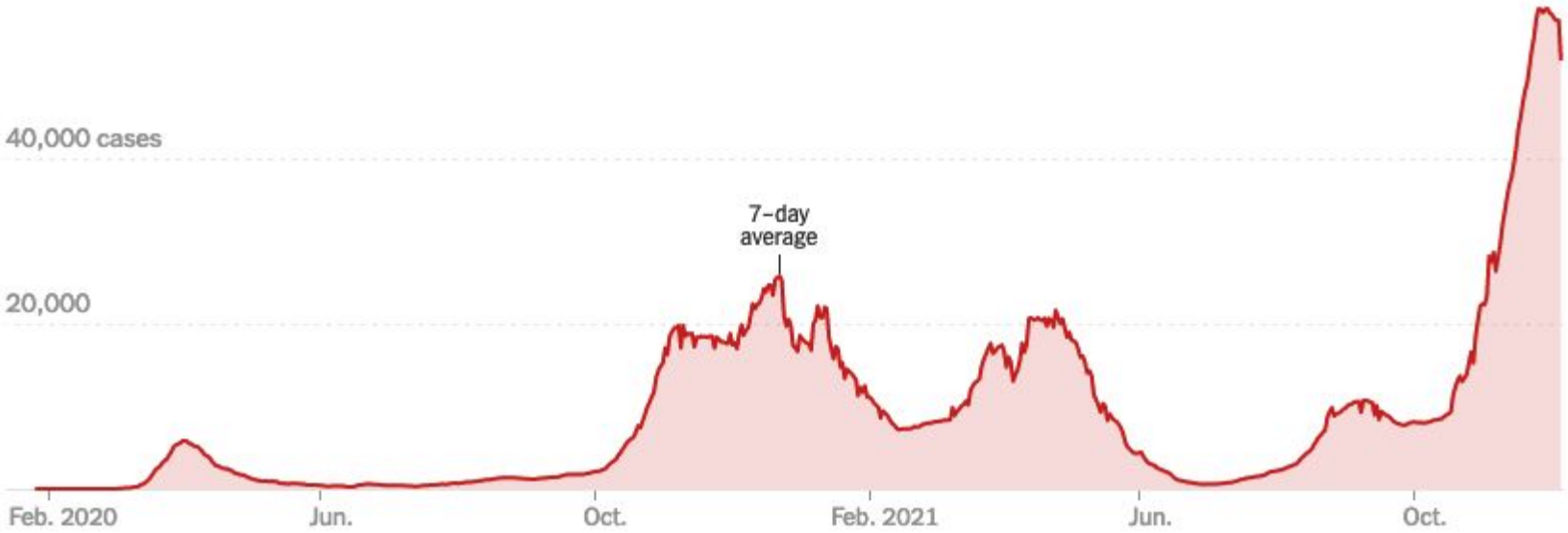
Controlling Epidemic Diffusion with Temporal Graph and Graph Neural Network

CS599 - Graph Analytics

New reported cases

All time | Last 90 days



Motivations for Epidemic Modeling

How can we leverage graph information to test efficiently and accurately the most high impact nodes to control epidemic diffusion?

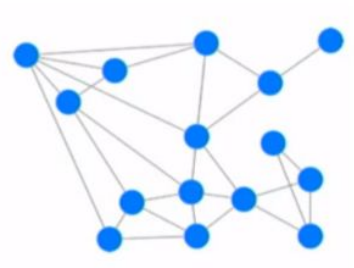
Considering:

- *Hub nodes, or nodes with high degree may be a super-superspreader*
- *We have incomplete information of the full interaction graph in a real-life setting*
- *We may not know all initial infected nodes*

Key Idea of Graph Neural Network

Graph G :

- V is the vertex set
- A is the adjacency matrix (assume binary)
- $X \in R^{m \times |V|}$ is a matrix of node features



Main Tasks:

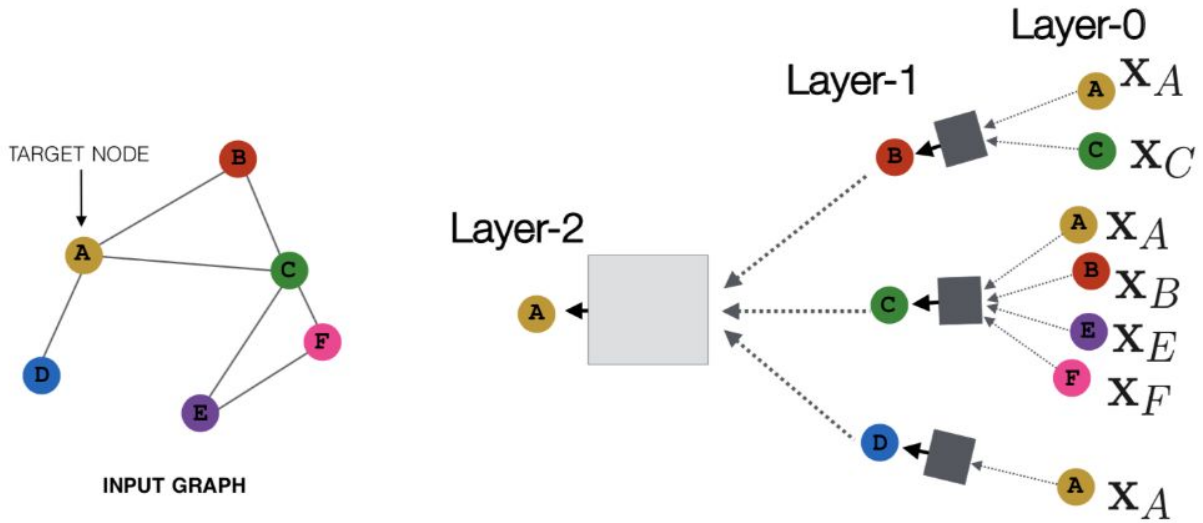
- Node classification
- Graph classification
- Graph generation

