Controlling COVID-19 Spreading using Probablistic Graphical Models with Reinforcement Learning and Graphical Neural Networks

Guojin Tang (gt1188); Xiaocheng Li (xl3119); Fanghao Zhong (fz477); Denglin Jiang (dj1369)

Abstract

The emergence of COVID-19 brings up a new topic within the machine learning (ML) community: how can we utilized ML tools to perform effective disease control. In this paper, we review a recent work utilizing reinforcement learning and graph neural network to perform adaptive disease control. We find that incorporating graph information yields better predictions of probability of getting infected, and utilizing reinforcement learning network helps evaluate strategic importance over nodes, leading to higher change of stopping the spread of epidemic. However, the setting of the task still fails to mention several real world cases, which could lead to our future research on adaptive control in a more complicated setting.

1 Introduction

Covid-19 shapes every aspect of our lives, so our team selects the topic of controlling COVID-19 spreads by graphical modeling. In this paper, we did a comprehensive review of the work on this problem (Meirom et al., 2020), which tried to solve one question: in fighting COVID spreading, how can we rank candidates and prioritize testing to prevent the disease from spreading? This question seems a bit outdated at current stage where the disease is spreading so fast and wide, but if people know the answer to this question early on, things might be different. So this is really a worth studying problem.

If we view this problem from a probablistic graphical model perspective, the question becomes: how to control a diffusive process on a temporally evolving graph?

Three things need to be solved for this question: **1)**How to form the problem using mathematical modeling? We think of each person as a node, and if there's contact between two people, there would be an edge. This way, COVID problem becomes a partially-observed, because we don't have all the information, temporal graph, because the interpersonal contact changes over time. **2)**Then we need to transform the problem: The ultimate goal would be to select a subset of nodes to test. **3)** Finally, we achieve this by designing a ranking algorithm using an actor-critic Reinforcement learning Graphical Neural Network. We will elaborate more on this ranking algorithm in later sections.

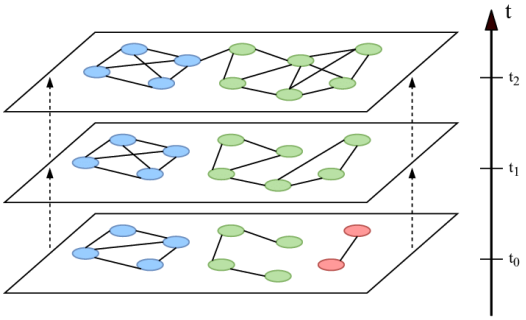


Figure 1: An example of a temporal graph (Lotito and Montresor, 2020), where each node represents a person and an edge exists between two individuals if there's a contact in between.

2 Background

This paper is the first work that applied deep reinforcement learning in the context of a temporally evolving graph. Prior work in deep reinforcement learning, Graphical Neural Networks (GNN) and dynamic graphs are covered in this section.

Dynamic process on graphs Dynamic graphical problems study how graphs change and how we can describe those changes using mathematical expressions. In the context of modern compartmental models, each node may represent a person which has different states, say in SIR

model (Bjørnstad et al., 2002), all possible states would be (susceptible, infected and removed). In SEIR model (Li and Muldowney, 1995), you have not only "susceptible", "infected" and "removed" states but also state "exposed", which is contagious but not yet having symptoms. This is a nasty and tricky property of COVID 19, so this is also the compartmental model the author adopts.

Grahical Neural Networks(GNN) There is a large family of GNN (Scarselli et al., 2008), including graphical convolutional neural network (Tompson et al., 2014) and etc. One simple but interesting example is Message Passing Neural Networks (Gilmer et al., 2020), which operates by repeatedly updating the feature vector of each node by aggregating information from its neighbourhood (Meirom et al., 2020).

Ranking on Graphs A well-known case of ranking on graphs is Google's Page Rank algorithm (Rogers, 2002).

Deep Reinforcement Learning Reinforcement learning (RL) is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them (Sutton and Barto, 2018). Basic elements of RL include: 1) a policy, which defines the learning agent's way of behaving at a given time; 2)a reward function, which defines the goal; 3) a value function, which specifies what is good in the long run and 4) a model of the environment, which is something mimics the behavior of the environment (Sutton and Barto, 2018). This framework is intended to be a simple way of representing essential features of the AI problem (Sutton and Barto, 2018).

Since neural networks are well suited for dealing with high-dimensional sensory inputs (such as times series, frames, etc.) and can be trained incrementally and make use of additional samples obtained as learning happens (François-Lavet et al., 2018), the combination of reinforcement learning and deep learning, Deep RL, becomes popular.

