Source detection on networks using spatial temporal graph convolutional networks

Hao Sha
Comp. and Info. Science
IUPUI
Indianapolis, USA
haosha@iupui.edu

Mohammad Al Hasan
Comp. and Info. Science
IUPUI
Indianapolis, USA
alhasan@iupui.edu

George Mohler
Comp. and Info. Science
IUPUI
Indianapolis, USA
gmohler@iupui.edu

***Abstract*—Detecting the source of an outbreak cluster during a pandemic like COVID-19 can provide insights into the transmission process, associated risk factors, and help contain the spread. In this work we study the problem of source detection from multiple snapshots of spreading on an arbitrary network structure. We use a spatial temporal graph convolutional network based model (SD-STGCN) to produce a source probability distribution, by fusing information from temporal and topological spaces. We perform extensive experiments using popular compartmental simulation models over synthetic networks and empirical contact networks. We also demonstrate the applicability of our approach with real COVID-19 case data.**

***Index Terms*—source detection, COVID-19 epidemic, spatial temporal graph convolutional network**

I. Introduction

By early January 2021, the number of confirmed COVID-19 cases has reached 83.6 millions world-wide, and over 1.8 million people have lost their lives. One important method for limiting transmission of an infectious disease consists of identifying epidemic cluster sources and isolating them from the population. Epidemiologists conduct source detection by analysing the genetic evolution of virus strains [1] or by contact tracing [2], which can be time-consuming and labor-intensive. However, COVID-19 has demonstrated limits to contact tracing when prevalence is widespread, for example in the United States, and methods are needed for source detection in such situations.

In real life, most individuals have high probability of contact with only a small portion of the population. It is thus realistic to model an epidemic as a spreading process on an interpersonal network, where infection can only transmit from an individual to its neighbors. Moreover, the interpersonal network can be obtained through IoT technology [3]. For example, [4] reconstructs contact networks through mobile phone communication data. So in this work, we solve the source detection problem in a network setting and assume the availability of the network.

Source detection on networks is a well studied problem with wide applications [5]. Existing solutions include centrality based methods [6]–[8], Dynamic Message Passing (DMP) [9] and Label Propagation based Source Identification (LPSI) [10]. Recently, [11] proposed using graph convolutional network (GCN) to solve source detection with improved efficiency and accuracy. The method takes as input a snapshot of the spreading and outputs a probability for each node in the graph as to whether the node is the source. There appears to be some controversy [12] as to whether a GCN approach is valid compared to more standard (non-deep learning based) methods for source detection. In particular, some discussion on the 2021 ICLR open review portal questions whether a GCN approach can handle diverse and realistic transmission dynamics (beyond SIR), diverse network topology, and can even be solved using a single snapshot.

In practice, there can possibly be multiple snapshots observed at different stages of an epidemic; these snapshots can help revealing the temporal dynamics of the disease propagation. As suggested in [13], multiple independent snapshots can enhance detectability. So an ideal model should be able to take advantage of richer observations and at the same time exploit the underlining connectivity of the network. This motivates us to adopt a spatial temporal graph convolutional network (STGCN) architecture that combines the best of the two worlds. STGCNs were originally developed for the tasks of traffic forecasting [14] and action recognition [15], where the data contains sequences of temporal snapshots of route networks and skeleton networks, respectively. We adapt the particular form of STGCN proposed in [14] for source detection, and name it Source-Detection-STGCN (SD-STGCN).

In this work we address some of the issues raised in recent debate on GCN applicability to source detection [12]. We show that single snapshot GCNs [11] do not perform significantly better than simpler message passing algorithms, however multi-snapshot STGCNs do lead to significant accuracy improvements. We validate these findings using more realistic models of transmission, including non-Markovian epidemic simulations with delay, and more realistic network topology, including both synthetic and empirical contact networks. Additionally, we apply our model to real COVID-19 case data. The experiments demonstrate the superior performance of SD-STGCN.

II. Background

A. Epidemic Models

SD-STGCN is a deep learning model that requires abundant data to train. However, real infection records are very limited

This is the author's manuscript of the article published in final edited form as:

Sha, H., Al Hasan, M., & Mohler, G. (2021). Source detection on networks using spatial temporal graph convolutional networks. 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), 1–11. https://doi.org/10.1109/DSAA53316.2021.9564188

in the public domain. So we resort to epidemic models to generate simulations that resemble the real contagion processes. It is worth noting that although SD-STGCN is trained on simulation data, it is independent with the particular epidemic models, and thus can be applied to more complex and realistic cases.

SIR [16], [17] splits the population into three compartments - susceptible (S), infectious (I), and recovered (R). Let $S(t)$, $I(t)$, and $R(t)$ denote the fractions of individuals who are susceptible, infectious, and recovered, respectively, at time t. They follow the transformation rule: $S \to I \to R$, and satisfy $S(t)+I(t)+R(t) = 1$, assuming a close system. The ordinary differential equations of the system are given by:

$$\frac{dS}{dt} = -\beta IS, \qquad \frac{dI}{dt} = \beta IS - \gamma I, \qquad \frac{dR}{dt} = \gamma I \tag{1}$$

where β is the transmission rate ($S \to I$) and γ is the recovery rate ($I \to R$). Given β, γ, and initial conditions (S_0, I_0, R_0), one can solve $S(t)$, $I(t)$, and $R(t)$ at any t from Eq. 1.

Network SIR [11], [18] assumes that the population form a static contact network G of N nodes, with each node representing an individual. An infectious node j can transmit the disease to a node i if and only if i is susceptible and is a neighbor of i, i.e. $i \in \mathcal{N}(j)$. Let A be the adjacency matrix of G, with $A_{ij} = 1$ if $i \in \mathcal{N}(j)$, $A_{ij} = 0$ otherwise. Let $S_i(t)$, $I_i(t)$, and $R_i(t)$ be the probabilities of node i being in each of the states at time t, with $S_i(t)+I_i(t)+R_i(t) = 1$. Let I_0 denote the initial infected persons at time $t = 0$, where $I_{0,i} = 1$ if i is initially infectious (i.e. patient zero), otherwise $I_{0,i} = 0$. Let $\mathcal{P}(t)$ denote the probability of remaining infectious at later time t after becoming infectious. $\mathcal{P}(t)$ is monotonic decreasing with $\mathcal{P}(0) = 1$ and $\lim_{t\to\infty} \mathcal{P}(t) = 0$. Consequently, S_i, I_i, and R_i obey the following differential equations (for derivation see Appendix):

$$\begin{aligned}
\frac{dS_i(t)}{dt} &= -\beta \sum_j A_{ij} S_i(t) I_j(t) \\
\frac{dI_i(t)}{dt} &= \beta \sum_j A_{ij} S_i(t) I_j(t) + I_{0,i} \mathcal{P}'(t) \\
&\quad + \beta \int_0^t \sum_j A_{ij} S_i(x) I_j(x) \mathcal{P}'(t-x) dx \\
\frac{dR_i(t)}{dt} &= -I_{0,i} \mathcal{P}'(t) - \beta \int_0^t \sum_j A_{ij} S_i(x) I_j(x) \mathcal{P}'(t-x) dx
\end{aligned} \tag{2}$$

where β is the transmission rate and $\mathcal{P}'(t) = \frac{d\mathcal{P}(t)}{dt}$. Note that a similar derivation can be found in [19], but there they assume that the population is homogeneously mixed and an individual can have contact with anyone else. In contrast, here we derive the system assuming that the population is in a network and an individual has only limited contacts.

Standard network SIR model For standard SIR, we have $\mathcal{P}(t) = e^{-\gamma t}$, i.e. the probability for an I node to stay infectious decays exponentially [19]. Eq. 2 becomes:

$$\begin{aligned}
\frac{dS_i(t)}{dt} &= -\beta \sum_j A_{ij} S_i(t) I_j(t) \\
\frac{dI_i(t)}{dt} &= \beta \sum_j A_{ij} S_i(t) I_j(t) - \gamma I_i(t) \\
\frac{dR_i(t)}{dt} &= \gamma I_i(t).
\end{aligned} \tag{3}$$

In the early stage, we have $S_i(t) \approx 1$. Plugging it in Eq. 3, we can obtain the basic reproduction number R_0 of the following form:

$$R_0 = \frac{\beta \lambda_1}{\gamma}, \tag{4}$$

where β is the transmission rate, γ is the recovery rate, and λ_1 is the largest eigenvalue of the adjacency matrix A. For detailed derivation, see Appendix.