} Based on Node embedding

Slide credit: Jure Leskovec, Stanford CS224

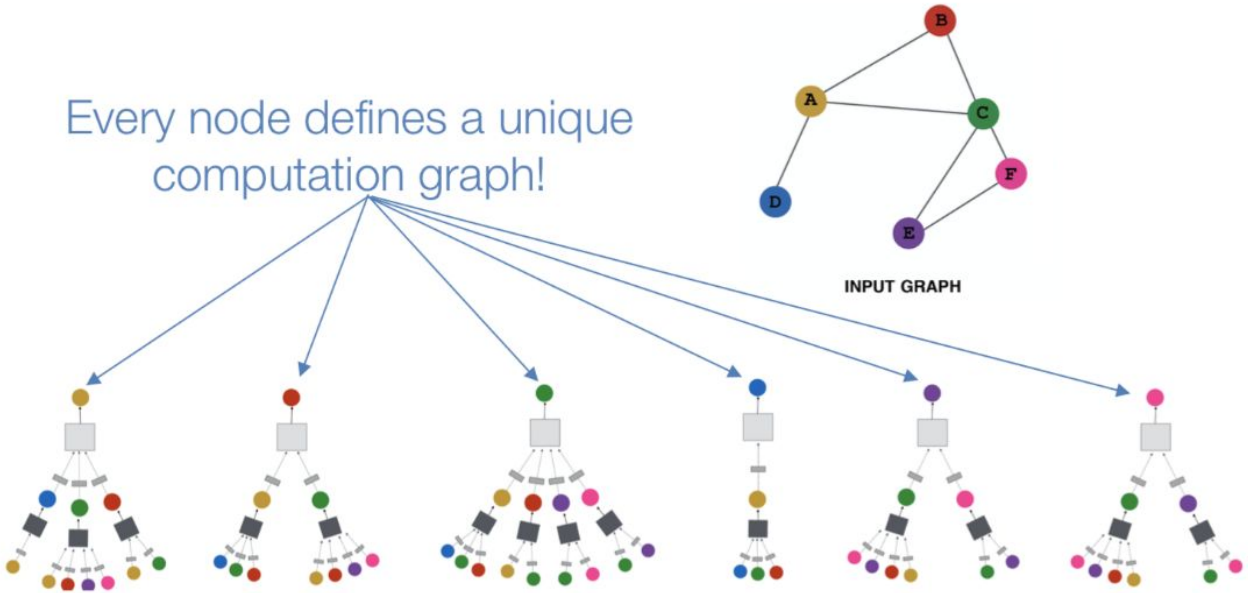
Key Idea of Graph Neural Network

Generate node embeddings based on local neighborhoods



Slide credit: Jure Leskovec, Stanford CS224

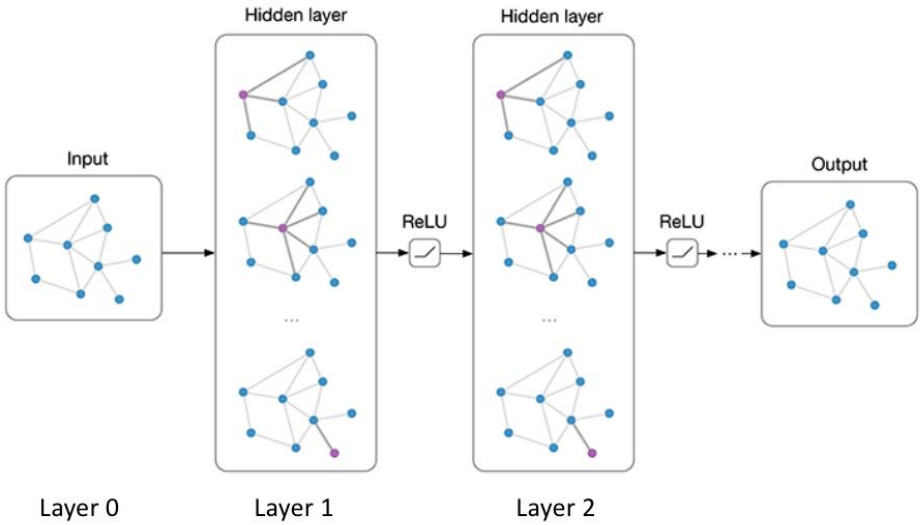
Key Idea of Graph Neural Network



Slide credit: Jure Leskovec, Stanford CS224

Key Idea of Graph Neural Network

Basic approach: Average neighbor information and apply a neural network

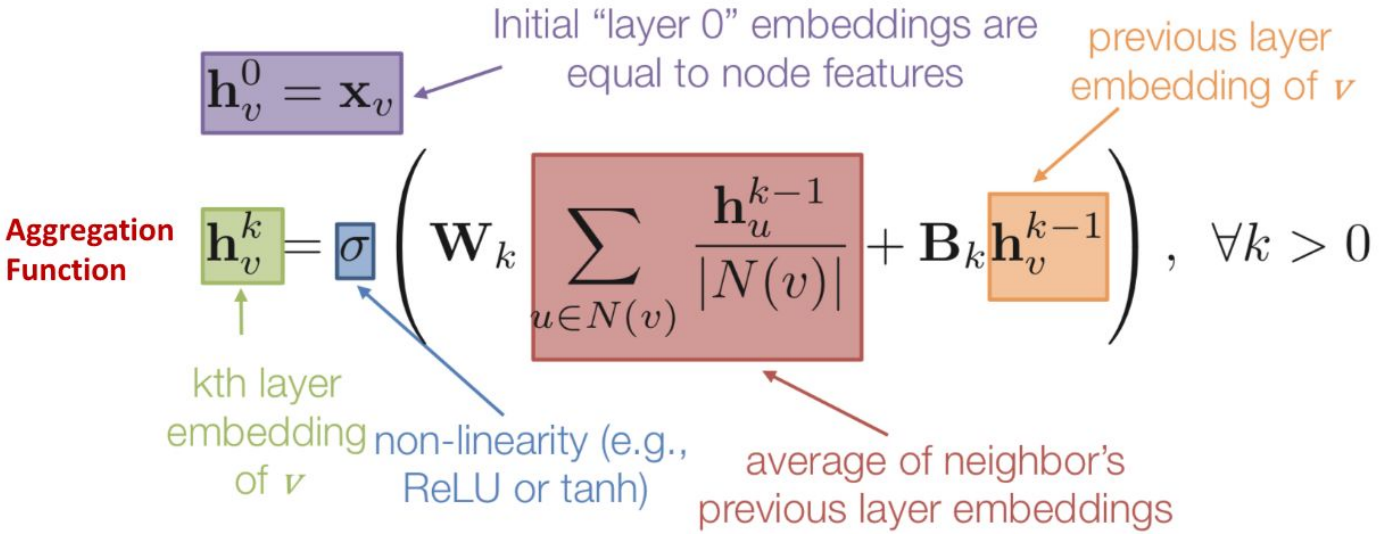


- Nodes have embeddings at each layer.
- Model can be arbitrary depth.
- Layer 0 embedding of node u is its input feature, i.e. x_u.

Slide credit: Jure Leskovec, Stanford CS224

Key Idea of Graph Neural Network

Basic approach: Average neighbor information and apply a neural network

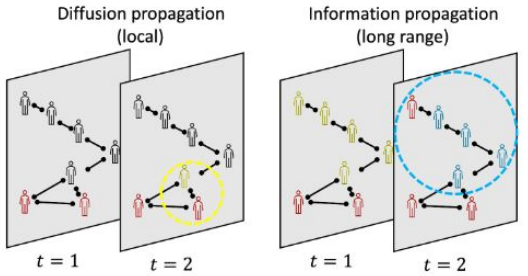


Slide credit: Jure Leskovec, Stanford CS224

Using GNN for Epidemic Prediction and Control

Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks (Meirom et al, 2021)

- Set up a framework for epidemic prediction and control on temporal graph using RL with Graph Neural Network
