Deep Reinforcement Learning-based Approach to Tackle Competitive Influence Maximization

Tzu-Yuan Chung*
tzuyuan.chung@gmail.com
CITI, Academia Sinica
Taipei, Taiwan

Khurshed Ali*
TIGP-SNHCC Program,
Academia Sinica, Taipei
ISA, NTHU, Hsinchu, Taiwan
khurshedmemon@gmail.com

Chih-Yu Wang
CITI, Academia Sinica
Taipei, Taiwan
cywang@citi.sinica.edu.tw

ABSTRACT

Competitive Influence Maximization (CIM) problem studies the competition among multiple parties where each party aims to maximize their profit while competing against other parties. Recently, Reinforcement-Learning based models have been proposed to address the CIM problem. However, such models are unscalable and incapable of handling changes in the network structure. Motivated by the recent success of Deep Reinforcement Learning models and their capability to handle complex problems, we propose a novel Deep Reinforcement learning based framework (DRIM) to address the multi-round competitive influence maximization problem. DRIM framework considers the community structure of the social network for budget allocation and feature extraction with deep Q network in order to reduce the computational time of seed selection. The proposed framework employs the quota-based ϵ-greedy policy to explore the optimality of influence maximization strategies and budget allocation for each community. Experimental results show that the proposed DRIM framework performs better than the state-of-art algorithms to tackle the multi-round CIM problem.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Influence Maximization, Competitive Influence Maximization, Deep Reinforcement Learning, Social Networks, Q-Learning

ACM Reference Format:
Tzu-Yuan Chung, Khurshed Ali, and Chih-Yu Wang. 2019. Deep Reinforcement Learning-based Approach to Tackle Competitive Influence Maximization. In *Proceedings of 2019 ACM SIGKDD conference (KDD19 (MLG Workshop)) (MLG'19).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.475/123_4

*Both authors contributed equally to this research.

MLG'19, August 2019, Anchorage, Alaska, USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

Viral marketing has been recognized as one of the most effective marketing tactics in advertising and branding today. It exploits social media, such as Facebook, Twitter, and YouTube, to promote products or brands to the public. The companies select key customers in a social network or media using such tactic in order to spread their product's attractive features in pursuit of the "word-of-mouth" effect. Such problem, identification of key customers to maximize the "word-of-mouth" effect, is referred as *Influence Maximization* (IM), one of the widely studied research topic in the social network [10, 17, 29].

Competitive influence maximization (CIM) problem, which considers multiple parties competing for their influences in the social network, is a natural but more practical extension to traditional IM problem. Specifically, CIM problem aims to select the best seeds in response to the other competitors' decisions with the goal of maximizing their influence. In a CIM model, the influences from each party are allowed to cascade simultaneously among the social network, which may interfere with each other.

There have been increasing research efforts in addressing competitive diffusion optimization with heuristic methods under the extensions of IM propagation models [3]. However, most of them fail to consider community structures in social networks. Efficient competitive influence algorithm should properly identify and utilize the hidden community structure within the network in order to maximize the influence spread with the limited budget investment. Despite the significant developments on the incorporation of community structures for a traditional single party IM[9, 31], to our best knowledge, the study on CIM considering community factor is still lacking.

To illustrate community-based CIM, let us consider the social network in Figure 1. Given the social network where two parties are competing against each other for the influence. We assume that nodes 1, 3 and 5 in community 1 and nodes 12 and 13 in community 2 have been activated by party A (blue) and B (green), respectively. If party A has enough budget to select two more seeds, it should select node 4 to expand the influence within community, and then select node 11 to block the influence of party B. If party A selects node 7 as the seed instead of node 11, then the influence will flow to node 2, 6, 8 and 9 which will be activated by node 4, leading to influence overlap. Even though there are three communities in the network, community 3 may be too small worth spending budget (i.e., selecting seed node) since it may only activate three nodes at most. This example shows that the community structure indeed affects the seed selection strategy of parties in CIM problem.

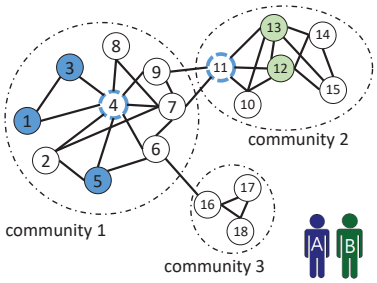


Figure 1: The community structure of an social network

In light of these concerns, we aim to address the following challenges in community-aware CIM problem. (1) How shall we determine the number of seeds to be allocated in each community? (2) What kind of seed-selection strategy shall we adopt for each community given the occupation status? To address the aforementioned community-aware CIM issues, we extend the CIM problem to the community structures of a network. First, instead of selecting seeds only in the first round, we consider a more realistic setup in which the parties might keep taking actions based on the current network state and the expected reactions of other parties within given rounds. Second, each party can spend a limited amount of budget in each round on key nodes. The key (or influential) nodes are selected according to different strategies based on the network state of communities.

Consequently, we address a *multi-round competitive influence maximization* (MRCIM) problem. In MRCIM problem, multiple parties compete against each other in a way that maximizes their rewards considering budget and community structures. Each party would prefer to find an optimal combination of strategies to utilize their budget efficiently for each community in such a competitive environment. However, it has been proved that the selection of seeds is an NP-hard problem even with single party IM. The data-driven approach has been proposed [23] to leverage real-time, history, or simulated data input for reinforcement learning (RL) to determine the optimal strategy based on the pre-defined network states. Nevertheless, this approach is not scalable due to the complexity of state space and incapable of handling network structural or settings change in the network, such as budget adjustment.