Delay network SIR model For delay SIR, following [19], we assume $\mathcal{P}(t) = \Theta(t - T)$, a step function, with $\mathcal{P}(t) = 1$ for $0 \le t \le T$ and $\mathcal{P}(t) = 0$ for $t > T$, then $\mathcal{P}'(t - x)$ is $-\delta(t - x - T)$. Here T is the delay time for recovery, i.e. an infectious node would recover after T units of time. Now Eq. 2 becomes:

$$\begin{aligned}
\frac{dS_i(t)}{dt} &= -\beta \sum_j A_{ij} S_i(t) I_j(t) \\
\frac{dI_i(t)}{dt} &= \beta \sum_j A_{ij} S_i(t) I_j(t) - \beta \sum_j A_{ij} S_i(t-T) I_j(t-T) \\
\frac{dR_i(t)}{dt} &= \beta \sum_j A_{ij} S_i(t-T) I_j(t-T).
\end{aligned} \tag{5}$$

Such delay differential equations are known to be associated with non-Markovian dynamics [19], [20]. From Eq. 5, we can derive the basic reproduction number R_0 of the following form (for derivation see Appendix in the supplementary material):

$$R_0 = \beta \lambda_1 T, \tag{6}$$

where β is the transmission rate, λ_1 is the largest eigenvalue of the adjacency matrix A, and T is the delay time for recovery.

SEIR is another popular epidemic model [16], [17] that is well-suited for modeling the infectious diseases with an exposed/latent period. During such period, the pathogen in the host is in low numbers, so that the host is infected but cannot transmit it to others. In addition to S, I, and R, the model further assumes an exposed (E) compartment, to account for the latent period. The compartments follow the transformation rule: $S \to E \to I \to R$, and obey the following differential

equations:

$$\begin{aligned}\frac{dS_i(t)}{dt} &= -\beta \sum_j A_{ij} S_i(t) I_j(t) \\ \frac{dE_i(t)}{dt} &= \beta \sum_j A_{ij} S_i(t) I_j(t) - \alpha E_i(t) \\ \frac{dI_i(t)}{dt} &= \alpha E_i(t) - \gamma I_i(t) \\ \frac{dR_i(t)}{dt} &= \gamma I_i(t),\end{aligned} \tag{7}$$

where β is the transmission rate, γ is the recovery rate, and α is the rate from E to I (or equivalently $1/\alpha$ is the mean latent period). The basic reproduction number is of the same form as Eq. 4.

B. STGCN

Although STGCN is separately developed in [15] and [14], here we adopt the one designed for traffic forecasting [14]. The model combines graph convolutional network (GCN) [21], [22] and convolutional neural network (CNN) for extracting spatial and temporal information. The model takes a traffic network (i.e. a network of sensor stations) and a sequence of sensor data (e.g. traffic speed, volume, density) as input, and predicts the future traffic status. The core of STGCN is a stack of the so-called spatio-temporal convolution (ST-Conv) blocks. An ST-Conv block consists of a spatial layer sandwiched by two temporal layers. A temporal layer contains a 1-D CNN along time axis followed by a gated linear unit (GLU), to capture the temporal dynamics. The spatial layer is a GCN implemented using the Chebyshev polynomials approximation [21] or the 1st-order approximation [22]. As suggested in [14], such sandwich architecture allows jointly processing graph-structured time series; the spatial layer in the middle can serve as a bottleneck to help achieve scale compression and feature squeezing.

III. Methodologies

Problem description In general, a contact network during a contagion process can be temporal, directed, and weighted, but here we concentrate on a static undirected and non-weighted graph $G = (V, E)$, where V and E are the sets of nodes and edges; the number of nodes in G is $N = |V|$. For a contagion process on G, we observe a sequence of k snapshots $\mathcal{X} = \{\mathbf{x}_{t_1}, \dots \mathbf{x}_{t_k}\}$ at time steps $\{t_1, t_2, \dots t_k\}$. Note that the snapshots are not necessarily for consecutive time value. A snapshot $\mathbf{x}_{t_k}$ contains the states of all the nodes of G at t_k, i.e. $\mathbf{x}_{t_k} = \{x_{t_k,1}, \dots x_{t_k,N}\}$, with each node's state $x_{t_k,i} \in \{S, I, R\}$ for the SIR model or $\in \{S, E, I, R\}$ for the SEIR model. The problem of source detection is to find the set of initially infected nodes $\mathcal{Y} = \{i | x_{t=0,i} = I, i \in V\}$ by solving the following objective:

$$\mathcal{Y}^\star = argmax_{\mathcal{Y}} P(\mathcal{X}|\mathcal{Y}, G) \tag{8}$$

where $P(\mathcal{X}|\mathcal{Y}, G)$ is the likelihood of observing $\mathcal{X}$ with $\mathcal{Y}$ being the source.

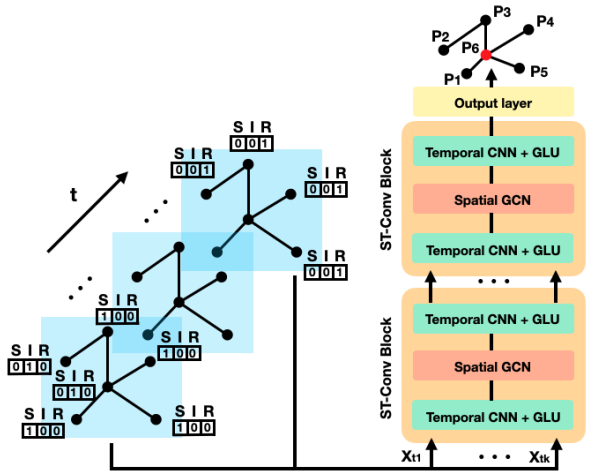


Fig. 1: SD-STGCN architecture. The blue areas on the left represent the input snapshots, which are one-hot encoded network states at multiple time steps. The orange areas on the right illustrate the model architecture consisting of a stack of ST-Conv blocks followed by an output layer. The output is a list of probabilities of each node being the source.

Model description Our SD-STGCN model is built upon the STGCN architecture. Fig. 1 is an illustration of SD-STGCN. The input is a sequence of k snapshots $\mathcal{X} = \{\mathbf{x}_{t_1}, \dots \mathbf{x}_{t_k}\}$ (the blue areas in Fig. 1). A snapshot $\mathbf{x}_{t_k} = \{x_{t_k,1}, \dots x_{t_k,N}\}$ contains the states of all the nodes of G, with $x_{t_k,i}$ being a one-hot encoded vector of the states. So the input $\mathcal{X}$ is of the shape $k \times N \times C_{in}$, where k is the number of snapshots, N is the number of nodes, C_{in} is the number of channels ($= 3$ for SIR or 4 for SEIR). $\mathcal{X}$ then goes through a series of ST-Conv blocks (the orange areas in Fig. 1), each consisting of two temporal layers and one spatial layer.

A temporal layer contains a 1-D CNN followed by a GLU. The CNN has a kernel of size K_t and applies to every node of G without padding, thus compressing $\mathcal{X}$ along time axis by $K_t - 1$. This part of the operation can be summarized as:

$$\mathcal{Z}_1 = GLU(CNN(\mathcal{X})) \in \mathbb{R}^{(k-K_t+1)\times N \times C_h}, \tag{9}$$

where $\mathcal{Z}_1$ is the output of the layer and C_h is the number of channels.

$\mathcal{Z}_1$ is then fed to the spatial layer, which is a GCN followed by a rectified linear unit (ReLU). The GCN has a kernel of size K_s. If the GCN is implemented by Chebyshev polynomials approximation [21], then $K_s - 1$ is the order of the truncated expansion; if the GCN is implemented by 1st-order approximation [22], then K_s is the number of successive convolutional layers. The spatial layer can effectively encode the information in the spatial domain by aggregating the signals in the K_s neighborhood of each node in each snapshot. Let the graph convolution preserve the channel dimension, the spatial layer can be summarized as:

$$\mathcal{Z}_2 = ReLU(GCN(\mathcal{Z}_1)) \in \mathbb{R}^{(k-K_t+1)\times N \times C_h}, \tag{10}$$

where $\mathcal{Z}_2$ is the output of the layer.