There are mainly three types of RL algorithms. First, the value-based class of algorithms aims to build a value function, which subsequently lets us define a policy. Examples are famous Q-learning, and deep Q-network (DQN) algorithm (Mnih et al., 2015). The second class of algorithms is the model-based approach that relies on a model of the environment (dynamics and reward function) in conjunction with a planning algorithm. The final class of algorithms is policy-based, which aims to optimize a performance objective (typically the expected cumulative reward) by finding a good policy (e.g a neural network parameterized policy). Famous example is Trust Region Optimization (TRPO) (Schulman et al., 2015). The DRL technique this paper adopts is policy-based.

3 A motivating example

Most epidemic models aim to find optimal groups of people that need interventions such as testings and quarantines under limited medical and human resources, since the deployment of these interventions are crucial to control the epidemic from fast spreading.

A natural way of finding the optimal group of people is 1) to predict the probability of each node to get infected and 2) to choose the group of people with the highest probabilities. However, this method is sub-optimal, as we can see from the following example in figure 2.

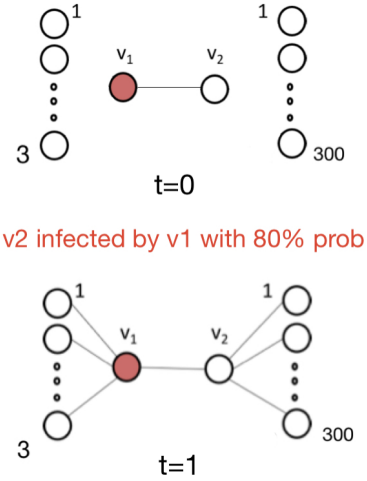


Figure 2: A double star configuration

At time = 0, v_1 is infected and v_2 is not infected. v_2 have interaction with v_1 and we assume that v_2 will be infected by v_1 with 80% probability. Then at time = 1, v_1 interacts with 3 people while v_2 interacts with 300 people. Here we have two testing choices either to test v_1 or to test v_2. If we test v_1, we protect his three neighbors. But v_2 has 80% probability to get infected, so we sacrifice v_2's 300 neighbors with 80% probability. Therefore, the cost of testing v_1 is 300 x 80% = 240. If we test v2, we may protect his 300 neighbors with

80% probability but we may also fail to affect the dynamics with 20% probability. Also we sacrifice v_1's 3 neighbors. So the cost of testing v_2 is 300 x 20% + 3 = 63. Therefore, choosing v1, the most likely infected node, is not optimal since we need to consider the cost of our actions.

Here, we need an algorithm that can take into consideration the strategic importance of a node in a large dynamical process with constant external interventions. This calls for a reinforcement learning framework that can use simulations and knowledge from interplay between dynamics and actions to find the optimal testing trajectory.

4 Problem Formulation

This paper defines a temporal graph G over a group of nodes(people) V characterized with features $\zeta_v(t)$ (e.g, age, sex etc.). The edge $\varepsilon(t)$ between two nodes exists if the nodes interact at time t. Each edge is characterized with interaction features (e.g, duration, distancing etc.) and a transmission probability $p_e(t)$ calculated using the interaction features.

This paper deploys the widely-used SEIR model. Each node can be in one of the following states: Infected, Susceptible, Exposed, Removed. We initially have some infected nodes and each remaining node v is susceptible. The node v will remain susceptible with probability $\prod_{e \in E_v(t)}(1 - p_e(t))$ where $E_v(t)$ contains all edges connecting v with its infected neighbor at t-1. Otherwise the node v will become exposed at time t and this time t is recorded as exposed time T_v. Then the node v will change to infected if $t \geq T_v + D_v$ where D_v is the random variable representing the latency period length. A node is removed only if it is tested and the result is positive.

The objective of this paper's task is to minimize the number of infected people over time. And at each time t, all previous info of interactions and test results are available for the decision maker.

5 Approach

The model combines reinforcement learning (RL) and graph neural network (GNN) (Fig.3). The agent receives inputs including static node features, dynamic node features, and dynamic edge features. Static node features include random node features and topological graph centralities such as betweeness, closeness, eigenvector and degree centralities. Dynamic node features include all test results up to the current time step. Dynamic edge features include the structure of the temporal graph and all previous interactions up to the current time step such as the transmission probability after interaction.

The agent generates scores for each node, and then it will sample K nodes based on nodes' ranking module. These K selected nodes are tested, so the action in reinforcement learning is taking tests. After having new test results for these K nodes, both dynamic node features and dynamic edge features will be updated, and the agent receives new information and will sample another K nodes to be tested.