Recently, the *deep reinforcement learning* (DRL) technique has demonstrated huge success in playing Atari and Go games [24, 32]. It shows a powerful data-driven method for handling more complex problems. Inspired by the enormous success of DRL methods in various domains, We propose a DRL based competitive Influence Maximization framework (DRIM), with the capability to learn an optimal multi-party IM strategy by considering budget allocation and community structures in it. Our framework achieves high scalability, flexibility and solution quality by (1) addressing the competing process of CIM using community structure of the network, (2) utilizing existing influence maximization strategies in communities as the action space, and (3) developing quota-based ϵ-greedy policy for training and evaluation with deep Q network. Further, we propose three different models, namely **DRIM-seq**, **DRIM-com** and **DRIM-Q** to tackle proper budget allocation and seed-selection issues in the proposed problem.

Our major contributions are summarized as follows:

- To the best of our knowledge, we are the first to apply the DRL technique to CIM problem. In this approach, we propose a DRIM framework with the definitions of network state, community-based strategy action, and deep Q network for reward function approximation.
- We propose a quota-based ϵ-greedy policy to determine the strategies and budget allocation for each community.
- Experiment results indicate that DRIM model achieves better results than the state-of-art algorithm (STORM) [23].

2 RELATED WORK

In this section, we specifically review the related research efforts on CIM problem, which considers influence diffusion of multiple competing products or concepts that interfere with each other. For the general knowledge of influence maximization problem, interested readers can refer to [21] for the comprehensive introduction. Among the rich body of literature, we briefly review the IM models that consider the network's structural properties to solve it.

Bharathi *et al.* [2] first modeled the CIM problem with the extension of the independent cascade model and resort to game theory to analyze the "first mover" strategies in terms of the expected diffusion. Carnes *et al.* [7], on the other hand, propose two influence diffusion models to simulate the competing process and consider the CIM problem from the followers' perspective. Both of these works formulate the CIM problem considering competitors' known strategy and show that at least $1 - 1/e$ approximation guarantee can be achieved with a greedy strategy against the competitor conditioned on competitor's known strategy or predicted accurately.

Some efforts such as [6, 13] proposed a popular strategy known as the influence blocking maximization, that the parties will try to block the effect of the other competitors and minimize their influence. It is a variant of CIM and these studies have provided greedy solutions for this problem. Borodin *et al.* [3] presented several models and proved the proposed models as NP-hard to achieve an approximation.

Community structure is a unique feature of social networks. It is a subset of nodes in the graph with strong connections between them and fewer edges to nodes in different communities. There exist several efficient algorithms[4, 9, 12, 14, 31] handling the IM problem by exploring the community structure. Community structure is also exploited to solve the CIM problem [5].

Recently, Lin *et al.* [23] proposed a reinforcement learning based framework to tackle the CIM problem. Different from previous works, Lin *et al.* [23] considers a more complex but realistic setup that each competitor makes multiple decisions to select seed(s) in multiple rounds. Either in case of known or unknown opponents' strategies, the proposed framework, STORM[23], is capable of learning an optimal policy based on the network state. However, they did not consider the community structure of the social networks and the budget allocation for each community. In addition, their handcrafted state design could lead to inefficient state representation and therefore reach a suboptimal solution.

3 PROBLEM FORMULATION

We model a social network, $G = (\mathcal{V}, \mathcal{E})$, as a weighted and directed graph, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges, respectively.

We then assume there is a set of parties $\mathcal{P}$ who are interested in promoting the ideas to the individuals in the social network through social relations. The ideas proposed by each party are incompatible, that is, one individual will only accept at most one idea from these parties. We define the acceptance of an individual to an idea of party l as the occupation status of the node in graph G. A node can be activated if it accepts the idea of the party. We denote the occupation status of a node u as $s_u \in \mathcal{P} \cup \{0\}$, in which $s_u = 0$ means the node remains inactive. Once a node u accepts the idea of a party l, it cannot switch its occupation status to other parties.

The parties select the seed node(s) in each round. At the beginning of a round t, each party l selects the seeds in turns by determining a set of nodes $\mathcal{F}_t^l$ from those who are still not occupied. The budget of each party is k, that is, each of them can select at most k seeds. After each party has selected seeds, the ideas diffuse at the same time following the competitive diffusion process. We denote the nodes occupied by party l at the end of the diffusion process as $\mathcal{V}^l = \{u | s_u = l\}$. In this paper, we use the competitive linear threshold (CLT) model, but it is applicable to use competitive independent cascade (CIC) model as well.

Definition 3.1. COMPETITIVE LINEAR THRESHOLD (CLT) Given the graph $G = (\mathcal{V}, \mathcal{E})$ and set of party $\mathcal{P}$, each node v picks an activation threshold ρ_v. At round t, the node v is activated by party $l \in \mathcal{P}$ if the total weight of its active in-neighbors exceeds the threshold, i.e., $\sum_{u \in O_t^l} w_{u,v} > \rho_v$, where $O_t^l \subseteq \mathcal{V}$ is the activated nodes set of party l before t^{th} round and $w_{u,v}$ is the edge weight from node u to v.