Following the spatial layer is another temporal layer. It performs the same operation as the first temporal layer, and further reduces the time dimension by $K_t - 1$. Unlike the spatial layer, this layer does not preserve the channel dimension, but instead magnifies it, so the spatial layer becomes a bottleneck. The formula for this layer is:

$$\mathcal{Z}_3 = GLU(CNN(\mathcal{Z}_2)) \in \mathbb{R}^{(k-2K_t+2)\times N\times C_{out}}, \qquad (11)$$

where $\mathcal{Z}_3$ is the output of the ST-Conv block and C_{out} is the number of channels ($C_{out} > C_h$). As in Fig. 1, $\mathcal{Z}_3$ then goes through more ST-Conv blocks that repeat the same sequence of operations. Assuming that M ST-Conv blocks are used, the output of the last ST-Conv block has the shape $\mathcal{Z}_3 \in \mathbb{R}^{(k-2MK_t+2M)\times N\times C_{out}}$.

Finally, we send $\mathcal{Z}_3$ to an output layer, which contains a temporal layer followed by a fully-connected (dense) layer with softmax activation. The temporal layer is a 1-D CNN with a kernel of size $K_o = k - 2MK_t + 2M$, so that the time dimension becomes 1. The fully-connected layer further reduces the channel dimension from C_{out} to 1, and the softmax function normalizes the output across the nodes to represent the source probability distribution. The output layer can be summarized as:

$$\mathcal{P} = Softmax(Dense(CNN(\mathcal{Z}_3))) \in \mathbb{R}^N, \qquad (12)$$

where $\mathcal{P} = \{P_1, \ldots, P_N\}$, with $\sum_{i=1}^{N} P_i = 1$. We then pick the set of nodes with top P_i as $\mathcal{Y}^\star$. For single-source detection (i.e. $|\mathcal{Y}^\star| = 1$), we select the node $i^\star = argmax_i P_i$ as the source. We perform single-source detection in Sec. V-A-V-B, and multi-source detection in Sec. V-D.

The model parameters Θ are learned in a supervised manner. In specific, we generate S(E)IR simulations on G with random sources $\mathcal{Y}_{true}$ (the ground truth), and sample k random snapshots $\mathcal{X}$ from the simulations. We then minimize the following cross-entropy loss:

$$\Theta = argmin_\Theta - \sum_{i\in V} y_i log(P_i) \qquad (13)$$

where $\mathcal{Y}_{true} = \{y_1, \ldots y_N\}$ is the one-hot encoded source set; $\mathcal{P} = \{P_1, \ldots, P_N\}$ is the output of SD-STGCN.

Training protocol We adopt two ST-Conv blocks in SD-STGCN. We set the block dimensions as $(C^1_{in}, C^1_h, C^1_{out}) = (3, 36, 144)$ and $(C^2_{in}, C^2_h, C^2_{out}) = (144, 36, 72)$ for SIR model; $(C^1_{in}, C^1_h, C^1_{out}) = (4, 36, 144)$ and $(C^2_{in}, C^2_h, C^2_{out}) = (144, 36, 72)$ for SEIR model. To train the model, we perform batch gradient descent with RMSProp optimizer, a batch size of 16, and a learning rate of 0.001. In each pass, a series of $k = 16$ snapshots are sampled uniformly at random from every simulation. We examine different configurations - Chebyshev polynomials approximation [14] vs. 1st order approximation [22], spatial kernel size K_s in $\{2, 3, 4\}$, and temporal kernel size K_t in $\{2, 3, 4\}$. In our experiments, for each graph, we generate $2,000$ simulations and split into 80% training, 10% validation, and 10% testing. Grid search on validation data suggests that using 1st order approximation with $K_s = 4$ and $K_t = 3$ renders the best performance. In the following experiments, we adhere to this setup.

IV. Related works

There have been a surge of works on various topics related to the COVID-19 pandemic [19], [23], [24]. To the best of our knowledge, there was only one work [11] before us using deep learning to solve the source detection problem of COVID-19, although the general problem of identifying the propagation sources in networks [5] is well studied. Early methods resort to graph-centrality measures such as rumor center [6], [25]–[27], eigenvector center [7], [28], [29], and Jordan center [8]. However, these methods are heuristic and only provide suboptimal solutions. Alternatively, [9] proposes a method named dynamic message passing (DMP) that provides near-optimal solution. However, this method has high computational complexity and requires the propagation time, which is in general not available in practise. [10] performs multi-source detection based on the idea of source prominence and label propagation, nonetheless, its convergent version has $O(N^3)$ complexity. Recently, [11] proposes using graph neural networks (GCN) [21], [22] for single-source detection without knowing the propagation time. But this method utilizes only one snapshot, while multiple snapshots may be observed in reality. Moreover, it has been shown in [13] that using multiple independent snapshots can improve source detection accuracy. In the fields of traffic forecasting [14] and action recognition [15], different forms of spatial temporal graph convolutional networks were developed for prediction involving spatial and temporal signals. Inspired by these works, especially [14], we propose SD-STGCN, a spatial temporal graph convolutional network for source detection that utilizes multiple snapshots.

V. Experiments

Data We run standard and delay S(E)IR simulations with synthetic random graphs [30] and empirical contact networks. We generate different types of random graphs using Erdös–Rényi (**ER**), Barabási–Albert (**BA** & **BA-Tree**) [31], and Random Geometric Graph (**RGG**) [32] models. We adopt three empirical contact networks - **Bernard and Killworth Fraternity** (Frat) [33], [34], **SFHH conference** (Conf) [35], and **High school** (High) [36], which are static networks obtained by aggregating dynamic contact sequences. Fraternity network describes the interactions between students living in a fraternity at a West Virginia college. An edge is added if two students are spotted engaged in a conversation. Conference network describes the face-to-face interactions of participants of the SFHH 2009 conference. High-school data contains close proximity records of students, teachers, and other persons in an American high school. We add an edge if two people are in close proximity for more than 5 minutes. The statistics of the networks are listed in Table I. Note in addition to the entries in Table I, we also test our model on ER graphs with $5,000$ and $10,000$ nodes. We defer the description of the **real COVID-19 cases data** to Sec. V-D.

TABLE I: Network statistics. The columns from left to right are the network name, number of nodes $|V|$, number of edges $|E|$, average degree $\overline{d}$, and clustering coefficient C [30].

network	$\|V\|$	$\|E\|$	$\overline{d}$	C
ER	1000	10128	20.3	0.020
BA	1000	9900	19.8	0.062
BA-Tree	1000	999	2.0	0
RGG	1000	9326	18.7	0.618
Frat	58	967	33.3	0.747
Conf	403	9565	47.5	0.282
High	774	7992	20.7	0.186

Baseline methods We compare our SD-STGCN with two popular baseline methods - Dynamic Message Passing (DMP) [9] and a graph convolutional network based model (GCN) [11]. We choose DMP as a representative of the non-deep learning based methods, as it has been shown in [9] to considerably outperform other popular methods like rumor center [6] and Jordan center [8]. We select the GCN model as the second baseline, as it also utilizes deep learning for source detection. We show that by leveraging multiple snapshots, our model can achieve significant improvement over this model. We notice that the algorithm proposed in [10] can perform multi-source detection, however their code is not publicly available.

DMP [9] is a source detection model based on the message passing framework [37]. For a single source SIR on a network $G = \{V, E\}$, it predicts the source given a snapshot $\mathcal{O}$ at time t. Assuming an arbitrary node i as the source, it first estimates the marginal probabilities for any node to be in each of the three states S, I, R, at time t. It then approximates the joint likelihood $P(\mathcal{O}|i)$ as the product of the marginal probabilities of the observed nodes being in the observed states. It then goes through all $i \in V$ and picks the one that maximizes $P(\mathcal{O}|i)$ as the source. The time complexity of the model is $\mathcal{O}(tN^2\overline{d})$, where t is the time step of the observations, N is the number of nodes, and $\overline{d}$ is the average degree of the network. It is thus computationally expensive for very large networks that are strongly connected [5], [11]. Besides, the method requires the propagation time t of the observations, which is generally unknown.

GCN [11] is a graph convolutional network [22] based model for source detection. Similar to DMP, it predicts the source using one snapshot of the network. But unlike DMP, it does not need to know the time of the snapshot. Moreover, it can be applied to both SIR and SEIR. The model takes one-hot encoded node states as features and outputs the probabilities of each node being the source. In [11], the authors actually proposed three GCN models (i.e. GCN-S, GCN-R, GCN-M) varied by propagation rules (i.e. symmetric, random walk, mixture). We adopt GCN-S as it appears to have the best performance among the three.

