How to Stop Epidemics: Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks

By - Eli A. Meirom, Haggai Maron, Shie Mannor, and Gal Chechik
Presented by - Sripath Mishra, Vishnu Devarakonda, Jason Kao, Boya Ouyang

Introduction

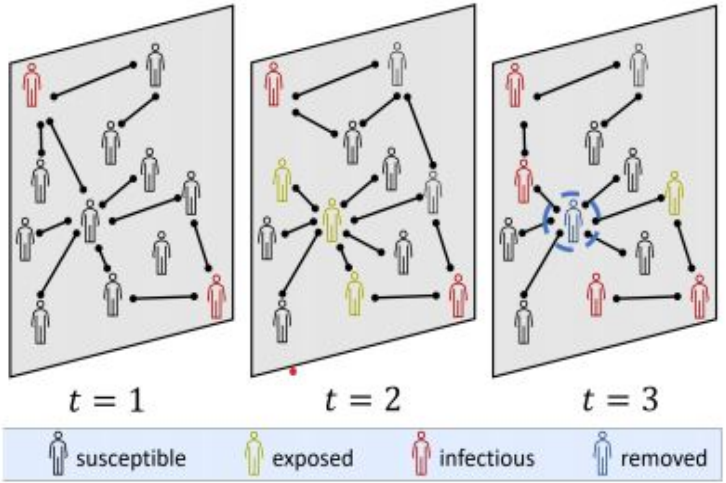
This paper makes the following contributions:

(1) A new framework for controlling the dynamics of diffusive processes over graphs. Namely, learning to perform local interventions to steer the global dynamics of a graph-based dynamical system.

(2) A new architecture for this problem, and a way to train a decision-making agent using reinforcement learning to prioritize interventions on the temporal multi-graph.

(3) An observation of the interplay between the dynamics of graph states and how information flows over the graph for a decision making agent, which motivates the design of our deep network architecture.

(4) A set of benchmarks and strong baselines for this problem. This includes statistics collected from real-world contact tracing data for COVID-19.



Related work

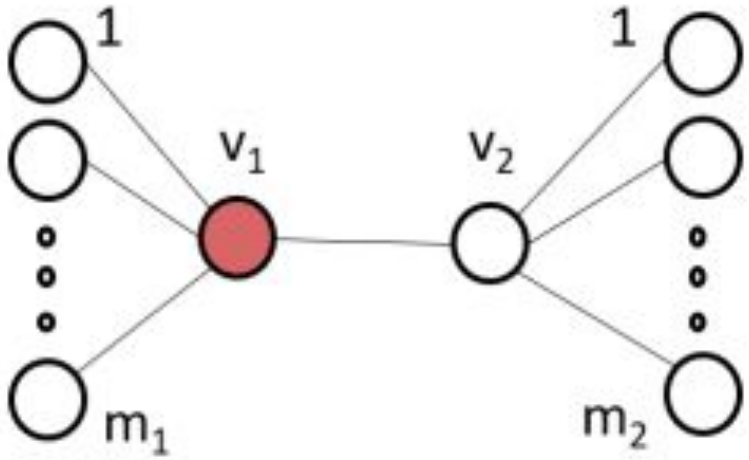
This paper differentiates itself in the following ways:

(1) It does not assume that the infected people can be vaccinated.

(2) It does not assume that the any person can be quarantined without a positive-test result.