- Provided baselines on a range of synthetic and real world datasets

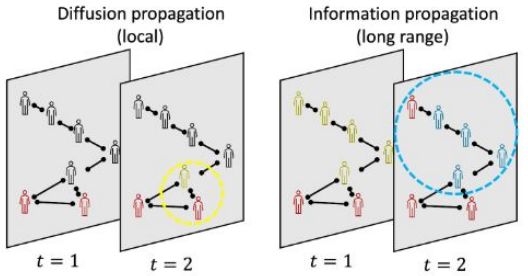


Project Overview

Github: https://github.com/zorache/Controlling-Graph-Dynamics/

- Provided an implementation of the epidemic spread framework, heuristic methods, and supervised learning with GNN

- Experimented on synthetic preferential attachment networks, and on the GR-QC collaboration network (Leskovec et al, 2007)

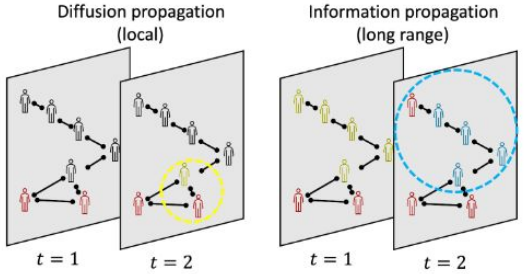


Problem Definition

SEIR Epidemic Spread

Every node (person) can be in one of the following states:

- **susceptible** – a healthy, yet uninfected person (S state)
- l**atent** – infected but cannot infect others (L state)
- **infectious** – may infect other nodes (I state)
- **removed** - self-quarantined and isolated from the graph (R state)



If a susceptible node interacts with an infectious node, the probability of it remaining healthy is

$$\prod_{e \in E_v(t)} (1 - p_e(t))$$

$p_e(t)$ Transmission prob of an given edge at time t

D_v Incubation period

Both are assigned randomly in experiments; transmission prob between 0.5 and 1, incubation is random in {1,2,3,4,5}

