Influence maximization in unknown social networks: Learning Policies for Effective Graph Sampling

Harshavardhan Kamarthi[1] Priyesh Vijayan[2] Bryan Wilder[3] Balaraman Ravindran[1,4] Milind Tambe[3]
[1]Dept. of Computer Science, Indian Institute of Technology Madras
[2]School of Computer Science,McGill University and Mila
[3]Center for Research on Computation and Society, Harvard University
[4]Robert Bosch Centre for Data Science and AI, Indian Institute of Technology Madras

1 INTRODUCTION

Social network interventions are used across a wide variety of domains to disseminate information or inspire changes in behavior; application areas range from substance abuse [24], to microfinance adoption [1], to HIV prevention [27, 28]. Such processes are computationally modelled via the *influence maximization* problem, where the goal is to select a subset of nodes from the network to spread a message, such that the number of people it eventually reaches is maximized. Several algorithmic approaches have been proposed for influence maximization [4, 10, 12, 16].

However, real-world applications of influence maximization are often limited by the high cost of collecting network data. In many domains, for instance, those arising in public health, a successful intervention requires information about the face-to-face interactions with the members of a population. This information is typically gathered via in-person surveys; conducting such surveys requires substantial effort on the part of the organization deploying an intervention. We are motivated in particular by the problem of using influence maximization for HIV prevention among homeless youth, where algorithms have been successfully piloted in real-world settings [27, 28]. In the HIV prevention domain, gathering the social network of the youth who frequent a given homeless centre requires a week or more of effort by social workers, which is not feasible for a typical community agency.

The method proposed in [27] addresses this by introducing the CHANGE algorithm which uses a simple yet effective sampling method: surveying random nodes and one of their neighbours to discover parts of the social network. Each node that is surveyed reveals its neighbours, allowing us to observe a subgraph of the entire social network. The subgraph is used to pick influential set of nodes by applying a influence maximization algorithm on it. Thus the discovered subgraph acts as a surrogate to the entire network for deciding on influential actors. Existing network discovery algorithms are entirely hand-designed [25, 27], typically aiming to exploit a specific property of graphs such as community structure or the friendship paradox [8] to get the surrogate graph.

This raises the question of whether it is possible to *automatically* train an agent for network discovery; exploring from the entire space of network discovery policies may yield more effective results than hand-engineered approaches. Hence, we employ reinforcement learning (RL) to discover better heuristics on a given network dataset. Instead of hard-coding a reliance on particular structural network features (as in previous works [25, 27]), our agent learns from scratch how to represent the social network in a way that is conducive to effective sampling policies. This allows it to exploit additional structural properties that were not used in previous approaches, ultimately enabling better performance.

In particular, our approach leverages the availability of historical network data from similar populations (for instance, when deciding which youths to survey for an HIV prevention intervention, we can deploy policies learnt from training using networks gathered at other centres).As a consequence, our agent learns more nuanced policies which can both be fine-tuned better to a particular distribution of graphs and which can adapt more precisely over the course of the surveying process itself (since the trained policy is a function of a graph discovered so far). Even if training networks are not available from the specific domain of interest, we show that comparable performance can also be obtained by training on standard social network datasets or datasets generated synthetically using community structure information and transferring the learned policy unchanged to the target setting.

The main contributions of our work can be summarized as follows:

(1) We formulate the process of network discovery for influence maximization as a sequential decision problem via a Markov Decision Process (MDP). To solve this MDP, we propose a neural network architecture called Geometric-DQN and a training algorithm that uses Deep Q-learning to learn policies for network discovery by extracting relevant graph properties from the training dataset. To the best of our knowledge, this is the first work to use deep reinforcement learning for network discovery to aid influence propagation in unknown networks.
(2) One key feature of Geometric-DQN is its ability to learn both global state representations for the entire discovered graph and local action representations for the individual nodes; capturing information at both scales allows our agent to learn nuanced policies. Our RL algorithm can accommodate an arbitrary number of actions (nodes) that can vary depending on the state (graph). We also enable training across multiple graphs by designing an appropriately scaled reward scheme that is applicable across different kinds of networks. This allows Geometric-DQN to benefit from training on multiple similar graphs resulting in superior performance compared to the baselines.
(3) Motivated by the benefits of training on multiple networks, we propose to use synthetically generated graphs from a similar population to augment existing networks. This allows the model to obtain improved performances, especially when there are limited number of training networks. To this end, we design

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand

https://doi.org/doi

algorithms that leverage community structure from available training graphs to generate graphs with similar properties.

(4) We experimentally evaluate our RL based network discovery algorithm on social networks from four different domains. Our model outperforms previous state-of-the-art sampling approaches by a substantial margin of 7-23%.

2 PROBLEM DESCRIPTION

Our problem is ultimately motivated by goal of choosing a set of seed nodes in a social network who will disseminate information to as many other nodes as possible. To model information diffusion over the network, we use the standard independent cascade model (ICM) [11], which is the most commonly used model in the literature. In the ICM, every node is either active or inactive. At the start of the process, every node is inactive except for the seed nodes, S. The process unfolds over a series of discrete time steps. 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..

Our algorithm is concerned with discovering a subgraph of the social network such that the $|S|$ seed nodes chosen from the discovered subgraph maximize the number of activated nodes at the end of the process. The subgraph is discovered as follows: at each step, we query a node from the already discovered subgraph. The queried node then reveals its neighbors, expanding the discovered subgraph. This process goes on for fixed number of steps after which we use the final discovered subgraph as input to an influence maximization algorithm. This process is illustrated in Figure 1.

