References

[1] Olalekan Adeyinka. Internet attack methods and internet security technology. In *2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pages 77–82, 2008.

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[3] Suman Banerjee, Mamata Jenamani, and Dilip Kumar Pratihar. A survey on influence maximization in a social network. *Knowledge and Information Systems*, 62:3417–3455, 2020.

[4] J. Behrmann, W. Grathwohl, Rtq Chen, D. Duvenaud, and J. H. Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, 2019.

[5] Marián Boguná, Romualdo Pastor-Satorras, Albert Díaz-Guilera, and Alex Arenas. Models of social networks based on social distance attachment. *Physical review E*, 70(5):056122, 2004.

[6] Zongmai Cao, Kai Han, and Jianfu Zhu. Information diffusion prediction via dynamic graph neural networks. In *2021 IEEE 24th international conference on computer supported cooperative work in design (CSCWD)*, pages 1099–1104. IEEE, 2021.

[7] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

[8] Gojko Čutura, Boning Li, Ananthram Swami, and Santiago Segarra. Deep demixing: Reconstructing the evolution of epidemics using graph neural networks. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 2204–2208. IEEE, 2021.

[9] Ming Dong, Bolong Zheng, Nguyen Quoc Viet Hung, Han Su, and Guohui Li. Multiple rumor source detection with graph convolutional networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 569–578, 2019.

[10] Fei Gao, Jiang Zhang, and Yan Zhang. Neural enhanced dynamic message passing. In *International Conference on Artificial Intelligence and Statistics*, pages 10471–10482. PMLR, 2022.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[12] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[13] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[14] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*, 2022.

[15] Qiang He, Hui Fang, Jie Zhang, and Xingwei Wang. Dynamic opinion maximization in social networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):350–361, 2021.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[18] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[19] William Ogilvy Kermack and Anderson 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*, 115(772):700–721, 1927.

[20] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017.

[21] Daniel B Larremore, Bailey K Fosdick, Kate M Bubar, Sam Zhang, and Yonatan H Grad. Estimating sars-cov-2 seroprevalence and epidemiological parameters with uncertainty from serological surveys. *eLife Sciences*, 10, 2021.

[22] Chen Ling, Junji Jiang, Junxiang Wang, and Zhao Liang. Source localization of graph diffusion via variational autoencoders for graph inverse problems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1010–1020, 2022.

[23] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

[24] Wuqiong Luo and Wee Peng Tay. Estimating infection sources in a network with incomplete observations. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 301–304. IEEE, 2013.

[25] Wuqiong Luo and Wee Peng Tay. Estimating infection sources in a network with incomplete observations. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 301–304. IEEE, 2013.

[26] Wuqiong Luo, Wee Peng Tay, and Mei Leng. How to identify an infection source with limited observations. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):586–597, 2014.

[27] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[28] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[29] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[30] B Aditya Prakash, Jilles Vreeken, and Christos Faloutsos. Spotting culprits in epidemics: How many and which ones? In *2012 IEEE 12th International Conference on Data Mining*, pages 11–20. IEEE, 2012.

[31] Maciel M Queiroz, Dmitry Ivanov, Alexandre Dolgui, and Samuel Fosso Wamba. Impacts of epidemic outbreaks on supply chains: mapping a research agenda amid the covid-19 pandemic through a structured literature review. *Annals of operations research*, pages 1–38, 2020.

[32] Jonathan M. Read, Jessica R. E. Bridgen, Derek A. T. Cummings, Antonia Ho, and Chris P. Jewell. Novel coronavirus 2019-ncov (covid-19): early estimation of epidemiological parameters and epidemic size estimates. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 376(1829):20200265–, 2021.

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[34] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[35] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[36] Hao Sha, Mohammad Al Hasan, and George Mohler. Source detection on networks using spatial temporal graph convolutional networks. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–11. IEEE, 2021.

[37] Chintan Shah, Nima Dehmamy, Nicola Perra, Matteo Chinazzi, Albert-László Barabási, Alessandro Vespignani, and Rose Yu. Finding patient zero: Learning contagion source with graph neural networks. *CoRR*, abs/2006.11913, 2020.

[38] Xincheng Shu, Bin Yu, Zhongyuan Ruan, Qingpeng Zhang, and Qi Xuan. Information source estimation with multi-channel graph neural network. In *Graph Data Mining*, pages 1–27. Springer, 2021.

[39] Tristan SW Stevens, Jean-Luc Robert, Faik C Yu, Jun Seob Shin, and Ruud JG van Sloun. Removing structured noise with diffusion models. *arXiv preprint arXiv:2302.05290*, 2023.

[40] Xiaolu Tang, Changcheng Wu, Xiang Li, Yuhe Song, Xinmin Yao, Xinkai Wu, Yuange Duan, Hong Zhang, Yirong Wang, Zhaohui Qian, et al. On the origin and continuing evolution of sars-cov-2. *National science review*, 7(6):1012–1023, 2020.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[42] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

[43] Junxiang Wang, Junji Jiang, and Liang Zhao. An invertible graph diffusion neural network for source localization. In *Proceedings of the ACM Web Conference 2022*, pages 1058–1069, 2022.

[44] Zheng Wang, Chaokun Wang, Jisheng Pei, and Xiaojun Ye. Multiple source detection without knowing the underlying propagation model. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 217–223. AAAI Press, 2017.

[45] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[46] Wenwen Xia, Yuchen Li, Jun Wu, and Shenghong Li. Deepis: Susceptibility estimation on social networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 761–769, 2021.

[47] KiJung Yoon, Renjie Liao, Yuwen Xiong, Lisa Zhang, Ethan Fetaya, Raquel Urtasun, Richard Zemel, and Xaq Pitkow. Inference in probabilistic graphical models by graph neural networks. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 868–875. IEEE, 2019.

[48] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

[49] Wenyu Zang, Peng Zhang, Chuan Zhou, and Li Guo. Locating multiple sources in social networks under the sir model: A divide-and-conquer approach. *Journal of Computational Science*, 10:278–287, 2015.

[50] Laijun Zhao, Hongxin Cui, Xiaoyan Qiu, Xiaoli Wang, and Jiajia Wang. Sir rumor spreading model in the new media age. *Physica A: Statistical Mechanics and its Applications*, 392(4):995–1003, 2013.

[51] Linhua Zhou and Meng Fan. Dynamics of an sir epidemic model with limited medical resources revisited. *Nonlinear Analysis: Real World Applications*, 13(1):312–324, 2012.

[52] Kai Zhu, Zhen Chen, and Lei Ying. Catch'em all: Locating multiple diffusion sources in networks with partial observations. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1676–1682. AAAI Press, 2017.

[53] Linhe Zhu, Gui Guan, and Yimin Li. Nonlinear dynamical analysis and control strategies of a network-based sis epidemic model with time delay. *Applied Mathematical Modelling*, 70:512–531, 2019.

[54] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018.

