Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks

Eli A. Meirom [1] **Haggai Maron** [1] **Shie Mannor** [1] **Gal Chechik** [1]

Abstract

We consider the problem of controlling a partially-observed dynamic process on a graph by a limited number of interventions. This problem naturally arises in contexts such as scheduling virus tests to curb an epidemic; targeted marketing in order to promote a product; and manually inspecting posts to detect fake news spreading on social networks.

We formulate this setup as a sequential decision problem over a temporal graph process. In face of an exponential state space, combinatorial action space and partial observability, we design a novel tractable scheme to control dynamical processes on temporal graphs. We successfully apply our approach to two popular problems that fall into our framework: prioritizing which nodes should be tested in order to curb the spread of an epidemic, and influence maximization on a graph.

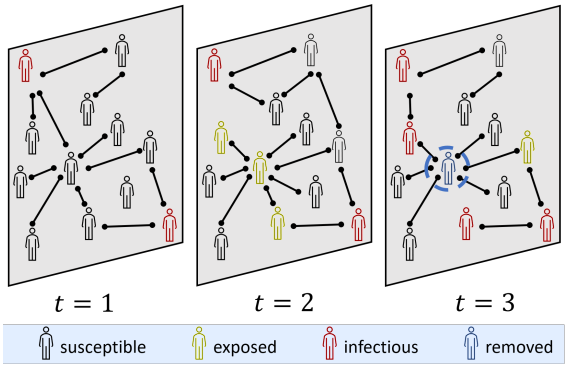

Figure 1. A viral infection process on a graph and an intervention aimed to stop its spread. Here, graph nodes represent people and edges represent interactions. At $t = 1$ only two people are infected (red). At $t = 2$ several interactions resulted in new *exposed* people (yellow); At $t = 3$ the blue node was selected to be quarantined to stop the viral spread. This paper presents a general framework for learning how to control such dynamic processes on graphs.

1. Introduction

Consider an epidemic spreading in the population. To contain the disease and prevent it from spreading, it becomes critical to detect infected carriers and isolate them; see Fig. 1 for an illustration. As the epidemic spreads, the demand for tests outgrows their availability, and not all potential carriers can be tested. It becomes necessary to identify the most likely epidemic carriers using limited testing resources. How should we rank candidates and prioritize vaccines and tests to prevent the disease from spreading? As a second example, imagine a seemingly very different problem, where one would like to promote an opinion or support product adaption by advertisements or information sharing on a social graph. If an impactful node is convinced, it may influence other nodes towards the desired opinion, creating a cascade of information diffusion.

These two problems are important examples of a larger family of problems: controlling diffusive processes over networks through nodal interventions. Other examples include viruses inflicting computer networks or cascades of failures in power networks. In all these cases, an agent can steer the dynamics of the system using interventions that modify the states of a (relatively) small number of nodes. For instance, infected people can be asked to self-quarantine, preventing the spread of a disease, and key twitters may be targeted with coupons. However, a key difficulty is that the current state is often not fully observed, for example, we don't know the ground truth infection status for every node in the graph.

More formally, we consider a graph $G(t) = (\mathcal{V}, \mathcal{E}(t))$ whose structure changes in time. $\mathcal{V}$ is the set of nodes and $\mathcal{E}(t) = \{e_{uv}(t)\}$ is the set of edges at step t. The state of a node $v \in \mathcal{V}$ is a random variable that depends on the interactions between v and its neighbors. At each turn, the agent may select a subset of nodes and attempt to change their state. The goal is to minimize an objective that depends on the number of nodes in each state. For example, consider a setup where the agent tries to promote its product or opinion. At each step, the agent may select a set of seed

[1]NVIDIA Research, Israel. Correspondence to: Eli Meirom <emeirom@nvidia.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

nodes and attempt to influence them by presenting relevant information or ads. If those nodes are convinced, they may spread the information through future contacts. The optimization goal, in this case, is to maximize the number of influenced nodes.

The problem of controlling the dynamics of a system using localized interventions is very hard, and for several reasons. First, it requires making decisions in a continuously changing environment with complex dependencies. Second, to solve the problem one must assess the potential downstream ripple effect for any specific node that becomes affected, and balance it with the probability that the node indeed becomes affected. Finally, models must handle noise and partial observability. In particular, it is well known that even the single-round, non-sequential, influence maximization problem is computationally hard (Kempe et al., 2003).

Current approaches for solving this problem can be divided into two main families: (1) Monte Carlo simulation that estimates the utility of each decision (see e.g. Goyal et al., 2011). These approaches can find good solutions for small to moderate-sized ($\sim 10^3$ nodes) graphs, but do not scale to larger graphs. (2) Heuristics based on topological properties of the known graph. For example, act on nodes with a high degree (e.g. Liu et al., 2017). These approaches can be scaled to very large graphs, but are often sub-optimal. In addition to these two families, learning approaches have been used to mix different heuristics (Chung et al., 2019; Tian et al., 2020).

We pose the problem of controlling a diffusive process on a temporally evolving graph as a partially-observed Markov decision process (POMDP). We then formulate the problem of selecting a subset of nodes for dynamical intervention as a *ranking* problem, and design an actor-critic RL algorithm to solve it. We use the observed changes of nodes states and connections to construct a temporal multi-graph, which has time-stamped interactions over edges, and describe a deep architecture based on GNNs to process it.

The main challenge in our setup is that the underlying dynamics is not directly and fully observed. Instead, partial information about the state of some nodes is given at each point in time. While the diffusive process spreads by point contacts, new node information may impact our belief on the state of a node a few hops away from the source of new information. For example, consider an epidemic spreading on a network. Detecting an infected person directly modifies the probability that nodes that are connected to it by a path in the temporal graph are also infected (Fig. 2). To address this issue, our architecture contains two separate GNN modules, one updates the node representation according to the dynamic process and the other is in charge of long range information propagation. These GNNs take as input a multi-graph over the nodes, where edges are time-stamped with the time of interactions. In addition, we show that combining RL with temporal graphs requires stabilizing information aggregation from other neighbors when updating nodes hidden states, and control how actions are sampled during training to ensure sufficient exploration. We show empirically the benefits of these components.

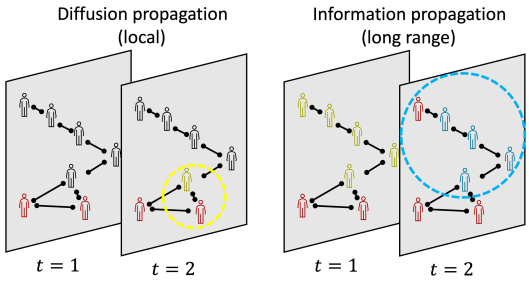

Figure 2. The difference between two types of data propagation on the graph. Red nodes are positively tested epidemic carriers. Yellow nodes are undetected epidemic carriers. Blue nodes are inferred to be infected. Left - infection propagation involves only direct neighbours. Right - long range information propagation: The top node is detected as infected at time $t = 2$. As it must have been infected by its neighbor, our belief regarding the infection state of people on this long infection chain, including those that are found many hops away, change. We use two separate GNNs in order to model these two processes.