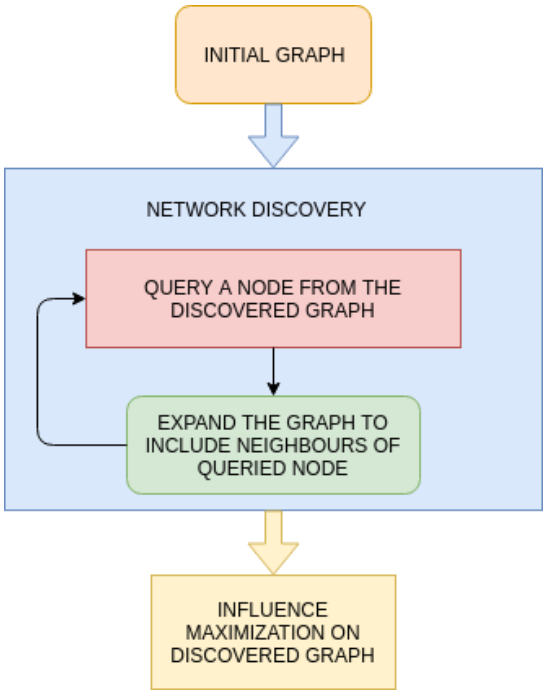

Figure 1: Network discovery for Influence maximization

We can define the problem more formally as a sequential decision problem. Let the entire unknown graph be $G^* = (V^*, E^*)$. Let $X \subseteq V^*$ denote a vertex set. Let $G[X]$ denote a sub-graph of G^* induced by X. Let $V(G)$ be the vertex set of a graph G and $E(G)$ be edge set of G. Let $N_G(u)$ be neighbors of vertex u in a graph G. $E(X, Y)$ be all direct edges that connect a node in X and a node in Y.

Initially, we are given a set of seed nodes, S, whose connections are observed. When no prior information is available, these seeds could be obtained by querying nodes at random. The agent has a budget of T queries to gather additional information. When we query a node, we discover the neighbours of the queried node. Let G_t be the sub-graph discovered after t queries with vertex set, $V_t = V(G_t)$. Let $G_0 = (S \cup N_{G^*}(S), E(S, N_{G^*}(S)))$. During the $(t+1)^{\text{th}}$ query, we choose a node u_t from G_t and observe $G_{t+1} = (V_t \cup N_{G^*}(u_t), E(G_t) \cup E(N_{G^*}(u_t), \{u_t\}))$. For any observed graph, we can use the *greedy influence maximization algorithm* [12] (which assumes that the entire graph is known) as an oracle to determine the best set of nodes to activate based on the available information. Let $\mathcal{A} = \mathcal{O}(G)$ be the output of this oracle on graph G, and let $I_G(\mathcal{A})$ be the expected number of influenced nodes in G on choosing $\mathcal{A}$ as the set of initial active nodes. The task is to find a sequence of queries $(u_0, u_1, \ldots, u_{T-1})$ such that the discovered graph G_T maximizes $I_{G^*}(\mathcal{O}(G_T))$, i.e, we need to discover a subgraph G_T such that the nodes selected by $\mathcal{O}$ in G_T maximizes the number of nodes influenced in the entire graph, G^*.

There is no existing method to directly optimize $I_{G^*}(\mathcal{O}(G_T))$ given G_0. Therefore, we employ reinforcement learning by converting the problem into a Markov Decision Process (MDP), to explore different heuristics and generate improved sampling policies over time. Since we use deep reinforcement learning methods that require the states and actions to be real valued vectors, we need to extract vector representations that summarize important properties of individual nodes or of the graph as a whole. We show how we can can automatically learn effective representations in tandem with process of discovering policies via a Q-learning based algorithm.

3 METHODOLOGY

Previous work has used entirely hand-designed heuristics for network discovery. In order to automatically produce more effective agents for the problem, we introduce a series of methodological contributions. *First*, we provide a MDP formulation of the problem, with appropriately defined rewards to enable training across multiple graphs. *Second*, we propose a step-reward function to alleviate reward sparsity. *Third*, we propose a combination of local and global embeddings as the state and action representation, which both solves unique difficulties in the way our problem's action space and provides more nuanced contextual information to our agent. Previous work on graph reinforcement learning [13] did not use a distinct global representation, which we found to be an important part of success in our domain. *Fourth*, we propose a data augmentation technique based on fitting a probabilistic model and generating additional synthetic graphs, allowing our agent to perform well when real-world network data is scarce. The following sections detail these contributions.

3.1 Markov Decision Process Formulation of Network Discovery problem

We start by formalizing the network discovery problem as a MDP.