A Broader impacts

Overall, our work offers valuable insights into how to limit the spread of malicious information. For example, by tracing the spread of infectious diseases, we can identify potential contacts of infected individuals, effectively containing the outbreak. However, it is important to consider the potential negative implications of this approach for society, such as compromising the privacy of individuals living with infectious diseases like HIV. Addressing these concerns should be important.

B Derivation of forward and backward processes

Derivation of forward processes. Extending Equation 4 by means of the Markov property:

$$\begin{aligned}
q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_0^i\right) &= \sum_{\boldsymbol{x}_{1:t-1}^i} \prod_{k=1}^{t} q\left(\boldsymbol{x}_k^i \mid \boldsymbol{x}_{k-1}^i\right) \\
&= \sum_{\boldsymbol{x}_{1:t-1}^i} \prod_{k=1}^{t} \boldsymbol{x}_{k-1}^i Q_k^i \boldsymbol{x_k^i}^T \\
&= \sum_{\boldsymbol{x}_{1:t-1}^i} \boldsymbol{x}_0^i Q_1^i \boldsymbol{x_1^i}^T \cdots \boldsymbol{x_{k-1}^i} Q_k^i \boldsymbol{x_k^i}^T \cdots \boldsymbol{x}_{t-1}^i Q_t^i \boldsymbol{x_t^i}^T \\
&= \boldsymbol{x}_0^i Q_1^i \left(\sum_{\boldsymbol{x}_1^i} \boldsymbol{x_1^i}^T \boldsymbol{x}_1^i\right) \cdots \left(\sum_{\boldsymbol{x}_{t-1}^i} \boldsymbol{x_{t-1}^i}^T \boldsymbol{x}_{t-1}^i\right) Q_t^i \boldsymbol{x_t^i}^T \\
&= \boldsymbol{x}_0^i Q_1^i I Q_2^i \cdots I Q_k^i I \cdots I Q_t^i \boldsymbol{x_t^i}^T \\
&= \boldsymbol{x}_0^i \bar{Q}_t^i \boldsymbol{x_t^i}^T \sim \operatorname{Cat}\left(\boldsymbol{x}_t^i ; \boldsymbol{p}=\boldsymbol{x}_0^i \bar{Q}_t^i\right)
\end{aligned} \tag{18}$$

Where I is the identity matrix.

Derivation of backward processes. The detailed derivation of Equation 6 is as follows:

$$\begin{aligned}
q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{x}_0^i\right) &= \frac{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_{t-1}^i, \boldsymbol{x}_0^i\right) q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_0^i\right)}{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_0^i\right)} \\
&= \frac{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_{t-1}^i\right) q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_0^i\right)}{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_0^i\right)} \\
&= \frac{\boldsymbol{x}_{t-1}^i Q_t^i \boldsymbol{x_t^i}^T \cdot \boldsymbol{x}_0^i \bar{Q}_{t-1}^i \boldsymbol{x_{t-1}^i}^T}{\boldsymbol{x}_0^i \bar{Q}_t^i \boldsymbol{x_t^i}^T} \\
&= \frac{\left(\boldsymbol{x}_t^i {Q_t^i}^T \boldsymbol{x_{t-1}^i}^T\right) \cdot \left(\boldsymbol{x}_0^i \bar{Q}_{t-1}^i \boldsymbol{x_{t-1}^i}^T\right)}{\boldsymbol{x}_0^i \bar{Q}_t^i \boldsymbol{x_t^i}^T} \\
&= \frac{\left(\boldsymbol{x}_t^i {Q_t^i}^T\right) \odot \left(\boldsymbol{x}_0^i \bar{Q}_{t-1}^i\right) \left(\boldsymbol{x_{t-1}^i}^T\right)}{\boldsymbol{x}_0^i \bar{Q}_t^i \boldsymbol{x_t^i}^T} \\
&\sim \operatorname{Cat}\left(\boldsymbol{x}_{t-1}^i ; \boldsymbol{p} = \frac{\left(\boldsymbol{x}_t^i {Q_t^i}^T\right) \odot \left(\boldsymbol{x}_0^i \bar{Q}_{t-1}^i\right)}{\boldsymbol{x}_0^i \bar{Q}_t^i \boldsymbol{x_t^i}^T}\right)
\end{aligned} \tag{19}$$

From the Bayesian formula, it follows that:

$$
\begin{aligned}
q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{x}_0^i\right) = q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i\right) &= \frac{\sum_{\boldsymbol{x}_0^i} q\left(\boldsymbol{x}_{t-1}^i, \boldsymbol{x}_t^i, \boldsymbol{x}_0^i\right)}{q\left(\boldsymbol{x}_t^i\right)} \\
&= \frac{\sum_{\boldsymbol{x}_0^i} q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{x}_0^i\right) q\left(\boldsymbol{x}_0^i \mid \boldsymbol{x}_t^i\right) q\left(\boldsymbol{x}_t^i\right)}{q\left(\boldsymbol{x}_t^i\right)} \\
&= \sum_{\boldsymbol{x}_0^i} q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{x}_0^i\right) q\left(\boldsymbol{x}_0^i \mid \boldsymbol{x}_t^i\right) \\
&= \mathbb{E}_{q\left(\boldsymbol{x}_0^i \mid \boldsymbol{x}_t^i\right)} q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{x}_0^i\right)
\end{aligned}
\tag{20}
$$

Fitting this distribution using a neural network:

$$
\begin{aligned}
q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i\right) \approx p_\theta\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i\right) &= \mathbb{E}_{\boldsymbol{x}_0^i \sim p_\theta\left(\boldsymbol{x}_0^i \mid \boldsymbol{x}_t^i\right)} q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_t^i, \boldsymbol{x}_0^i\right) \\
&= \mathbb{E}_{p_\vartheta\left(\boldsymbol{x}_0^i \mid \boldsymbol{x}_t^i\right)} \frac{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_{t-1}^i, \boldsymbol{x}_0^i\right) q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_0^i\right)}{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_0^i\right)} \\
&= \mathbb{E}_{p_0\left(\boldsymbol{x}_0^i \mid \boldsymbol{x}_t^i\right)} \frac{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_{t-1}^i\right) q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_0^i\right)}{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_0^i\right)} \\
&= \frac{\sum_j \left[q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_{t-1}^i\right) q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_{\mathbf{0}}^{\boldsymbol{i}\,(j)}\right) p_\theta\left(\boldsymbol{x}_{\mathbf{0}}^{\boldsymbol{i}\,(j)} \mid \boldsymbol{x}_t^i\right)\right]}{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_0^i\right)} \\
&= \frac{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_{t-1}^i\right) \left[\sum_j q\left(\boldsymbol{x}_{t-1}^i \mid \boldsymbol{x}_{\mathbf{0}}^{\boldsymbol{i}\,(j)}\right) p_\theta\left(\boldsymbol{x}_{\mathbf{0}}^{\boldsymbol{i}\,(j)} \mid \boldsymbol{x}_t^i\right)\right]}{q\left(\boldsymbol{x}_t^i \mid \boldsymbol{x}_0^i\right)}
\end{aligned}
\tag{21}
$$

C Propagation rule constraint of information diffusion reconstruction

To investigate the circumstances under which the propagation rules may be violated, let's revisit Equation 6. Note that the denominator serves as the normalization term, while the numerator is composed of two key terms - $\left(\boldsymbol{x}_t^i {Q_t^i}^T\right)$ and $\left(\boldsymbol{x}_0^i \bar{Q}_{t-1}^i\right)$ - that are crucial in preserving the propagation rule.