We test our approach on two very different problems, Influence Maximization and Epidemic Test Prioritization, and show that our approach outperforms state-of-the-art methods, often significantly. Our framework can be possibly further extended for problems beyond the ones mentioned here, e.g. traffic control, active sensing for complex scenes, etc.

This paper makes the following contributions: **(1)** A new RL framework for controlling *partially-observed diffusive processes over graphs*. We present a novel formulation of two challenging problems: the *testing allocation* problem and the *partially-observed influence maximization* problem. **(2)** A new architecture for controlling the dynamics of diffusive processes over graphs. Our architecture prioritizes interventions on a temporal multi-graph by leveraging deep Graph Neural Networks (GNNs). **(3)** A set of benchmarks and strong baselines, including network-based real-world contact tracing statistical data for COVID-19. Our RL approach achieves superior performance over these datasets.

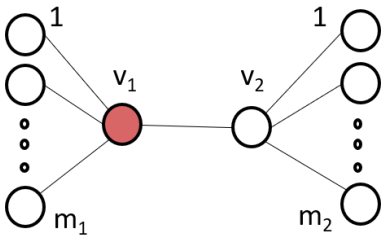

Figure 3. A double star configuration. The state of v_2 is unknown at the $t = 1$. v_1 is infected at $t = 0$.

2. A motivating example

We begin with an example to illustrate the trade-offs of the problem (Figure 3). In this example, our goal is to minimize the number of infected nodes in a social interactions graph.

Given a list of time-stamped interactions between nodes, we form a discrete time-varying graph as follows. If u and v interact at time t, then the edge $e = (u, v)$ exists at time t. Each interaction is characterized by a transmission probability $p_e(t)$, meaning that a healthy node that interacts with an infected node at time t becomes infected with probability $p_e(t)$.

For the purpose of this example, assume that we can test a single node only at odd timesteps. If the node is positively tested as infected, it is quarantined and cannot further interact with other nodes. Otherwise, we do not perturb the dynamics and it may interact freely with its neighbors.

Consider the "two stars" network in Figure 3. The left hub (node v_1) has m_1 neighbors, and the right hub (v_2) has m_2. At $t = 0$, only the edge (v_1, v_2) is present with transmission probability p. For all $t \geq 1$, all edges depicted in Figure 3 exist with transmission probability 1. Assume that this is known to the agent, and that at $t = 1$ we suspect that v_1 was infected at $t = 0$. Clearly, we should either test v_1 or v_2. It is easy to compute the expected number of infected nodes in both cases (details in Appendix A). The decision would be to test v_2 if $2p \geq 1 + m_1/m_2$ and otherwise test v_1.

This example illustrates that an optimal policy must balance two factors: *the probability that the dynamics is affected* - that a test action yields a "positive", and the future consequences of our action - the *strategic importance* of selecting v_1 *vs.* v_2, expressed by the ratio m_1/m_2. A policy targeting likely-infected nodes will always pick node v_1, but since it only focuses on the first term and ignores the second term, it is clearly suboptimal.

3. Problem Formulation

We start with a general formulation of the control problem, and then give two concrete examples from different domains: Epidemic test prioritization, and dynamic influence maximization. Formal definitions are given in Appendix B.

3.1. General formalism

Consider a graph $G(t) = (\mathcal{V}, \mathcal{E}(t))$ whose structure changes in time. $\mathcal{V}$ is the set of nodes and $\mathcal{E}(t) = \{e_{uv}(t)\}$ is the set of edges at step t. Each edge $e_{uv}(t)$ is associated with features $\phi_{uv}(t)$ which may vary in time, and each node v is characterized with features $\zeta_v(t)$.

The state of a node $v \in \mathcal{V}$ is a random variable $ST_v(t)$ which can have values in $\mathcal{Y} = \{y_1, y_2, ..\}$. The node's state dynamic depends on the interactions between v and its neighbors, its state and the state of those neighbors, all at time $t - 1$. At each step, the agent selects a subset $a(t)$ of k nodes, and attempt to change the state of any selected node $v \in a(t)$, namely, apply a stochastic transformation on a subset of the nodes. Selecting nodes and setting their states defines the action for the agent, and plays the role of a knob for controlling the global dynamics of the process over the graph. The action space consists of all possible selections of a subset $a(t)$ of k nodes $a(t) \subset V$. Even for moderate graph, with $\sim 100 - 1000$ and small k the action space $\binom{|\mathcal{V}|}{k}$ is huge.

The optimization criterion depends only on the total number of nodes in state y_i, $c_i(t)$. The objective is therefore of the form $\max \sum_t \gamma^{t-t_0} g(c_1(t), c_2(t), ..)$, where future evaluations are weighted by a discount factor $\gamma \leq 1$. Additionally, the agent may be subject to constraints written in a similar manner $\sum_i f_i(c_1(t), c_2(t), ..) \geq z_i(t)$.

3.2. Epidemic test prioritization

We consider the recent COVID-19 outbreak that spreads through social contacts. The temporal graph G is defined over a group of nodes (people) $\mathcal{V}$, and its edges $\mathcal{E}(t)$ are determined by their daily social interactions. An edge (u, v) between two nodes exists at time t iff the two nodes interacted at time t. Each of these interactions is characterized by features $e_{uv}(t)$, including its duration, distancing and environment (e.g., indoors or outdoors). Additionally, each node v has features $\zeta_v(t)$ (e.g., age, sex etc.).

The SEIR model dynamics (Lopez & Rodo, 2020). Every node (person) can be in one of the following states: *susceptible* – a healthy, yet uninfected person (S state), *exposed/latent* – infected but cannot infect others (L state), *infectious* – may infect other nodes (I state), or *removed* – self-quarantined and isolated from the graph (R state).

A healthy node can become infected by interacting with its neighbors. The testing intervention changes the state of a node. If infected or exposed, its state is set to R, otherwise it remains as it is. More details can be found in the appendix.

Optimization goal, action space. The objective is to minimize the spread of the epidemic, namely, minimize the number of infected people (in either L, R or I states), over time. Our setup differs from previous work (e.g., (Hoffmann et al., 2020; Wang et al., 2020)) in two important aspects. First, we do not assume a node can be vaccinated or immunized against the epidemic. Second, we do not assume a node can be quarantined or disconnected from the graph without justification, namely, without a positive test result. Often, nodes perform required social functionality. Isolating a high-degree node from the network, like putting a bus-driver in quarantine, will either deteriorate the transportation network quality, or will require using a replacement driver that will have the same interactions pattern. A preemptive node removal would either not affect the network connectivity or impair the network functionality.

Observation space. At each time t, the agent is exposed to all past interactions between network nodes, $\{\mathcal{E}(t')|t' < t\}$. In addition, we are given partial information on the nodes state. The agent is provided with information on a subset of the infectious nodes at $t = 0$. At every $t > 0$, the agent observes all past test results, i.e, for every $v \in a(t'), t' < t$ we observe if node s was healthy at t' or not.