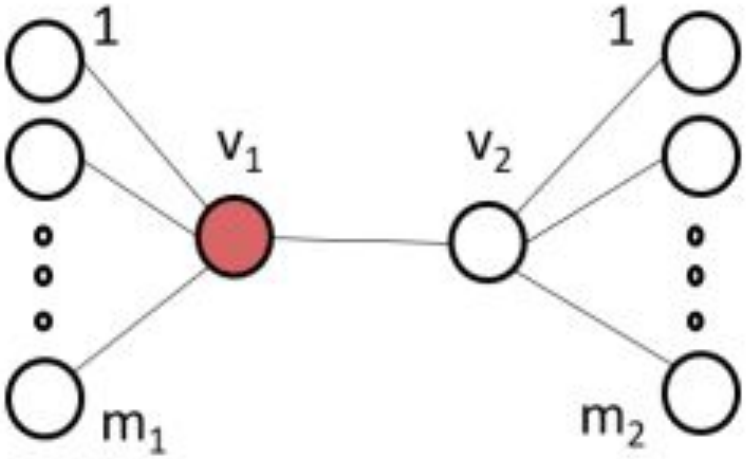
(3) The agent needs to decide at each time step on its testing policy, given the information available at $t \geq 0$

(4) The agent needs to know where to test, but also when to test each node.



A Motivating Example

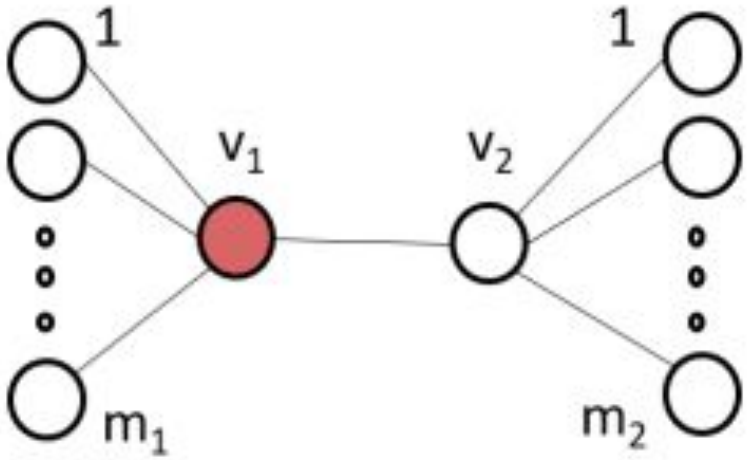
- A natural algorithm would be to simply act on nodes that we know are infected. However, this approach is suboptimal.
- Suppose we have a graph with nodes *u* and *v*. An edge exists between *u* and *v* if they interact at some time *t*.
- Each interaction has a probability of transmission of infection $p_e(t)$.
- Finally, assume that we can test nodes at odd timesteps and infected nodes will be removed from the graph.



In this figure, the state of v2 is unknown at t=1 while v1 is infected at t=0

The Two Scenarios

- Given e = (v1,v2), p(e,t=0)=p. For t>=1, v1 was infected at t=0
- We can either test v1 or v2.
- Test V2: If it's infected, it can be removed at the cost of infecting v1's neighbors. Or, the test is negative (1-p) and the infection spreads. Cost: (1-p)*m2+m1
- Test V1: We block the spread to v1's neighbors but sacrifice all m2 neighbors of v2 with probability p. Cost: p*m2.
- The decision would therefore be to test V2 if 2p>=1+m1/m2.
- This illustrates how an optimal policy must balance two factors.



In this figure, the state of v2 is unknown at t=1 while v1 is infected at t=0

Problem Formulation

- Temporal Graph G(t) = *G(V,E(t))*
- Edge *e* = *(u,v)* exists iff the two nodes interacted at time t.
- *O(x)* - Nodes features(age, sex, etc.) - $\zeta_v(t)$
- *P(x)* - Edge features(duration, distance, environment, etc.) - $\Phi_{uv}(t)$
- The SEIR Model is used:
  - 4 state: susceptible (*S*), exposed/latent (*L*), infectious (*I* state), removed (*R*).
- $ST_v(t)$: State of node *v* at time *t*.
- Define *I(t)*, *L(t), R(t), S(t)*.
- $p_e(t)$ - Transmission probability of edge *e* at time *t*
- Define $E_v(t)$ - impinging edge on node v at time *t*.
- p(node *x* remains healthy, t) = 1 - $\Pi_{e \in Ev(t)} (1-p_e(t))$.
- If infected or exposed then nodes state is set to R.