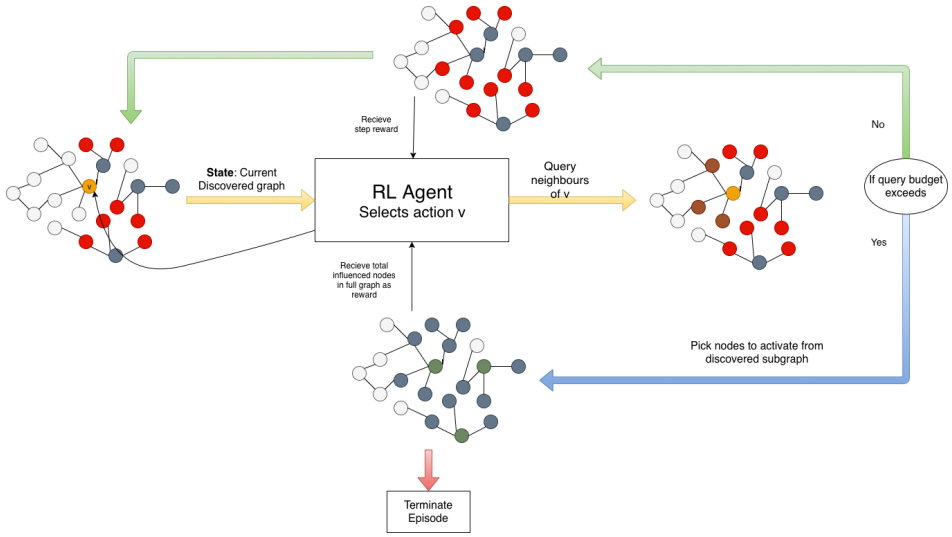

Figure 2: RL agent pipeline. (Red: Nodes in current action set, Yellow: Node selected by RL agent, Grey: Other nodes from discovered graph, Green: Nodes picked to activate by greedy influence maximization algorithm)

State: The current state is the discovered graph G_t.

Actions: Given a sub-graph G_t, we can query any of the nodes in G_t which are not yet queried. Thus, the action space is $V_t/\{S \cup_{i \leq t} u_i\} \forall t > 0$ or is $N_{G^*}(S)$ if $t = 0$.

Rewards: The reward we get after T steps is the number of nodes influenced in the entire graph, G^* using the discovered graph, G_T which is $I_{G^*}(O(G_T))$. We denote this reward, which the model receives at the end of an episode as R_i.

Rewards when using multiple graphs: When we train simultaneously on multiple graphs from the dataset, the task is no longer an MDP since the next state depends on which graph from the training dataset we are currently training on, which is unknown to the agent. Hence, the problem becomes a POMDP (Partially observable MDP) [17] where the distribution over next state on choosing an action is also conditionally dependent on graph used along with previous state.

The range of rewards values are highly dependent on size and structure of the graphs considered. Therefore, we can't directly use this reward signal to train with multiple graphs simultaneously without appropriate reward normalization. We solve this problem by scaling the reward. Specifically, we compare the influence reward $I_{G^*}(O(G_T))$ with reward from using the CHANGE algorithm [27], a state-of-art algorithm that also serves as our baseline. We also keep the range of reward between 0 and 1 to improve the stability of DQN training [18]. We normalize the influence reward as:

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

where $CHANGE(G^*)$ is the average number of influenced nodes when the graph is sampled using CHANGE [27] and OPT is the number of influenced nodes when we select the active nodes given the knowledge of entire graph, i.e, $I_{G^*}(O(G^*))$.

We use $CHANGE$ for bounding R_s in Equation 1 because of three reasons. Firstly, $CHANGE$ is a powerful method for graph discovery and dominates other state-of-art sampling methods in terms of performance. Secondly, it is computationally inexpensive to simulate sampling according to $CHANGE$ on large networks. Hence we can pre-compute performance of $CHANGE$ for multiple training graphs before training the RL model. Finally, it is hard to find a feasible alternative to Equation 1 that uses network graph properties like size and density to estimate the range of influence rewards to scale it appropriately for different domains of networks.

Step-rewards: While these rewards are well-scaled and translate across networks, they are only available at the end of each episode. Hence, we also add *step-rewards* at each step to help alleviate reward sparsity and encourage the agent to learn to find larger graphs. The step-reward $R_{p,t}$ at time t is given as:

$$R_{p,t} = \frac{|V(G_t)| - |V(G_{t-1})|}{|V(G^*)|} \tag{2}$$

It was observed that while the step-rewards are about two orders of magnitude less than the influence reward in general, they are usually essential for initial stable learning. While we found that our method tends to discover larger graphs than the baselines, there was no correlation between a model's performance and the size of the graph it discovers. We suspect that the step-rewards are important during initial phase of training, after which the final reward signal dominates.

3.2 Representation learning for Geometric-DQN

The MDP formulation presents us with challenges atypical of most reinforcement learning problems. A social network is a very structured object that can vary in size and complexity. To condition the decision making process on the graph, we need a good vector representation of the graph. The actions, which are nodes of the networks yet to be queried, need to be represented as vectors that encode structural information of the node in the context of the discovered network.

Structure of Geometric-DQN network: The network discovery problem imposes new challenges on how representations are structured, which are not encountered in other RL problems. First, reinforcement learning problems usually have a fixed set of actions. Even when the action space is continuous, the range of action is known a priori and bounded. However, in this case, the size of action set (number of queryable nodes) depends on current graph. Second, the node representation is a function of the current state, i.e, the same node could have very different neighborhood properties and hence different embeddings in context of two different graphs. Therefore, the original DQN architecture [18], which takes only the state representation as input and outputs action values, can't be used. To resolve these challenges, we input both state (high-level graph representation) and action representation (node embeddings) to the DQN and train it to predict the state-action value. This allows Geometric-DQN to augment the global information from graph embeddings with the neighbourhood information of node embeddings, adapting its actions to the information obtained so far.

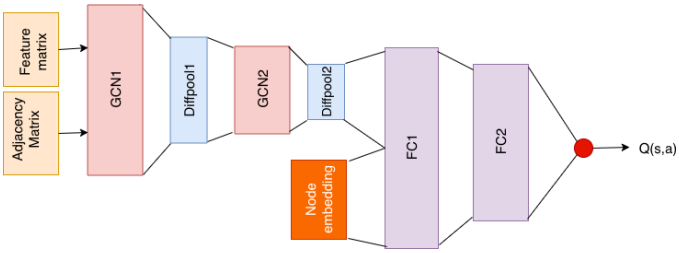

Figure 3: Geometric-DQN Architecture

Graph and Node Representation: Next, we discuss how we extract the state and action embeddings from structure of the network.