Otherwise, it becomes latent and the latent node stays latent at time t if

$$t < T_v + D_v$$

Heuristic Methods

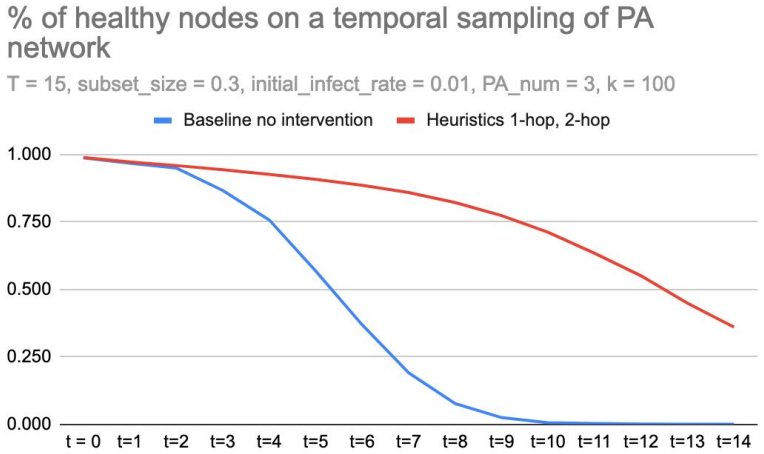
Rank Nodes based on # of infected neighbors in 1-hop and 2-hop
Rationale: if you are in an epidemic hotspot, you are more likely to be infected at the next time step. Prioritize testing nodes that have high numbers of infected neighbors, and high number of total neighbors of neighbors who are infected
Con: in real life this calculation may be difficult to obtain

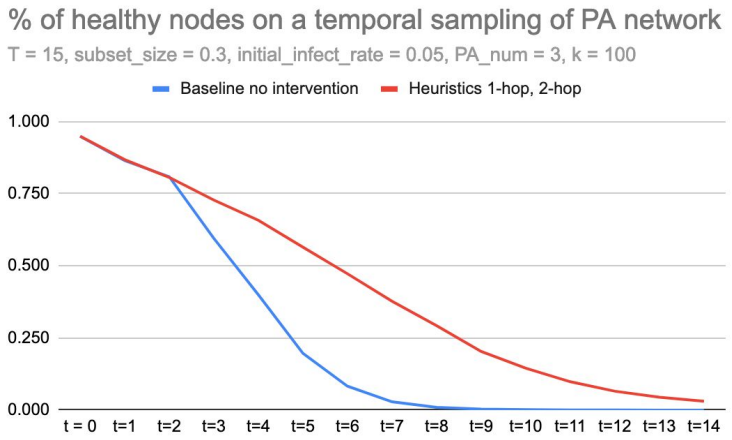
Implementation

- Input: edge list of the graph
- Graph representation: doubly linked linked list, as implemented for assignment
- Fill a (#num nodes, 2) shaped array storing # of 1-hop infected neighbor, # of 2-hop infected neighbor
- Lexicological sort the array to get the top k index of nodes to be tested

  Why is this a good graph representation?
- For large graphs, isolated nodes and their edges can be removed from the graph easily

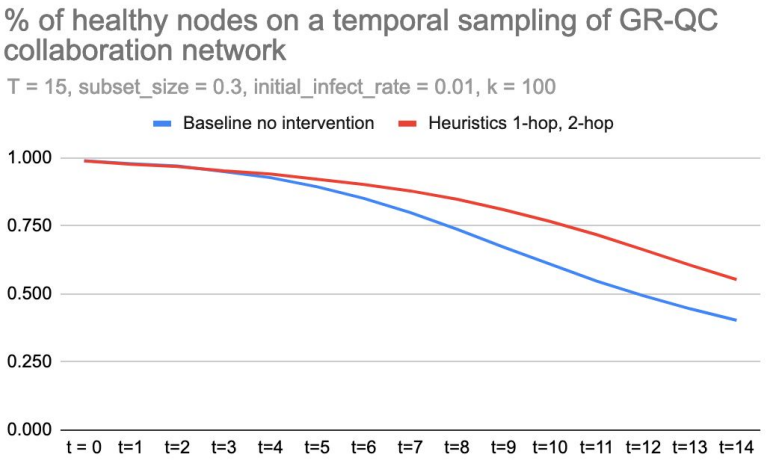
Heuristic Method (hop) Experiments

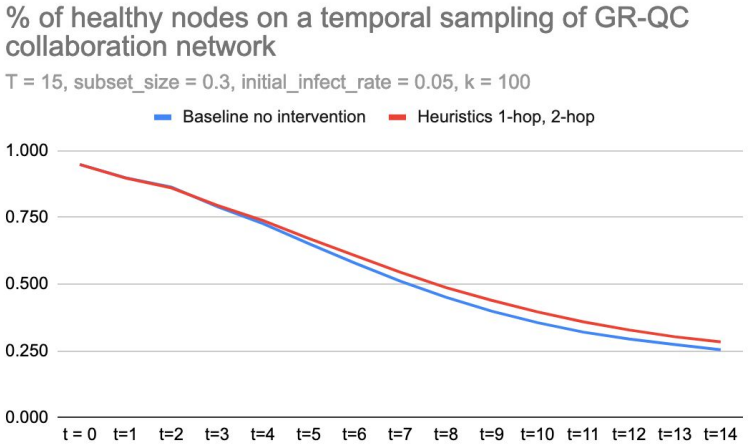




5 Trials for each experiment setting, variance is less than 0.02 for all no intervention baselines
Average percent of healthy nodes across all trials is graphed here for each time step for each setting

Heuristic Methods Experiments





5 Trials for each experiment setting, variance is less than 0.02 for all no intervention baselines
Average percent of healthy nodes across all trials is graphed here for each time step for each setting

