Modeling Attention Flow on Graphs

Xiaoran Xu[1], **Songpeng Zu**[1], **Chengliang Gao**[2]*, **Yuan Zhang**[2]*, **Wei Feng**[1]
[1]Hulu Innovation Lab, Beijing, China
{xiaoran.xu, songpeng.zu, wei.feng}@hulu.com
[2]School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{gaochengliang, yuan.z}@pku.edu.cn

Abstract

Real-world scenarios demand reasoning about process, more than final outcome prediction, to discover latent causal chains and better understand complex systems. It requires the learning algorithms to offer both accurate predictions and clear interpretations. We design a set of trajectory reasoning tasks on graphs with only the source and the destination observed. We present the *attention flow mechanism* to explicitly model the reasoning process, leveraging the relational inductive biases by basing our models on graph networks. We study the way attention flow can effectively act on the underlying information flow implemented by message passing. Experiments demonstrate that the attention flow driven by and interacting with graph networks can provide higher accuracy in prediction and better interpretation for trajectory reasoning.

1 Introduction

Many practical applications have the need to infer latent causal chains or to construct interpretations for observations or some predicted results. For example, in a physical world, we want to reason the trajectories of moving objects given very few observed frames; in a video streaming system, we wish recommendation models that track the evolving user interests to provide personalized recommendation reasons linking users' watched videos to the recommended ones. Here, we focus on graph-based scenarios, and aim to infer latent chains that might cause observed results described by nodes and edges in a given graph.

Graph networks [1, 2, 3, 4, 5] are a family of neural networks that operate on graphs and carry strong relational inductive biases. It is believed that graph networks have a powerful fitting capacity to deal with graph-structured data. However, its black-box nature makes it less competitive than other differentiable logic-based reasoning [6, 7, 8] when modeling the reasoning process with interpretations provided. In this work, we develop a new attention mechanism on graphs, called *attention flow*, to model the reasoning process to predict the final outcome with the interpretability. We use the message passing algorithm in graph networks to derive a transition matrix evolving with time steps to drive the attention flow. We also let the attention flow act back on the passed messages, called *information flow*. To evaluate the models, we design a set of trajectory reasoning tasks, where only the source and destination ends of trajectories are observed.

Our contributions are two-fold. First, our attention flow mechanism, built on graph networks, introduces a new way to construct interpretations and increase the transparency when applying graph networks. Second, we show how attention flow can effectively intervene back in message passing conducted in graph networks, analogous to that of reinforcement learning where actions taken by agents would affect states of the environment. Experiments demonstrate that the graph network models with the explicit and backward-acting attention flow compare favorably both in prediction accuracy and in interpretability against those without it.

Related Works. Graph networks[3, 4, 5], dating back to a decade ago [1, 2], are thought to support relational reasoning and combinatorial generalization over graph-structured representations. Recently, this area has grown rapidly and many versions of graph networks have been proposed, including Gated Graph Neural Networks [9], Interaction Networks [10], Relation Networks [11], Message Passing Neural Networks [12], Graph Attention Networks [13], Non-Local Neural Networks [14], and the graph convolutional network family [15], spectral [16, 17, 18, 19] or non-spectral [20, 21, 22, 23, 24]. From a unified perspective, [12]

*Work done during the internship in Hulu

NIPS 2018 Workshop on Relational Representation Learning, Montréal, Canada.

introduces the message passing mechanism to generalize computation frameworks on graphs; [3] uses the term of *graph networks* to generalize and extend several lines in this area. While graph networks give reasoning over graphs more fitting capacity, we look back on the old-fashioned logic and rules-based reasoning to seek the interpretability. Recent probabilistic logic programming, such as TensorLog [6, 7] and NeuralLP [8], develops differentiable reasoning based on a knowledge graph, learning soft logic rules in an end-to-end style, and the process is much like a rooted random walk computing conditional probabilities based on paths. People also studied reasoning over paths or graphs using reinforcement learning to deal with discrete actions of choosing nodes or edges, such as MINERVA [25], Structure2Vec Deep Q-learning [26], and Neural Combinatorial Optimization [27]. Attention mechanisms, derived from sequence-based tasks [28], developed in [29] referred to as *self-attention*, have been brought in graphs recently by attending over neighbors of each node [13, 30] or non-local areas [14]. Here, we present the attention flow mechanism not only for the computation need but also for the interpretation purpose.

2 Tasks

Real-world scenarios often demand reasoning about process, that is, constructing interpretations by listing a series of causal connections linking an opening to an outcome. We need a simulation system to generate a trajectory of events with the dynamics governed by latent factors, such as the trajectory of a moving object controlled by an external force. Instead of full observation, we only allow events at the source and the destination observed, treating the task as a trajectory reasoning problem.