5.1 Agent Ranking Module

There are four modules within the agent. They are epidemic graph neural network E, information graph neural network I, hidden state update network G, and score module F. These four modules work together to update agent's internal representation for nodes and generate scores used for sampling nodes.

5.1.1 Epidemic GNN E

Epidemic GNN module models the spread of epidemic through point contact at current time step (based on current step's test results). It is a one-layer graph neural network because the epidemic can only spread to the node's immediate neighbors in one time step. Epidemic GNN takes inputs from both the node and its immediate neighbors' static and dynamic node features as well as dynamic edge features between the node and its immediate neighbors. Epidemic GNN outputs a epidemic node feature vector denoted by $e_v(t)$ for the node v:

$$e_v(t) = \sum_{v' \sim_t v} p_{v'v}(t) M_e(\zeta_v(t), \zeta_{v'}(t))$$

where $v^{'}$ is the immediate neighbors for node v at time step t, $p_{v'v}(t)$ represents the probability of transmission during the interaction, and M_e is a multilayer perceptron, modeling the interaction between node v and $v^{'}$ based on their static and dynamic node features. Thus, epidemic GNN mimics the epidemic transition rule and speeds up learning process.

5.1.2 Information GNN I

Information GNN module represents the information state for each node. In this module, it as-

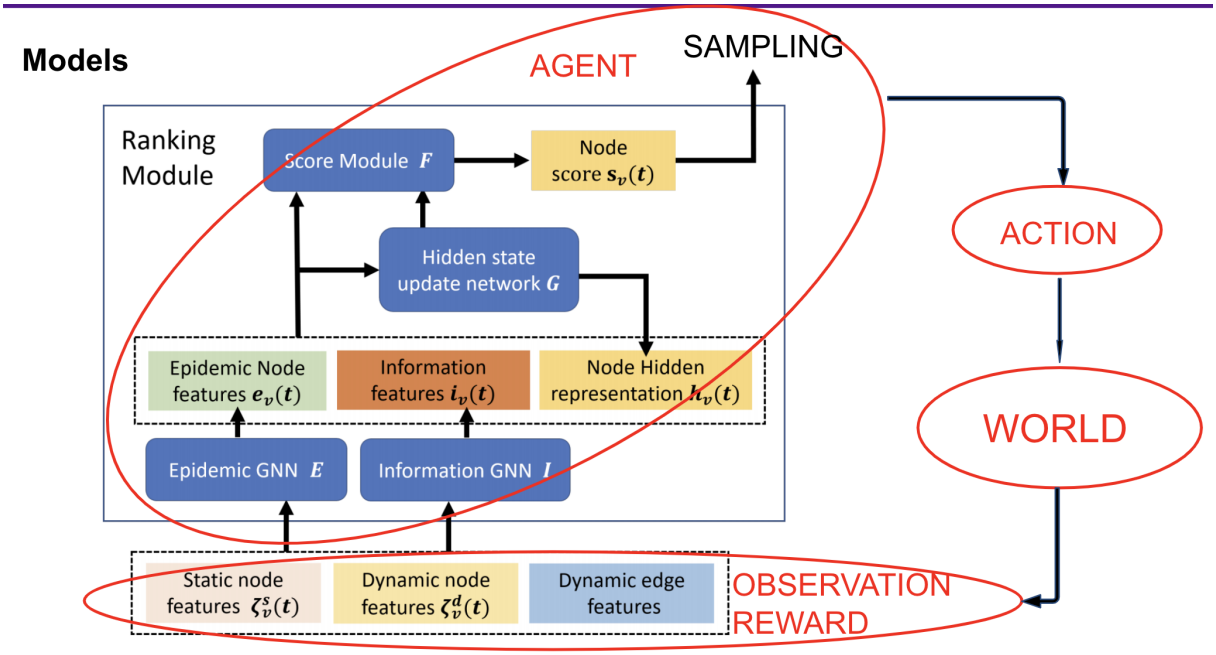


Figure 3: Model

sumes that information can spread long distance in a graph immediately, instead of spreading only to the node's immediate neighbors like in epidemic GNN. For example, if we have a chain of highly dependent nodes (in other words, they are closely interacted), and all of them are untested, then the probability of infection for each node is 50%. However, if we know there is a node that shows infection in the test, then the probability of infection for these nodes will be dramatically changed. This node's immediate neighbors' probability of infection will be increased to close to 1, and all nodes' probability of infection will exceed 50% because they are closely interacted to each other in previous steps. Thus, if we know the information about node u, it will significantly change information about node v which are several steps away from node u.