3.3. Dynamic influence maximization

The classical multi-round influence maximization problem (Domingos & Richardson, 2001; Kempe et al., 2003; Lei et al., 2015) assumes the agent knows the groundtruth state of every node at every turn. More often than not, that is an unrealistic assumption. The agent can only know if a person is influenced if the person *actively* signals it, for example by using a coupon code. Furthermore, there might be a substantial delay from the time the information was presented to the time a feedback was received. Therefore, we extend this setup to include partial observability.

Model Dynamics. Each node is either *Influenced* or *Susceptible*. Influenced nodes try to influence their neighbors, following a dynamic generalization of two canonical models: Linear Threshold (LT) and Independent Cascades (IC). In an IC model, if u is Influenced and $(u, v) \in \mathcal{E}_t$, then u may influence v according to a probabilistic model. In a LT model, each node v is associated with a threshold w_v, and each edge e carries an impact weight of q_e. If the sum of edge weights, the cumulative "peer pressure", of neighboring infected nodes exceeds w_v, node v is influenced. See Appendix B for details on these models.

Optimization goal, action space. The goal is to maximize the number of *Influenced* nodes. All nodes start at the *Susceptible* state. At each step the agent selects a seed set $a(t)$ of k nodes, and attempts to influence them. Each attempt succeeds with some probability q independently for every $v \in a(t)$.

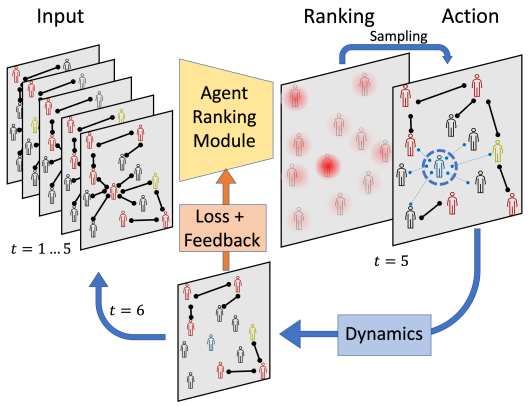

Figure 4. Schematic of our approach. The Ranking module receives as input a sequence of graphs and outputs scores over nodes. Scores are then used to sample actions, selecting nodes for intervention. Here, the person circled in blue is selected for quarantine and its connections are canceled (dashed blue lines). The downstream effect on epidemic progression is then fed as a loss to the ranking module.

Observation space. At every step, an influenced node may reveal that it is influenced, e.g. by clicking on ads, with some probability η. The set of these signals at previous times along with past interactions between nodes consists the observation space.

4. Approach

This section introduces our main contribution. Our goal is to select a subset of nodes for influencing the dynamics. The direct approach would be to perform a Monte Carlo simulation of the diffusive process for every possible action at every step, and choose the best performing action. However, this approach does not scale, and is unfeasible even for moderate networks (see Liu et al., 2017; Banerjee et al., 2020, and Appendix C for discussion). An alternative popular approach uses predefined heuristics or greedy approaches (e.g., (Yang et al., 2020; Preciado et al., 2014; Murata & Koga, 2018)), but this is arbitrary and often sub-optimal.

We propose a learning-based approach, which generalizes from past patterns collected during training. Since our goal is to maximize an objective over time in a dynamic environment, RL is a natural choice (Figure 4).

Yet, even with a learning approach, solving the general case of the *subset selection* problem would be combinatorially hard (Kempe et al., 2003) and is difficult to scale to large graphs. At the other extreme, a simple approximated solution can be achieved by scoring each node independently and then selecting the top-ranked nodes. Unfortunately, this approximation would potentially be far from optimal because it neglects correlations across nodes that are crucial.

Therefore, it is important that node selection would consider other nodes, at least locally. For example, creating tight clusters of *Influenced* nodes is critical in Influence Maximization under the Linear Threshold model (see Appendix B). Assume that the intervention budget is sufficient for establishing a single cluster but there exist two equally beneficial regions to promote such cluster. The agent should learn to focus on one region rather than spread on two regions. This requires learning to choose optimal subsets rather than choosing nodes independently.

Our approach takes a mid-road: We use a graph neural network to compute per-node scores, where each node is exposed to the features of nodes in its extended m-hop neighborhood (where m is the depth of the GNN). This way, agent can learn to take into account complex correlations, and to select high-quality subsets by ranking nodes by their scores.

4.1. The Ranking Module

Overview. In our approach, an RL agent receives as input the node and edge features of the temporal graph, and scores each node. The module that performs that scoring is called the *ranking module* (Figure 5). Scores are used to generate a probability distribution over nodes, and then for sampling a subset of k nodes for testing. Namely, the scores encode the agent policy. The ranking module also updates the internal representation of each node, which aggregates past observations and information.

The score of a node is affected both by propagation dynamics and by information available to the agent. One may hope that on a short time scale the node score would only be affected by its neighboring nodes. Unfortunately, information can propagate long distances in the graph almost instantaneously, because revealing the state of one node in a long chain affects other nodes.To handle this effect, the ranking module contains two GNNs (see Fig. 5). (1) A local diffusion component D updates the diffusion process state; and (2) a long-range information component I updates the information state.

We use Proximal Policy Optimization (PPO) (Schulman et al., 2017) to optimize our agent. We sequentially apply the suggested action, log the (state, action) tuple in an experience replay buffer, and train our model based on the PPO loss term. We further motivate our framework and extended the discussion on our design choices in Appendix C.

4.2. Modules

Input. The input to the ranking module consists of three feature types: (1) *Static node features* $\zeta_v^s(t)$: e.g., topological graph centralities (betweenness, closeness, eigenvector, and degree centralities) and random node features. (2) *Dynamic node features* $\zeta_v^d(t)$: All intervention results up to the current timestamp. We denote all nodes features as a concatenation $\zeta_v(t) = [\zeta_v^s(t), \zeta_v^d(t)]$. (3) *Edge features* and the structure of the temporal graph $\mathcal{E}(t)$: All previous interactions up to the current step, including the transmission probability for each interaction. All these features are scalars, except the dynamic node features, which are encoded as one hot vectors. Figure 5 illustrates the basic data flow in the ranking module.

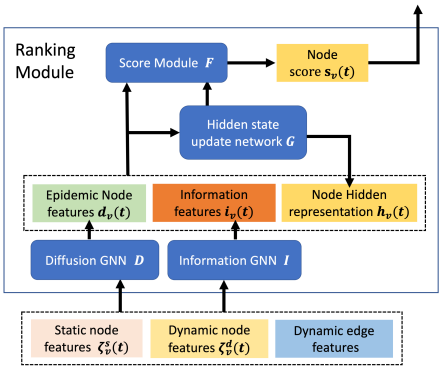

Figure 5. The ranking module. It is composed of 4 neural networks I,D,G,F, which update the nodes scores and hidden states at each time step.