We build a corrupted $N \times N$ grid world with a small fraction of nodes or edges randomly removed. There are 8 types of directed edges at most for each node, connecting it to its neighbors, such as *east (E)* and *northeast (NE)*. Picking an arbitrary node as the source v^0, we draw a sequence of consecutive nodes to construct a trajectory $(v^0, v^1, \ldots, v^T)$ and obtain the final node v^T as the destination. Each node on the trajectory except the source is chosen from the neighborhood of the previous node $v_{x,y}$ by drawing one of the 8 edge types e from the distribution below driven by latent direction function $\vec{d}_{t,x,y}$ varying with time t and location (x, y):

$$\begin{aligned} &P(e) \propto \exp\left(\langle \vec{d}_e/\|\vec{d}_e\|, \vec{d}_{t,x,y}/\|\vec{d}_{t,x,y}\|\rangle/\sigma^2\right), \quad e \in \{E, NE, N, NW, W, SW, S, SE\} \\ &\vec{d}_{t,x,y} = (a_1 \cos\theta_{t,x,y} + b_1, a_2 \sin\theta_{t,x,y} + b_2), \quad \theta_{t,x,y} = \omega t + \lambda_1 x + \lambda_2 y + \phi \end{aligned} \tag{1}$$

The trajectory terminates by either choosing a non-existent edge or reaching the maximal steps. To be specific, we generate four types of trajectories governed by:

- $\vec{d}_{t,x,y} = (1, 0.4)$, a straight line with a slope.
- $\vec{d}_{t,x,y} = (1, \sin(0.4t + 1.6))$, a sine curve with directions varying with time t.
- $\vec{d}_{t,x,y} = (\cos\theta_{x,y}, \sin\theta_{x,y})$, $\theta_{x,y} = 0.2x + 0.2y$, directions varying with the current location.
- $\vec{d}_{t,x,y} = (\cos\theta_{x,y}, \sin\theta_{x,y})$, $\theta_{x,y} = 0.2\max\{x_i\} + 0.2\max\{y_i\}$, depending on location history.

Instead of learning a latent model to solve the trajectory reasoning problem, we use a supervised setting. Considering only the source and the destination available, we train a discriminative model to predict the destination node by inputting the source node. We leverage the graph structure in the corrupted grid world, as the problem implies strong inductive biases on graphs, relational (*sequences of consecutive nodes*) and non-relational (*latent direction functions depending on time, location and history*). The trajectory reasoning problem is difficult considering that many candidate paths link the source to the destination. The only clue we can observe is the destination nodes resulting from the blocked trajectories caused by removed nodes or edges. Note that we do not look for the shortest paths but the true trajectory pattern governed by some latent dynamics. The evaluation criteria should be based on both the accuracy of prediction and the human readability of interpretation.

3 Models

Modeling attention flow on graphs. We view the problem of predicting the destination given the source as predicting the output attention distribution over nodes given the input attention distribution. We use the term of attention distribution to represent the probability distribution of attending over nodes. For each pair (v_{src}, v_{dst}), the input attention distribution has all the probability mass concentrated on the source node. After a series of computation on the graph, the resulting output attention distribution predicts the most likely node to be the destination. Attention transfers from the source to the destination, implying a flow through the graph mimicking latent causal chains. The followings attempt to model the attention flow on graphs from three different perspectives.

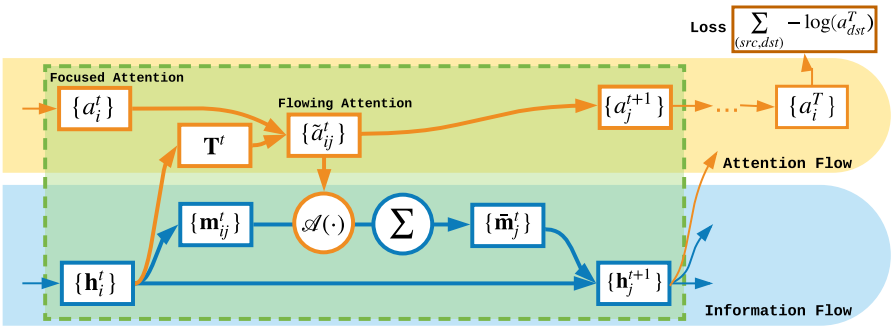

Figure 1: The two-flow model architecture.