To better represent information for node v at time step t, information GNN uses a k-layer graph neural network to aggregate all nodes' information that are within k steps away from the node v. To update l^{th}-layer in information GNN, we use

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

where M^l is a multilayer perceptron as before. $x_v^0(t)$ contains both static and dynamic node features for node v at time step t, and $\phi_{v'v}(t)$ contains all edge features information in the previous τ steps (from time $t-\tau$ to t), which includes all interactions and transmission probability during the interaction between node v and v'. It assumes that nodes' information will only depend on nodes' interaction and transmission in previous τ steps. The output for information GNN is $x_v^k(t) = i_v(t)$, which is the node information features aggregated information from neighbors within k hops away.

5.1.3 Hidden State Update Network G

After updating epidemic node features from epidemic GNN and information features from information GNN, agent will update its internal representation $h_v(t)$ for node v by incorporating information from previous step's internal representation, epidemic node fetures, information features, and node features (including both static and dynamic):

$$h_v(t) = G(h_v(t-1), e_v(t), i_v(t), \zeta_v(t))$$

where G is a function, multilayer perceptron or recurrent neural network, to update the internal representation of node v.

5.1.4 Score Module F

The node v's score will be calculated from both before and after updated internal representation and node features (including both static and dynamic) by a neural network F:

$$s_v(t) = F(h_v(t), h_v(t-1), \zeta_v(t))$$

where F could be another multilayer perceptron. And the scores have encoded the agent's policy.

5.1.5 Sampling

Given the score calculated for each node, the agent will convert the score to probability distribution and then sample K nodes from probability distribution without replacement to test them.

The score can be converted to a probability distribution by a softmax function, Gibbs distribution, or weight method. Both softmax function and Gibbs distribution involves exponential term in conversion equation, which discourages exploration of one side of nodes that have low or high-scores and limits sensitivity to the another side of nodes. Thus, it is better to convert the score to probability distribution by the weight of scores:

$$Pr(a_i) = \frac{x_i^{'}}{\sum x_i^{'}}$$

$$x_i^{'} = x_i - \min_i x_i + \epsilon$$

where $\{x_i\}$ is the set of nodes' score. ϵ is a constant to control the probability difference between low-score and high-score's nodes. If ϵ is large, then each node will be uniformly selected.

After having probability distribution of nodes, agent will sample a node based on this probability distribution. Then the agent will remove this node and re-calculate probability distribution for the remaining nodes, and repeat this process K times until K nodes are selected.

5.2 Design Choice

The reinforcement learning is a policy gradient algorithm, because using an action-value approach will results in an extremely large action space even for selecting a small subset of k nodes out of a small number of n (total) nodes. Given this, we leverage on Proximal Policy Optimization (PPO), an actor-critic algorithm, to optimize our agent.

Because hidden state update network G could be a recurrent neural network (RNN), gradient vanishing or exploding are common issues during the training. It can be solved by using GRU/LSTM to replace RNN or directly normalizing the node's internal representation (output) from RNN. After trying with various normalization method in the experiment, L_2 normalization works best.

6 Experimental Details

The paper (Meirom et al., 2020) perform experiments on two types of data, one type as generated random networks, and the other as real-world data, in order to test model's performance on graphs of a variety of characteristics. Moreover, the paper compare the network based on Reinforcement Learning (RL) and supervised learning (SL) GNN is compared with baseline models including

- simple probablistic models
- rule-based models
- SL-based models
- RL-based models

All models are tested based on two evaluation metrics: the proportion of health individuals in one simulations, and the chance of successfully containing infected percentage under 40%.