Problem Formulation

- The objective is to simply minimize the number of infected people over time given by: $\amalg_{STv(t)\in\{L,D\}}$
- Assuming that testing is limited to k tests per day, the optimization goal becomes min $\sum\gamma^t \amalg_{STv(t)\in\{L,D\}}$. Gamma here is the discounting factor.
- The action space consists all possible selection of a subset *a(t)* of k nodes from *V*.
- At time t, the agent is exposed to all interactions between network nodes.
- In additional, partial information on the node states are given.
- At t=0, the agent is given information on the subset of infected nodes.
- At t>0, the agent observes all past test results and observe if a node was infected or not.

Problem Formulation

- Aggregating all neighborhood information into a random variable:

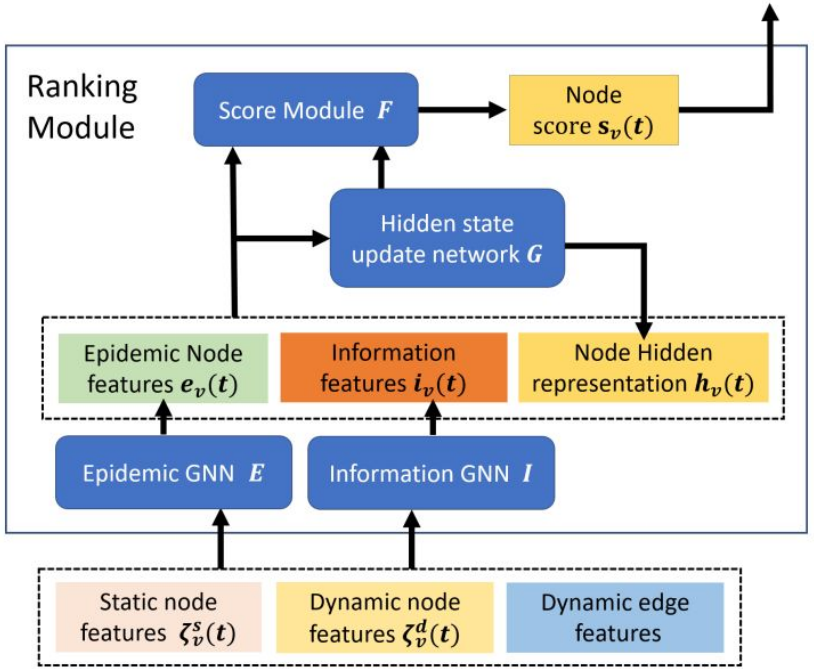
$$N_v(t) = \{(O(t),P(t),ST_u(t)(t-1))|u,e_{vu \in E(t)}\}$$

- $ST_v(t)$ depends on the previous state and on node features.

$$ST_v(t)=f(ST_v(t-1),P(t),N_v(t))$$

- At each timestep we select a subset of k nodes and change the state of the nodes. This affects the global dynamics of the graph. The optimization objective should be invariant to permutation of the nodes. It is assumed that it depends only on the number of nodes in state *I*.
- With this, we have establish the states, actions, “reward” which together define the mathematical framework that constitute a markov decision process.

Network Block Diagram



Input

- Dynamic Node Features: All test results perform up to the current timestep.
- Static Node Features: Betweenness, Closeness, Eigenvector and Degree Centralities.
- Dynamic Edge Features: All interaction up to the current timestep, include the transmission probabilities.

Epidemic GNN

- Single layer is used to model the spread of the epidemic.
- Output is calculated as:

$$e_v(t) = \sum_{u \sim_t v} p_{vv'}(t) \cdot M_e(\zeta_v(t), \zeta_{v'}(t); \theta_{m_e})$$

- $p_{vv'}(t)$ denotes the transmission probability and M_e denotes a multi-layer perceptron.