Performance metrics For single-source detection, we adopt three types of metrics - top-1 accuracy, mean reciprocal rank (MRR), and hit rates. Top-1 accuracy (Top-1 Acc) examines whether the highest ranking node is aligned with the true source. MRR is the average reciprocal ranks of the true source given by the model, and a greater MRR indicates better performance. In addition, we evaluate hit rates at k (Hit@k) to see if the true source is among the top k candidates given by the model. For multi-source detection, besides Hit@k, we adopt Jaccard Similarity (JS) and normalized discounted cumulative gain (nDCG). JS and nDCG are estimated between the true source set $\mathcal{Y}^\star$ and the predicted set $\mathcal{Y}$ of the same size, i.e. $|\mathcal{Y}^\star| = |\mathcal{Y}|$. These metrics can measure the model's ranking quality.

TABLE II: Performance of SD-STGCN and GCN trained and tested on SIR simulations using $R_0 = 2.5$ and $\gamma = 0.4$, over random graphs of different types. The scores are evaluated over five graphs per type and five runs per graph. The format is mean (standard deviation).

Type	Model	Top-1 Acc	MRR	Hit@5
ER	SD-STGCN	**0.694** (0.017)	**0.816** (0.012)	**0.974** (0.006)
	GCN	0.302 (0.029)	0.375 (0.025)	0.450 (0.028)
	DMP	0.258 (0.009)	0.291 (0.008)	0.328 (0.006)
BA	SD-STGCN	**0.818** (0.027)	**0.892** (0.016)	**0.981** (0.005)
	GCN	0.523 (0.035)	0.598 (0.029)	0.685 (0.031)
	DMP	0.383 (0.013)	0.415 (0.011)	0.451 (0.010)
BA-Tree	SD-STGCN	**0.908** (0.042)	**0.948** (0.023)	**0.994** (0.004)
	GCN	0.753 (0.029)	0.834 (0.019)	0.935 (0.018)
	DMP	0.781 (0.016)	0.854 (0.011)	0.949 (0.014)
RGG	SD-STGCN	**0.724** (0.020)	**0.839** (0.012)	**0.984** (0.008)
	GCN	0.413 (0.041)	0.517 (0.037)	0.629 (0.043)
	DMP	0.362 (0.042)	0.439 (0.050)	0.529 (0.057)

Model reproducibility All experiments were conducted on a server with Nvidia Tesla V100-PCIE-16GB GPU. We make our datasets and code available at https://github.com/anonymous-anuthor/SD-STGCN.

A. Experiments with standard S(E)IR simulations

In this section, we examine our SD-STGCN model on standard S(E)IR simulations obeying Eq. 3 and 7 over random graphs and empirical contact networks.

SIR on random graphs We generate standard SIR simulations (Eq. 3) with $R_0 = 2.5$ (the basic reproduction number of COVID-19 [23]), $\gamma = 0.4$ on ER, BA, BA-tree, and RGG networks of $1,000$ nodes. The transmission rate β can be calculated using Eq. 4. To compare with the results in [11], we adopt a similar setup with the simulation length fixed at 30 time steps. SD-STGCN infers the source using 16 randomly sampled snapshots, while DMP and GCN use only one. We evaluate the average performance across five independent runs per graph, and five graphs per type. The results are shown in Table II. We can see that SD-STGCN outperforms DMP and GCN by a significant margin, which indicates the advantage of leveraging multiple snapshots. Like the two baselines, the performance of SD-STGCN varies across graph types: the highest top-1 accuracy ~ 0.908 in BA-Tree; the lowest ~ 0.694 in ER.

SEIR on random graphs In addition to SIR, we evaluate SD-STGCN with SEIR simulations on ER, BA, BA-Tree and RGG random graphs with $1,000$ nodes. The simulations are generated using $R_0 = 2.5$, $\gamma = 0.4$, and $\alpha = 0.5$. Note that DMP is designed for SIR and not applicable for SEIR [9], so

TABLE III: Performance of SD-STGCN and GCN trained and tested on SEIR simulations using $R_0 = 2.5$, $\gamma = 0.4$ and $\alpha = 0.5$, over random graphs of different types. The scores are evaluated over five graphs per type and five runs per graph. The format is mean (standard deviation).

Type	Model	Top-1 Acc	MRR	Hit@5
ER	SD-STGCN	**0.775** (0.028)	**0.849** (0.019)	**0.954** (0.012)
	GCN	0.126 (0.019)	0.192 (0.019)	0.252 (0.023)
BA	SD-STGCN	**0.802** (0.029)	**0.870** (0.021)	**0.959** (0.012)
	GCN	0.154 (0.024)	0.226 (0.025)	0.291 (0.033)
BA-Tree	SD-STGCN	**0.983** (0.011)	**0.990** (0.007)	**0.997** (0.003)
	GCN	0.439 (0.053)	0.583 (0.043)	0.754 (0.035)
RGG	SD-STGCN	**0.775** (0.032)	**0.846** (0.026)	**0.942** (0.025)
	GCN	0.207 (0.038)	0.319 (0.042)	0.435 (0.059)

we evaluate SD-STGCN against GCN only. The results across different types of random graphs are listed in Table III. We can see that SD-STGCN is the clear winner over GCN by more than two folds in most of the cases. We also observe a similar trend like in the SIR case, where the performance varies across graph types, with BA-Tree the easiest to predict. Overall, the results here highlight that by using multiple snapshots, our SD-STGCN outperforms GCN for not only SIR but also SEIR.

SIR on empirical contact networks Empirical contact networks can exhibit different characteristics from random graphs. For example, as shown in Table I, the empirical contact networks under consideration show higher clustering coefficients than the random graphs except for RGG, which indicate that the nodes in these empirical networks are more likely to form clusters. It is thus important to examine SD-STGCN over contagion processes on empirical contact networks.

For data generation, we run standard SIR simulations on Frat, Conf and High networks. The training set is generated using random $R_0 \in [1, 15)$ and $\gamma \in [0.1, 0.2)$ for Frat and $\gamma \in [0.1, 0.3)$ for Conf and High. The ranges are determined such that the simulations are longer than 20 iterations. The test set is generated using $R_0 = 2.5$ and $\gamma = 0.2$ for Frat and Conf, and $\gamma = 0.4$ for High. We evaluate the performance of SD-STGCN against GCN and DMP on the test set, and the results are listed in Table IV. We can see that SD-STGCN outperforms the competing methods by a significant margin, in particular SD-STGCN achieves above 90% Hit@5 rates higher than the runner-up by about 30%. This demonstrates that our method is effective for not only random graphs but also empirical contact networks.

B. Experiments with delay SIR simulations

The experiments up to this point are based on simulations that are Markovian, while in reality the disease diffusion process is better described as non-Markovian [38]. To generate non-Markovian simulations, we adopt a delay SIR model [39], [40] that follows the dynamics in Eq. 5. In particular, it assumes that a constant delay period T between infectious and recovery states. Although, in real epidemics, T can vary from one individual to another, for simplicity, we adopt a constant T representing the average length of delay across population, and most importantly, using constant T already induces non-Markovian property.

TABLE IV: Performance of SD-STGCN evaluated against baseline methods over standard SIR simulations on empirical contact networks. The scores are evaluated across five runs per network. The format is mean (standard deviation).