Background: Graph Convolutional Networks (GCNs): GCNs learn refined node features of a graph by passing and aggregating node features from the neighbors of each nodes. Given a graph G with adjacency matrix $A \in \mathbb{R}^{n\times n}$ and a node feature matrix, $F^{(k-1)} \in \mathbb{R}^{n\times d}$ in layer $k-1$ where n is the number of nodes and d is the number of features, a Graph Convolutional layer derives node features using a transformation function $F^{(k)} = M(A, F^{(k-1)}; W^{(k)})$ where $W^{(k)}$ are weights of kth layer. In this work, we use the formulation in [14] for GCN layers. Let $\tilde{A} = A + I$ and $D = \sum_j \tilde{A_{ij}}$.

$$F^{(k)} = ReLU(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}F^{(k-1)}W^{(k)}) \tag{3}$$

Global state representation: While GCNs refine node features by message passing and aggregating, we use differential pooling (Diffpool) [29] to learn a global representation of the graph by aggregating node features in a hierarchical manner. Diffpool learns hierarchical representations of the graph in an end-end differentiable manner by iteratively coarsening the graph, using GCNs as a building block. Diffpool can be used to map a graph to a single finite dimensional representation by iteratively coarsening the input graph to a graph with a single node and extracting features for this new one node graph.

We used DeepWalk embeddings [19] as node features $\phi = F^{(0)}$ for the input layer. DeepWalk is a random-walk based node embedding model. It learns node representations that are similar to other nodes that lie within a fixed proximity on multiple random walks.

In Geometric-DQN, the DeepWalk embeddings, as well as the GCN and DiffPool layers' output dimensions, were set to be size 100.

Local action representation: We also utilized DeepWalk node embeddings, ϕ for representing nodes as action input in Q-network. We used deepwalk embeddings rather than randomly initializing $F^{(0)}$ since it was found to be vital for stable training of Geometric-DQN. Using GCN node features for actions resulted in unstable training as the input action space was non-stationary. We used neighbourhood-based DeepWalk embeddings as it performed much better and was computationally less expensive than structure based encodings [6, 7].

Algorithm 1: Train Network

Input : Train Graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_K\}$, number of episodes N, Query budget T, number of random seeds $|S|$

1 Initialize DQN Q_θ and target DQN $Q_{\theta'}$ with $\theta = \theta'$;
2 Initialize Prioritized Replay Buffer B;
3 **for** *episode = 1 to N* **do**
4 $G = sample(\mathcal{G})$;
5 $S = sample(G)$;
6 $V_0 = S \cup N_G(S)$;
7 Initial graph is $G_0 = G[V_0]$;
8 $F_0^0 = DeepWalk(G_0)$ and $S_0 = (F_0^0, A_0)$;
9 $X \leftarrow N_G(S)$;
10 **for** *t = 0 to T − 1* **do**
11 With probability ϵ select a random node v_t from X else select node $v_t \leftarrow \underset{v\in X}{argmax} Q_\theta(S_t, \phi(v))$;
12 Query node v_t and observe new graph G_{t+1};
13 Set $R \leftarrow R_{p,t}$ [Eqn. 2].;
14 **if** $t = T - 1$ **then**
15 $R \leftarrow I_{G^*}(O(G_T))$;
16 $R_t = \frac{R-CHANGE(G^*)}{OPT(G^*)-CHANGE(G^*)}$
17 **end**
18 $F_{t+1}^0 = DeepWalk(G_{t+1})$ and $S_{t+1} = (F_{t+1}^0, A_{t+1})$;
19 Add $(S_t, \phi(v_t), R_t, S_{t+1})$ to prioritized replay buffer ;
20 Sample from B and update Q_θ;
21 X is the set of nodes not yet queried in G_{t+1};
22 **end**
23 Update target network $Q_{\theta'}$ with parameters of Q_θ at regular intervals;
24 **end**

3.3 Model training and deployment

For training, we can use single or multiple graphs. Algorithm 1 summarizes the training steps. At the start of every episode, we sample a graph at random from available training graphs (Line 4). Every time we expand the discovered graph, for timestep t, we compute node embeddings for all vertices in G_t (Lines 8, 18).

The Deepwalk representation for node v, denoted as $\phi(v)$, is a d dimensional vector. Then we create the feature matrix $F_t^0 \in \mathbb{R}^{|V_t| \times d}$ whose rows are the embeddings of nodes in V_t. F_t^0 is fed along with adjacency A_t as input for Geometric-DQN. The state is represented by $S_t = (F_t^0, A_t)$. For each of the nodes v which are not queried yet until t, we get the state-value $Q(S_t, \phi(v_t))$ and choose the node v_t that has maximum estimated state-value to query next (Line 12). We receive the reward as discussed and observe the new graph G_{t+1} (Line 13). We also store the experience $(S_t, \phi(v_t), R_t, S_{t+1})$ in replay buffer. Since we use prioritized replay [22], we also compute the TD error which is used to determine the importance of the experience in training. Since the rewards are scaled (Equation 1) we can combine experiences from different graphs, enabling the agent to generalize better. After we train the DQN, we can deploy it on a new network after freezing the weights of Q-network. The deployment algorithm is similar to the training algorithm in that we compute Deepwalk embeddings for nodes at each step and find the next node to query based on the estimated state-values of all nodes which are not queried yet.