The influence from each party will be propagated at the same time after the seed nodes have been activated. When the conflict happens, meaning that if more than one party have the right to activate the same node v, we adopt the majority rule that the node v will be activated by party p^i whose total influence is highest on v; in other words, $\sum_{u \in O_t^i} w_{u,v} > \sum_{u \in O_t^j} w_{u,v}$ for any party j. We now formally define the MRCIM problem as follows.

Definition 3.2. MULTI-ROUND COMPETITIVE INFLUENCE MAXIMIZATION (MRCIM) The MRCIM problem consists of T rounds. In each round, the parties begin in turn by selecting k seeds among G from nodes which are not occupied. The influence propagation is performed with the CLT model at each round and will be continued till T rounds or until no more nodes can be activated. **The goal of party l in MRCIM is to maximize its overall influence after T rounds, that is, $_{\mathcal{F}_t^l, 1 \le t \le T} |\mathcal{V}^l|$ subject to competitive linear threshold model.**

4 PRELIMINARIES

4.1 Reinforcement Learning and Deep Q Networks

Reinforcement learning refers to reward-oriented algorithm, in which an agent learns how to solve tasks based on a scalar reward signal through interacting with an *environment* $\mathcal{E}$. During interaction with the environment, the agent learns a policy $\pi(s)$, a function that maps states to actions, to define the way of behaving in a certain situation. An optimal policy aims to maximize the accumulative reward $\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where $\gamma \in [0, 1]$ is a discount factor that controls the weight it gives to future reward. RL models value function V and action function Q to estimate how good the policy π is in maximizing the reward in the long run.

The common model-free algorithm, Q-learning, does not require a model of the environment and allows us to approximate the V and Q functions even having no knowledge beforehand regarding state transition probabilities. The Q value is updated by the following equation as proposed in [34]:

$$Q_{t+1}(s_t, a_t) = (1 - \alpha) Q_t(s_t, a_t) + \alpha(r_{t+1} + \gamma V(s_t + 1)) \quad (1)$$

Q-learning uses the greedy policy $\pi(s) =_a Q(s, a)$ to maximizes the expected total reward.

Though reinforcement learning has achieved certain success in various applications, it is generally intractable in large scale problems due to the high dimensional state and action space. The advent of deep learning helps us to reduce the burden of feature extraction to approximate the value functions. Instead of learning action values in all states, we can learn a parameterized value function $Q(s, a; \theta)$ constructed by a neural network that returns an approximate Q value with the parameters θ. A deep Q network (DQN) was first proposed by Mnih et al. [24, 25]. DQN consists of a neural network with multiple layers and given an n-dimensional state; it returns a vector of action value $Q(s, \cdot; \theta)$ in m size. In light of the instability of Q estimation with an online network, they also proposed target Q network that is the same as the online network except that the parameters θ^- are frozen for a fixed number of iterations and updated from the online network, i.e., $\theta^- = \theta$. The target network is helpful for estimating target $Q(s', a'; \theta^-)$ and stabilize the algorithm. The target Y^Q in DQN is:

$$Y^Q = r + \gamma \max_{a'} Q(s', a'; \theta^-) . \quad (2)$$

Another key component of DQN is the experience replay [22]. During training, the agent observes and collects the experience in the form of $e = (s, a, r, s')$ over times and sampled a batch of experience uniformly at random to train the network.

4.2 Community Detection

Several approaches have been proposed to uncover community structures in a social network [15, 19, 27]. Most of these works consider the spectral properties of a network to infer communities and its structure. However, such techniques do not perform well to detect communities on sparse networks [18, 26]. Recently, the non-backtracking methods [1, 18] have shown significant performance in detecting communities and their structure even on sparse networks. Inspired by its performance, we adopt spectral algorithms based on a non-backtracking walk [18] to detect the communities in a social network.

5 PROPOSED FRAMEWORK

Since the multi-round competitive influence maximization (MRCIM) can be proved as an NP-hard problem by reducing it to a broad family of competitive influence[3]. We refer to data-driven or learning-based models to look for an approximated solution that can maximize the influence in the long run. Although traditional RL had success to tackle MRCIM [23], previous approaches lack scalability and were limited to fairly small networks. When the network gets

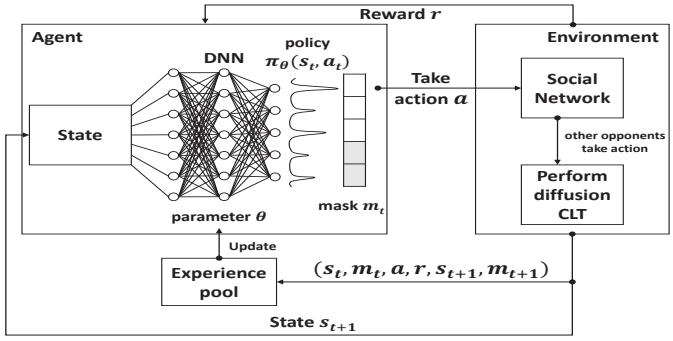


Figure 2: Deep Q-learning framework in training stage

complicated, this problem becomes intractable since the knowledge space would be too large to store each state-action pair. In this paper, we propose a Deep Reinforcement Learning based competitive Influence Maximization framework (DRIM) to handle the complexity by combining two techniques: deep reinforcement learning (DRL) and community detection. The most important property of DRL is that deep neural networks can automatically find compact low-dimensional features of high-dimensional data without compromise of hand-crafted features. In addition, we observe that the influence diffusion highly depends on the inherent community structure in the social network. The community information should be extracted and treated as features in the proposed learning algorithm. Figure 2 illustrates the flow of our framework. The agent observes the state of a network and takes a set of community-based actions derived from the deep Q network to achieve the maximum influence spread. The influence propagation is performed after the parties selected k seeds among the network and then the state transition will be stored in the experience pool. To handle budget allocation problem, we further propose a Q-learning algorithm with the quota-based ϵ-greedy policy to determine the number of seeds to be allocated in a given community and determine which nodes to select for this community in the evaluation and training of Q-value at each round.