Information GNN

- Information can propagate in a graph instantaneously which gives rise to the need for a cumulative multi-layer graph.
- Previous interaction can also affect the agent's belief on the state of a node *v*.
- The output is formulated as the following:

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

- M is another MLP and ϕ encapsulates the interaction delay and the transmission probabilities.

Score, Hidden State Update, and Sampling

- The hidden state modeled as either a MLP or a recurrent module like GRU.
- After hidden state update, the node score is calculated using a MLP.
- Sampling is done without replacement and the sampled score is mapped to a probability distribution.

Design Choices

- Action space can be prohibitively large, so a parameterized model is used.
- Critic module is constructed with an architecture similar to the ranking module with a different output layer.
- Softmax score-to-probability is not ideal so the following mapping is used:

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

- L_2 normalization is utilized at each node hidden state.
- Transition probabilities is supplied by the government but could be optional.

Experiments

- Methods from three categories are compared: Programmed Baselines, Supervised Learning (SL), and Reinforcement Learning (RL).
- Programmed baselines fail to utilize all the given information.
- Various combination of supervised learning algorithms are tested.
- Reinforcement learning tested is similar to the algorithm described previous with some differences.

Details and Evaluations

- RL and SL are trained by generating random networks and randomly initialized infected nodes.
- The performance of the model is evaluated by the percentage of healthy nodes at the end and the probability of containing the epidemic.
- These metrics reflects real life objective of “flattening the curve”.

Network Datasets and Model

- Community-based networks: nodes clustered into densely-connected communities and sparse connection across communities.

%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
RLGN (VANILLA)	66 ± 10	7 ± 5
RLGN FULL (OURS)	**88±1**	**53±13**

Table 2: Probability (in %) of containing an epidemic in community-based networks. Each community has 30

Network Datasets and Model

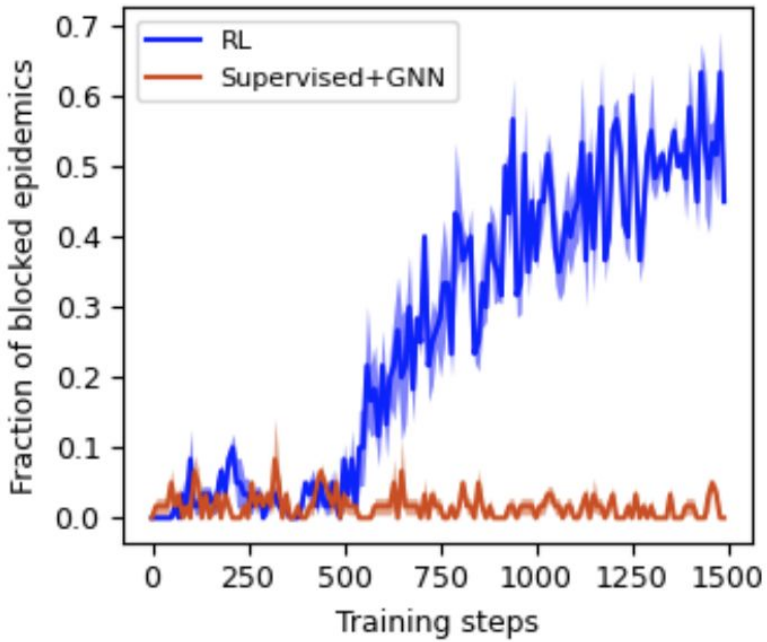
- Preferential Attachment: networks exhibit a node-degree distribution that follows a power-law.
- Contact-tracing Network: anonymized high-level statistical information about real contact tracing networks that included the distribution of node degree, transmission probability and mean number of interactions per day, collected during April 2020.