6.1 Data

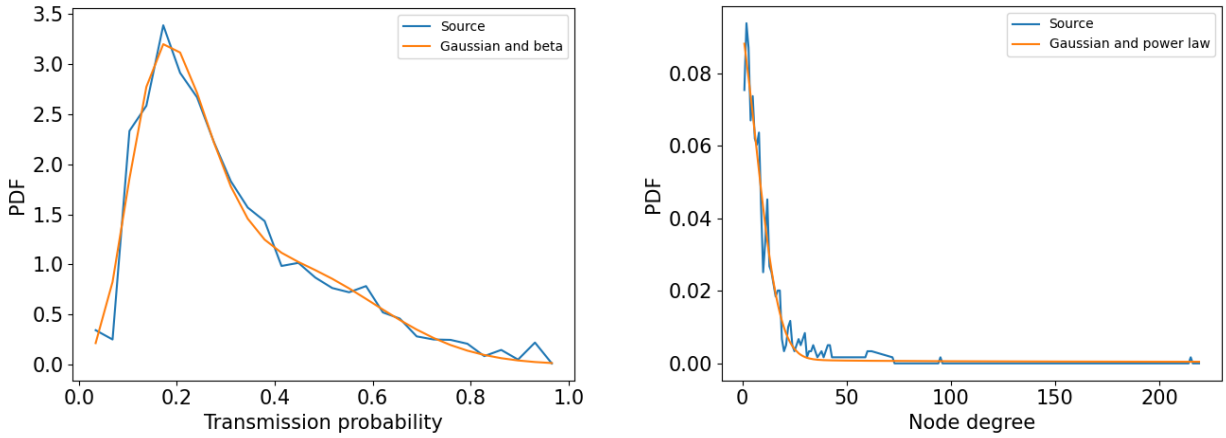
6.1.1 Generated Random Networks

The first type of data contains generated random networks. The paper tests two types of generated random networks, one based on communities and one more densely-connected.

Community-based networks have nodes separated into densely-connected communities, while between communities there are only sparse connections. By testing model performance on community-base networks it might shows how models stops the spread of epidemic within and between communities. However, this type of data is less similar to real-world models of epidemic spreading, where each individual several connections, while the whole graph are more densely-connected with a weak sign of "community". The paper only tests the case of two and three communities.

Preferential attachment (PA) networks are described by a node-degree distribution following a power-law (scale-free), which is more similar to real-world data with more dense connections.

The paper uses dual-Barabási–Albert (DBA) model (Moshiri, 2018). The DBA model has two properties: the expected degree and the degree distribution of networks sampled under the model. It also include statistical properties such as average path length and clustering coefficients. However, the DBA model is not guaranteed to generate power-law degree distributions.



(a) (b)

Figure 4: Statistics of the contract tracing graph (Meirom et al., 2020). (a) The empirical transition probability $P(p_e)$ on a contact tracing network and our suggested curve fit. (b) The degree distribution on the contact tracing network, along with its fit.

6.1.2 Real-world data

Contract-tracing networks are based on the high-level statistical information collected by disease control agencies. It tracks some key properties of COVID-19 spread in April, 2020. However, this data is not publicly available. 4 shows the distribution of the contract-tracing network, in which node degree follows the power-law.

The data is not used directly. Instead, all random networks are generated based on parameters derived from the data. Other types of real-world data include data of social network, academic collaboration, etc.. We skip the introduction of these types of data as properties of these networks might be different from networks of epidemic spread, which makes results on these graphs less convincing.

6.2 Baseline models

Simple probablistic models. Probabilstic models in the paper utilizes tree-based belief propagation to update probability of getting infected in time t. For more details one can visit the Appendix B of the paper. (Meirom et al., 2020).

Rule-based models. Rules are similar to real-world instructions of stopping epidemic spread. Three models are built based on three features repectively, i.e. i) number of known infected nodes in 2-hop neighborhood (neighbors and their neighbors), ii) node degrees, and iii) spectral graph properties like eigenvalue centrality. These are simple versions of real-world instructions in stopping epidemic spread, while always lack adaptivity.

SL-based and RL-based models. SL-based models include vanilla model as a deep neural network (DNN) and a GNN-based model. RL-based models include a vanilla model which doesn't introduce graph structure.

7 Results and Discussion

7.1 Simulated Networks

Tables 1 and 2 shows the results on the community-based networks and PA networks. Clearly the RLGN model outperforms all baseline models on both networks. We then take a closer look into the results by taking a deeper look of the case of 3-community networks 5. The figures suggests that SL-based GNN detects infected nodes more effectively than RLGN (right), it fails to control the epidemic spread by failing to take the strategic importance of nodes (i.e. nodes that connect communities) into account, which is utilized by the RLGN.

The analysis performed by the paper on PA network is on testing model performances on either densely or sparsely connected networks. R_0 coefficient in Figure 6 (a) is the mean number of infected nodes per node. Figure 6 (a) shows that RL has a significant advantage over the SL-based GNN for R_0 between 2.0 and 2.9. Figure 6 (b) shows the depicts a robustness analysis of RLGN for variations in the epidemiological model, showing that the trained model can sustain up to 40%

%CONTAINED	2 COMMUNITIES	3 COMMUNITIES
TREE-BASED MODEL	15 ± 35	0 ± 0
COUNTER MODEL	19 ± 39	1 ± 4
DEGREE CENTRALITY	23 ± 1	0 ± 0
EIGENVECTOR CENTRALITY	19 ± 3	0 ± 0
SUPERVISED (VANILLA)	24 ± 11	2 ± 2
SUPERVISED +GNN	27 ± 10	2 ± 2
SUPERVISED +DEGREE	29 ± 10	1 ± 2
SUPERVISED +GNN+DEG	24 ± 10	2 ± 02
(VANILLA)	66 ± 10	7 ± 5
FULL (OURS)	**88±1**	**53±13**

Table 1: Results from the paper (Meirom et al., 2020). Probability (in %) of containing an epidemic in community-based networks. Each community has 30 densely connected nodes.