Local diffusion GNN. The spread through point contact is modeled by a GNN D. As the diffusive process spreads by only one hop per step, it is sufficient to model the spread with a single GNN layer. Formally, denote by $u \sim_t v$ an interaction between u and v at time t, and by p_{vu} the probability of transmission during this interaction. For each v, the output of $D(\cdot)$ is a feature vector denoted by $d_v(t)$:

$$d_v(t) = \sum_{u \sim_t v} p_{vu}(t) \cdot M_e(\zeta_v(t), \zeta_u(t); \theta_{m_e}),$$

where M is multilayer perceptron (MLP). Rather than considering the probability as an edge feature, this component mimics the dynamic process transition rule to accelerate learning.

Long-range information GNN. GNN I computes the *information state* of each node. As discussed above, updated information on a node u a few hops away from node v may abruptly change our belief on the state of v. Furthermore, this change may occur even if v and u did not interact in the last time step but rather a while ago. To update the information state, we construct a cumulative multi-graph G' where the set of edges between nodes v and u at time t are all the interactions that occurred during the last τ steps. The features of each edge $\phi_{vu}(t')$ at time t' are the interaction delay $t - t'$ and the transmission probability $p_{v,v'}(t')$. The information features are the output of k-layer GNN; the l^{th}

layer is:

$$x_v^l(t) = \sum_{v' \sim_t v} M^l(x_v^{l-1}(t), x_{v'}^{l-1}(t), \phi_{vv'}(t); \theta_M^l).$$

As before, M^l is an MLP, with $x_v^0(t) = \zeta_v(t)$ and $i_v(t) = x_v^k(t)$ are the final node features.

Score and hidden state update. For every node we hold a hidden state $h_v(t)$, updated according to a neural network G,

$$h_v(t) = G(h_v(t-1), \zeta_v(t), d_v(t), i_v(t); \theta_g) \quad (1)$$

After updating the new node hidden states, we use them to calculate the node scores using a neural network F,

$$s_v(t) = F(h_v(t), h_v(t-1), \zeta_v(t); \theta_f) \quad (2)$$

Here, F and G are two additional components (see Fig. 5). F is an MLP, while G can be either an MLP or recurrent module such as GRU.

4.3. Sampling and scoring

During inference, we pick the top k scored nodes. During training, to encourage exploration we use the score per node $s_v(t)$ to sample k nodes. We (1) map the score of n nodes to a probability distribution (2) sample a node, and (3) adjust the distribution by removing its weight. We repeat this process k iterations (sample without replacement).

Score-to-probability. Usually, node scores are converted to a distribution over actions using a softmax. As demonstrated in (Mei et al., 2020), this approach is problematic as node probabilities decay exponentially with their scores, leading to two major drawbacks: it discourages exploration of low-score nodes, and also limits sensitivity to the top of the distribution rather than at the k-th ranked node. Instead, we set the probability to sample an action a_i to

$$\Pr(a_i) = \frac{x_i'}{\sum x_i'} \quad \text{, with } x_i' = x_i - \min_i x_i + \epsilon, \quad (3)$$

where $\{x_i\}$ is the set of scores and ϵ a constant. The probability difference between low scoring nodes and high scoring nodes becomes less extreme than softmax. Furthermore, the parameter ϵ controls the initial exploration ratio. We compare our approach with the recent escort transform (Mei et al., 2020) that is considered to be a state-of-the-art score-to-probability method. As shown in Appendix D, our method outperforms the escort transform in this problem.

5. Experiments

We evaluated our approach in two tasks: (1) Epidemic test prioritization, and (2) Dynamic influence maximization. More experiments and details are in Appendix D.

Real-World Datasets. We tested our algorithm and baselines on graphs of different sizes and sources, ranging from 5K to over 100K nodes. **(1) CA-GrQcA** A research collaboration network (Rossi & Ahmed, 2015). **(2) Montreal**, based on WiFi hotspot tracing(Hoen et al., 2015). **(3) Portland:** a compartment-based synthetic network (Wells et al., 2013; Eubank et al., 2004). **(4) Email:** An email network (Leskovec et al., 2007) **(5) GEMSEC-RO:** (Rozemberczki et al., 2019), friendship relations in the Deezer music service. All these networks have been extensively used in previous works, in particular in epidemiological studies, as key networks models (Sambaturu et al., 2020; Yang et al., 2020; Herrera et al., 2016; Wells et al., 2013; Eubank et al., 2004). Table S4 summarizes the datasets.

Synthetic Datasets. We considered three synthetic, random network families: **(1) Community-based networks** have nodes clustered into densely-connected communities, with sparse connections across communities. We use the *Stochastic Block Model* (SBM, (Abbe, 2017)), for 2 and 3 communities. **(2) Preferential attachment (PA)** networks exhibit a node-degree distribution that follows a power-law (scale-free), like those found in many real-world networks. We used the dual Barbarsi-Albert model (Moshiri, 2018). **(3) Contact-tracing networks.** We received anonymized high-level statistical information (see Appendix D) about real contact tracing networks, collected during April 2020.

Generating temporal graphs. For all networks except CT graphs, at each time step t we select uniformly at random a subset of edges $\mathcal{E}(t)$ and then assign to each edge a transmission probability $q_e(t)$ sampled uniformly in $[0.5, 1]$. We use a different methodology for the CT graphs, See Appendix D for details.

Training procedure. Algorithms were trained on randomly generated PA networks with 1000 nodes. Each experiment was performed with at least three random seeds.

5.1. Epidemic test prioritization

5.1.1. Baselines

We compare methods from three categories.

A. Preprogrammed heuristic (no-learning) baselines. Rank nodes based on: *(1) Infected neighborhood*: Number of known infected nodes in their 2-hop neighborhood (Meirom et al., 2015; 2018). *(2) Probabilistic risk*: Probability of infection at time $t-1$. Using dynamic programming to analytically solve the probability propagation. *(3) Degree centrality* (Salathé & Jones, 2010; Sambaturu et al., 2020). *(4) Eigenvector centrality:* (Preciado et al., 2014; Yang et al., 2020).

B. Supervised learning. Learn the risk per-node using features of the temporal graph, its connectivity, and infection

	PA	CT
TREE-BASED (2)	10 ± 7	11 ± 3
COUNTER MODEL (1)	7 ± 7	14 ± 5
DEGREE (3)	30 ± 2	16 ± 1
EIGENVECTOR (4)	30 ± 1	16 ± 1
SL (VANILLA) (5)	13 ± 3	17 ± 1
SL + GNN (6)	34 ± 3	32 ± 2
SL + DEG (7)	15 ± 3	18 ± 1
SL + DEG + GNN (8)	33 ± 3	32 ± 1
RL (VANILLA) (9)	17 ± 1	16 ± 1
RLGN (OURS)	**52±2**	**40±1**

Table 1. % of healthy nodes achieved on a preferential attachment (PA) network, and contact tracing (CT) network. Here, two nodes were selected for testing at each step, $k = 2$.

state. Each time step t and node v_i is a sample, and its label is determined by the next step. *(5) Supervised (vanilla).* Features include a static component described in Section 4.1, and a dynamic part that contains the number of infected neighbors and their neighbors. *(6) Supervised (+GNN).* Like #5, the input is the set of all historic interactions of v_i's and its d-order neighbors.*(7) Supervised (+weighted degree).* Like #6, the loss weighs nodes are by their degree. *(8) Supervised (+weighted degree +GNN).* Like #6 above, using degree-weighted loss like #7.