3.4 Synthetic graph generation

In many real domains, only a small number of actual networks are available. To alleviate this difficulty, we propose a data augmentation strategy which uses known structural properties of the family of networks to synthetically generate graphs for training. Even if a large number of training graphs are available, adding synthetic graphs provides the model with a richer training set and encourages robustness to small perturbations in the graph structure.

Our strategy is based on Stochastic Block Models (SBMs), which originated in sociology [9] and can generate graphs that emulate structural properties seen in real world social networks. SBMs generate random graphs containing densely connected communities. The individual communities generated are Erdos-Renyi graphs with edge probability p_{in}. The communities are interconnected sparsely by assigning an edge between each pair of nodes of different communities with a low probability p_{out}.

In this work, for each training graph, we construct a stochastic block model based on size of densely connected components of the graph called communities, determined using the louvain community detection algorithm [3]. The Louvain community detection algorithm maps each node of input graph to one of the communities. For each community, we find the maximum liklihood estimate p_{in} since we know the actual number of edges within the community. Similarly, using edge information of actual graph, we find the maximum liklihood estimate p_{out} for the allocated community structure. Thus, we have a SBM model with a distinct p_{in} for each community.

For retweet networks, we make a small change to the SBM model, which we refer to as the Stochastic Star Model (SSM): We assume each community is a star-graph instead of a Erdos-Renyi graph (and estimate only p_{out}). This is justified by the observation that retweet networks usually [15, 21] looks like interconnected network of star graphs. During training we generate SBM or SSM graphs using parameters estimated on the real graphs.

4 EXPERIMENT SETUP

Datasets: We evaluate the effectiveness of our model on datasets from four different domains: 1) Rural Networks - networks gathered by [1] to the study diffusion of micro-finance in Indian rural households, 2) Retweet Networks from twitter [20], 3) Animal Interaction Networks - networks are a part of the wildlife contact networks collected by [5] to study the physical interactions between Voles, 4) Homeless Networks -from various HIV intervention campaigns organized for homeless youth in Los Angeles [25, 27]. For each of the 4 families of networks, we randomly divide them into train and test data as shown in Table 1. We use Algorithm 1 to train models on networks from training set and deploy them on the networks from test set for each network family.

Network category	Train networks	Test Networks
Rural	rural1,rural2	rural3,rural4
Animal	voles1,voles2	voles3, voles4
Retweet	copen, occupy	isreal,damascus
Homeless	a1,spy,mfp	b1,cg1,node4, mfp2,mfp3,spy2,spy3

Table 1: Train and test split for different sets of networks

Sampling baseline: For selecting the best baseline, we tested four sampling based graph discovery methods which are used in [24, 25]: CHANGE, SNOWBALL, RECOMMEND and RANDOM-GREEDY. We found that Random greedy, Snowball and recommend are 3.9%, 42.1% and 41.7% worse than CHANGE respectively. Hence, *we only use CHANGE as the baseline.* CHANGE is a recent state-of-art method that was used for effective HIV intervention campaign. It uses a simple yet powerful sampling method: *For each of the random seeds, we query one of its neighbors picked at random.* The model is inspired by *friendship paradox* which states that the expected degree of a random node's neighbor is larger than the expected degree of a random node.

Environment parameters: We assume that information flow is modelled by independent cascade model [12]. We assign it the same parameters to the model as done in [26] since it is similar to the real-world deployment setting. We fix the diffusion probability p for edges as 0.1. We fix the budget for the number of nodes to be activated at 10. Similar to setting in [26], before we start network discovery, we are given 5 random seed nodes $\mathcal{R}$ and their neighbourhood is revealed. We have $T = 5$ queries to discover the graph G_T using which we find the 10 nodes to activate.

Performance metrics: We employ the greedy algorithm, which we denote as the oracle O, on the discovered graph to pick the nodes to activate. We remark that greedy gives the optimal approximation ratio unless $P = NP$ [12]. OPT is the influence score when we have the entire network for O to choose from. This value can be represented as $I_{G^*}(O(G^*))$ (note that we approximate OPT via the greedy algorithm since exact optimization is NP-hard.) We use the following performance metrics to validate our models:

(1) *influence score* or *influence reward*: We deploy our model on graphs from the test set and consider the *average number of nodes influenced* over 100 runs as the performance metric.

	Rural		Animals		Retweet		Homeless						
Train\Test	rural3	rural4	voles3	voles4	damascus	israel	b1	cg1	node4	mfp2	mfp3	spy2	spy3
OPT	25.2	45.4	110.6	115.7	195.4	115.2	24.7	17.02	15.84	20.56	23.45	21.26	21.71
CHANGE	17.4	31.5	33.7	58.9	95.24	30.6	19.1	14.17	12.85	14.6	16.5	16.01	16.09
Geometric-DQN	**18.95**	**35.7**	**45.8**	**80.2**	**119.3**	**43.6**	**20.2**	**14.98**	**13.9**	**15.95**	**17.7**	**17.4**	**18.2**

Table 2: Comparison of influence score of CHANGE and best performing Geometric-DQN model for each test network

	Rural		Animals		Retweet		Homeless						
Train\Test	rural3	rural4	voles3	voles4	damascus	israel	b1	cg1	node4	mfp2	mfp3	spy2	spy3
Avg. individual	**13.14**	14.86	13.07	29.05	11.58	8.16	4.76	12.51	25.75	11.74	7.18	**21.40**	31.61
Multiple	10.26	**18.71**	**14.84**	**35.21**	**15.54**	**15.37**	**14.29**	**22.11**	**35.12**	**12.92**	**18.10**	20.76	**35.77**

Table 3: Comparing average *improve percent* using individual network with that for training with multiple networks

(2) *improve percent*: percentage reduction with respect to the gap between *OPT* and a given method. (This is the scaled reward from Equation 1). This is useful to aggregate performance over multiple networks since the range of values for actual *influence scores* for each network vary depending on network properties.