	%CONTAINED		%HEALTHY	
	PA	CT	PA	CT
TREE BASED MODEL	1 ± 1	0 ± 0	10 ± 7	11 ± 3
COUNTER MODEL	0 ± 0	0 ± 0	7 ± 7	14 ± 5
DEGREE CENTRALITY	22 ± 4	0 ± 1	30 ± 2	16 ± 1
EIGENVECTOR CENTRALITY	21 ± 1	0 ± 1	30 ± 1	16 ± 1
SL (VANILLA)	2 ± 2	0 ± 0	13 ± 3	17 ± 1
SL + GNN	27 ± 6	15 ± 4	34 ± 3	32±2
SL + DEG	3 ± 3	0 ± 1	15 ± 3	18 ± 1
SL + DEG + GNN	26 ± 5	16 ± 5	33 ± 3	32 ± 1
RL (VANILLA)	2 ± 2	1 ± 1	17 ± 1	16 ± 1
RLGN (OURS)	**78±4**	**45±6**	**52±2**	**40±1**

Table 2: Results from the paper (Meirom et al., 2020). %contained epidemics and %healthy nodes achieved on a preferential attachment (PA) network, and contact tracing (CT) network. In both cases, networks had 200 nodes.

deviation at test time in this key parameter.

7.2 Robustness of Results

In table 3, $2 * n$ denotes the number of nodes, and k denotes available tests proportion with respect to the population. All graphs are generated as PA networks.

As table 3 shows, RLGN outperforms SL-based GNN in all graph sizes and initial infection sizes, given only a limited amount of tests. Especially in the case of high initial infection size, RLGN even outperforms SL with 2 times available tests, while maintaining high disease containment rate, which is more obvious in large-size graph settings.

8 Future Work

We list two cases where might be the shortcomings of the currently developed models, and can be applied as future work. However, the real-world cases are always complicated, for which we can only approximate how the real world works and can hardly understanding it comprehensively. We then propose our possible research idea.

8.1 Other Real-world Cases

One case is detecting fake news and confining them, which is similar to the case of epidemic spread control in objectives. Another case is advertising through graphs, in which the objective turns out to be maximize the spread through the networks.

8.2 Adaptive Control with More Flexibility

The paper only tests networks that are based on a fixed number of tests with stable transmission probability. However, in reality the cases are much more complicated, as number of tests usually comes with a cost in the consideration of numbers, and the transmission probability is subject to change due to other external factors including tem-

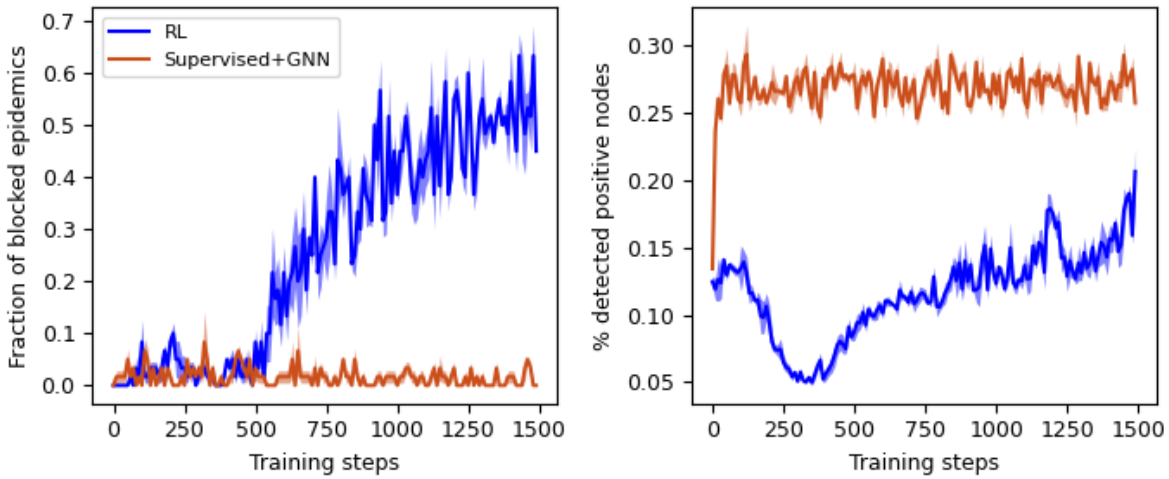


Figure 5: (Meirom et al., 2020) Supervised vs RL with 3-community networks. **Left:** RLGN successfully learns to contain the epidemic 60% of the time, while SL fails. **Right:** SL isolates many more infected nodes, but less important ones.

perature and humidity, and internal factors including possible mutations of the virus itself.