Data	Model	Top-1 Acc	MRR	Hit@5
Frat	SD-STGCN	**0.664** (0.002)	**0.805** (0.009)	**0.976** (0.004)
	GCN	0.457 (0.035)	0.561 (0.031)	0.670 (0.040)
	DMP	0.546 (0.161)	0.642 (0.157)	0.738 (0.180)
Conf	SD-STGCN	**0.540** (0.006)	**0.708** (0.003)	**0.926** (0.002)
	GCN	0.480 (0.038)	0.549 (0.031)	0.623 (0.027)
	DMP	0.414 (0.341)	0.448 (0.356)	0.475 (0.382)
High	SD-STGCN	**0.644** (0.022)	**0.781** (0.013)	**0.953** (0.007)
	GCN	0.408 (0.029)	0.482 (0.027)	0.564 (0.034)
	DMP	0.346 (0.310)	0.376 (0.326)	0.404 (0.359)

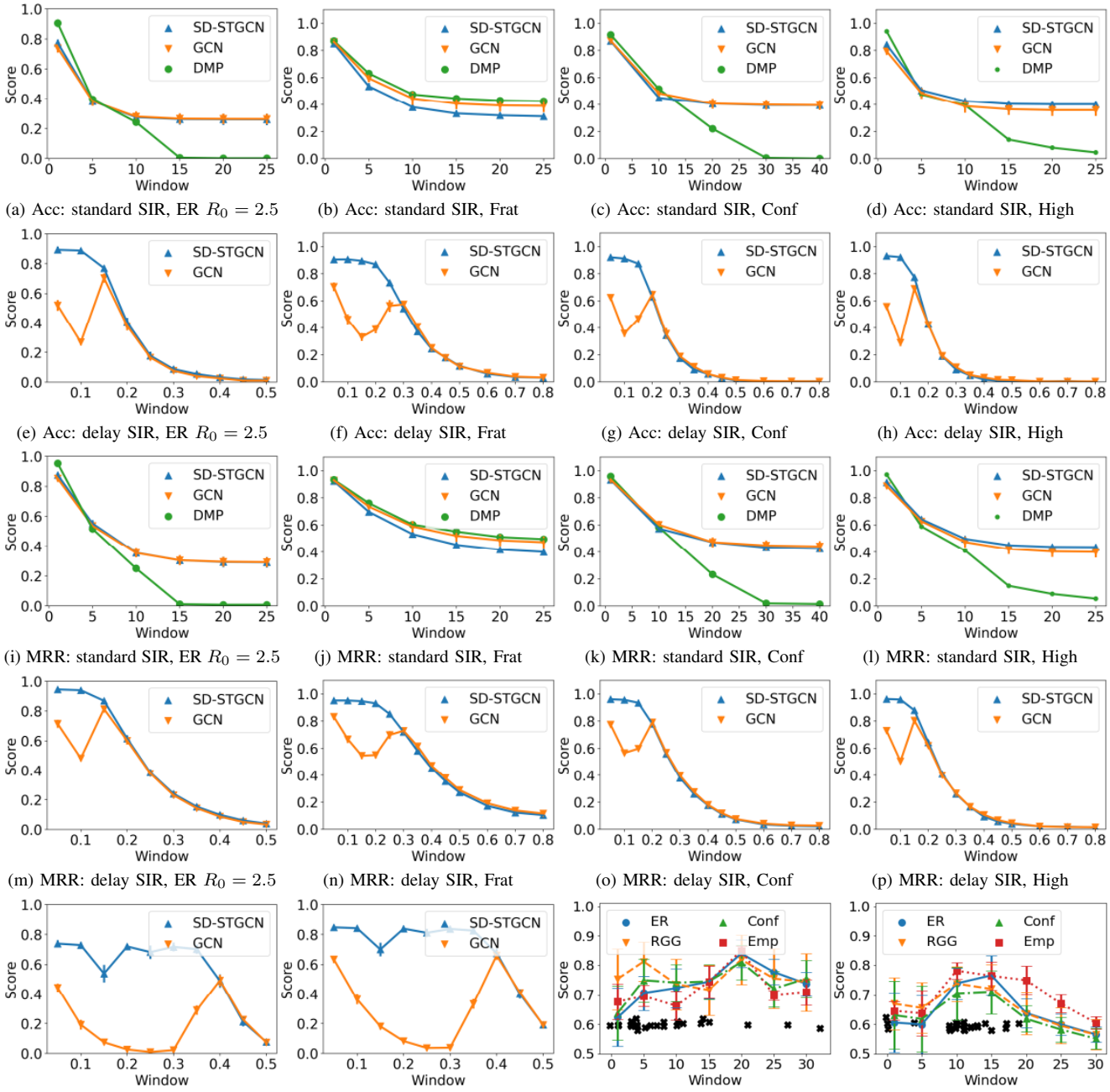
TABLE V: Performance of SD-STGCN evaluated against GCN over delay SIR simulations. The scores are evaluated across five runs per network. The format is mean (standard deviation).

Data	Model	Top-1 Acc	MRR	Hit@5
ER	SD-STGCN	**0.573** (0.165)	**0.708** (0.149)	**0.881** (0.139)
	GCN	0.130 (0.025)	0.203 (0.025)	0.285 (0.029)
Frat	SD-STGCN	**0.773** (0.012)	**0.875** (0.005)	**0.995** (0.000)
	GCN	0.231 (0.033)	0.367 (0.037)	0.504 (0.047)
Conf	SD-STGCN	**0.719** (0.007)	**0.833** (0.004)	**0.982** (0.002)
	GCN	0.145 (0.044)	0.230 (0.044)	0.306 (0.039)
High	SD-STGCN	**0.722** (0.013)	**0.828** (0.006)	**0.953** (0.005)
	GCN	0.141 (0.027)	0.214 (0.024)	0.280 (0.027)

We run delay SIR with $R_0 = 2.5$ and $T = 14$ (e.g. days) on random graphs (ER with $1,000$ nodes) and empirical contact networks (Frat, Conf, and High). Since no tractable form of DMP is known for the non-Markovian simulations [9], we only evaluate SD-STGCN against GCN. The results are shown in Table V. We can see that SD-STGCN significantly outperforms GCN. Comparing to Table II, for ER, we find that the performance of GCN reduces significantly, whereas the performance of SD-STGCN only reduces slightly. For example, GCN's top-1 accuracy drops by $\sim 56\%$, in contrast, SD-STGCN's top-1 accuracy only decreases by $\sim$ 17%. Comparing to Table IV, for Frat, Conf and High, we observe that the performance of SD-STGCN improves, whereas the performance of GCN reduces by more than 50%. Overall, we can see that SD-STGCN remains effective for non-Markovian simulations, while GCN deteriorates, which highlights the fact that SD-STGCN is better than GCN by utilizing multiple snapshots instead of only one.

C. Sliding windows

In the previous sections, we perform source detection based on randomly sampled snapshots distributed over different stages of the spread. In real life contact tracing, we may observe consecutive snapshots in a time window of limited length. Therefore, in this section, we mimic such scenario and place sliding windows across standard and delay SIR simulations on ER random graph and Frat, Conf and High contact networks, and examine SD-STGCN's performance at these windows.



(a) Acc: standard SIR, ER $R_0 = 2.5$ (b) Acc: standard SIR, Frat (c) Acc: standard SIR, Conf (d) Acc: standard SIR, High

(e) Acc: delay SIR, ER $R_0 = 2.5$ (f) Acc: delay SIR, Frat (g) Acc: delay SIR, Conf (h) Acc: delay SIR, High

(i) MRR: standard SIR, ER $R_0 = 2.5$ (j) MRR: standard SIR, Frat (k) MRR: standard SIR, Conf (l) MRR: standard SIR, High

(m) MRR: delay SIR, ER $R_0 = 2.5$ (n) MRR: delay SIR, Frat (o) MRR: delay SIR, Conf (p) MRR: delay SIR, High

(q) Acc: delay SIR, ER $R_0 = 10$ (r) MRR: delay SIR, ER $R_0 = 10$ (s) nDCG: Singapore (t) nDCG: Tianjin

Fig. 2: Top-1 accuracy and MRR across sliding windows with standard and delay SIR simulations on different networks. (a)-(d) Top-1 accuracy and (i)-(l) MRR for standard SIR simulations. The horizontal axis represents the first frame in each window. (e)-(h) Top-1 accuracy and (m)-(p) MRR for delay SIR simulations. The horizontal axis represents percentage windows. (q)-(r) Top-1 accuracy and MRR for delay SIR on ER graph with $R_0 = 10$. (s)-(t) nDCG across sliding windows on Singapore and Tianjin datasets. The black crosses represent the source cases, which are jittered to avoid overlapping.