The three rows of ${Q_t^i}^T$ correspond to the distribution of $q(x_t^i|x_{t-1}^i, x_0^i)$ when x_t^i is in one of three states: S, I, and R. Similarly, the three rows of $\bar{Q}_t^i$ represent the distribution of $q(x_{t-1}^i|x_0^i)$ when x_0^i is in one of three states: S, I, and R. Let $\left[\bar{Q}_t^i\right]_{12}$ and $\left[\bar{Q}_t^i\right]_{13}$ be denoted by q_t^a and q_t^b respectively, then $q_1^a = \gamma_1^i, q_1^b = 0$.

$$
\bar{Q}_t^i = \begin{bmatrix} \prod_{k=1}^t (1-\beta_k^i) & \prod_{k=1}^{t-1}(1-\beta_k^i)\beta_t^i + q_{t-1}^a(1-\gamma_t^i) & q_{t-1}^a(\gamma_t^i) + q_{t-1}^b \\ 0 & \prod_{k=1}^t (1-\gamma_k^i) & \sum_{j=1}^t \prod_{k=1}^{t-j}(1-\gamma_k^i)\gamma_{k+1}^i \\ 0 & 0 & 1 \end{bmatrix}
\tag{22}
$$

When x_t^i and x_0^i are in different states, the results of $q(x_{t-1}^i|x_t^i, x_0^i) = \left(\boldsymbol{x}_t^i {Q_t^i}^T\right) \odot \left(\boldsymbol{x}_0^i \bar{Q}_{t-1}^i\right)$ are shown in Table 4. The table reveals that $q(x_{t-1}^i|x_t^i, x_0^i) = [0, 0, 0]$ when the propagation rule is violated. Since we are predicting x_0^i, there are only two possible states for x_0^i: S (Susceptible) and I (Infected), with $x_0^i = R$ (Recovered) being excluded. In such instances, $q(x_{t-1}^i|x_t^i, x_0^i)$ can be set to $[1, 0, 0]$ to resolve the issue. Furthermore, if $x_t^i = R$ and $x_0^i = S$, then $x_{t-1}^i = S$ would violate the propagation rules. However, as shown in the $9th$ row of Table 4, the probability of $x_{t-1}^i = S$ is much smaller than that of $x_{t-1}^i = I$ and $x_{t-1}^i = R$, and such situations will not lead to training failure. Hence, there is no need for any specific handling of this scenario.

To further minimize propagation rule violations during the training process, we incorporate supervision of the propagation rule. Specifically, when using this supervision function, nodes that have a state of R are set to I to enforce the propagation rule.

$$
L_{constrain1} = Relu\left(\mathbf{X_{t-1}} - (\mathbf{A} + \mathbf{I})\mathbf{X_0}\right)
\tag{23}
$$

Table 4: The distribution of the unnormalized $q(x_{t-1}^i | x_t^i, x_0^i)$ in different cases.

x_t^i	x_0^i	$q(x_{t-1}^i \| x_t^i, x_0^i)$		
		S	I	R
S	S	$\prod_{k=1}^{t}(1-\beta_k^i)\prod_{k=1}^{t-1}(1-\beta_k^i)$	0	0
	I	0	0	0
	R	0	0	0
I	S	$q_t^a \prod_{k=1}^{t-1}(1-\beta_k^i)$	$\prod_{k=1}^{t}(1-\gamma_k^i)q_{t-1}^a$	0
	I	0	$\prod_{k=1}^{t}(1-\gamma_k^i)\prod_{k=1}^{t-1}(1-\gamma_k^i)$	0
	R	0	0	0
R	S	$q_t^b \prod_{k=1}^{t-1}(1-\beta_k^i)$	$q_{t-1}^a \sum_{j=1}^{t}\prod_{k=1}^{t-j}(1-\gamma_k^i)\gamma_{k+1}^i$	q_{t-1}^b
	I	0	$\sum_{j=1}^{t}\prod_{k=1}^{t-j}(1-\gamma_k^i)\gamma_{k+1}^i \cdot \prod_{k=1}^{t-1}(1-\gamma_k^i)$	$\sum_{j=1}^{t}\prod_{k=1}^{t-j}(1-\gamma_k^i)\gamma_{k+1}^i$
	R	0	0	1

where $(\mathbf{A}+\mathbf{I})\mathbf{X_0}$ represents the total number of infected nodes in a given node's first-order neighborhood. Equation 23 penalizes $\mathbf{X_0}$ if the node is infected and there are no other infected nodes within its first-order neighborhood. To maintain the stability of inferred $\mathbf{X_0}$ generating $\mathbf{X_{t-1}}$, we apply the same constraint to the process. Specifically, we utilize the monotonicity regularization of information diffusion from [22]. If the source set $\mathbf{X_0^{(i)}}$ is a superset of $\mathbf{X_0^{(j)}}$, then the generated $\mathbf{X_{t-1}}$ resulting from their diffusion needs to satisfy the following equation.

$$L_{constrain2} = \left\| \max\left(\mathbf{0}, \mathbf{X_{t-1}^{(j)}} - \mathbf{X_{t-1}^{(i)}}\right) \right\|^2, \forall \mathbf{X_0^{(i)}} \supseteq \mathbf{X_0^{(j)}} \tag{24}$$

D Proofs of lemmas and theorems

D.1 The proof of theorem 3.1

Proof *Set the initial infection seed set as:* $\mathbb{S} = \{x_0^{s_1}, x_0^{s_2}, \dots x_0^{s_m}\}$. *At the initial moment, the infection status distribution of node* i *is:*

$$\begin{cases} P_S^{i=S_m}(0) = 0, P_I^{i=S_m}(0) = 1 \\ P_S^{i\neq S_m}(0) = 1, P_I^{i\neq S_m}(0) = 0 \\ P_R^i(0) = 0 \end{cases} \tag{25}$$

At time t*, the infection status distribution of node* i *is:*

$$\begin{cases} P_I^i(t) = P_I^i(t-1)(1-\gamma_R^i(t-1)) + P_S^i(t-1)\left[1-\prod_j(1-\beta_I^j(t))A_{ij}P_I^j(t-1)\right] \\ P_S^i(t) = P_S^i(t-1)\left[\prod_j(1-\beta_I^j(t))A_{ij}P_I^j(t-1)\right] \\ P_R^i(t) = P_I^i(t-1)\gamma_R^i(t) \end{cases} \tag{26}$$

where A *is the adjacency matrix and* j *is the neighbor of node* i*. When dealing with a static graph* $\mathcal{G}$*,* A *is fixed, allowing for determination of the state distribution at any time based on the initial node state. Specifically, if both graph G and the seed node set* $\mathbb{S}$ *are known, it becomes possible to calculate the state distribution of each node at any given time using Equation 26. Utilizing Equations 2 and 3,* Q_t^i *can also be determined.*