5.1 Framework Structure

The goal of the agent in the proposed framework (DRIM) is to learn the optimal policy π for seed placement strategy that maximizes its expected accumulated rewards. At each step t, The agent observes a set of features that represent the network state $s_t \in \mathcal{S}$ and selects an action a_t from the set of legal actions $\mathcal{A}$.

Environment. The environment is the proposed competitive influence diffusion model discussed in Definition 3.1 which propagates the influence of active nodes based on the CLT model.

Reward. We evaluate the agent based on the number of activated nodes in the last round, which is designed as the delayed reward. The agent will receive a delayed reward r_t^l as, 0 if $t < T$ and $|\mathcal{V}^l|.|p|/v$ if $t = T$. Where $|\mathcal{V}^l|$ is the expected number of nodes activated by party l in the terminal state. Squashing the rewards to the range $[0, |\mathcal{P}|]$ confines the scale of the error derivatives and generalize across a wide variety of possible situations.

Action. We formulate the action space by extending the meta-learning approach proposed in [23] to include the properties of community structure in the target social network. Specifically, we couple the community selection and seed placement strategy into a pair-wise meta-learning framework. Each action determines which seed placement strategy to take in which community. When a certain strategy is selected for a certain community, the strategy is applied in subgraph defined by the community in order to select the seed node rather than on the whole social network. [1] Further, readers are requested to review the seed-selection strategies proposed in [23] except we apply those seed-placement strategies on a community level.

State. Existing learning-based framework [23] resorted a set of handcrafted features to represent the network state. The performance of the proposed framework [23] relies on the quality of the feature representation in the designed state space. Here, we take advantage of DQN to design more high-dimensional features to represent the current activation state and the network condition. The following are the feature vectors that we have designed based on the correlation between the principle of strategies and the rewards.

(1) D^{out}: Number of free out-degree
(2) W^{out}: Summation of free out-edge weight
(3) B^l: Summation of free out-edge weight for nodes which are the neighbors of party l
(4) O_x^l: Activation state by party l after x rounds, where $x \in \{0, 1, 2\}$
(5) C^m: identity of community c_m, where the community identity of each node is detected by the non-backtracking spectral method [18]

When learning from high dimensional feature vector observations, the different aspects of the observations may have various scale from network to network. This makes it difficult for the neural network to use the same hyper-parameters which generalize across the different environments. We address this issue by normalizing each feature vector to [0, 1] so as to reduce the error derivatives.

Deep Q network. The combination of each feature we defined implies a huge state space. Several methods have been proposed to represent the features in low dimension space, such as kernel-based method [28], randomized trees [11], and neural networks [30] to approximate the Q-value. In the DRIM framework, we create a neural network, which is similar to the neural network structure as in [25], to estimate the Q-value calculated by Eq. (1). As illustrated in Figure 3, we utilize the above features to represent the environment state as the input features of the deep Q network. The rectified linear unit (ReLU) is used as the activation function in hidden layers, and the linear layer is added for calculating action value as output. With one forward pass, the Q-value for all actions could be derived from the deep Q network, leading to great improvement in efficiency.

5.2 Training Setup of DRIM

Experience replay. To perform experience replay we store the experience $e_t = (s_t, m_t, \hat{\mathcal{A}}_t, r_t, s_{t+1}, m_{t+1})$ at each time step t in an experience pool $\mathcal{M} = \{e_1, ..., e_t\}$, where m denotes the action mask used to avoid the agent from selecting invalid action, i.e., the action that selects seeds in a community where the nodes are fully

[1] Notice that the proposed learning algorithm can include any kind of IM strategy as the candidate in meta-learning.

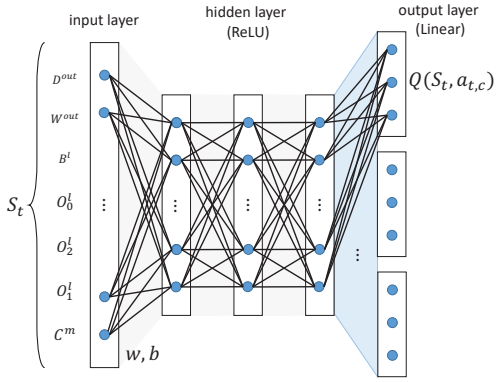


Figure 3: Neural network structure in DRIM framework

activated. During training, we perform mini-batches Q-learning updates to samples of experience $(s_t, m_t, \hat{\mathcal{A}}_t, r_t, s_{t+1}, m_{t+1}) \sim U(\mathcal{M})$, drawn uniformly at random from the experience pool.