5 RESULTS

The policies learned through reinforcement learning by our agent results in a significant increase in the number of nodes influenced. Table 4 shows the scores of the best-performing variant of our agent on each test set. Geometric-DQN improves the total number of nodes reached by 7-26% compared to CHANGE, allowing us to substantially improve the effectiveness of influence maximization interventions without increasing the budget for network discovery.

Network Family	improve %
Rural	23.76
Animals	26.6
Retweet	19.7
Homeless	7.91

Table 4: Summary of best scores averaged over test networks

In the following subsections, we discuss multiple ways we can improve robustness of training in the face of uncertainties such as choosing the right training graph or overcoming lack of actual real-world network to train on. First, we show that simultaneously training with multiple networks is on average better than using a single network. Then, we show the effectiveness of synthetic graphs both as an effective substitute for real networks and as a valuable method for data augmentation.

Training on Individual network vs Multiple networks: We can directly use the training networks to discover efficient policies for sampling from the unknown network of similar domain. One way to do this is to pick one of the train networks to train on and this indeed outperforms the baseline. However, using scaled rewards (eqn: 1) allows us to leverage information from multiple networks simultaneously. As shown in table 3, *training with multiple networks gives better performance gains compared to average scores obtained with training on single networks on all but one.*

Synthetic graphs can be valuable Synthetic graphs can be useful substitutes when we have limited access to real-world social networks. Since synthetic graphs are cheaper alternatives to collecting more real-world data, they can also be used along with available networks for training. We performed two different experiments to validate the usefulness of synthetic graphs: 1) Use only the synthetic graphs generated for training, 2) Use synthetic graphs along with training dataset. For each training graph we generated 5 synthetic graphs from SBM model (as discussed in Section 3.4). During training, we use synthetic graphs with probability 0.5 for an episode. Otherwise, we sample from the training dataset. We empirically found that this method gave best performance.

During training, to generate a synthetic graph we sampled a graph from training set, fit a SBM (or SSM for the retweet dataset) model, and sampled a synthetic graph. We compare *improve percent* for different families of graphs using these two settings along with other settings in figure 4. We see that the *improve percent* scores for models using only synthetic graphs are better than the average of the scores of models trained from individual networks. *This indicated that training on synthetic graph alone is a more robust method than training on individual real networks.* For Homeless, Retweet and Rural networks, data augmentation improved the performance on the model trained on only training graphs. In the case of Homeless networks using only synthetic graphs give the highest score on average. This may indicate that the SBM model generates graphs that model some of the Homeless networks very well. These experiments suggest that using synthetic graphs when we don't have access to real graph datasets can be very effective.

6 INSIGHTS ON POLICY LEARNT

We observe some of the characteristics of the policy learnt by the Geometric-DQN models and investigate some interpretable quirks of the policies that lead to improvement over CHANGE.

6.1 Size of the discovered graph

In general, we note that the Geometric-DQN policies almost always discover larger graphs than CHANGE though there is no correlation between influence scores and size of discovered graphs by

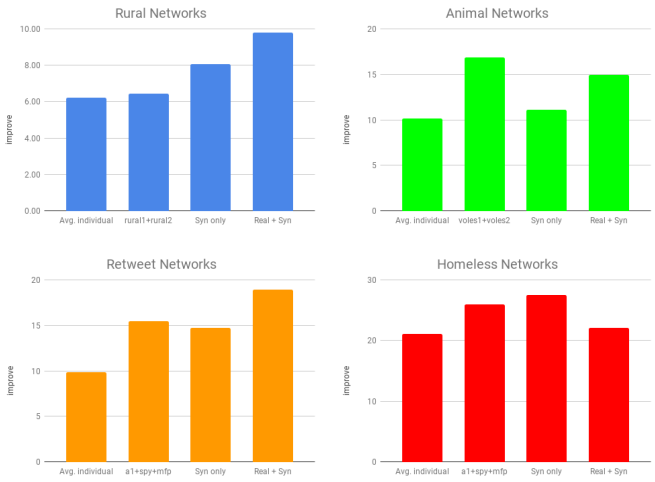

Figure 4: Variation of average *improve percent* using synthetic data. Syn only: Only synthetic graphs for training. Real+Syn: Use both synthetic and actual train graphs

different Geometric-DQN policies. Therefore, we further explore the properties of the nodes selected by the policy.

6.2 Observations on node selection

We observed two frequent events, labelled *O1* and *O2*, during deployment on test networks:
O1: The next node to be selected has *minimum* degree
O2: The node selected is from set of nodes most recently discovered.
We found that almost all the time, if *O2* occurs, *O1* also does.