D.2 Proof of lemma 3.1

Proof *The graph convolution layer with batch normalization* $\mathbf{BN}(g(\xi))$ *can be abbreviated as* $GConv$*. In our approach, we apply spectral normalization to both the linear transformation* $\mathbf{U}$ *and convolutional layers* $GConv$*. As a result, the weight parameters of both the networks* f_w *and* $GConv$ *possess 1-Lipschitz continuity after spectral normalization, as described in [28].*

Let $GConv : \mathbb{R}^n \to \mathbb{R}^m$, $x_1, x_2 \in \mathbb{R}^n$. Note that the nonlinear activation function $\sigma(\cdot)$ is set to $Mish(\cdot)$, which obviously possesses 1-Lipschitz continuity.

$$\begin{aligned}
&\|f_\theta(x_1) - f_\theta(x_2)\|_p = \|\sigma(GConv(x_1)) - \sigma(GConv(x_2))\|_p \\
&\le \|GConv(x_1) - GConv(x_2)\|_p \\
&< \|x_1 - x_2\|_p
\end{aligned} \tag{27}$$

where $\|\cdot\|_p$ represents the p-norm ($p = 2$ or $p = \infty$). Therefore, the Lipschitz constant of f_θ is less than 1.

D.3 Proof of theorem 3.2

Proof *To prove that $\mathbf{Y_T} = \mathbf{F}(f_w(\mathbf{X_0}))$ is reversible, it is necessary to ensure that F and f_w are reversible [43].*

$$\begin{cases} f_w(\mathbf{X_0}) = \mathbf{X_0} + f_w(\mathbf{X_0}) - \mathbf{X_0} = \xi \\ \xi + f_\theta(\xi) = \mathbf{Y_T} \end{cases} \Leftrightarrow \begin{cases} \mathbf{X_0} = \xi + \mathbf{X_0} - f_w(\mathbf{X_0}) \\ \xi = \mathbf{Y_T} - f_\theta(\xi) \end{cases} \tag{28}$$

We construct the following iterative formula:

$$\begin{cases} \mathbf{X_0^{k+1}} = \xi + \mathbf{X_0^k} - f_w(\mathbf{X_0^k}), \mathbf{X_0^0} = \xi \\ \xi^{k+1} = \mathbf{Y_T} - f_\theta(\xi^k), \xi^0 = \mathbf{Y_T} \end{cases} \Rightarrow \begin{cases} \lim_{k\to\infty} \mathbf{X_0^k} = \mathbf{X_0} \\ \lim_{k\to\infty} \xi^k = \xi \end{cases} \tag{29}$$

By Lemma 2, we can ensure that $L_w < 1$ and $L_\theta < 1$. Moreover, the Lipschitz constant of $(\mathbf{X_0^k} - f_w(\mathbf{X_0^k})$ is $1 - L_w$, which is less than 1. Thus, when the number of iterations k is sufficiently large, it follows that the transformation $\mathbf{Y_T} = \mathbf{F}(f_w(\mathbf{X_0}))$ is reversible according to the Banach fixed point theorem [4].

D.4 Proof of theorem 3.3

Proof *Specifically, Theorem 3.2 proves that $(\mathbf{F_1} \circ \mathbf{F_2} \circ \ldots \circ \mathbf{F_n})(\xi)$ is invertible when $n = 1$. Therefore, we are currently examining whether $(\mathbf{F_1} \circ \mathbf{F_2} \circ \ldots \circ \mathbf{F_n})(\xi)$ retains its reversibility for $n > 1$. To make the notation simpler, we denote $(\mathbf{F_1} \circ \mathbf{F_2} \circ \ldots \circ \mathbf{F_i})(\xi)$ as $\hat{\mathbf{F_i}}$.*

$$\hat{\mathbf{F_n}} = (\mathbf{F_1} \circ \mathbf{F_2} \circ \ldots \circ \mathbf{F_n})(\xi) = \underbrace{\mathbf{DP} \circ \cdots \circ \mathbf{DP}}_{n} \left[\xi + f_w(\xi) + \sum_{i=1}^{n-1} f_w \left(\hat{\mathbf{F_i}}(\xi) \right) \right] \tag{30}$$

The application of the dropout function $\mathbf{DP}$ will limit the Lipschitz constant of any function f to $1 - r$ times its original value: $L_{\mathbf{DP}(f)} = (1 - r)L_f$. Additionally, we have $L_{\hat{\mathbf{F_i}}} \le \prod_{j=1}^{i} \left(L_{F_j} \right) \le (1 - r)^i (1 + L_{f_w})^i$ [13]. Therefore, the Lipschitz constant of $\hat{\mathbf{F_n}}$ is expressed as:

$$\begin{aligned}
L_{\hat{\mathbf{F_n}}} &\le (1 - r)^n \left[1 + L_{f_w} + \sum_{i=1}^{n-1} L_{f_w} \cdot L_{\hat{\mathbf{F_i}}} \right] \\
&\le (1 - r)^n \left[1 + L_{f_w} \sum_{i=0}^{n-1} [(1 - r)(1 + L_{f_w})]^i \right] \\
&= (1 - r)^n \left[1 + L_{f_w} \cdot \frac{1 - [(1 - r)(1 + L_{f_w})]^n}{1 - (1 - r)(1 + L_{f_w})} \right]
\end{aligned} \tag{31}$$

Note that when $L_{f_w} < 1$ and $n > 1$, we only need to set $r = 1/2$ to ensure that $L_{\hat{\mathbf{F_n}}} < 1$, hereby guaranteeing that $\hat{\mathbf{F_n}}$ is reversible.

E DDMSL implementation details

The DDMSL approach has been previously explained, and we will now provide further details on the implementation of DDMSL. The linear transform $\mathbf{U}$ refers to a fully connected layer, and in Figure 2, the dense layer is composed of two fully connected layers that undergo spectral normalization. The final output of the nn_θ is represented by an $N \times 1$ matrix, indicating the probability that each node is in an infected state at $t = 0$. We designate nodes with an infection probability higher than a certain threshold as infected nodes.

Given a complete information diffusion instance $\mathbf{X} = \{\mathbf{X_0}, \ldots, \mathbf{X_T}\}$ where $\mathbf{X_t} = \{x_t^1, \ldots, x_t^N\}$, we sample $t \in \{1, \ldots, T\}$ to be included in the neural network nn_θ based on the probability distribution $p(t) \sim \frac{t}{\sum_{t=1}^{T} t}$, and use the sine-cosine position encoding [41] to embed t. The training and inference processes are shown in Algorithm 1 and Algorithm 2, respectively. Additionally, the variable T remains consistent with the information diffusion step size.