Heuristic Methods

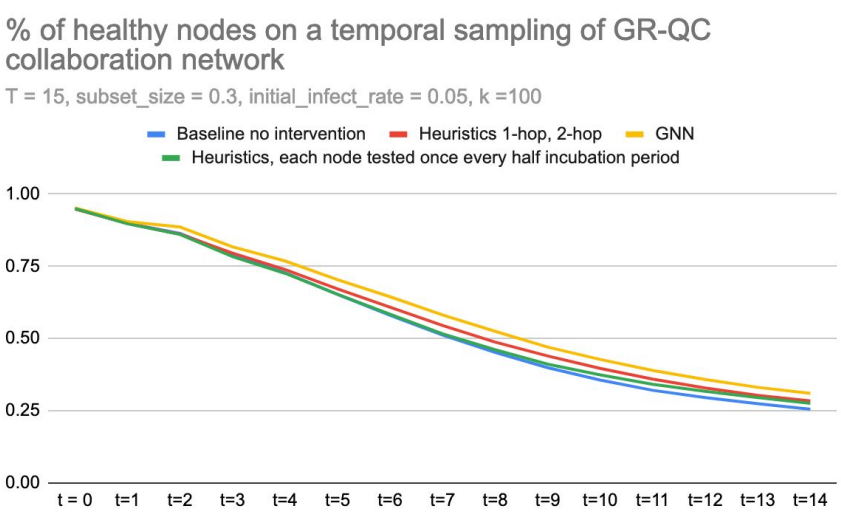
Testing nodes every half-incubation period (similar to University Testing)
Rationale: if the epidemic is controlled, testing nodes every half incubation period can efficiently control outbreak
Con: would not work well when epidemic is not controlled and rates of transmission is high in the population

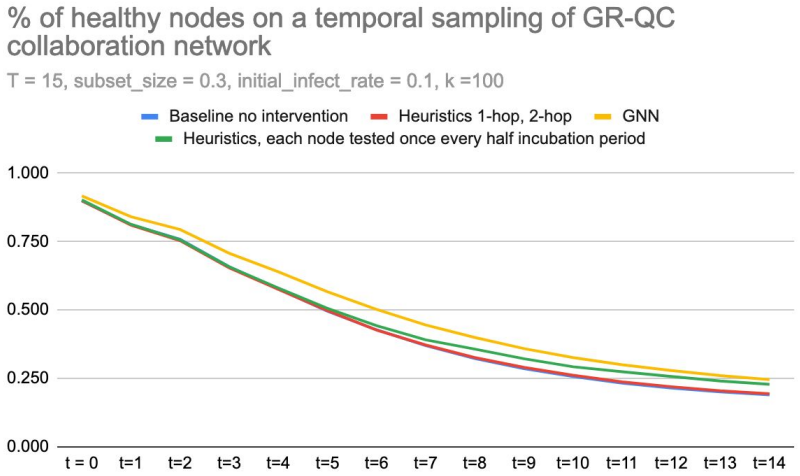
Implementation

- Each node has a testing schedule of every half incubation period, though only k randomly is chosen to test at each step
    - Keeping intervention numbers the same
    - Also may reflect real-life testing, where students may miss testing

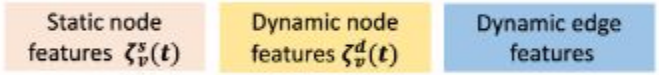
BU's Testing policy

Testing each node half of one incubation period appears to be another adequate heuristic





Data Processing for GNN



Static node

topological graph centralities (betweenness, closeness, eigenvector, and degree centralities) and random node features.

Dynamic node

one-hot vector of dimension 4, corresponding to either untested, tested positive, tested negative, or tested positive in a step before the prior step.

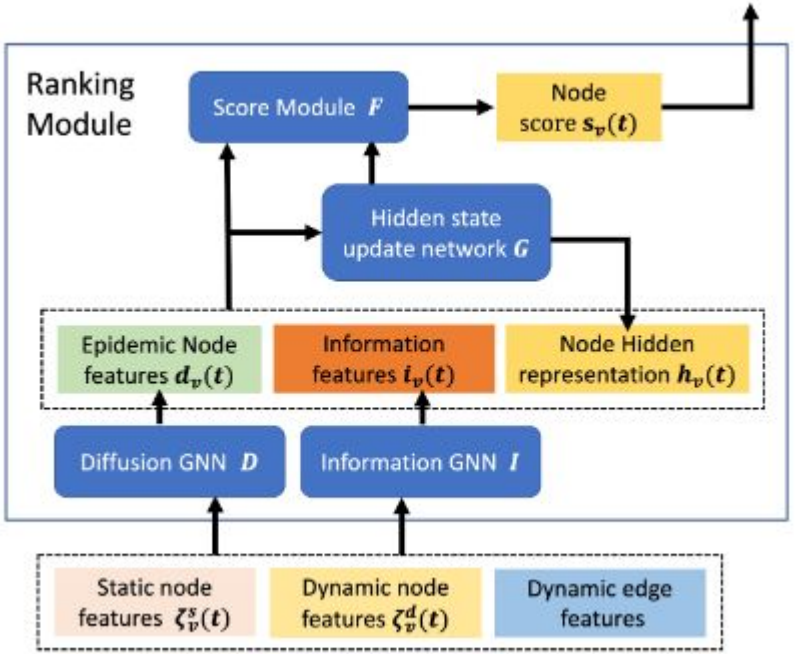
Dynamic Edge

multi-graph view of temporal change, where the edges are all the interactions for a window period prior and including the current time step, and the feature is a vector of dimension 2 with the time step difference of the edge compared to current time, and the edge attribute of transmission probability

GNN to Diffuse Locally and Encode Long-term Info

D: 1 layer graph convolution network to locally diffuse at time t
I: MLP on the multi-graph with all observed time edges
G: updates hidden node representation
F: scores each node's probability of being infected

Loss: binary cross entropy of predicted scores and the ground truth of the node at that time
Goal: predict nodes that are not healthy with high scores