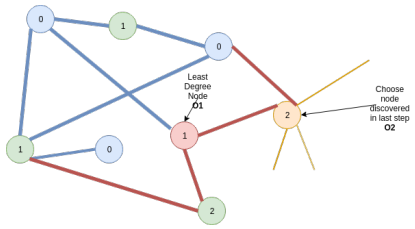

Figure 5: A toy example demonstrating observations *O1* and *O2*. The number on top of nodes denote the time-step at which they where discovered. Red nodes satisfy O1. Orange nodes satisfy O2. Green nodes are other potential actions. Blue denotes seed nodes

Heuristics based on observations. To verify that the behaviours *O1* and *O2* were beneficial for our task, we devised two heuristics.
H1: *Query only from the nodes with* minimum *degree in sub-graph*
H2: *Query only from the nodes with minimum degree from the set of node discovered in the* previous *step. If no new nodes are discovered in the previous step, choose unqueried node in the discovered graph.*
We break all ties by choosing uniformly at random. The performance of the heuristics is summarized in Table 5. We observe that H2 outperforms CHANGE by huge margins for Animals, Homeless and Retweet networks, whereas H1 performs comparable to

Graph	CHANGE	H1	H2	Geometric-DQN
rural3	17.4	17.36	17.3	**18.95**
rural4	31.5	32.39	32.6	**35.7**
voles3	33.7	38.7	39.6	**45.8**
voles4	58.9	61.9	72.8	**80.2**
damascus	95.2	93.7	104.9	**119.3**
israel	30.6	30.37	34.2	**43.6**
b1	19.1	19.0	19.4	**20.2**
spy2	16.01	15.877	16.4	**17.4**

Table 5: Comparisons of scores of heuristics with baselines and best of Geometric-DQN models for each graph

CHANGE in all networks. Geometric-DQN models still perform much better than heuristics. This indicates that the policies learned by Geometric-DQN that are adapted to the specific class of training networks are more nuanced than the heuristics we designed.

6.3 Properties of selected nodes

To further investigate why the heuristics and Geometric-DQN policy performs better, we look at **degree centrality** and **betweenness centrality** of the nodes queried in the true underlying graph, (including the nodes and edges not discovered yet).

We call betweenness and degree centrality of a node computed on the true graph as its *true betweenness centrality* and *true degree centrality* respectively. Picking nodes with high true degree centrality allows access to a larger number of nodes during discovery. For network discovery, nodes of high true betweenness centrality could act as a bridge between different strongly connected communities of nodes for further exploration. In relation to influence maximization, nodes with true high betweenness centrality can allow the flow of information between parts of the network which would otherwise be hard to access.

In particular, we report studies on *b1*, an interaction network depicting real-world community structure and *israel*, a retweet network with sparsely interconnected star-graphs. We compare the true degree centrality and true betweenness centrality of queried nodes using CHANGE, Geometric-DQN and the heuristics discussed above. As we can see from Figure 6, compared to other models, the Geometric-DQN can recognize nodes with high true degree centrality and true betweenness centrality. H2 also picks nodes with higher true betweenness centrality than CHANGE, but does not improve in terms of average true degree centrality.

We further investigate these properties vary across timesteps for H1, H2 and best Geometric-DQN model (see Figure 7). On average, Geometric-DQN finds nodes of high full betweenness centrality and degree centrality, especially in the last query. This may indicate that Geometric-DQN has leveraged the Deepwalk embeddings as well as the learned graph embeddings to find complex higher-order patterns in the graphs that enable it to find such nodes.

7 RELATED WORKS

Influence Maximization and network discovery. The existing literature on algorithmic aspects of influence maximization problems

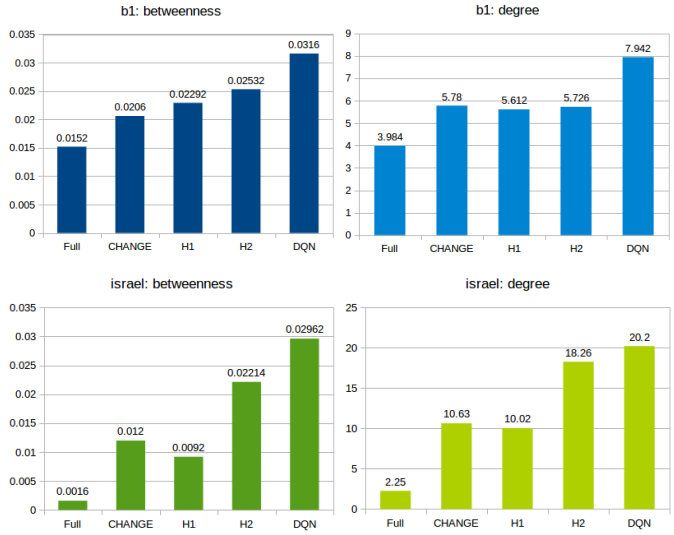

Figure 6: Average true betweenness (R) and degree(L) centrality of Full graph, nodes queried by CHANGE, H1, H2 and best Geometric-DQN

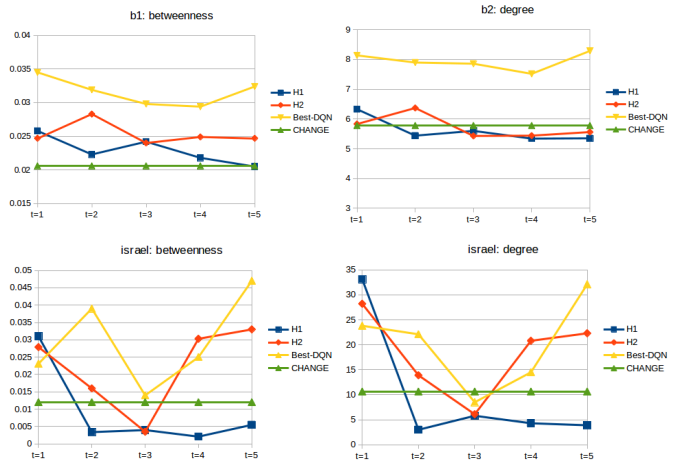

Figure 7: Average true betweenness(L) and degree(R) centrality across timesteps for H1,H2 and best Geometric-DQN. (We have added corresponding CHANGE values for reference)