C. RL algorithms: RLGN is our algorithm described in Section 4. The input to **(9) RL-vanilla** is the same as in (#1) and (#6) above. Correspondingly, the GNN module described in Section 4 is replaced by a DNN similar to (#6).

Evaluation Metric. The end goal of quarantining and epidemiological testing is to minimize the spread of the epidemic. Our success metric is therefore the percent of nodes kept healthy throughout the simulation. An auxiliary metric we sometime used was **%contained:** The probability of containing the epidemic. This was computed as the fraction of simulations having cumulative infected nodes smaller than a fraction $\alpha = 0.4$.

5.1.2. RESULTS

In the first set of experiments, we compared RLGN with the 9 baselines described in Section 5.1.1 on the synthetic networks described above. The results reported in Table 1 show that RLGN outperforms all baselines on all network types. We selected the top-performing algorithms and evaluated them on the large, real-world networks dataset.

Table 2 compares the performance of the RLGN and the best baseline (SL) on the large-scale datasets. We included the centralities baselines (#3,#4) in the comparison as they are heavily used in epidemiological studies. Table 2 shows that RLGN consistently performs better than the baselines, and the gap is clearly statistically significant. We also evaluated the performance of RLGN on a Preferential Attachment network with $50,000$ nodes (mean degree $= 2.8$), as this random network model is considered a reasonable approximation for many other real-world networks. The mean percentile of healthy nodes at the end of the episode was 51 ± 1 for RLGN, while for the SL+GNN it was only 21 ± 2, a difference of more than 15 STDs.

Analysis. To gain insight into these results, Figure 6 traces the fraction of contained epidemics and infected nodes during training in 3-community networks. Supervised learning detects *substantially more infected nodes* than RLGN (right panel), but these tend to have a lower future impact on the spread, and it fails to contain the epidemic (left). A closer look shows that RLGN, but not SL, successfully learns to identify and neutralize the *critical nodes that connect communities* and prevent the disease from spreading to another community. See a video highlighting these results online [1].

When would RLGN be successful? In sparsely connected networks, it is easy to cut long infection chains, and both approaches succeed. In densely connected networks, there are no critical nodes, because there are many paths between any two nodes. This can also be viewed in terms of the R_0 coefficient, the mean number of nodes infected by a single diseased node. The greater R_0, the more difficult it is to contain the epidemic. Therefore, we expect RLGN to excel in intermediate regimes. Fig. S1(a) indeed shows that RL has a significant advantage over supervised+GNN for a range of R_0 values between 2.0 and 2.9.

We have deepened our analysis, and investigated: (1) Can we quantify the algorithm by their ability to reduce R_0, the mean number of nodes infected by a single diseased node? Can we quantify the performance by the number of tests required to achieve the same level of performance, measuring their effective test utilization? (2) How robust the trained algorithms to variations in the epidemiological parameters? (3) How does the performance gap between the algorithms scale with the network size? Due to lack of space, we expand on these topics in Appendix D.

Appendix D also includes a comparison between RLGN and the best performing baselines across a range of network sizes, initial infection sizes and testing capacities (Table S6).

Ablation studies. We assess the importance of key elements in our framework using ablation studies. First, to quantify the contribution of the information module, we removed it completely from our DNN module, keeping only the epidemic module. The full DNN module achieved a contained epidemic score of 0.77 ± 0.06, while the ablated DNN module corresponding score was 0.62 ± 0.10, a degradation of more than 20%. This shows that the information module has a critical role in improving the performance of the RLGN framework.

Second, in the opposite direction, one may wonder: Why separate local and long-range GNNs, rather than a single higher-capacity network? We found that using a local-

[1] Link: https://youtu.be/Rhqy7YY9gX8

Table 2. Mean percentile of healthy nodes after 20 steps. RLGN perform better on all datasets. In all cases, std < 0.1. 1% of nodes are tested at each step.

	CA-GrQc	Montreal	Portland	Enron	GEMSEC-RO
Degree	25.5	12.8	0.7	71.1	2.4
E.vector	25.4	8.1	0.04	55.1	2.4
SL	29.8	23.1	1.6	68.5	4.3
RLGN	**42.7**	**39.7**	**3.71**	**89.2**	**6.5**

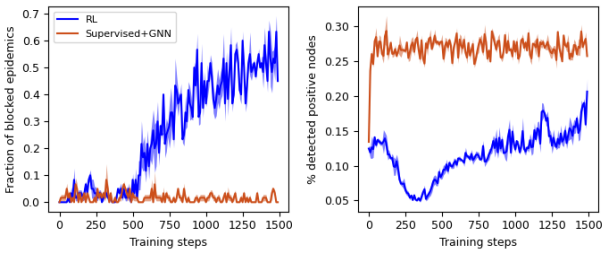

Figure 6. Supervised *vs.* RL with 3-community networks. **Left:** RLGN successfully learns to contain the epidemic 60% of the time (see containment definition in Appendix D), while SL fails. **Right:** SL isolates many more infected nodes, but less important ones.

diffusion GNN training converges faster (Fig. S3). Presumably, because it models the process more closely to the true spreading.

Appendix D contains additional ablation studies of key elements in our framework: (A) Our score-to-probability function outperforms the popular softmax distribution and escort transform. (B) Internal state normalization in scale-free networks accelerates training substantially.

5.2. Influence Maximization

Baselines. Unlike the epidemic test prioritization, in this problem there is no supervised signal; there is no immediate feedback that may be used for supervision. We compare our RLGN framework against the state-of-the-art scalable algorithms. *(1) LIR* (Liu et al., 2017) is an algorithm for top-k ranking for the IM problem. It was shown to achieve similar performance to MC based methods. *(2) LIR (filtered):* LIR was designed for a fully observable setup. We extend this algorithm to a partially observed setup and filter out nodes with an identified *influenced* neighbor. The motivation is that it is likely that such nodes are already influenced or likely to be influenced soon. *(3) Degree discounted* (Chen et al., 2009) is a topology-based algorithm that was shown to achieve a state-of-the-art performance on some networks, and *w*as recently extended to temporal graphs (Murata & Koga, 2018). *(4) Degree Centrality* and *(5) Eigenvector Centrality*, defined previously, were also used extensively (Lei et al., 2015; Chen et al., 2014; Bozorgi et al., 2016).

Results. We have compared RLGN against the aforementioned baselines on the real-world datasets in Table 3. We included an additional (CA-HEPTh) network that was frequently used as a benchmark for this problem.

Table 3 shows that RLGN performs remarkably in this domain as well. It achieves state-of-the-art performance, often with a considerable gap. Additional experiments and experimental details appear in Appendix D.