8.3 Our Research Idea

We propose a research idea in an adversarial setting. Usually in today's disease control bureaus rely on social media or advertisements on streaming platforms to send information of the disease and guidance of preventing illness or self-relieving of illness. This often comes with a setting where misinformation could emerge from a variety of sources. Our research idea is to maximize the spread of true information in the setting of the massive spread of misinformation. However, this requires the definition of objectives and a clear descriptions of experimental settings.

9 Conclusions

We review a recently released paper which utilizes frameworks of reinforcement learning and graph neural networks to perform disease control over time. We find that by taking into account the graphical information and the strategic importance of nodes from a sequential decision setting, the model is more likely to yield successful results in "flattening the curve". However, the proposed method failed to mention some more complicated settings in real-world disease control, which could be interesting for us to investigate in the future. At the end, we propose an research idea of maximizing spread of true information in the context of massive spread of misinformation.

Bibliography

Emmanuel Abbe. 2018. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18(177):1–86.

S. Agarwal. 2006. Ranking on graph data. In *ICML '06*.

Ottar N Bjørnstad, Bärbel F Finkenstädt, and Bryan T Grenfell. 2002. Dynamics of measles epidemics: estimating scaling of transmission rates using a time series sir model. *Ecological monographs*, 72(2):169–184.

S. Fang, Qi Zhang, Gaofeng Meng, S. Xiang, and C. Pan. 2019. Gstnet: Global spatial-temporal network for traffic flow prediction. In *IJCAI*.

Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. 2018. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug. PMLR.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2020. Message passing neural networks. In *Machine Learning Meets Quantum Physics*, pages 199–214. Springer.

S. Guo, Youfang Lin, Ning Feng, Chao Song, and H. Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*.

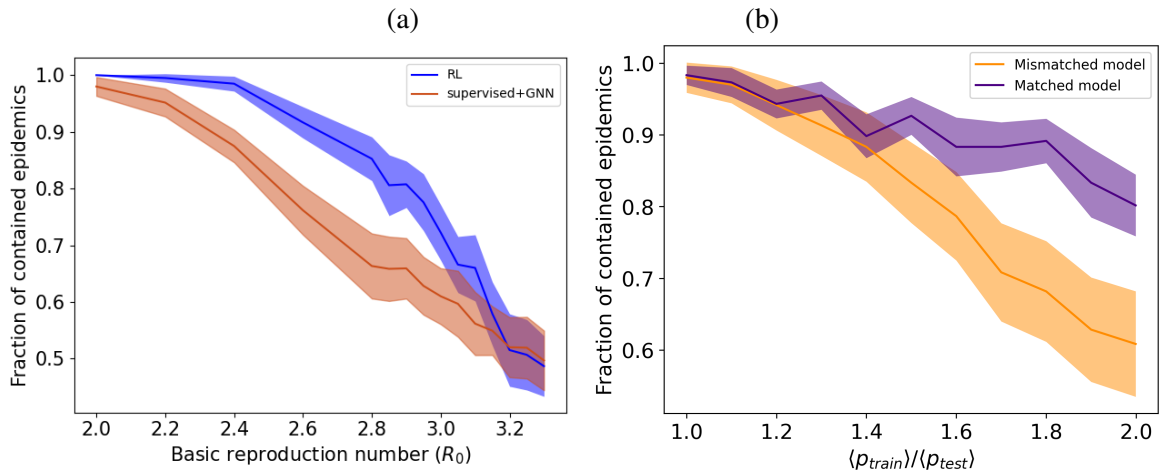


Figure 6: **Stability analysis** (Meirom et al., 2020) **: (a)** The contained epidemic fraction as a function of the basic reproduction number R_0 on a PA network. RLGN outperforms SL over a large range of R_0 values. **(b)** Stability against test-time shift in transmission probability. *Orange:* The performance of RLGN deteriorates when the mean transmission probability at test time is higher more than 40% than train time. *Purple:* As a baseline, training and testing with the same higher transmission probability.

J. Hoffmann, Matt Jordan, and C. Caramanis. 2020. Quarantines as a targeted immunization strategy. *ArXiv*, abs/2008.08262.

Michael Y Li and James S Muldowney. 1995. Global stability for the seir model in epidemiology. *Mathematical biosciences*, 125(2):155–164.