Algorithm 1: Training

```
Input: $\mathbf{X_0}$, $\mathcal{G}$, $\mathbf{X_t}$, threshold $\alpha$
1  repeat
2  $\mathbf{Q_t} \leftarrow$ Equation 2 and Equation 3
3  $q(x_{t-1}|x_t) \leftarrow$ Calculate Equation 6 using $\mathbf{X_t}$ and $\mathbf{Q_t}$
4  $t \sim \mathrm{P}(\{1, \ldots, T\}), P(t) = \frac{t}{\sum_{t=1}^{T} t}$
5  $\mathbf{X_0^{t-1}} = nn_\theta(\mathcal{G}, \mathbf{X_t}, t)$
   // Using data from $t$, $nn_\theta$ infers the source node $\mathbf{X_0^{t-1}}$, which is then
      used to reconstruct the diffusion graph at $t-1$.
6  $\mathbf{X_0^{t-1}}[\mathbf{X_0^{t-1}} > \alpha] = 1$
7  $\mathbf{X_0^{t-1}}[\mathbf{X_0^{t-1}} != 1] = 0$
8  $\mathbf{Q_t^{'}} \leftarrow$ Equation 2 and Equation 3
   // $\mathbf{Q_t^{'}}$ is generated by $\mathbf{X_0^{t-1}}$.
9  $P_\theta(x_{t-1}|x_t) \leftarrow$ Calculate Equation 8 using $\mathbf{X_t}$, $\mathbf{X_0^{t-1}}$ and $\mathbf{Q_t^{'}}$
10 $\mathbf{X_{t-1}} \leftarrow Gumbel - Softmax(P_\theta(x_{t-1}|x_t))$
11 Take gradient descent step on:
12 $\nabla_\theta(L_{simple} + L_{constrain})$
```

Algorithm 2: Infering

```
Input: $\mathbf{X_t}$, $\mathcal{G}$, Empty set $\mathbb{X}$, threshold $\alpha$
Output: $\mathbb{X}$
1  for $t$ starts from $T$ to $1$ do
2      $\mathbf{X_0^{t-1}} = nn_\theta(\mathcal{G}, \mathbf{X_t}, t)$
3      $\mathbf{X_0^{t-1}}[\mathbf{X_0^{t-1}} > \alpha] = 1$
4      $\mathbf{X_0^{t-1}}[\mathbf{X_0^{t-1}} != 1] = 0$
5      $\mathbf{Q_t^{'}} \leftarrow$ Equation 2 and Equation 3
6      $P_\theta(x_{t-1}|x_t) \leftarrow$ Calculate Equation 8 using $\mathbf{X_t}$, $\mathbf{X_0^{t-1}}$ and $\mathbf{Q_t^{'}}$
7      $\mathbf{X_{t-1}} \leftarrow Gumbel - Softmax(P_\theta(x_{t-1}|x_t))$
8      $\mathbb{X}[t] \leftarrow \mathbf{X_{t-1}}$
9  end
```

In Algorithm 1, we have $\mathbf{Q_t} = \{Q_t^1, \ldots, Q_t^N\}$, where Q_t^i can be calculated using Equations 2 and 3. Additionally, the $P_S^i(t)$, $P_I^i(t)$, and $P_R^i(t)$ in Equation 3 can be computed using various methods. Monte Carlo simulations provide the most accurate results, but require at least 10^5 simulations to be sufficiently precise, leading to a high time complexity. An alternative approach is to utilize a neural network model [10, 47] to learn $\left[P_S^i(t), P_I^i(t), P_R^i(t)\right]$, which significantly reduces the training time of the model. When applied to the SI model, DDMSL utilizes the state transfer matrix by:

$$Q_t^i = \begin{bmatrix} 1 - \beta_I^i(t) & \beta_I^i(t) \\ 0 & 1 \end{bmatrix} \tag{32}$$

where $\mathbf{X_0^{t-1}}$ represents the predicted source node using $\mathbf{X_t}$, while $Q_t^{'}$ is generated using the same method as above. Ultimately, $P_\theta(x_{t-1}|x_t)$ is calculated using $\mathbf{X_0^{t-1}}$, $\mathbf{X_t}$, and $\mathbf{X_0^{t-1}}$. We obtain $\mathbf{X_{t-1}}$ by sampling from $P_\theta(x_{t-1}|x_t)$, and label nodes as $0(S)$, $1(I)$, or $2(R)$ based on their state. Algorithm 2 proceeds in a similar manner to the process described above.

F Additional algorithms and dataset parameters

F.1 Hyperparameters of the algorithms

The hyperparameters for each algorithm have been set according to the values shown in Table 5. Any parameter that is not stated as default is common to both SI and SIR models. In the updated version of SLVAE, the original three-layer MLP network encoder was replaced with a three-layer GCN network, resulting in improved performance. For hyperparameters and implementation details of other algorithms, please refer to the corresponding original papers. DDMSL and deep learning comparison

Table 5: Hyperparameter settings of different algorithms.

Algorithms	**Hyper-parameter**	**karate**	**jazz**	**cora_ml**	**power grid**	**PGP**	**Search space**	**Description**
DDMSL	Initial learning rate	2×10^{-3}	2×10^{-3}	2×10^{-3}	2×10^{-3}	3×10^{-3}	$[2 \times 10^{-3}, 4 \times 10^{-3}, 5 \times 10^{-3}]$	
	Learning Rate Decline Interval	[1200,1500]	[200,1000]	[500, 1200]	[500, 1200]	[200,500,800,1200]	Determined by the $LOSS$ curves of the training and validation sets.	Learning rate decreases by 0.97 times the set epoach.
	n	6	6	6	6	8	$[min = 3, max = 9, step = 1]$	Number of residual blocks
	Dropout rate	0.5	0.5	0.5	0.5	0.5	Determined by Theorem \3.3	
	α in SIR model	0.4	0.6	0.4	0.4	0.45	$[min$=0.3,max=0.7,$step$=0.05]	Threshold
	α in SI model	0.4	0.45	0.4	0.4	0.45	$[min$=0.3,max=0.7,$step$=0.05]	
	Epoch	2000	1600	1600	1600	1600	Determined by the $LOSS$ curves of the training and validation sets.	
SLVAE	α	0.55	0.5	0.55	0.55	0.45	$[min$=0.3,max=0.7,$step$=0.05]	Threshold
	Learning rate	2×10^{-3}	2×10^{-3}	2×10^{-3}	2×10^{-3}	3×10^{-3}	$[2 \times 10^{-3}, 4 \times 10^{-3}, 5 \times 10^{-3}]$	
	GCN-based encoder parameters	[64,128,256]	[64,128,256]	[64,128,256]	[64,128,256]	[64,128,256]	[64,128,256],[128,256,512]	The hidden dim of the encoder
	MLP-based decoder parameters	[256,128,1]	[256,128,1]	[256,128,1]	[256,128,1]	[256,128,1]	[256,128,1],[512,256,1]	The hidden dim of the decoder
	Epoch	200	200	200	200	200	\	
DDMIX	α	0.5	0.5	0.5	0.5	0.5	$[min = 3, max = 9, step = 1]$	Threshold
	Learning rate	2×10^{-3}	2×10^{-3}	2×10^{-3}	2×10^{-3}	2×10^{-3}	$[2 \times 10^{-3}, 4 \times 10^{-3}, 5 \times 10^{-3}]$	
	Epoch	100	100	100	100	100	\	

algorithms are both running on Windows 10 systems and trained using a 4090 graphics card. The code for DDMSL is already open source, please refer to: `https://github.com/Ashenone2/DDMSL`.