5.3 Deep Reinforcement Learning Framework and Value Function Approximation

Quota-based ϵ-greedy policy. In the MRCIM model, we assume that each party can invest a limited amount of budget, k, on the potential nodes at each round. Ideally, we can train a Q-network for estimating the value of the combination of strategies, but this requires a large action space of $|\mathcal{A}|^K$, which could make learning challenging. To come up with a model that is capable of finding the global maximum without searching in large action space, we further propose *quota-based ϵ-policy* during the training phase to tackle the budget allocation issue by exploring the combinatorial strategy that comprises the basic strategies defined in Section 5.1. The motivation of quota-based ϵ-policy is based on the observation[9] that placing too many seeds in the same community is inefficient due to influence overlapping. In order to determine how much budget to allocate in a given community, we design a quota formula based on the number of free nodes in communities. Let $C = \{c_1, c_2, ..., c_m\}$ denote a set of communities in the network and each c_i has n_i number of free nodes. Then the quota of seeds nodes allocated in c_i with a budget k is defined as:

$$quota(c_i) = \frac{k \times n_i}{\sum_{j=1}^{m} n_j}. \tag{3}$$

After the quota for communities is determined at each step, the agent gathers a set of strategies based on the quota-based ϵ-greedy policy:

$$\pi_c(s) = \begin{cases} \{a_c^i | a_c^i \in \mathcal{A}_c; i = random(n); c \in C, quota(c) > 0\} & \text{probability } \epsilon \\ \{a | a =_{a_c \in \mathcal{A}_c} Q(s, a_c); c \in C, quota(c) > 0\} & \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{A}_c$ is the valid action set in a given community c and includes the seeds by the selected action in c to complete for the quota of c. Quota-based ϵ-greedy policy greatly reduces the action space for budget k and prevent the curse of dimensionality. For example, the naive model that each party has k budget at each step leads to a combinatorial action space with dimensionality: $|\mathcal{A}|^k$, while the model based on quota-based ϵ-policy reduces the action space size to $|\mathcal{A}| \times |C|$, which is significantly smaller than the former.

In the training phase, the agent selects the valid strategy under the mask using quota-based ϵ-policy derived from action-value function Q. It updates the parameters of Q-network for a fixed number of steps using a sample of experience drawn uniformly at random from the experience pool. Since MRCIM is a finite round game, $Q_{t+1}(s_t, a_t)$ is guaranteed to converge to the optimal policy through Q-learning [33]. The target vector of the neural network can be calculated by Eq. (2) and then we can train the Q-network with parameters θ_i at each iteration i by minimizing a sequence of loss functions.

$$\mathbb{E}_{(s,m,\hat{\mathcal{A}}_t,r,s',m') \sim U(\mathcal{M})}\left[\left(Y^Q - \sum_{a_c \in \hat{\mathcal{A}}_t} Q(s, a_c; \theta_i) \times \frac{quota(c)}{k}\right)^2\right] \tag{5}$$

We calculate the loss function by normalizing the Q value estimation with respect to the allocated quota. If we directly use the summation of meta-action values that participate in strategy combination, it may overestimate the action values and result in a large variance. The parameters of the target network are only changed every given step and keep fixed between individual updates for more stable training. The training process then enters the updated state s_{t+1}. More details are provided in Algorithm 1.

Algorithm 1 DQN for multi-round competitive influence maximization

```
Initialize neural network Q with random weights
Copy neural network Q and store as Q̂(·|θ⁻)
Initialize experience pool M to capacity N
while training is not terminal do
    s_t ← s_0, m_t ← m_0
    for t = 1, T do
        Select a set of valid strategies Â_t according to the
            quota-based ε-greedy policy and m_t
        Perform each strategy in Â_t
        Simulate the opponents' strategy
        Propagate influence and observe state s_{t+1},
            mask m_{t+1} and reward r_t
        Store transition (s_t, m_t, Â_t, r_t, s_{t+1}, m_{t+1})
            in M
        s_t ← s_{t+1}, m_t ← m_{t+1}
        if t mod training frequency == 0 then
            Sample random minibatch of transitions
                from M
            Update the value function Q approximator
                by (5)
        end if
    end for
end while
```

5.4 Estimation Models of Combinatorial Strategy for Budget Allocation

Ideally, we could train the deep Q network with various budget settings for estimating the value of the combination of meta-actions, but this involves a large multiplicative computational overhead that grows with the number of players and budgets, which could make the Q-learning challenging. For more general models, we leverage a pre-trained Q-network with a given budget setting to approximate the Q-value of combinatorial action. We propose three

models, namely DRIM-seq, DRIM-com, and DRIM-Q to determine the budget allocation and strategies for each community with pre-trained Q-network.

DRIM-seq The naive way to determine the k seeds is to sequentially pick a particular meta-action with the maximum Q-value in following k steps, where each meta-action selects a single node as seed without propagation. We can then estimate the action-value of the combination of meta-action $Q(s_t, u_t)$, where $u_t = \{a_t, a_{t+1}, ..., a_{t+k}\}$. $Q(s_t, u_t^i)$ represents the expected accumulated rewards by performing the combinatorial strategy. That is, we learn a deterministic policy $\pi'(s)$ defined as:

$$\pi'(s_t) = \{\pi(s_t^0), \pi(s_t^1), ..., \pi(s_t^k)\}, \tag{6}$$

where $\pi(s_t^i)$ is the deterministic policy for the action choice, i.e. $\pi(s_t^i) = a_t^i$, and the next state s_t^{i+1} is determined after the agent activates the seed selected by action a_t^i. A greedy policy from Q-learning is used to compute these policies:

$$\pi(s_t^i) = _{a_t^i \in \mathcal{A}^i} \ Q(s_t^i, a_t^i), \tag{7}$$

where $\mathcal{A}^i$ is the set of legal actions in such state s_t^i.