For standard SIR, we train SD-STGCN using simulations with random $R_0 \in [1, 15)$ and $\gamma \in [0.1, 0.4)$. For testing, we generate simulations using $R_0 = 2.5$ and $\gamma = 0.4$, with a fix size of 30 iterations. We then place sliding windows of size 16 at step 1, 5, 10, 15, 20, and 25, and evaluate the top-1 accuracy and MRR. As comparison, we apply GCN and DMP at the same windows with the first frame of each window as input. The results are illustrated in Fig. 2 (a)-(d) (top-1 accuracy) and (i)-(l) (MRR). We can see that the performances of all the three models decrease as the window moving away from the starting point. Nevertheless, SD-STGCN and GCN outperform DMP significantly after 10 iterations except for the Frat case in Fig. 2 (b) where the network size (58 nodes) is smaller than the others (Table I). Also notice that SD-STGCN and

TABLE VI: COVID-19 case data network statistics. The columns from left to right are the network name, number of nodes $|V|$, number of edges $|E|$, clustering coefficient (C) [30].

network	$\|V\|$	$\|E\|$	C
Emp-HighSchool	774	7992	0.172
Singapore-ER	1000	9999 (84)	0.021 (0.000)
Singapore-RGG	1000	9463 (75)	0.601 (0.009)
Singapore-Conf	1000	10311 (79)	0.029 (0.001)
Tianjin-ER	1000	10074 (88)	0.020 (0.000)
Tianjin-RGG	1000	9439 (104)	0.589 (0.004)
Tianjin-Conf	1000	10348 (79)	0.029 (0.001)

GCN give very close scores. This is because the standard SIR simulations following Eq. 3 are Markovian - the subsequent snapshots are conditionally independent of the source given the first snapshot in the window [13]. Therefore, SD-STGCN essentially operates like a GCN in this case. In the following, we will demonstrate that this is not the case for delay SIR simulations which are non-Markovian.

For delay SIR, we run simulations using $R_0 = 2.5$ and $T = 14$. Since the simulations generated in this way have different sizes, we place windows at fix percentages rather than at fix time steps, to better represent the various stages of the spread. Note that DMP is not available for non-Markovian simulations, so we only compare SD-STGCN against GCN. The results are illustrated in Fig. 2 (e)-(h) (top-1 accuracy) and (m)-(p) (MRR). We can see that GCN suffers a performance degradation in the early stage (near the 10% window). In contrast, our SD-STGCN achieves above 80% top-1 accuracy in this period. This is because in the early stage many newly infected nodes emerge while none is yet recovered, so GCN has to pick the source out of many I nodes in one snapshot. In contrast, SD-STGCN can look ahead for multiple snapshots (e.g. 16 frames) in which the source and other early infected nodes recover, so it only needs to select the source from the fewer recovered nodes. In the later stage, as more and more nodes recover, the difference between one snapshot and multiple snapshots becomes less significant, and therefore we can see that SD-STGCN and GCN have similar performance.

To verify this reasoning, we further compare SD-STGCN against GCN on sliding windows using delay SIR with $R_0 = 10$ on ER graph. The results are shown in Fig. 2 (q)-(r). With a greater R_0, more nodes become infectious in the early stage, making it more difficult for GCN to predict the source. Therefore, we can see that the difference between the two curves here is even more significant. Before the 40% window, SD-STGCN maintains high accuracy and MRR, whereas the corresponding scores of GCN drop to near zero.

D. Case study: real COVID-19 case data

In this experiment, we assess SD-STGCN on two real world datasets of COVID-19 cases in Singapore and Tianjin. The **Singapore** dataset comprises 93 confirmed COVID-19 cases in Singapore from Jan 19, 2020 - Feb 26, 2020; the **Tianjin** dataset contains 135 confirmed cases in Tianjin, a city in the northeast of China, from Jan 21, 2020 - Feb 27, 2020 [41]. In both datasets, the initial cases were imported from Wuhan (or Hubei province), with later cases being caused by local transmission. The datasets provide temporal information like date of onset symptoms, date of confirmation, and date discharged for those who recovered (or date of death). Assuming SIR type, we use the date of onset symptoms as the step when an individual turns from S to I; while the date of onset symptoms is not available, we use the date of confirmation instead. For the recovered/death cases, we assume that they turned from I to R on the date discharged or date of death.

The datasets also provide links between the cases that are related (e.g. by family or location). We can thus connect these cases and form a network G_0. However, G_0 is likely a subset of a larger network G, as the confirmed cases may have unidentified contacts. To model this, we overlay G_0 onto a greater network G of $1,000$ nodes. Note that the size of G is arbitrary, and a different value can be used. We generate G using ER, RGG, and a configuration model [18] with the degree distribution of the High school network. In addition, we simply adopt the high school network as G. The network statics are listed in Table VI.

For training, we generate $2,000$ SIR simulations per network, using random $R_0 \in [1, 15)$ and $\gamma \in [0.1, 0.4)$. For testing, we project the states of the confirmed cases to a sequence of daily snapshots of the network (treating the unknown cases as susceptible), rendering 38 and 40 consecutive snapshots for Singapore and Tianjin, respectively. For prediction, we take 16 random snapshots as input, and rank the nodes by the probability of being the source. We take the set of the top k candidates and evaluate how much it overlaps with the set of the initial cases from Wuhan (or Hubei province). The results are shown in Table VII. Note the scores are evaluated over five runs and five networks for each graph type. As the sources here are more than one, we utilize Hit@10, Jaccard Similarity (JS), and normalized discounted cumulative gain (nDCG) as performance measure. The Hit@10 scores indicate that $\sim 60\%$ (Singapore) and $\sim 40\%$ (Tianjin) of the top 10 predictions are overlapped with the reported sources. The JS scores (between the set of sources and the set of top predictions of the same size) are around 0.4 for Singapore and 0.2 for Tianjin. The nDCG scores are above 0.7 for Singapore and 0.6 for Tianjin. We can also see that the performance does not vary significantly between different networks. In Fig. 2 (s)-(t), we plot nDCG scores at different sliding windows. The curves here are not monotonically decreasing as time increases. This is likely because multiple sources emerge at different time steps. The black crosses in Fig. 2 (s)-(t) mark the time when a source emerges. We can see that the nDCG scores are high at the steps when the source cases cluster, especially in the Tianjin case.

E. Impact of graph and simulation related factors

Effect of graph sizes We also examine SD-STGCN on ER graphs of different sizes. In addition to graphs of $1,000$ nodes, we generate ER graphs of $5,000$ and $10,000$ nodes. We train and test SD-STGCN using standard SIR simulations

TABLE VII: Performance of SD-STGCN over real COVID-19 cases. The cases are projected onto random networks generated by ER, RGG, and the Configuration (Conf) models, and an empirical contact network (Emp). The scores are evaluated across five runs and five networks per model. The format is mean (standard deviation).

		Singapore			Tianjin		
Data	Model	Hit@10	JS	nDCG	Hit@10	JS	nDCG
ER	SD-STGCN	**0.624** (0.076)	**0.398** (0.034)	**0.738** (0.058)	**0.400** (0.000)	**0.198** (0.076)	**0.644** (0.095)
	GCN	0.528 (0.151)	0.280 (0.099)	0.724 (0.089)	0.312 (0.170)	0.138 (0.077)	0.609 (0.086)
RGG	SD-STGCN	**0.632** (0.112)	**0.425** (0.044)	**0.802** (0.060)	**0.428** (0.053)	**0.219** (0.079)	**0.688** (0.102)
	GCN	0.464 (0.176)	0.248 (0.085)	0.724 (0.077)	0.216 (0.164)	0.130 (0.092)	0.579 (0.056)
Conf	SD-STGCN	**0.656** (0.070)	**0.393** (0.061)	**0.771** (0.051)	**0.400** (0.000)	**0.185** (0.090)	**0.625** (0.118)
	GCN	0.524 (0.166)	0.287 (0.099)	0.757 (0.088)	0.284 (0.138)	0.129 (0.086)	0.601 (0.085)
Emp	SD-STGCN	0.560 (0.150)	**0.396** (0.034)	0.712 (0.042)	0.400 (0.000)	**0.276** (0.013)	**0.733** (0.038)
	GCN	**0.572** (0.137)	0.261 (0.065)	**0.797** (0.062)	**0.468** (0.122)	0.193 (0.080)	0.684 (0.069)

TABLE VIII: Performance of SD-STGCN trained and tested on SIR simulations using $R_0 = 2.5$ and $\gamma = 0.4$, over ER graphs of different sizes. The scores are evaluated over five graphs per size and five runs per graph. The format is mean (standard deviation).

$\|V\|$	Top-1 Acc	MRR	Hit@5
1000	0.541 (0.038)	0.704 (0.022)	0.928 (0.018)
5000	0.532 (0.044)	0.689 (0.030)	0.896 (0.019)
10000	0.441 (0.018)	0.616 (0.016)	0.845 (0.032)

with $R_0 = 2.5$ and $\gamma = 0.4$ on these graphs. The performance metrics are shown in Table VIII. We observe a slightly decrease in performance as the graph size increases, which is understandable as the model has to pick the correct source out of more candidates. Nonetheless, SD-STGCN achieves above 44% top-1 accuracy for a network of $10,000$ nodes.