Quintino Francesco Lotito and Alberto Montresor. 2020. Efficient algorithms to mine maximal span-trusses from temporal graphs. *arXiv preprint arXiv:2009.01928*.

J. Medlock and A. Galvani. 2009. Optimizing influenza vaccine distribution. *Science*, 325:1705 – 1708.

Eli A. Meirom, Haggai Maron, Shie Mannor, and Gal Chechik. 2020. How to stop epidemics: Controlling graph dynamics with reinforcement learning and graph neural networks.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Niema Moshiri. 2018. The dual-barabási-albert model.

Masaki Ogura and V. Preciado. 2016. Optimal containment of epidemics in temporal and adaptive networks. *ArXiv*, abs/1612.06832.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1999. The pagerank citation ranking : Bringing order to the web. In *WWW 1999*.

V. Preciado, M. Zargham, Chinwendu Enyioha, A. Jadbabaie, and George J. Pappas. 2014. Optimal resource allocation for network protection against spreading processes. *IEEE Transactions on Control of Network Systems*, 1:99–108.

Ian Rogers. 2002. The google pagerank algorithm and how it works.

Sudip Saha, Abhijin Adiga, B. Prakash, and A. Vullikanti. 2015. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *SDM*.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. 2014. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807.

Chenyan Xiong, R. Power, and J. Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. *Proceedings of the 26th International Conference on World Wide Web*.

$2*n = 300$	Init. infection size 5%		Init. infection size 7.5%		Init. infection size 10%	
	%healthy	%contained	%healthy	%contained	%healthy	%contained
SL, $k = 1\%$	27 ± 2	15 ± 5	21 ± 2	4 ± 2	18 ± 1	1 ± 1
SL, $k = 1.33\%$	41 ± 3	37 ± 6	27 ± 2	12 ± 4	24 ± 2	6 ± 3
SL, $k = 2\%$	66 ± 4	76 ± 6	48 ± 3	55 ± 7	37 ± 2	32 ± 6
RLGN, $k = 1\%$	50 ± 2	78 ± 7	43 ± 2	58 ± 1	40 ± 1	48 ± 6

$2*n = 500$	Init. infection size 5%		Init. infection size 7.5%		Init. infection size 10%	
	%healthy	%contained	%healthy	%contained	%healthy	%contained
SL, $k = 1\%$	24 ± 2	7 ± 4	20 ± 1	2 ± 1	19 ± 1	0 ± 1
SL, $k = 1.6\%$	48 ± 3	54 ± 6	35 ± 2	27 ± 7	29 ± 1	11 ± 1
SL, $k = 2\%$	67 ± 3	83 ± 5	46 ± 2	53 ± 4	38 ± 2	37 ± 7
RLGN, $k = 1\%$	52 ± 1	97 ± 2	44 ± 2	75 ± 11	42 ± 1	66 ± 6

$2*n = 1000$	Init. infection size 5%		Init. Infection size 7.5%		Init. infection size 10%	
	%healthy	%contained	%healthy	%contained	%healthy	%contained
SL, $k = 1\%$	25 ± 2	5 ± 3	21 ± 1	0 ± 1	19 ± 1	0 ± 0
SL, $k = 1.5\%$	42 ± 2	49 ± 6	30 ± 1	10 ± 3	27 ± 1	4 ± 2
SL, $k = 2\%$	66 ± 1	84 ± 5	45 ± 2	59 ± 5	37 ± 1	30 ± 1
RLGN, $k = 1\%$	52 ± 1	97 ± 2	44 ± 2	75 ± 11	42 ± 1	66 ± 6

Table 3: (Meirom et al., 2020) A comparison between RLGN and SL+GNN (the best baseline). RLGN performance is highlighted. The number of additional resources needed to surpass the RLGN performance in a given metric is also highlighted. In many cases, even using SL+GNN with twice as many resources than RLGN performs worse than RLGN. The evaluation was performed on a preferential attachment network with mean degree 2.8. The number of nodes is indicated at the top of each table.

Shenghao Yang, Priyabrata Senapati, Di Wang, C. Bauch, and K. Fountoulakis. 2020. Targeted pandemic containment through identifying local contact network bottlenecks. *ArXiv*, abs/2006.06939.

Ting Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*.

K. Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and T. Başar. 2018. Fully decentralized multi-agent reinforcement learning with networked agents. *ArXiv*, abs/1802.08757.

L. Zhao, Yujiao Song, C. Zhang, Y. Liu, P. Wang, T. Lin, Min Deng, and H. Li. 2020. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21:3848–3858.