F.2 Additional details of the datasets

The description of the data sets used for the experiments and their statistics are shown as below:

- **Karate** [48]: It includes a network of interrelationships between 34 members of the Karate club, comprising 34 nodes and 78 edges. The Karate dataset is a real dataset, widely employed in complex network community discovery research.
- **Jazz** [12]: The Jazz dataset is a network dataset that captures the collaborative relationships between jazz musicians. It comprises 198 nodes and 2,742 edges, and has been extensively used in research on complex network community discovery, node importance metrics, and other related studies.
- **Cora-ML** [27]: Cora-ML is a citation network dataset containing papers from the field of machine learning. Nodes represent papers and edges represent citation relationships between papers.
- **Power Grid** [45]: The Power Grid dataset is a network dataset describing the topology of the Northeastern US power grid, containing 4,941 nodes and 6,594 edges.
- **PGP** [5]: It is a User network of the Pretty-Good-Privacy algorithm for secure information exchange, consisting of 10,680 nodes and 24,316 edges.

Table 6: Statistics of the five datasets.

Datasets	#Nodes	#Edges	#Avg(degree)	#Average clustering coefficient	#Density	#Diameter
Karate	34	78	2.29	0.57	0.14	5
Jazz	198	2,742	27.70	0.62	0.14	6
Cora_ml	2,810	7,981	5.68	0.28	0.002	17
Power Grid	4,941	6,594	1.33	0.08	0.005	46
PGP	10,680	24,316	4.55	0.27	0.0004	24

G Additional experiments

Experiments on Real Diffusion Datasets. In order to gauge the efficacy of DDMSL on real-world propagation datasets, we opted for the Twitter [6] and Douban [6] datasets, encompassing 12,627 nodes with 309,631 edges, and 23,123 nodes with 348,280 edges, respectively. The detailed performance metrics can be found in Table 7.

Table 7: Additional experiments on real diffusion datasets.

	Twitter				**Douban**			
Methods	**PR**	**RE**	**F1**	**AUC**	**PR**	**RE**	**F1**	**AUC**
DDMSL	0.445	0.286	0.313	0.625	0.484	0.324	0.381	0.622
SLVAE	0.310	0.317	0.253	0.578	0.412	0.140	0.209	0.547

Generalization Performance. We conducted extensive tests on datasets of varying scales to assess the generalization performance of both DDMSL and SLVAE algorithms. The comparative results are presented in Table 8, revealing that DDMSL demonstrates commendable generalization performance across a majority of scenarios.

Table 8: Additional generalization experiments: Test results on different network topologies after one training on a real network, where the original performance represents the test performance of DDMSL on real networks.

Training data		Cora Ml				Power Grid				PGP				Twitter				Douban			
Network		PR	RE	F1	AUC	PR	RE	F1	AUC	PR	RE	F1	AUC	PR	RE	F1	AUC	PR	RE	F1	AUC
Original performance	DDMSL	**0.790**	**0.908**	**0.845**	**0.941**	**0.763**	**0.966**	**0.852**	**0.966**	**0.754**	**0.887**	**0.815**	**0.928**	**0.445**	**0.286**	**0.313**	**0.625**	**0.484**	**0.324**	**0.381**	**0.622**
	SLVAE	**0.721**	**0.765**	**0.852**	**0.908**	**0.908**	**0.719**	**0.803**	**0.856**	**0.817**	**0.721**	**0.766**	**0.851**	**0.310**	**0.317**	**0.253**	**0.578**	**0.412**	**0.140**	**0.209**	**0.547**
Small World	DDMSL	0.732	0.826	0.776	0.896	0.812	0.499	0.62	0.744	0.987	0.684	0.808	0.841	0.375	0.299	0.290	0.618	0.409	0.295	0.301	0.624
	SLVAE	0.824	0.576	0.678	0.781	0.626	0.335	0.436	0.656	0.982	0.539	0.696	0.769	0.308	0.241	0.227	0.572	0.375	0.124	0.186	0.544
ER	DDMSL	0.722	0.584	0.645	0.779	0.349	0.687	0.463	0.773	0.997	0.614	0.632	0.731	0.439	0.289	0.309	0.624	0.320	0.367	0.307	0.614
	SLVAE	0.894	0.586	0.708	0.789	0.747	0.409	0.528	0.697	0.956	0.555	0.703	0.776	0.310	0.311	0.285	0.569	0.368	0.118	0.179	0.537
BA Tree	DDMSL	0.482	0.832	0.609	0.866	0.872	0.774	0.82	0.881	0.947	0.672	0.786	0.834	0.327	0.377	0.323	0.612	0.451	0.289	0.314	0.623
	SLVAE	0.961	0.577	0.721	0.787	0.939	0.399	0.560	0.698	0.993	0.548	0.708	0.774	0.252	0.285	0.193	0.517	0.312	0.126	0.180	0.525
BA Dense	DDMSL	0.749	0.683	0.715	0.829	0.654	0.354	0.459	0.667	0.972	0.598	0.741	0.798	0.369	0.305	0.288	0.622	0.423	0.289	0.304	0.623
	SLVAE	0.662	0.611	0.635	0.788	0.731	0.441	0.550	0.712	0.997	0.446	0.617	0.723	0.286	0.405	0.315	0.567	0.374	0.132	0.195	0.553

Time Complexity. Lastly, we conducted a comparative evaluation of the time complexity between DDMSL and baseline algorithms across diverse datasets, revealing the outcomes illustrated in Table 9. Owing to the gradual inference of diffusion state for each time step, the time complexity of DDMSL tends to be substantial. However, optimization through parallel computing can effectively mitigate this disparity.

Table 9: Additional Time complexity experiment.

Test time	**Cora-Ml**	**Power-Grid**	**PGP**
DDMSL	15.84s	22.19s	22.72s
SLVAE	4.77s	7.28s	11.29s
DDMIX	9.34s	15.7s	23.26s
GCNSI	1.4s	8.14s	15.41s
LPSI	2m14s	1m51s	21m37s
OJC	6m11s	50m17s	2h41m52s
NetSleuth	2m40s	4m39s	21m24s
Training Time	**Cora-Ml**	**Power-Grid**	**PGP**
DDMSL	10m36s	16m57s	32m03s
SLVAE	18s	39s	1m10s
DDMIX	16m5s	24m17s	37m53s
GCNSI	3m53	20m6s	34m26s

H Visualization

H.1 Visualization of reconstructing diffusion paths

To conserve space, we displayed the actual node states and corresponding prediction results every 20% of the time. Figures 5 to 9 showcase the results, where the blue nodes denote susceptible ones, the red nodes denote infected nodes, and the green nodes denote recovered nodes. The findings indicate that DDMSL accurately restored the node states at different times. In contrast, DDMIX could only restore infected nodes, revealing that DDMSL far surpasses DDMIX in its expression capability.