Effect of basic reproduction numbers R_0 In this experiment, we evaluate SD-STGCN with standard SIR and SEIR simulations using different R_0. [17] provides a list of estimated reproduction numbers for some well-known diseases, ranging from 1.5 (Spring wave) to 14.5 (Measles in Ghana). We thus train and test SD-STGCN over simulations with $R_0 = 1.5, 2.5,$ 5 and 10 on ER graphs of $1,000$ nodes. For fair comparison, we keep the simulation length roughly the same for different R_0 (~ 40 for SIR and ~ 70 for SEIR), by adjusting γ and α. The results are shown in Table IX. We can see that the performance reduces as R_0 increases. This is likely because when R_0 is large, the number of I nodes reaches a large value in a relatively short period of time, thus making the back-tracking more difficult.

F. Training without pre-knowledge of epidemics

In real-world scenarios, we may not know the true R_0 and γ during training. In this case, we train our model with bunch of R_0 and γ combinations in the range of the well-known diseases [17]. In specific, we train SD-STGCN on simulations generated using R_0 and γ sampled uniformly at random in $[1, 15)$ and $[0.1, 0.4)$, respectively. The range of R_0 is based on the estimated reproduction numbers of well-known diseases [17]; the range of γ is determined such that the simulations are at least 20 iterations. To evaluate the model trained in this way against that trained with known parameters, we use the same test sets described in the previous section. The results are listed in Table X. Compared to Table IX, the results here are slightly better, except for the case when $R_0 = 10$. Therefore, in practise, we can train SD-STGCN in this way and apply it to real epidemics with unknown parameters. It is worth pointing out that in some of the experiments earlier in this work, we have already adopted this approach.

TABLE IX: Performance of SD-STGCN trained and tested on SIR and SEIR simulations using different R_0. The scores are evaluated over five ER graphs and five runs per graph. The format is mean (standard deviation).

	R_0	Top-1 Acc	MRR	Hit@5
SIR	1.5	0.597 (0.035)	0.735 (0.024)	0.924 (0.008)
	2.5	0.541 (0.038)	0.704 (0.022)	0.928 (0.018)
	5	0.483 (0.029)	0.655 (0.019)	0.885 (0.024)
	10	0.369 (0.036)	0.533 (0.028)	0.735 (0.032)
SEIR	1.5	0.883 (0.019)	0.929 (0.011)	0.988 (0.004)
	2.5	0.887 (0.012)	0.933 (0.007)	0.993 (0.005)
	5	0.837 (0.021)	0.902 (0.014)	0.986 (0.006)
	10	0.814 (0.023)	0.878 (0.015)	0.965 (0.009)

TABLE X: Performance of SD-STGCN trained on SIR simulations using random R_0 and γ, and tested on simulations with different R_0. The scores are evaluated over five ER graphs and five runs per graph. The format is mean (standard deviation).

R_0	Top-1 Acc	MRR	Hit@5
1.5	0.619 (0.029)	0.749 (0.026)	0.930 (0.021)
2.5	0.568 (0.029)	0.725 (0.016)	0.931 (0.013)
5	0.504 (0.035)	0.675 (0.021)	0.896 (0.014)
10	0.359 (0.039)	0.526 (0.027)	0.738 (0.014)

VI. Conclusion

In this work, we tackled the problem of identifying the source(s) of epidemics. We considered the problem in the framework of source detection on networks and solved it using SD-STGCN - a model that extracts both spatial and temporal features of a contagion process. We conducted a series of experiments using standard and non-Markovian epidemic simulations, on synthetic and empirical contact networks. We compared our model with two state-of-the-art baselines - DMP and GCN. The results suggest that SD-STGCN outperforms the baselines for randomly sampled snapshots and consecutive snapshots at sliding windows. Lastly, we applied SD-STGCN

to two real COVID-19 cases datasets with multiple sources, and we found that the prediction was well aligned with the ground truth. For future work, we like to examine SD-STGCN on simulations not limited to fix reproduction number and extend SD-STGCN to directed acyclic graph (DAG) and subgraph with a fraction of the nodes observed.

ACKNOWLEDGMENT

This research is sponsored by the National Science Foundation (NSF) under Grant Numbers SCC-1737585, ATD-1737996, ATD-2124313, and IIS-1909916.

REFERENCES

[1] X. Tang, C. Wu, *et al.*, "On the origin and continuing evolution of SARS-CoV-2," *National Science Review*, vol. 7, pp. 1012–1023, 03 2020.

[2] D. Auerbach, W. Darrow, H. Jaffe, and J. Curran, "Cluster of cases of the acquired immune deficiency syndrome. patients linked by sexual contact," *Journal of Urology*, vol. 132, no. 2, pp. 421–421, 1984.

[3] V. Jahmunah, V. K. Sudarshan, *et al.*, "Future iot tools for covid-19 contact tracing and prediction: A review of the state-of-the-science," *International Journal of Imaging Systems and Technology*, vol. 31, no. 2, pp. 455–471, 2021.

[4] K. Farrahi, R. Emonet, and M. Cebrian, "Epidemic contact tracing via communication traces," *PloS one*, vol. 9, pp. e95133–e95133, 05 2014.

[5] J. Jiang, S. Wen, S. Yu, Y. Xiang, and W. Zhou, "Identifying propagation sources in networks: State-of-the-art and comparative studies," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 465–481, 2017.

[6] D. Shah and T. Zaman, "Detecting sources of computer viruses in networks: Theory and experiment," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, p. 203–214, June 2010.

[7] B. A. Prakash, J. Vreeken, and C. Faloutsos, "Spotting culprits in epidemics: How many and which ones?," in *2012 IEEE 12th International Conference on Data Mining*, pp. 11–20, 2012.

[8] K. Zhu and L. Ying, "Information source detection in the sir model: A sample-path-based approach," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 408–421, 2016.

[9] A. Y. Lokhov, M. Mézard, H. Ohta, and L. Zdeborová, "Inferring the origin of an epidemic with a dynamic message-passing algorithm," *Phys. Rev. E*, vol. 90, p. 012801, Jul 2014.

[10] Z. Wang, C. Wang, J. Pei, and X. Ye, "Multiple source detection without knowing the underlying propagation model," AAAI'17, p. 217–223, 2017.

[11] C. Shah, N. Dehmamy, N. Perra, M. Chinazzi, A.-L. Barabási, A. Vespignani, and R. Yu, "Finding patient zero: Learning contagion source with graph neural networks," 2020.

[12] Openreview. https://openreview.net/forum?id=xQnvyc6r3LL, 2021.

[13] Z. Wang, W. Dong, W. Zhang, and C. W. Tan, "Rumor source detection with multiple observations: Fundamental limits and algorithms," SIGMETRICS '14, p. 1–13, 2014.

[14] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[15] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *AAAI'18*, 2018.

[16] W. O. Kermack and A. G. McKendrick, "A contribution to the mathematical theory of epidemics," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 115, no. 772, pp. 700–721, 1927.

[17] P. van den Driessche, "Reproduction numbers of infectious disease models," *Infectious Disease Modelling*, vol. 2, no. 3, pp. 288–303, 2017.

[18] M. E. J. Newman, *Networks: an introduction*. 2010.

[19] L. Dell'Anna, "Solvable delay model for epidemic spreading: the case of covid-19 in italy," *medRxiv*, 2020.

[20] I. Z. Kiss, G. Röst, and Z. Vizi, "Generalization of pairwise models to non-markovian epidemics on networks," *Phys. Rev. Lett.*, vol. 115, p. 078701, Aug 2015.

[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, vol. 29, pp. 3844–3852, 2016.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[23] M. Chinazzi, J. T. Davis, *et al.*, "The effect of travel restrictions on the spread of the 2019 novel coronavirus (covid-19) outbreak," *Science*, vol. 368, no. 6489, pp. 395–400, 2020.

[24] H. Sha, M. Al Hasan, G. Mohler, and P. J. Brantingham, "Dynamic topic modeling of the COVID-19 Twitter narrative among U.S. governors and cabinet executives," *arXiv e-prints*, Apr. 2020.

[25] D. Shah and T. Zaman, "Rumors in a network: Who's the culprit?," *IEEE Trans. Inf. Theory*, vol. 57, p. 5163–5181, Aug. 2011.

[26] W. Dong, W. Zhang, and C. W. Tan, "Rooting out the rumor culprit from suspects," in *2013 IEEE International Symposium on Information Theory*, pp. 2671–2675, 2013.