6. Previous work

Deep Learning on graphs. Graph neural networks (GNNs) are deep neural networks that can process graph-structured data (Sperduti, 1993; 1994; Sperduti & Starita, 1997; Pollack, 1990; Küchler & Goller, 1996; Kipf & Welling, 2016; Gilmer et al., 2017; Duvenaud et al., 2015; Hamilton et al., 2017; Veličković et al., 2017). Several works combine recurrent mechanisms with GNNs to learn temporal graph data, (Liu et al., 2019; Rossi et al., 2020; Liu & Zhou, 2020; Pareja et al., 2019). Further information can be found in (Kazemi et al., 2020).

Ranking on graphs. The problem of ranking on graphs is a fundamental CS problem. It has various applications such as web page ranking (Page et al., 1999; Agarwal, 2006) and knowledge graph search (Xiong et al., 2017).

Reinforcement learning and graphs studies can be split into two main categories: leveraging graph structure for general RL problems (Zhang et al., 2018a; Jiang et al., 2018), and applying RL methods for graph problems. Our work falls into the latter. An important line of work uses RL to solve NP-hard combinatorial optimization problems on graphs. (Zhu et al., 2019; Dai et al., 2017; Wei et al., 2021).

Manipulation of dynamic processes on graphs. The problem of node manipulation (e.g., vaccination) for controlling epidemic processes on graphs was intensively studied (Hoffmann et al., 2020; Medlock & Galvani, 2009). This problem is often addressed in the setup of the fire-fighter problem and its extensions (Finbow & Macgillivray, 2009; Tennenholtz et al., 2017; Sambaturu et al., 2020). Other work considered the problem of vaccination assignments, and cast this problem into a minimal cover problem (Wang et al., 2020; Song et al., 2020; Wijayanto & Murata, 2019). Other common approaches include developing centrality measures designed to highlight bottleneck nodes (Yang et al., 2020), or using spectral methods for allocating resources (Saha et al., 2015; Preciado et al., 2014; Ogura & Preciado, 2017). Alternative line of research (Miller & Hyman, 2007; Cohen et al., 2002) developed heuristics for the same task.

In most previous work setups a single decision is taken. In our multi-round setup, the agent performs a sequential decision making. The agent needs to balance between retrieving

Table 3. Influence Maximization: Mean percentile of influenced nodes after 15 steps .

	CA-GrQc	Montreal	Enron	GEMSEC-RO	CA-HEPTh
LIR (Liu et al., 2017)	7.3 ± 0.3	86.2 ± 0.7	29 ± 0.3	0.25 ± 0.02	$\mathbf{9.2 \pm 0.3}$
LIR (filtered)	8.0 ± 0.2	$\mathbf{86.4 \pm 0.7}$	28.8 ± 0.3	0.22 ± 0.02	8.5 ± 0.3
Degree	8.4 ± 0.2	85.5 ± 0.8	$\mathbf{31.6 \pm 0.6}$	0.07 ± 0.01	$\mathbf{9.2 \pm 0.3}$
Degree Discounted (Murata & Koga, 2018)	8.7 ± 0.2	85.6 ± 0.7	26.7 ± 0.6	0.05 ± 0.01	8.4 ± 0.2
Eigenvector (Bozorgi et al., 2016)	8.3 ± 0.2	82.9 ± 0.8	$\mathbf{31.8 \pm 0.5}$	0.07 ± 0.01	2.2 ± 0.2
RLGN (ours)	$\mathbf{10.2 \pm 0.6}$	$\mathbf{87.4 \pm 0.5}$	$\mathbf{31.3 \pm 0.6}$	$\mathbf{5.8 \pm 0.3}$	$\mathbf{9.1 \pm 0.5}$

information (for better informed future decisions), maximizing the probability that the intervention will be successful, and optimizing the long-term goal.

Influence Maximization, (IM) is a canonical optimization problem of dynamical processes on graphs. IM was first presented in (Kempe et al., 2003), and proved to be NP-Hard and hard to approximate. Key approximation algorithms were derived in (Goyal et al., 2011; Nguyen et al., 2016), but since they do not scale to large graphs, many alternative heuristics were developed (Murata & Koga, 2018; Liu et al., 2017). For surveys, see Banerjee et al. (2020); Li et al. (2018). Multi-armed Bandit was used for estimating model parameters (Vaswani et al., 2017; Lei et al., 2015). The IM formulation was extended to a multi-round framework by Lin et al. (2015). Chung et al. (2019); Tian et al. (2020); Lin et al. (2015) used RL to find the optimal combination of heuristics from a short list of hand-designed features.

These approaches are limited by the small number of preselected heuristics and by the problem-specific, hand-crafted features. In contrast, our approaches is general and it is not limited to reweighting or a predefined subset of policies, neither uses hand-designed, problem-specific features. Our agent learns a policy from scratch and uses GNNs to generalize to different domains (Yehudai et al., 2021).

7. Conclusions

This paper shows that combining RL with GNNs provides a powerful approach for controlling diffusive processes on graphs. Our approach handles an exponential state space, combinatorial action space and partial observability, and achieves superior performance on challenging tasks on large, real-world networks.

The approach and model discussed in this paper can be applied to important problems other than epidemic control and influence maximization. For example, fake news can be maliciously distributed, and spread over the network. A decision maker can verify the authenticity of items, but only verify a limited number of items per a time period. The objective would be to minimize the total number of nodes that observe fake items.

Our approach assumes that a decision taken by considering only k hops neighborhood of each node is a fairly good approximation to the optimal policy which takes into account the whole graph. If long range correlations exists, this may deteriorate performance. As such, it is sufficient to train our model on small graphs and infer on a larger graph. An interesting question is the ability of our approach to address edge cases that may result from this training protocol, and the generalization ability of our model as a function of long-range correlations in the data.

A key concern for real world application is privacy preservation of individual nodes. Our approach requires local aggregated information about the node's neighborhood, compared to other approaches (e.g., (Kempe et al., 2003; Yang et al., 2020) which required detailed information on the complete graph. Furthermore, recent papers (Zhou et al., 2021) have shown that it is possible to use graph neural network while preserving privacy, and we leave it for future research to apply such approaches in this setup.

References

Abbe, E. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

Agarwal, S. Ranking on graph data. In *Proceedings of the 23rd international conference on Machine learning*, pp. 25–32, 2006.

Banerjee, S., Jenamani, M., and Pratihar, D. K. A survey on influence maximization in a social network. *Knowledge and Information Systems*, 62(9):3417–3455, sep 2020. ISSN 02193116. doi: 10.1007/s10115-020-01461-4. URL https://link.springer.com/article/10.1007/s10115-020-01461-4.

Barabási, A. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, oct 1999. ISSN 00368075. doi: 10.1126/science.286.5439.509. URL http://www.sciencemag.org/content/286/5439/509.abstract.

Bozorgi, A., Haghighi, H., Sadegh Zahedi, M., and Rezvani, M. INCIM: A community-based algorithm for influence

maximization problem under the linear threshold model. *Information Processing and Management*, 52(6):1188–1199, nov 2016. ISSN 03064573. doi: 10.1016/j.ipm.2016.05.006.

Chen, W., Wang, Y., and Yang, S. Efficient influence maximization in social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 199–207, New York, New York, USA, 2009. ACM Press. ISBN 9781605584959. doi: 10.1145/1557019.1557047. URL http://portal.acm.org/citation.cfm?doid=1557019.1557047.