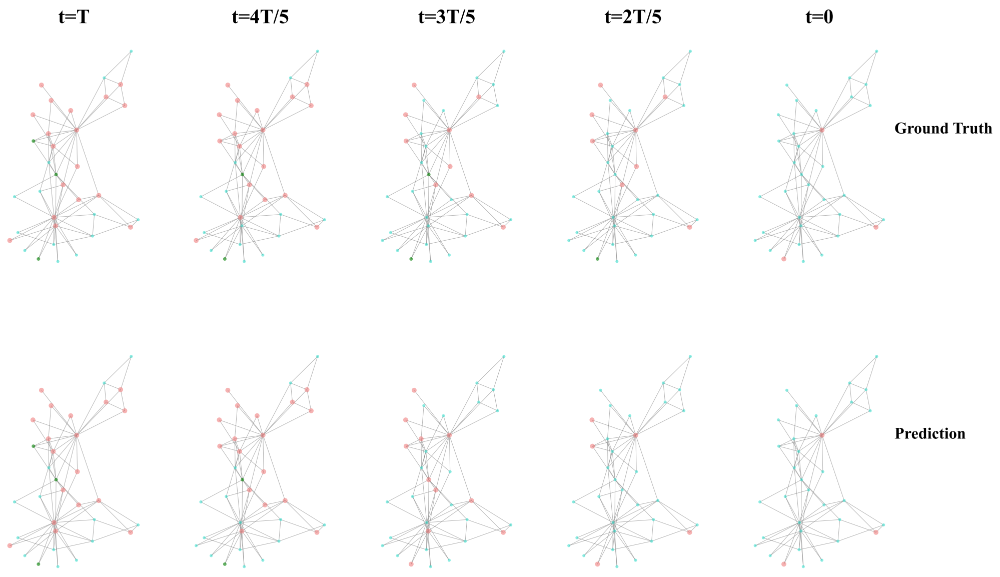

Figure 5: DDMSL reconstructs SIR diffusion on Karate.

H.2 Visual comparison of source localization

Due to spatial limitations, we only presented the source localization results of DDMSL and benchmark algorithms on the Karate and Jazz datasets. The results are depicted in Figures 10 and 11. The baseline algorithm's performance is unsatisfactory, as evidenced by significant positioning errors in the source nodes. On the contrary, DDMSL outperforms other benchmark algorithms in accurately identifying source nodes. Moreover, even when misidentifying source nodes, DDMSL locates them near the actual nodes.

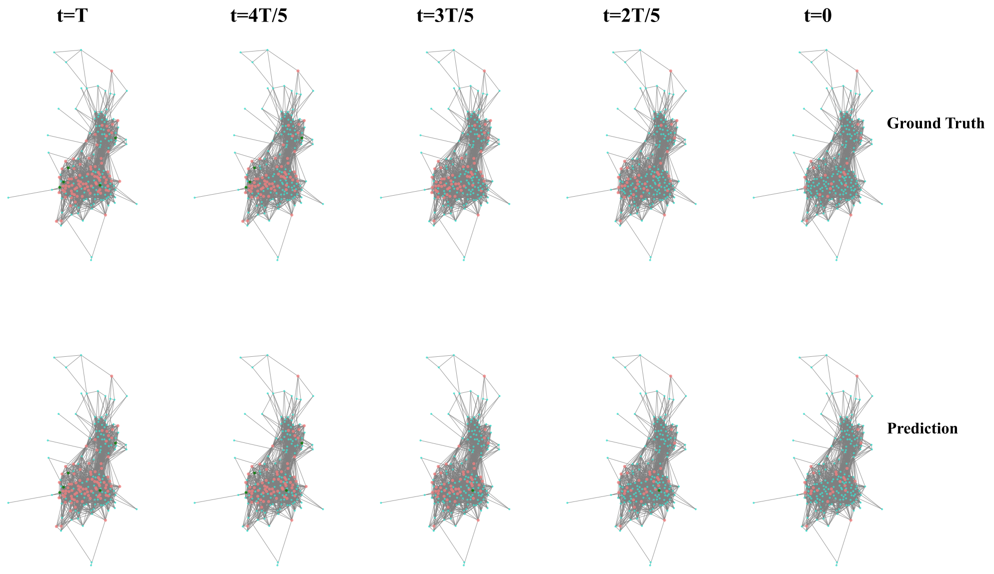

Figure 6: DDMSL reconstructs SIR diffusion on Jazz.

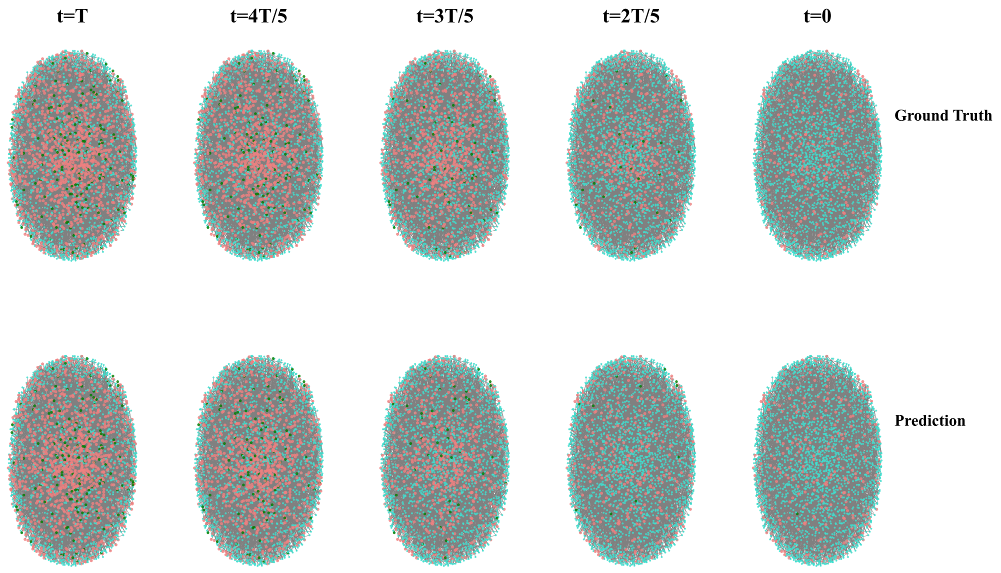

Figure 7: DDMSL reconstructs SIR diffusion on Coral ml.