DRIM-com Another direct approach to utilize the pre-trained Q-network is to adopt the community quota formula to select $quota(c)$ seeds for each community c based on the meta-action with the maximum Q-value. This is basically the quota allocation of a quota-based e-greedy algorithm we proposed in Section 5.3, that is, we allocated the budget according to Eq. (3), and select the action of each community according to the output of deep Q network given by Eq. (4)

DRIM-Q Finally, We present a quota formula that adapts our DRIM models for budget allocation. In the testing phase, instead of using the quota formula based on the number of free nodes in communities, we dynamically assign the quota according to the maximum Q-value in each community. The quota formula based on the Q-value is calculated by Eq. (8), which is proportional to the maximum Q-value in each community c_i.

$$quota_Q(c_i) = \frac{k \times \max_{a_{c_i} \in \mathcal{A}_{c_i}} Q(s, a_{c_i})}{\sum_{j=1}^{m} \max_{a_{c_j} \in \mathcal{A}_{c_j}} Q(s, a_{c_j})} \tag{8}$$

Although the action value is not exactly a linear function of the quota for communities, we can still heuristically estimate the expected reward contributed by meta-action in each community and assign corresponding quota to it by Eq. (8). After the quota is determined by Eq. (8), we may select the action of each community derived from the maximum Q-value for each community.

6 EXPERIMENTS

6.1 Experimental Setup

We conducted experiments to evaluate the efficiency of the proposed models in terms of influence spread. The datasets we used consists of two real-world social networks and two synthetic networks. The real social networks are obtained from the Stanford Large Network Dataset Collection website [20] while the synthetic datasets in our experiments are generated using the stochastic block model [16] to produce graphs containing communities. The statistics of networks can be found in Table 1. In the experiments, we use the CLT model as the diffusion model. While the edge weights and the activation threshold of nodes are set randomly in a range between 0 and 1. With respect to the Q-learning agent, fully-connected neural networks are used for the state representation learning. The neural network has three hidden layers, and each layer is comprised of 512 neurons. In the training phase, the experience replay is used and the memory size is set to 35,000 tuples of (s, m, a, r, s', m'). We use RMSProp algorithm for learning with stochastic gradient descent where the batch size is set to 32, while the learning rate is held fixed to 0.00025. The value of ϵ used in quota-based ϵ-greedy is annealed down from 1 to 0.1 throughout training and we assign 0.99 to the discount factor.

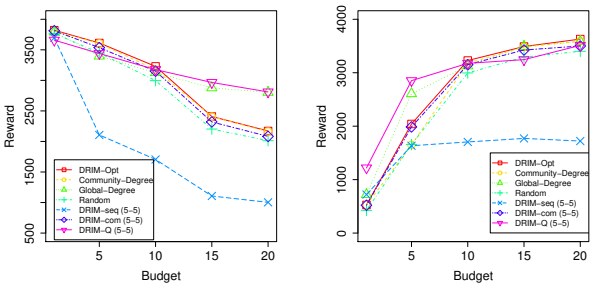
Table 1: Datasets

Name	#Nodes	#Edges	#Comm	Description
Facebook	4,039	88,234	10	Social circles from Facebook
C-GrQc	5,242	14,496	9	Collaboration network of Arxiv General Relativity
SBM-10	3,000	14,921	10	Synthetic network generated by stochastic block model
SBM-1	3,000	14,789	1	Synthetic network generated by stochastic block model

In the experiments, DRIM-seq, DRIM-com, and DRIM-Q are evaluated in different budget settings, k. In addition, the state-of-art algorithm STORM [23] for CIM is also implemented for comparison. Random strategy, in which the strategies are selected randomly, serves as the baseline of non-strategic seed selection. Two naive methods, Community Degree and Global Degree, are implemented as the baseline performance of the fixed strategy. Specifically, Global Degree is the traditional maximum degree algorithm while Community Degree is the community-aware version of maximum degree algorithm [8]. Finally, DRIM-Opt is the optimal DRIM model which represents the performance upper bound of DRIM when the budget in training and testing stages are the same, and the budget allocation is according to the quota of each community. We train those models for a total of 10,000 episodes against the community-based degree strategy in four networks and in each episode the parties take actions and influence is propagated for $T = 7$ rounds. All of these evaluations are measured by running a workload of 500 episodes. The experiments are implemented by C++ on a server with Intel XEON E5 2.3GHz 36 cores processors and 512GB RAM.

6.2 Experimental Results

6.2.1 Evaluation on budget setting. In the first experiment, we examine the effectiveness of the proposed models' performances in terms of reward by assuming both parties have different budget settings during testing. In this case, we have trained the models in advance by assuming both parties have the same budget on a facebook network, that is, $k = 5$. Figure 4 (a) illustrates the reward comparison of DRIM models with fixed budget (i.e., $k = 10$) while varying competitors' budget. Moreover, Figure 4 (b) present the reward comparisons of fixed competitor's budget with varying