Chen, Y. C., Zhu, W. Y., Peng, W. C., Lee, W. C., and Lee, S. Y. CIM: Community-based influence maximization in social networks. *ACM Transactions on Intelligent Systems and Technology*, 5(2):1–31, apr 2014. ISSN 21576912. doi: 10.1145/2532549. URL https://dl.acm.org/doi/10.1145/2532549.

Chung, T.-Y., Citi, A. S., Taipei, T., Khurshed, A., Wang, C.-Y., and Ali, K. Deep Reinforcement Learning-based Approach to Tackle Competitive Influence Maximization. Technical report, 2019. URL https://doi.org/10.475/123{_}4.

Cohen, R., Havlin, S., and Ben-Avraham, D. Efficient Immunization Strategies for Computer Networks and Populations. *Physical Review Letters*, 91(24), jul 2002. doi: 10.1103/PhysRevLett.91.247901. URL http://arxiv.org/abs/cond-mat/0207387http://dx.doi.org/10.1103/PhysRevLett.91.247901.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. Learning Combinatorial Optimization Algorithms over Graphs. *Advances in Neural Information Processing Systems*, 2017-Decem:6349–6359, apr 2017. URL http://arxiv.org/abs/1704.01665.

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

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Eubank, S., Guclu, H., Kumar, V. S., Marathe, M. V., Srinivasan, A., Toroczkai, Z., and Wang, N. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180–184, may 2004. ISSN 00280836. doi: 10.1038/nature02541. URL www.nature.com/nature.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Finbow, S. and Macgillivray, G. The Firefighter Problem: A survey of results, directions and questions. *The Australasian Journal of Combinatorics [electronic only]*, 43, 2009.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Goyal, A., Lu, W., and Lakshmanan, L. V. CELF++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web, WWW 2011*, pp. 47–48, New York, New York, USA, 2011. ACM Press. ISBN 9781450305181. doi: 10.1145/1963192.1963217. URL http://portal.acm.org/citation.cfm?doid=1963192.1963217.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems*, 2017-Decem:1025–1035, jun 2017. URL http://arxiv.org/abs/1706.02216.

Herrera, J. L., Srinivasan, R., Brownstein, J. S., Galvani, A. P., and Meyers, L. A. Disease Surveillance on Complex Social Networks. *PLOS Computational Biology*, 12(7):e1004928, jul 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004928. URL https://dx.plos.org/10.1371/journal.pcbi.1004928.

Hoen, A. G., Hladish, T. J., Eggo, R. M., Lenczner, M., Brownstein, J. S., and Meyers, L. A. Epidemic wave dynamics attributable to urban community structure: A theoretical characterization of disease transmission in a large network. *Journal of Medical Internet Research*, 17(7):e169, jul 2015. ISSN 14388871. doi: 10.2196/jmir.3720. URL https://www.jmir.org/2015/7/e169/.

Hoffmann, J., Jordan, M., and Caramanis, C. Quarantines as a Targeted Immunization Strategy. aug 2020. URL http://arxiv.org/abs/2008.08262.

Jiang, J., Dun, C., Huang, T., and Lu, Z. Graph Convolutional Reinforcement Learning. oct 2018. URL http://arxiv.org/abs/1810.09202.

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., and Poupart, P. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

Kempe, D., Kleinberg, J., and Tardos, É. Maximizing the spread of influence through a social network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146, New York, New York, USA, 2003. ACM Press. doi: 10.1145/956750.956769. URL http://portal.acm.org/citation.cfm?doid=956750.956769.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Küchler, A. and Goller, C. Inductive learning in symbolic domains using structure-driven recurrent neural networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1137, pp. 183–197. Springer Verlag, 1996. ISBN 3540617086. doi: 10.1007/3-540-61708-6_60. URL https://link.springer.com/chapter/10.1007/3-540-61708-6{_}60.

Lei, S., Maniu, S., Mo, L., Cheng, R., and Senellart, P. Online influence maximization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 2015-August, pp. 645–654, New York, NY, USA, aug 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783271. URL https://dl.acm.org/doi/10.1145/2783258.2783271.

Leskovec, J., Kleinberg, J., and Faloutsos, C. Graph evolution. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2–es, mar 2007. ISSN 15564681. doi: 10.1145/1217299.1217301. URL http://dl.acm.org/citation.cfm?id=1217299.1217301.

Li, Y., Fan, J., Wang, Y., and Tan, K. L. Influence Maximization on Social Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1852–1872, oct 2018. ISSN 15582191. doi: 10.1109/TKDE.2018.2807843.

Lin, S. C., Lin, S. D., and Chen, M. S. A learning-based framework to handle multi-round multi-party influence maximization on social networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 2015-Augus, pp. 695–704, New York, NY, USA, aug 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783392. URL https://dl.acm.org/doi/10.1145/2783258.2783392.

Liu, D., Jing, Y., Zhao, J., Wang, W., and Song, G. A Fast and Efficient Algorithm for Mining Top-k Nodes in Complex Networks. *Scientific Reports*, 7(1):1–8, feb 2017. ISSN 20452322. doi: 10.1038/srep43330. URL www.nature.com/scientificreports.

Liu, Z. and Zhou, D. Towards fine-grained temporal network representation via time-reinforced random walk. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Liu, Z., Zhou, D., and He, J. Towards explainable representation of time-evolving graphs via spatial-temporal graph attention networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2137–2140, 2019.

Lopez, L. and Rodo, X. A Modified SEIR Model to Predict the COVID-19 Outbreak in Spain and Italy: Simulating Control Scenarios and Multi-Scale Epidemics. *SSRN Electronic Journal*, pp. 2020.03.27.20045005, apr 2020. ISSN 1556-5068. doi: 10.1101/2020.03.27.20045005. URL https://doi.org/10.1101/2020.03.27.20045005.

Medlock, J. and Galvani, A. P. Optimizing influenza vaccine distribution. *Science*, 325 (5948):1705–1708, sep 2009. ISSN 00368075. doi: 10.1126/science.1175570. URL https://science.sciencemag.org/content/325/5948/1705https://science.sciencemag.org/content/325/5948/1705.abstract.

Mei, J., Xiao, C., Dai, B., Li, L., Szepesvári, C., and Schuurmans, D. Escaping the Gravitational Pull of Softmax. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

Meirom, E. A., Milling, C., Caramanis, C., Mannor, S., Shakkottai, S., and Orda, A. Localized epidemic detection in networks with overwhelming noise. *submitted to ACM SIGMETRICS 2015*, 2015.

Meirom, E. A., Caramanis, C., Mannor, S., Orda, A., and Shakkottai, S. Detecting Cascades from Weak Signatures. *IEEE Transactions on Network Science and Engineering*, 5(4):313–325, oct 2018. ISSN 2334-329X. doi: 10.1109/TNSE.2017.2764444.