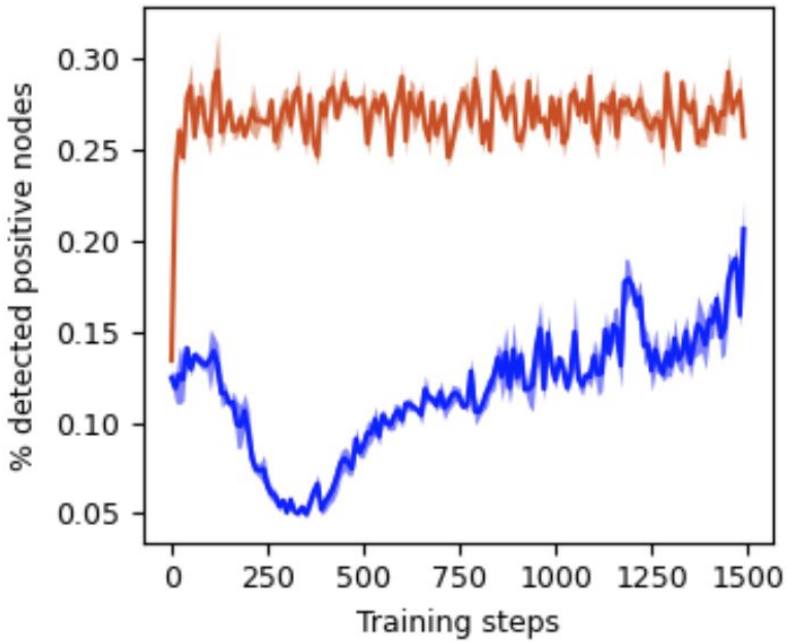
Network Datasets and Model

	%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**

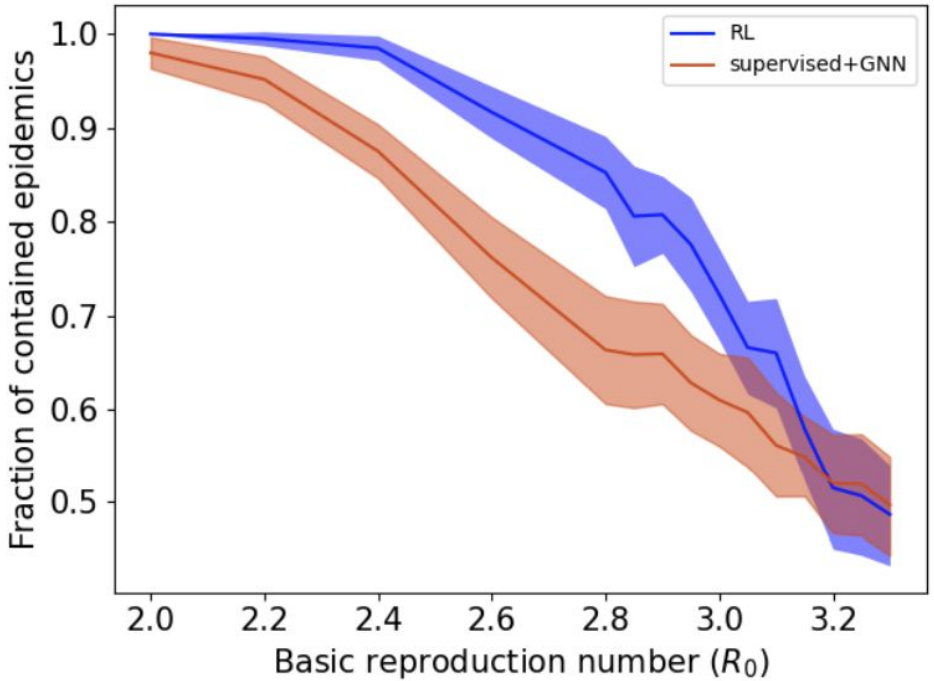
Table 3: %contained epidemics and %healthy nodes achieved on a preferential attachment (PA) network, and contact tracing (CT) network. In both cases, networks had 200 nodes.