GNN to Diffuse Locally and Encode Long-term Info

D:

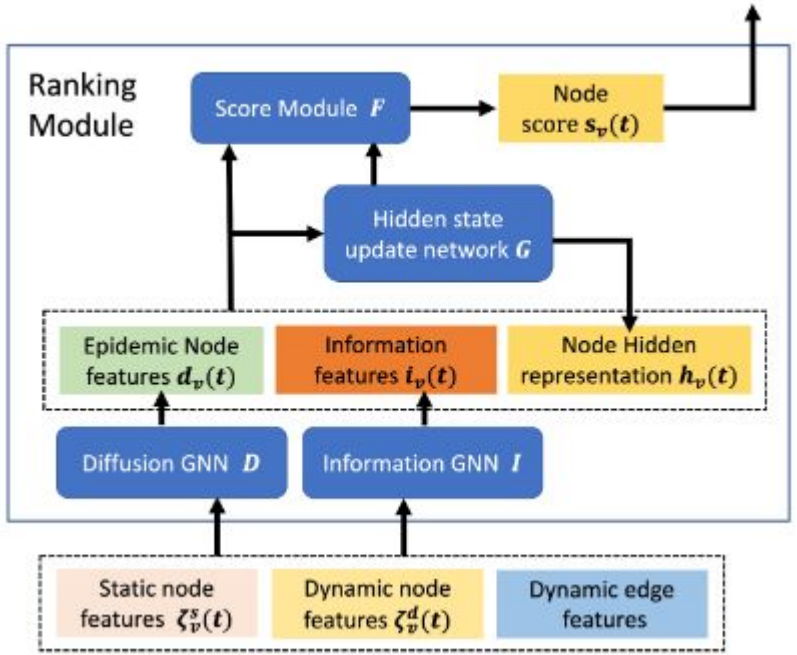
$$d_v(t) = \sum_{u \sim_t v} p_{vu}(t) \cdot M_e(\zeta_v(t), \zeta_u(t); \theta_{m_e}),$$

Transmission prob

Graph convolutional neural net

Neighboring nodes at time t's node features

Parameters for neural net



GNN to Diffuse Locally and Encode Long-term Info

l:

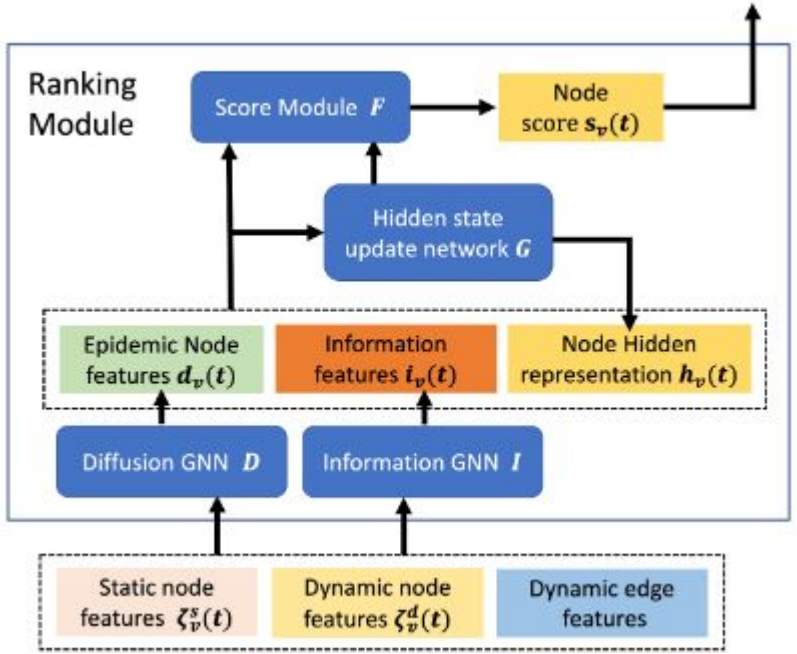
$$x_v^l(t) = \sum_{v' \sim_t v} M^l(x_v^{l-1}(t), x_{v'}^{l-1}(t), \phi_{vv'}(t); \theta_M^l).$$

Neural net

Neighboring nodes at time t's long-term feature from last time step

Parameters for neural net

Multi-graph edge attr



GNN to Diffuse Locally and Encode Long-term Info

G:

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

Neural net

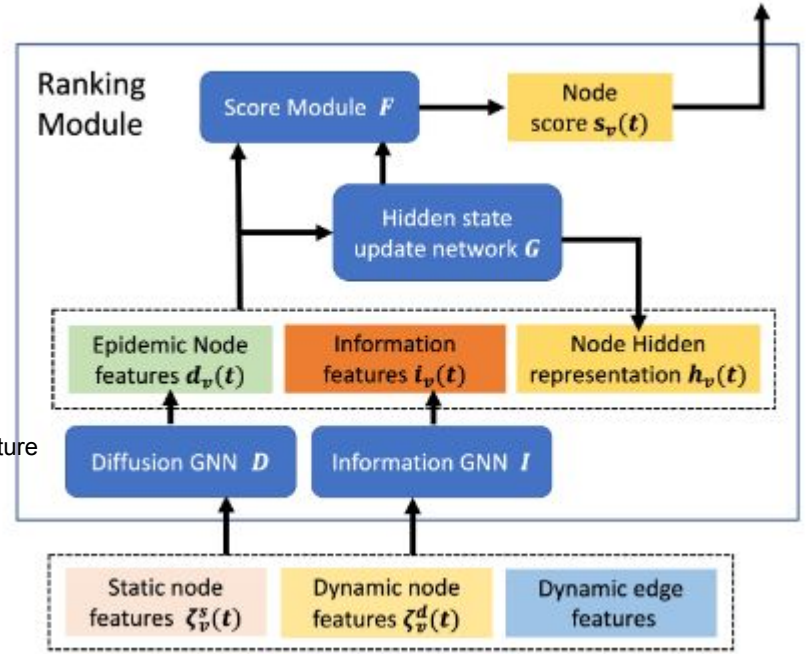
Last hidden feature

Node features

Diffusion node feature

Long-term node feature

Parameters for neural net



GNN to Diffuse Locally and Encode Long-term Info

F:

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

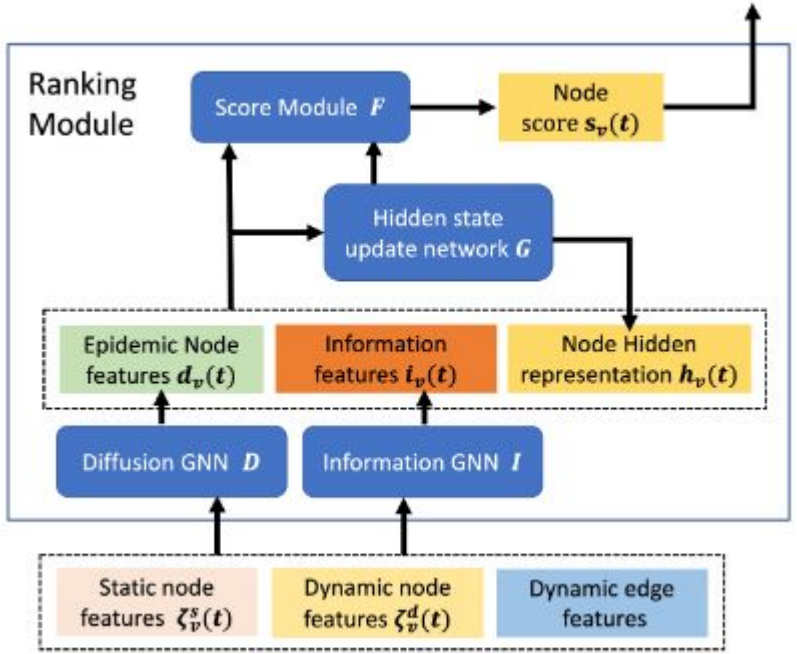
Neural net

Current hidden feature

Last hidden feature

Node features

Parameters for neural net



Experiments

Table 1: Dataset Summary

Method \Graph	avg_{deg}	nodes	edges
$PA_2(Generated)$	2	1000	—
$PA_5(Generated)$	5	1000	—
GR-QC	—	5242	28980

Table 2: Percentage of healthy nodes at t=14, initial infection rate =0.01

Method \Graph	PA_3	PA_5	GR-QC
No intervention	0	0	0.403
Heuristic-hop	**0.36**	0.186	**0.553**
Heuristic-half-incubation	0.008	0.009	0.42
GNN	0.271	**0.253**	0.496

Table 3: Percentage of healthy nodes at t=14, initial infection rate =0.05

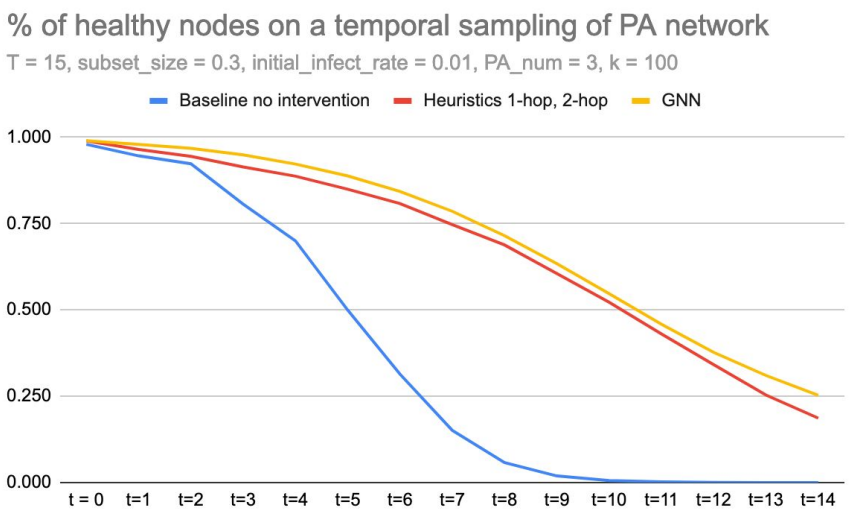
Method \Graph	PA_3	PA_5	GR-QC
No intervention	0	0	0.255
Heuristic-hop	0.031	0.023	0.284
Heuristic-half-incubation	0.01	0.009	0.275
GNN	**0.195**	**0.146**	**0.310**
GNN, no init given	0.092	0.13	0.287