(a) Fixed AI budget to 10 (b) Fixed OP budget to 10

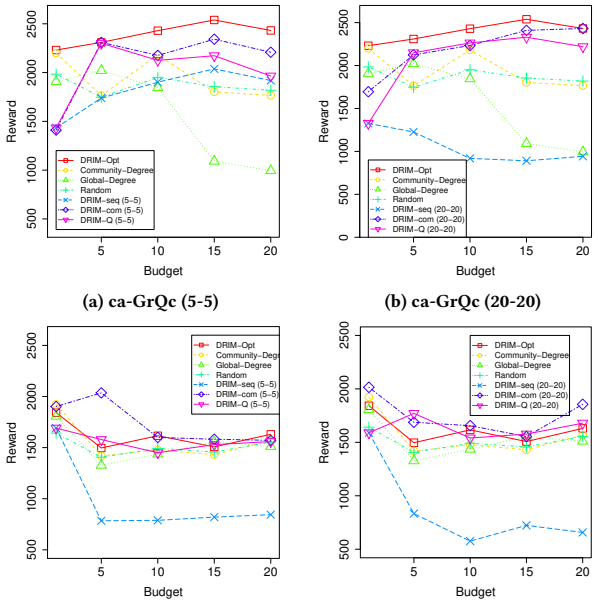
Figure 4: Evaluation on varying budget setting on Facebook

agent's budget. It can be seen from Figure 4 that DRIM-Opt, DRIM-com, and DRIM-Q achieve better performance than the baseline models on Facebook network in all aforementioned budget constraint settings. In addition, DRIM-seq performs poorly in most settings. This is due to the over-investment in a single community. In DRIM-seq, the agent chooses to select each seed according to the current state after a previous selection. When the budget is small, it is likely that the current selection does not change the state much. Therefore, DRIM-seq will select the same action in the same community repeatedly which results in inefficient budget allocation. Finally, we observe that DRIM-Q performs closely or even outperforms DRIM-Opt in almost all budget settings. This suggests that budget allocation according to Q-value is a better estimation than pure quota-based estimation.

6.2.2 Evaluation on networks with different structures. Next, we illustrate the performance of the proposed framework on networks with different structures. We conduct the experiments on ca-GrQc, SBM-10 and SBM-1, which contain 9, 10, and 1 communities, respectively. In this experiment, we use the models with the budget settings of 5 and 20 for both parties. From Figure 5, the result shows that DRIM-Opt, DRIM-com, and DRIM-Q outperforms DRIM-seq and other baseline methods no matter in either network with more or few communities in the social networks.

6.2.3 Evaluation on edge-weight setting. Third, we examine the effect of edge-weight setting in the range of [0.1, 0.4] and [0.4, 0.7] on the SBN-10 network. The influence will diffuse further in a higher value of edge weight; that is, the seeds would affect more nodes. Here again, we use the models with the budget setting of 5 and 20 for both parties. Figure 6 illustrate the performance comparison in various budget setting. It also shows that either DRIM-Opt, DRIM-com, and DRIM-Q outperforms others. In general, DRIM-Opt performs best while DRIM-com and DRIM-Q perform similarly. Besides, the performance gap of these methods to other baselines remains on the same scale. This suggested that the performance improvement brought by DRIM is robust to the changes in edge weights, or the diffusion speed.

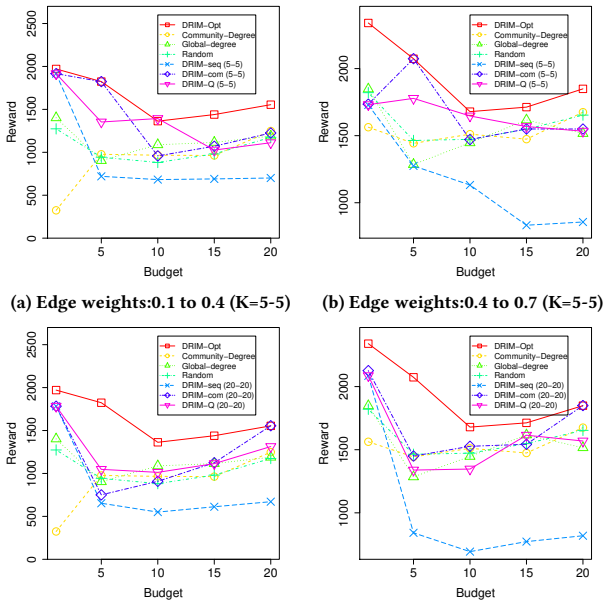
6.2.4 Performance comparison with STORM against known strategies. Finally, we conduct the experiments on SBM-10 network to measure the performance of DRIM and STORM algorithms against the four aforementioned baseline strategies. Here, we assume that the AI (DRIM-Opt and STORM) and the opponent have the 1 and



(a) ca-GrQc (5-5) (b) ca-GrQc (20-20)

(c) SBM-10 (5-5) (d) SBM-10 (20-20)

Figure 5: Evaluation on networks with different structures



(a) Edge weights:0.1 to 0.4 (K=5-5) (b) Edge weights:0.4 to 0.7 (K=5-5)

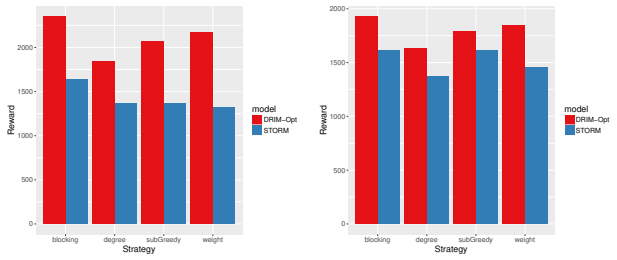
(c) Edge weights:0.1 to 0.4 (K=20-20) (d) Edge weights:0.4 to 0.7 (K=20-20)

Figure 6: Evaluation on networks with different weight setting

20 budget, respectively. It can be observed from Figure 7 that DRIM-Opt algorithm achieves better results than the STORM model competing against the opponent's every strategy in all budget settings.