Supervised vs RL with 3-community networks

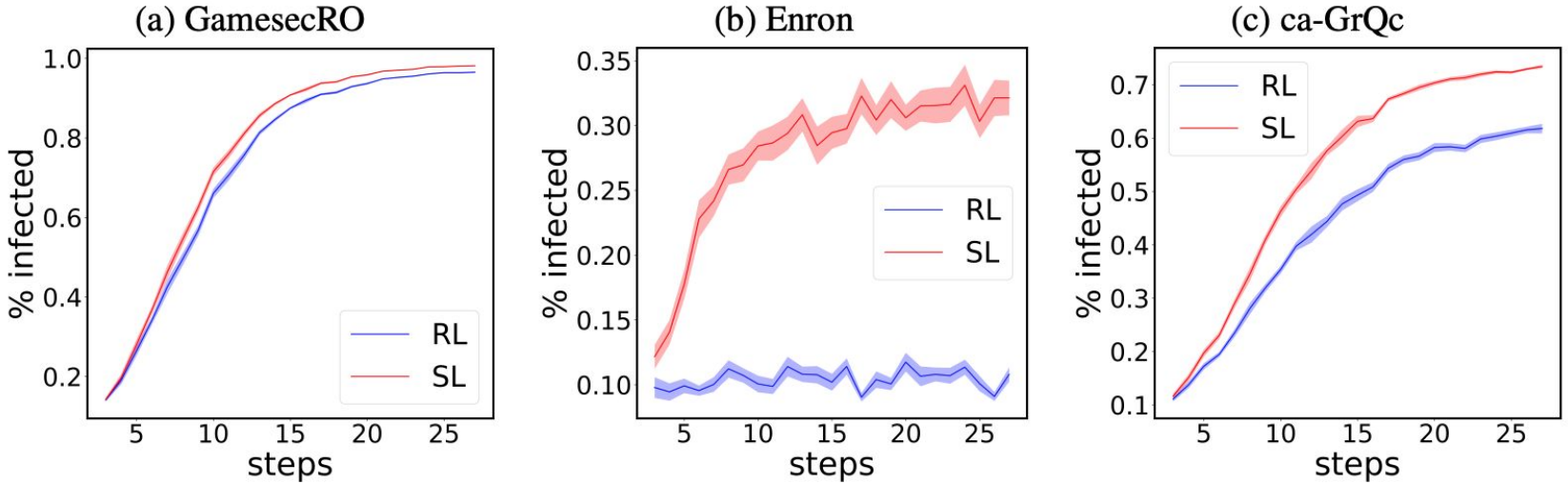




Stability analysis



Inference on large graphs and real-world networks



Comparison between RLGN and SL+GNN

$n = 300$	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

$n = 500$	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

$n = 1000$	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

The mean percentile of healthy nodes after a 20 steps

	CA-GrQc	Montreal	Portland	Enron	GEMSEC-RO
Degree Centrality	25.52 ± 0.01	12.83 ± 0.01	0.67 ± 0.01	71.14 ± 0.02	2.43 ± 0.01
Eigenvector Centrality	25.37 ± 0.01	8.06 ± 0.01	0.04 ± 0.01	55.10 ± 0.02	2.41 ± 0.01
SL	29.79 ± 0.02	23.09 ± 0.03	1.57 ± 0.01	68.45 ± 0.05	4.26 ± 0.01
RLGN	**42.69±0.03**	**39.68±0.03**	**3.71±0.01**	**89.19±0.02**	**6.52±0.01**

Extensions

The approach and model discussed in this paper can be applied to important problems other than epidemic control

- Influence maximization
- Fake news detection and confinement.
- Epidemic Control: Beyond Node Selection

Conclusions

- Combining RL with GNNs provides a powerful approach for controlling spreading processes on graphs.
- An RL+GNN approach allows us to confine the spread of an epidemic that is approximately 30% more contagious (i.e., R0 that is 30% higher) with the same resources as a standard supervised learning-based approach.
- Using RL on temporal graphs can increase the number of healthy people by 25% and contain the epidemic 30% more often than supervised approaches and 2.5× more often than non-learned baselines using the same resources