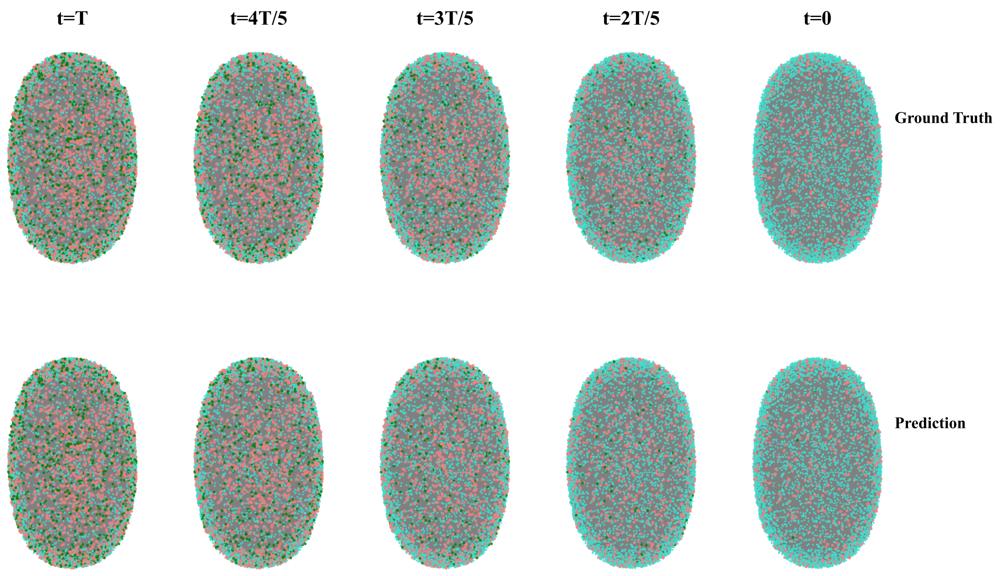

Figure 8: DDMSL reconstructs SIR diffusion on Power grid.

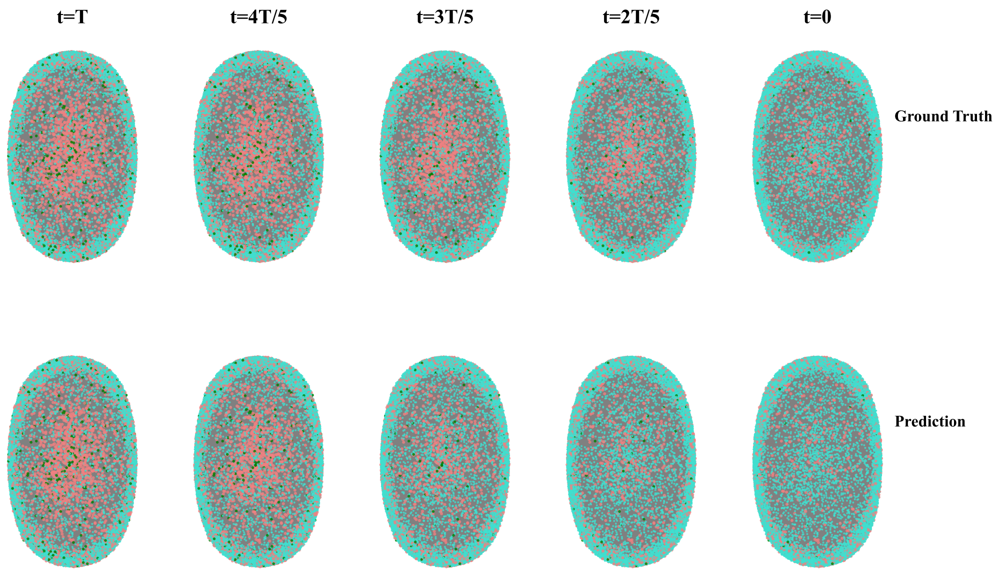

Figure 9: DDMSL reconstructs SIR diffusion on PGP.

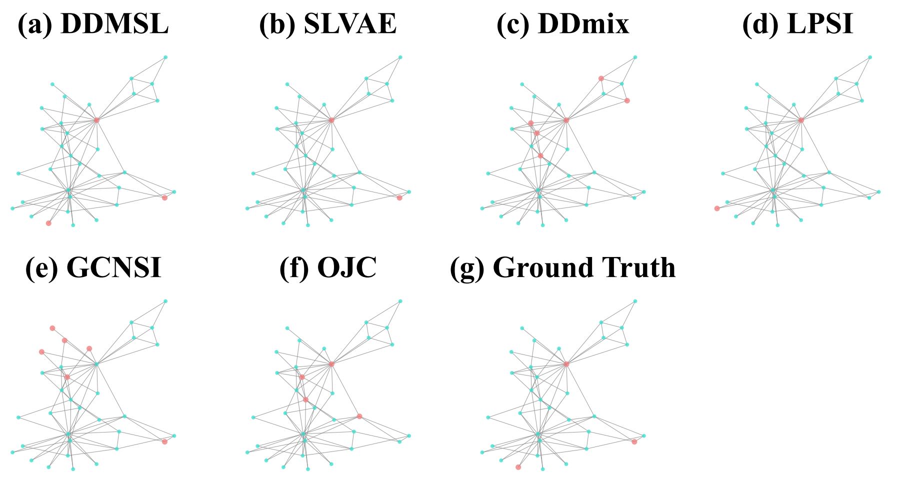

Figure 10: Visualization comparisons of source localization on Karate.

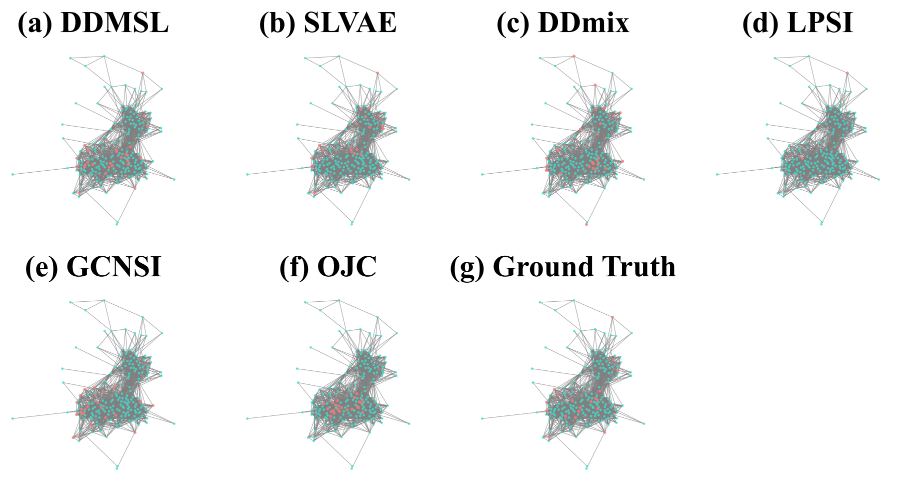

Figure 11: Visualization comparisons of source localization on Jazz.