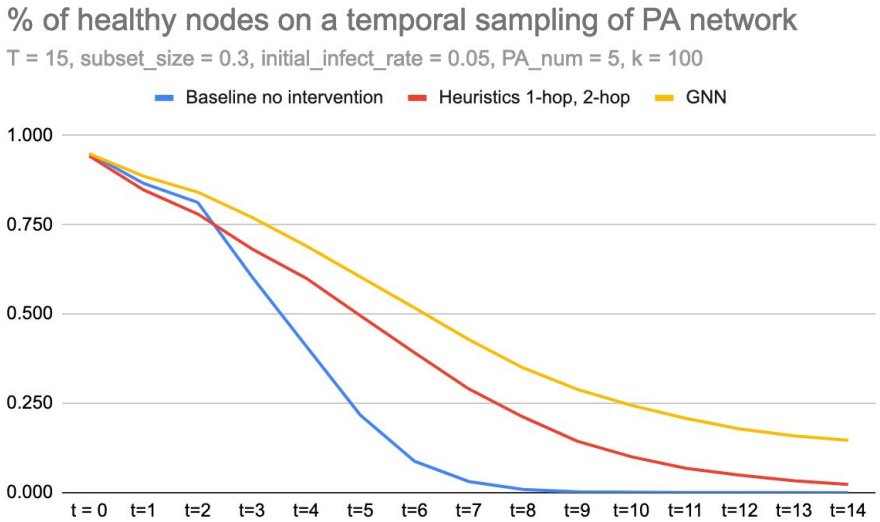
Table 4: Percentage of healthy nodes at t=14, initial infection rate =0.1

Method \Graph	PA_3	PA_5	GR-QC
No intervention	0	0	0.190
Heuristic-hop	0.005	0.011	0.194
Heuristic-half-incubation	0.006	0.005	0.228
GNN	**0.141**	**0.113**	**0.282**
GNN, no init given	0.078	0.103	0.228

GNN Experiments

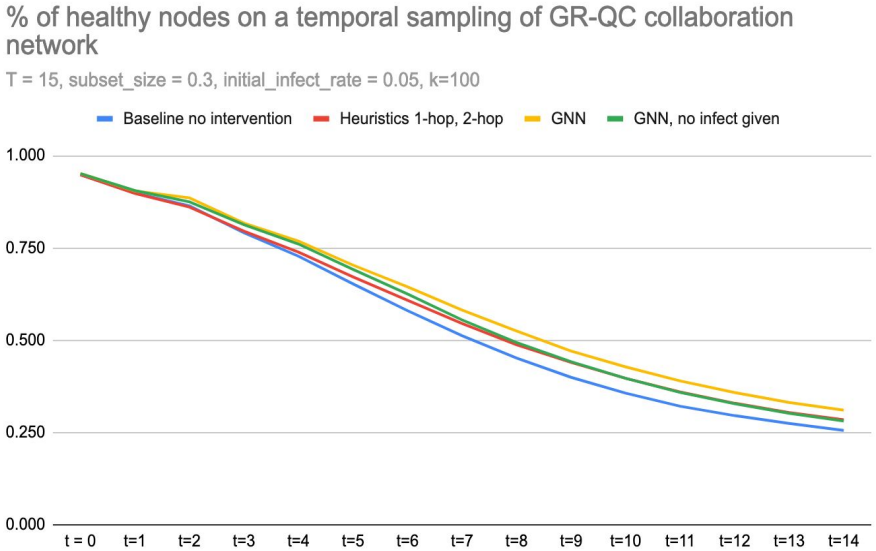
GNN outperforms heuristics for initial infect=0.05, 0.1

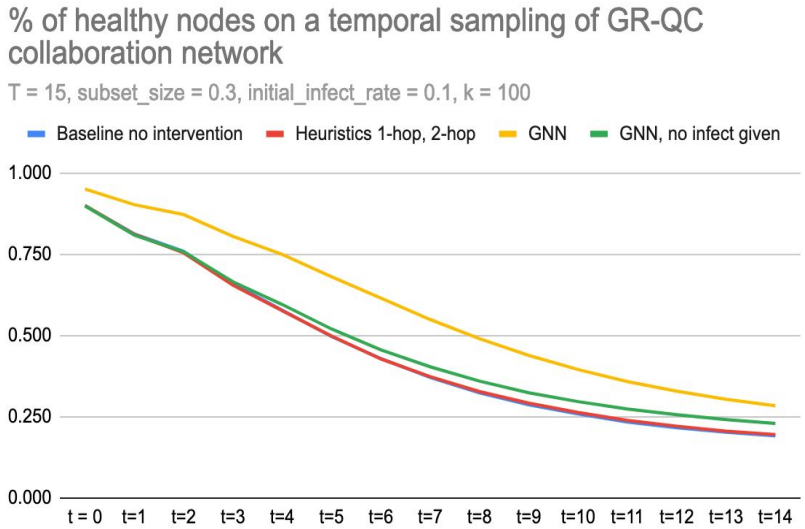




GNN Experiments

GNN outperforms heuristics even when no initial infect is given





Open Questions and Directions

This project has shown GNN outperforms heuristic methods at infect rates=0.05 and 0.1. GNN may also handle the case with no initial infect information.

How can we generalize for graphs we have yet to see?

- More compute and training on more graphs for more epochs
  - Ideally we also have parameters of the graph to generate synthetic networks that mimics the graph

Gap of current model lies in epochs trained, and complexity of ranking module, esp the recurrent module

- Though the experiments outperform the heuristics, it has not matched accuracy from the paper due to possible model differences (simpler model layers were chosen for consideration of GPU memory), and epochs and number of graphs trained on (~30 epoch for this model architecture is the bottleneck for my compute capacity)
  - Smaller batching would solve the issue at the cost of time

References

Jure Leskovec, Stanford CS224W: Machine Learning with Graphs, http://cs224w.stanford.ed

Meirom, Eli, et al. "Controlling Graph Dynamics with Reinforcement Learning and Graph Neural Networks." *International Conference on Machine Learning*. PMLR, 2021.