal. In the future, we will investigate different ways to generate a goal with proven upper bound.

7 CONCLUSION

In this work, we proposed a sample efficient reinforcemernt learning approach for network discovery and influence maximization problem. Through detailed experiments, we show that our approach outperforms existing approaches on real

world datasets. In future, we would like to extend this work to consider multiple queries at each timestep.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5048–5058. Curran Associates, Inc., 2017.

Abhijit Banerjee, Arun G Chandrasekhar, Esther Duflo, and Matthew O Jackson. The diffusion of microfinance. *Science*, 341(6144), 2013.

Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web*, pages 665–674, 2011.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Wei Chen, Chi Wang, and Yajun 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*, pages 1029–1038, 2010.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

Gerard Cornuejols, Marshall Fisher, and George L Nemhauser. On the uncapacitated location problem. In *Annals of Discrete Mathematics*, volume 1, pages 163–177. Elsevier, 1977.

Stephen Davis, Babak Abbasi, Shrupa Shah, Sandra Telfer, and Mike Begon. Spatial analyses of wildlife contact networks. *Journal of the Royal Society Interface*, 12 (102):20141004, 2015.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 12623–12634. Curran Associates, Inc., 2019.

Harshavardhan Kamarthi, Priyesh Vijayan, Bryan Wilder, Balaraman Ravindran, and Milind Tambe. Influence maximization in unknown social networks: Learning policies for effective graph sampling. *arXiv preprint arXiv:1907.11625*, 2019.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 137-146, 07 2003.

Hui Lin, Jeff Bilmes, and Shasha Xie. Graph-based submodular selection for extractive summarization. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 381–386. IEEE, 2009.

Meghna Lowalekar, Pradeep Varakantham, and Akshat Kumar. Robust influence maximization. *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, 2016.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in Neural Information Processing Systems*, 31:9191–9200, 2018.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014.

Ryan A Rossi and Nesreen K Ahmed. Networkrepository: A graph data repository with visual interactive analytics. In *29th AAAI Conference on Artificial Intelligence, Austin, Texas, USA*, pages 25–30, 2015.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Thomas W Valente and Patchareeya Pumpuang. Identifying opinion leaders to promote behavior change. *Health education & behavior*, 34(6):881–896, 2007.

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

Bryan Wilder, Nicole Immorlica, Eric Rice, and Milind Tambe. Maximizing influence in an unknown social network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4743–4750. AAAI Press, 2018a.

Bryan Wilder, Laura Onasch-Vera, Juliana Hudson, Jose Luna, Nicole Wilson, Robin Petering, Darlene Woo, Milind Tambe, and Eric Rice. End-to-end influence maximization in the field. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *AAMAS*, pages 1414–1422. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018b.

Amulya Yadav, Hau Chan, Albert Xin Jiang, Haifeng Xu, Eric Rice, and Milind Tambe. Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In *AAMAS*, volume 16, pages 740–748, 2016.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling, 2019.