Miller, J. C. and Hyman, J. M. Effective vaccination strategies for realistic social networks. *Physica A: Statistical Mechanics and its Applications*, 386(2):780–785, 2007. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2007.08.054. URL http://www.sciencedirect.com/science/article/pii/S037843710700948X.

Moshiri, N. The dual-Barabasi-Albert model. oct 2018. URL http://arxiv.org/abs/1810.10538.

Murata, T. and Koga, H. Extended methods for influence maximization in dynamic networks. *Computational Social Networks*, 5(1):8, dec 2018. ISSN 21974314. doi: 10.1186/s40649-018-0056-8. URL https://computationalsocialnetworks.springeropen.com/articles/10.1186/s40649-018-0056-8.

Newman, M. Networks: An Introduction. may 2010. URL http://dl.acm.org/citation.cfm?id=1809753.

Nguyen, H. T., Thai, M. T., and Dinh, T. N. Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 26-June-20:695–710, may 2016. URL http://arxiv.org/abs/1605.07990.

Ogura, M. and Preciado, V. M. Optimal Containment of Epidemics in Temporal and Adaptive Networks. pp. 241–266. Springer, Singapore, 2017. doi: 10.1007/978-981-10-5287-3_11. URL https://link.springer.com/chapter/10.1007/978-981-10-5287-3{_}11.

Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. B., and Leiserson, C. E. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. feb 2019. URL http://arxiv.org/abs/1902.10191.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Pollack, J. B. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105, nov 1990. ISSN 00043702. doi: 10.1016/0004-3702(90)90005-K.

Preciado, V. M., Zargham, M., Enyioha, C., Jadbabaie, A., and Pappas, G. J. Optimal resource allocation for network protection against spreading processes. *IEEE Transactions on Control of Network Systems*, 1(1):99–108, mar 2014. ISSN 23255870. doi: 10.1109/TCNS.2014.2310911.

Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., and Bronstein, M. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Rossi, R. A. and Ahmed, N. K. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*, 2015. URL http://networkrepository.com.

Rozemberczki, B., Davies, R., Sarkar, R., and Sutton, C. GEMSEC: Graph Embedding with Self Clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pp. 65–72. ACM, 2019.

Saha, S., Adiga, A., Prakash, B. A., and Vullikanti, A. K. S. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *SIAM International Conference on Data Mining 2015, SDM 2015*, pp. 568–576. Society for Industrial and Applied Mathematics Publications, 2015. ISBN 9781510811522. doi: 10.1137/1.9781611974010.64. URL https://epubs.siam.org/page/terms.

Salathé, M. and Jones, J. H. Dynamics and control of diseases in networks with community structure. *PLoS Computational Biology*, 6(4), apr 2010. ISSN 1553734X. doi: 10.1371/journal.pcbi.1000736.

Sambaturu, P., Adhikari, B., Prakash, B. A., Venkatramanan, S., and Vullikanti, A. Designing Effective and Practical Interventions to Contain Epidemics. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pp. 1187–1195, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. jul 2017. URL http://arxiv.org/abs/1707.06347.

Song, S., Zong, Z., Li, Y., Liu, X., and Yu, Y. Reinforced Epidemic Control: Saving Both Lives and Economy. *arXiv*, aug 2020. URL http://arxiv.org/abs/2008.01257.

Sperduti, A. Encoding Labeled Graphs by Labeling RAAM. Technical report, 1993.

Sperduti, A. Labelling Recursive Auto-associative Memory. *Connection Science*, 6(4):429–459, 1994. ISSN 13600494. doi: 10.1080/09540099408915733. URL https://www.tandfonline.com/doi/abs/10.1080/09540099408915733.

Sperduti, A. and Starita, A. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997. ISSN 10459227. doi: 10.1109/72.572108.

Tennenholtz, G., Caramanis, C., and Mannor, S. The Stochastic Firefighter Problem. nov 2017. URL http://arxiv.org/abs/1711.08237.

Tian, S., Mo, S., Wang, L., and Peng, Z. Deep Reinforcement Learning-Based Approach to Tackle Topic-Aware Influence Maximization. *Data Science and Engineering*, 5(1):1–11, mar 2020. ISSN 23641541. doi: 10.1007/s41019-020-00117-1. URL https://doi.org/10.1007/s41019-020-00117-1.

Vaswani, S., Kveton, B., Wen, Z., Ghavamzadeh, M., Lakshmanan, L. V. S., and Schmidt, M. Model-Independent Online Learning for Influence Maximization. Technical report, jul 2017. URL http://proceedings.mlr.press/v70/vaswani17a.html.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. pp. 1–12, 2017. URL http://arxiv.org/abs/1710.10903.

Wang, B., Sun, Y., Duong, T. Q., Nguyen, L. D., and Hanzo, L. Risk-Aware Identification of Highly Suspected COVID-19 Cases in Social IoT: A Joint Graph Theory and Reinforcement Learning Approach. *IEEE Access*, 8:115655–115661, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.3003750.

Wei, Y., Zhang, L., Zhang, R., Si, S., Zhang, H., and Carin, L. Reinforcement Learning for Flexibility Design Problems. jan 2021. URL http://arxiv.org/abs/2101.00355.

Wells, C. R., Klein, E. Y., and Bauch, C. T. Policy Resistance Undermines Superspreader Vaccination Strategies for Influenza. *PLoS Computational Biology*, 9(3):1002945, 2013. ISSN 1553734X. doi: 10.1371/journal.pcbi.1002945. URL www.nserc-crsng.gc.ca/.

Wijayanto, A. W. and Murata, T. Learning Adaptive Graph Protection Strategy on Dynamic Networks via Reinforcement Learning. In *Proceedings - 2018 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2018*, pp. 534–539. Institute of Electrical and Electronics Engineers Inc., jan 2019. ISBN 9781538673256. doi: 10.1109/WI.2018.00-41.

Xiong, C., Power, R., and Callan, J. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, pp. 1271–1279, 2017.

Yang, S., Senapati, P., Wang, D., Bauch, C. T., and Fountoulakis, K. Targeted Pandemic Containment Through Identifying Local Contact Network Bottlenecks. jun 2020. URL http://arxiv.org/abs/2006.06939.

Yehudai, G., Fetaya, E., Meirom, E., Chechik, G., and Maron, H. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*. PMLR, 2021.

Zhang, K., Yang, Z., Liu, H., Zhang, T., and Başar, T. Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. *35th International Conference on Machine Learning, ICML 2018*, 13:9340–9371, feb 2018a. URL http://arxiv.org/abs/1802.08757.

Zhang, Y., Vuong, Q. H., Song, K., Gong, X.-Y., and Ross, K. W. Efficient Entropy for Policy Gradient with Multidimensional Action Space. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*, jun 2018b. URL http://arxiv.org/abs/1806.00589.

Zhou, J., Chen, C., Zheng, L., Wu, H., Wu, J., Zheng, X., Wu, B., Liu, Z., and Wang, L. Vertically federated graph neural network for privacy-preserving node classification, 2021.

Zhu, S., Ng, I., and Chen, Z. Causal Discovery with Reinforcement Learning. jun 2019. URL http://arxiv.org/abs/1906.04477.