7 CONCLUSIONS

In this work, we propose a novel deep reinforcement learning-based framework (DRIM) to tackle the multi-round competitive



(a) 1 Budget **(b) 20 Budget**

Figure 7: DRIM-Opt and STORM algorithms results against known strategies on SBM-10 with 1 and 20 budget, respectively

influence maximization (MRCIM) problem considering network community structure. To our knowledge, this is the first DRL-based model for MRCIM problem that incorporates community structure for seed-selection strategies. DRIM framework incorporates the quota-based ϵ-greedy policy to determine the budget allocation and seed-selection strategies for each community of the network. Besides, we propose three different models of DRIM framework to tackle the budget allocation and seed selection for each community in MRCIM problem. The experimental results show that the DRIM models outperform state-of-the-art learning-based algorithms to tackle the MRCIM problem.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Science and Technology under Grant MOST 105-2221-E-001- 003-MY3 and the Academia Sinica under Grand Challenge Seed Grant AS-GC-108-01.

REFERENCES

[1] Emmanuel Abbe and Colin Sandon. 2015. Detection in the stochastic block model with multiple clusters: proof of the achievability conjectures, acyclic BP, and the information-computation gap. *arXiv preprint arXiv:1512.09080* (2015).

[2] Shishir Bharathi, David Kempe, and Mahyar Salek. 2007. Competitive influence maximization in social networks. In *International Workshop on Web and Internet Economics*. Springer, 306–311.

[3] Allan Borodin, Yuval Filmus, and Joel Oren. 2010. Threshold models for competitive influence in social networks. In *International Workshop on Internet and Network Economics*. Springer, 539–550.

[4] Arastoo Bozorgi, Hassan Haghighi, Mohammad Sadegh Zahedi, and Mojtaba Rezvani. 2016. Incim: A community-based algorithm for influence maximization problem under the linear threshold model. *Information Processing & Management* 52, 6 (2016), 1188–1199.

[5] Arastoo Bozorgi, Saeed Samet, Johan Kwisthout, and Todd Wareham. 2017. Community-based influence maximization in social networks under a competitive linear threshold model. *Knowledge-Based Systems* 134 (2017), 149–158.

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

[7] Tim Carnes, Chandrashekhar Nagarajan, Stefan M Wild, and Anke Van Zuylen. 2007. Maximizing influence in a competitive social network: a follower's perspective. In *Proceedings of the ninth international conference on Electronic commerce*. ACM, 351–360.

[8] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 199–208.

[9] Yi-Cheng Chen, Wen-Yuan Zhu, Wen-Chih Peng, Wang-Chien Lee, and Suh-Yin Lee. 2014. CIM: Community-based influence maximization in social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 2 (2014), 25.

[10] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 57–66.

[11] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6, Apr (2005), 503–556.

[12] Maoguo Gong, Chao Song, Chao Duan, Lijia Ma, and Bo Shen. 2016. An efficient memetic algorithm for influence maximization in social networks. *IEEE Computational Intelligence Magazine* 11, 3 (2016), 22–33.

[13] Xinran He, Guojie Song, Wei Chen, and Qingye Jiang. 2012. Influence blocking maximization in social networks under the competitive linear threshold model. In *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 463–474.

[14] Maryam Hosseini-Pozveh, Kamran Zamanifar, and Ahmad Reza Naghsh-Nilchi. 2017. A community-based approach to identify the most influential nodes in social networks. *Journal of Information Science* 43, 2 (2017), 204–220.

[15] Jianbin Huang, Heli Sun, Jiawei Han, Hongbo Deng, Yizhou Sun, and Yaguang Liu. 2010. Shrink: a structural clustering algorithm for detecting hierarchical communities in networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 219–228.

[16] Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical review E* 83, 1 (2011), 016107.

[17] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 137–146.

[18] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. 2013. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences* 110, 52 (2013), 20935–20940.

[19] Andrea Lancichinetti, Santo Fortunato, and János Kertész. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3 (2009), 033015.

[20] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

[21] 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* (2018).

[22] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8, 3-4 (1992), 293–321.

[23] Su-Chen Lin, Shou-De Lin, and Ming-Syan Chen. 2015. A learning-based framework to handle multi-round multi-party influence maximization on social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 695–704.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

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

[26] Raj Rao Nadakuditi and Mark EJ Newman. 2012. Graph spectra and the detectability of community structure in networks. *Physical review letters* 108, 18 (2012), 188701.

[27] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 026113.

[28] Dirk Ormoneit and Śaunak Sen. 2002. Kernel-based reinforcement learning. *Machine learning* 49, 2-3 (2002), 161–178.

[29] Matthew Richardson and Pedro Domingos. 2002. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 61–70.

[30] Martin Riedmiller. 2005. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*. Springer, 317–328.

[31] Jiaxing Shang, Shangbo Zhou, Xin Li, Lianchen Liu, and Hongchun Wu. 2017. CoFIM: A community-based framework for influence maximization on large-scale networks. *Knowledge-Based Systems* 117 (2017), 88–100.

[32] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and others. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.

[33] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[34] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.