[27] W. Luo, W. P. Tay, and M. Leng, "Identifying infection sources and regions in large networks," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2850–2865, 2013.

[28] B. A. Prakash, J. Vreeken, and C. Faloutsos, "Efficiently spotting the starting points of an epidemic in a large graph," *Knowl. Inf. Syst.*, vol. 38, no. 1, pp. 35–59, 2014.

[29] V. Fioriti and M. Chinnici, "Predicting the sources of an outbreak with a spectral technique," *Appl. Math. Sci.*, vol. 8, no. 135, p. 6775–6782, 2014.

[30] M. E. J. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 1, pp. 2566–2572, 2002.

[31] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002.

[32] J. Dall and M. Christensen, "Random geometric graphs," *Phys. Rev. E*, vol. 66, p. 016121, Jul 2002.

[33] B. Killworth and H. Bernard, "Informant accuracy in social network data," *Human Organization*, vol. 35, pp. 269–286, 1976.

[34] C. L. DuBois, E. S. Spiro, Z. Almquist, M. S. Handcock, D. Hunter, C. T. Butts, S. M. Goodreau, and M. Morris, "netdata: A collection of network data," 2003.

[35] M. G'enois and A. Barrat, "Can co-location be used as a proxy for face-to-face contacts?," *EPJ Data Science*, vol. 7, p. 11, May 2018.

[36] M. Salathé, M. Kazandjieva, J. W. Lee, P. Levis, M. W. Feldman, and J. H. Jones, "A high-resolution human contact network for infectious disease transmission," *Proceedings of the National Academy of Sciences*, vol. 107, no. 51, pp. 22020–22025, 2010.

[37] B. Karrer and M. E. J. Newman, "Message passing approach for general epidemic models," *Phys. Rev. E*, vol. 82, p. 016101, Jul 2010.

[38] N. Sherborne, J. C. Miller, K. B. Blyuss, and I. Z. Kiss, "Mean-field models for non-markovian epidemics on networks: from edge-based compartmental to pairwise models," *Journal of Mathematical Biology*, vol. 76, no. 3, pp. 755–778, 2018.

[39] I. Z. Kiss, J. C. Miller, and P. Simon, *(Book) Mathematics of epidemics on networks: from exact to approximate models*. Springer, 2017.

[40] J. C. Miller and T. Ting, "Eon (epidemics on networks): a fast, flexible python package for simulation, analytic approximation, and analysis of epidemics on networks," *Journal of Open Source Software*, vol. 4, no. 44, p. 1731, 2019.

[41] L. Tindale, M. Coombe, J. E. Stockdale, E. Garlock, W. Y. V. Lau, M. Saraswat, Y.-H. B. Lee, L. Zhang, D. Chen, J. Wallinga, and C. Colijn, "Transmission interval estimates suggest pre-symptomatic spread of covid-19," 2020.

APPENDIX

A. Derivation of the differential equations for network SIR

Let I_0 denote the initial infected persons at time $t = 0$, and $\mathcal{P}(t)$ denote the probability of remaining infectious at later time t after becoming infectious. $\mathcal{P}(t)$ is monotonic decreasing with $\mathcal{P}(0) = 1$ and $\lim_{t\to\infty} \mathcal{P}(t) = 0$. The probability for node i being in state I at time t is:

$$I_i(t) = I_{0,i}\mathcal{P}(t) + \beta \int_0^t \sum_j A_{ij} S_i(x) I_j(x) \mathcal{P}(t-x) dx \quad (14)$$

where β is the transmission rate. So the derivative of I_i is:

$$\begin{aligned} \frac{dI_i(t)}{dt} &= \beta \sum_j A_{ij} S_i(t) I_j(t) + I_{0,i}\mathcal{P}'(t) \\ &\quad + \beta \int_0^t \sum_j A_{ij} S_i(x) I_j(x) \mathcal{P}'(t-x) dx. \end{aligned} \quad (15)$$

As $\mathcal{P}(t)$ is non-increasing, $\mathcal{P}'(t)$ is non-positive, so the last two terms reduce the increase of infection, which corresponds to the increase of the recovered:

$$\frac{dR_i(t)}{dt} = -I_{0,i}\mathcal{P}'(t) - \beta \int_0^t \sum_j A_{ij} S_i(x) I_j(x) \mathcal{P}'(t-x) dx. \quad (16)$$

Given that $\frac{dS_i(t)}{dt} + \frac{dI_i(t)}{dt} + \frac{dR_i(t)}{dt} = 0$, we also have:

$$\frac{dS_i(t)}{dt} = -\beta \sum_j A_{ij} S_i(t) I_j(t). \quad (17)$$

Let $F_i(t)$ be the probability of node i being in either I state or R state, i.e. $F_i(t) = I_i(t) + R_i(t)$ and $S_i(t) + F_i(t) = 1$. We further have the derivative of $F_i(t)$ as:

$$\begin{aligned} \frac{dF_i(t)}{dt} &= \frac{dI_i(t)}{dt} + \frac{dR_i(t)}{dt} \\ &= \beta \sum_j A_{ij} (1 - F_i(t))(F_j(t) - R_j(t)) \end{aligned} \quad (18)$$

B. Derivation of basic reproduction number R_0

Standard SIR model Plugging $S_i(t) \approx 1$ and $\mathcal{P}(t) = e^{-\gamma t}$ in Eq. 15, $\frac{dI_i(t)}{dt}$ becomes:

$$\begin{aligned} \frac{dI_i(t)}{dt} &= \sum_j (\beta A_{ij} - \gamma \delta_{ij}) I_j(t) \\ &= \sum_j (\beta A - \gamma 1)_{ij} I_j(t) \end{aligned} \quad (19)$$

where δ_{ij} is Kronecker delta. Solving Eq. 19, we have

$$\begin{aligned} I(t) &= exp((\beta A - \gamma 1)t) I_0 \\ &= exp((\beta Q \Lambda Q^T - \gamma Q Q^T)t) I_0 \\ &= exp(Q(\beta \Lambda t - \gamma 1 t) Q^T) I_0 \\ &= Q exp(\beta \Lambda t - \gamma 1 t) Q^T I_0 \\ &\approx \psi_1 exp((\beta \lambda_1 - \gamma)t) \psi_1^T I_0 \\ &= exp((\beta \lambda_1 - \gamma)t)(\psi_1^T I_0) \psi_1 \end{aligned} \quad (20)$$

where we expand A using the eigenvalue decomposition $A = Q\Lambda Q^T$ with Q and Λ being the eigen-vector and eigen-value matrices. λ_1 and ψ_1 are the largest eigen-value and the corresponding eigen-vector, respectively. Eq. 20 gives the basic reproduction number

$$R_0 = \frac{\beta \lambda_1}{\gamma}, \quad (21)$$

and we can see that when $R_0 > 1$ the disease will spread to form an epidemic.

Delay SIR model Plugging $\mathcal{P}(t) = \Theta(t - T)$, a step function, with $\mathcal{P}(t) = 1$ for $0 \le t \le T$ and $\mathcal{P}(t) = 0$ for $t > T$ in Eq. 15 and combing the result with $\frac{dF_i(t)}{dt} = \frac{dR_i(t)}{dt} + \frac{dI_i(t)}{dt} = \beta \sum_j A_{ij} S_i(t) I_j(t)$, we have $\frac{dR_i(t)}{dt} = \frac{dF_i(t-T)}{dt}$, which leads to $R_i(t) = F_i(t) + C$ where C is a constant. Since $R_i(t)$ and $F_i(t)$ are both monotonic increasing and $F_i(t)$ saturates at $t \to +\infty$, therefore $C = 0$, namely $R_i(t) = F_i(t - T)$. Plugging this into Eq. 18, we have

$$\frac{dF_i(t)}{dt} = T\beta \sum_j A_{ij} \frac{F_j(t) - F_j(t-T)}{T}, \quad (22)$$

or equivalently

$$\begin{aligned} \frac{dF(t)}{dt} &= T\beta Q \Lambda Q^T \frac{F(t) - F(t-T)}{T} \\ &\approx T\beta \psi_1 \lambda_1 \psi_1^T \frac{F(t) - F(t-T)}{T} \\ &= T\beta \lambda_1 (\psi_1^T \frac{F(t) - F(t-T)}{T}) \psi_1, \end{aligned} \quad (23)$$

which gives the basic reproduction number

$$R_0 = \beta \lambda_1 T. \quad (24)$$