[4, 10, 11, 16, 23] build on a greedy strategy for influence maximization, which iteratively adds the node with the largest marginal contribution to the objective until the budget is reached. Our work is situated along a more recent, largely orthogonal axis: developing algorithms which reduce the data collection requirements needed to deploy influence maximization in the real world. Wilder et al. [25] introduced the exploratory influence maximization problem, where the goal is to sample nodes from the network such that an influential seed set can be selected. They propose an algorithm motivated by community structure and prove theoretical guarantees for graphs from the stochastic block model. [27] introduced CHANGE, a more practical algorithm based on the friendship paradox [8].

Reinforcement Learning for Combinatorial Problems on graphs. There has been some recent work on the idea of using deep reinforcement learning methods to automatically learn heuristics to solve combinatorial problems on graphs that generalize well to graphs from a similar distribution. E.g., [2] use policy gradient methods to solve the Travelling Salesman Problem (TSP) and [13] use Q-learning to solve Minimum Vertex Cover, Maxcut and TSP. In contrast to these works, our setting deals with unknown, partially observed graphs, and we train an *adaptive* policy for deciding where in the graph to query based on what as been observed so far. This distinction in problem setting renders earlier architectures ineffective for our problem because the state and action representations change as more information about the graph is uncovered.

8 DISCUSSION AND CONCLUSION

We introduce a novel deep Q-learning based method to leverage structural properties of the available social networks to learn effective policies for the network discovery problem for influence maximization on undiscovered social networks. Our method can leverage information about structural properties of the networks and even use synthetic graphs as a robust alternative to training on real-world graphs or can be used along with training graphs as data augmentation technique. We showed that our trained models outperform current state-of-the-art algorithms on social networks across different domains. We observed 7-23% improvement over CHANGE (a state-of-art sampling algorithm). Our Q-learning based model uses the influence score on the entire network as the reward signal to learn the policy. An interesting direction for future work is to explore applications of our method to other network discovery problems to by altering the reward function.

We have also observed the graph embeddings learned was used to pick nodes with high betweenness centrality with respect to the entire network, which was key to discovering important portions of the network. A detailed investigation into complex patterns learned by our model could reveal insights about structural properties of social network pertaining to the identification of influential nodes.

REFERENCES

[1] Abhijit Banerjee, Arun G Chandrasekhar, Esther Duflo, and Matthew O Jackson. 2013. The diffusion of microfinance. *Science* 341, 6144 (2013), 1236498.

[2] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural Combinatorial Optimization with Reinforcement Learning. *ArXiv* abs/1611.09940 (2016).

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[4] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*. 1029–1038.

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

[6] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning Structural Node Embeddings via Diffusion Wavelets. In *KDD*.

[7] Alessandro Epasto and Bryan Perozzi. 2019. Is a Single Embedding Enough? Learning Node Representations that Capture Multiple Social Contexts. In *WWW*.

[8] Scott L Feld. 1991. Why your friends have more friends than you do. *Amer. J. Sociology* 96, 6 (1991), 1464–1477.

[9] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks* 5, 2 (1983), 109–137.

[10] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. Irie: Scalable and robust influence maximization in social networks. In *ICDM*. 918–923.

[11] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD*. 137–146.

[12] David Kempe, Jon Kleinberg, and Éva Tardos. 2005. Influential nodes in a diffusion model for social networks. In *International Colloquium on Automata, Languages, and Programming*. Springer, 1127–1138.

[13] Elias Boutros Khalil, Hanjun Dai, Yuyu Zhang, Bistra N. Dilkina, and Le Song. 2017. Learning Combinatorial Optimization Algorithms over Graphs. In *NIPS*.

[14] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[15] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

[16] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. 2018. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1852–1872.

[17] William S. Lovejoy. 1991. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research* 28 (1991), 47–65.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[20] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. http://networkrepository.com

[21] Eldar Sadikov and Maria Montserrat Medina Martinez. 2009. Information propagation on Twitter. *CS322 project report* (2009).

[22] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. *CoRR* abs/1511.05952 (2016).

[23] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*. ACM, 75–86.

[24] Thomas W Valente and Patchareeya Pumpuang. 2007. Identifying opinion leaders to promote behavior change. *Health Education & Behavior* 34, 6 (2007), 881–896.

[25] Bryan Wilder, Nicole Immorlica, Eric Rice, and Milind Tambe. 2018. Maximizing influence in an unknown social network. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[26] Bryan Wilder, Nicole Immorlica, Eric Rice, and Milind Tambe. 2018. Maximizing influence in an unknown social network. In *AAAI*. 4743–4750.

[27] Bryan Wilder, Laura Onasch-Vera, Juliana Hudson, Jose Luna, Nicole Wilson, Robin Petering, Darlene Woo, Milind Tambe, and Eric Rice. 2018. End-to-end influence maximization in the field. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1414–1422.

[28] Amulya Yadav, Bryan Wilder, Eric Rice, Robin Petering, Jaih Craddock, Amanda Yoshioka-Maxwell, Mary Hemler, Laura Onasch-Vera, Milind Tambe, and Darlene Woo. 2018. Bridging the Gap Between Theory and Practice in Influence Maximization: Raising Awareness about HIV among Homeless Youth.. In *IJCAI*. 5399–5403.

[29] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*. 4800–4810.