Implicit attention flow in graph networks. Modern graph networks (GNs) mostly employ the message passing mechanism implemented by neural network building blocks, such as MLP and GRU modules. Representations in GNs include node-level states $\mathbf{h}_i \in \mathbb{R}^d$, edge-level messages $\mathbf{m}_{ij} \in \mathbb{R}^d$ and sometimes graph-level global state $\mathbf{g} \in \mathbb{R}^d$. A GN framework has three phases: the initialization phase, the propagation phase, and the output phase. The model in the propagation phase includes:

- Message function: $\mathbf{m}_{ij}^t = f_{\text{msg}}(\mathbf{h}_i^t, \mathbf{h}_j^t, \mathbf{g}^t; \theta_{e_{ij}})$, where $\theta_{e_{ij}}$ is edge type-specific parameters.
- Message aggregation operation: $\bar{\mathbf{m}}_j^t = \sum_i \mathbf{m}_{ij}^t$, aggregating all received messages from neighbors.
- Node update function: $\mathbf{h}_i^{t+1} = f_{\text{node}}(\mathbf{h}_i^t, \bar{\mathbf{m}}_i^t, \mathbf{g}^t, \mathbf{u}_i)$, where $\mathbf{u}_i$ is stationary node embeddings.
- Global update function: $\mathbf{g}^{t+1} = f_{\text{global}}(\mathbf{g}^t, \bar{\mathbf{h}}^t, \bar{\mathbf{m}}^t)$, where $\bar{\mathbf{h}}^t = \frac{1}{\|\mathcal{V}\|}\sum_i \mathbf{h}_i^t$, $\bar{\mathbf{m}}^t = \frac{1}{\|\mathcal{V}\|}\sum_i \bar{\mathbf{m}}_i^t$.

To model the attention flow, we modify the initialization phase by defining $\mathbf{h}_i^0 := [\dot{\mathbf{h}}_i^0 : \ddot{\mathbf{h}}_i^0]$ where $\dot{\mathbf{h}}_i^0 \in \mathbb{R}^{d'}$ is attention channels and $\ddot{\mathbf{h}}_i^0 \in \mathbb{R}^{d-d'}$ auxiliary channels. We initialize $\dot{\mathbf{h}}_{src}^0 = \tilde{\mathbf{1}}$ for the source and $\dot{\mathbf{h}}_i^0 = \mathbf{0}$ for the rest where $\tilde{\mathbf{1}} := \mathbf{1}/\|\mathbf{1}\|$ acts as a reference vector for computing attention distributions, so that score $\langle \dot{\mathbf{h}}_i^0, \tilde{\mathbf{1}} \rangle$ is 1 on the source node and 0 on the rest. We set $\ddot{\mathbf{h}}_i^0 = f_{\text{init}}(\mathbf{u}_i)$. At the output phase, we compute the output attention distribution by $\text{softmax}(\langle \dot{\mathbf{h}}_i^T, \tilde{\mathbf{1}} \rangle_{i=1}^n)$.

This model wraps the attention flow in the messages passing process at the beginning and takes it out at the end. Neural network-based computation makes the propagation model a black box, lacking an explicit way to depict the attention flow, helpless for the interpretation purpose.

Explicit attention flow by random walks. To explicitly model the attention flow, we use random walks with learnable transition $\mathbf{T}$. The dynamics of attention flow are driven by $\mathbf{a}^{t+1} = \mathbf{T}\mathbf{a}^t$ where $\mathbf{a}^{t+1}$ and $\mathbf{a}^t$ represent two consecutive attention distributions. Here, we take two model settings:

- Stationary transition setting: $\tau_{ij} = f_{\text{trans}}(\mathbf{u}_i, \mathbf{u}_j; \phi_{e_{ij}})$ and do the row-level softmax to get entry T_{ij}. The transition $\mathbf{T}$ is stationary across inputs and steps.
- Dynamic transition setting: $\tau_{ij}^t = f_{\text{trans}}(\mathbf{h}_i^t, \mathbf{h}_j^t; \phi_{e_{ij}})$ where $\mathbf{h}_i^{t+1} = f_{\text{node}}(\mathbf{h}_i^t, \mathbf{g}^t, \mathbf{u}_i)$, $\mathbf{g}^{t+1} = f_{\text{global}}(\mathbf{g}^t, \bar{\mathbf{h}}^t)$ and $\bar{\mathbf{h}}^t = \sum_i a_i^t \mathbf{h}_i^t$. Note that no message passing is applied. The update on global state $\mathbf{g}^t$ is based on the weighted sum of node states affected by $\mathbf{a}^t$. We emphasize that attention distributions can act as more than output of internal states and effectively impact back on the internal. The graph context captured this way is still limited without leveraging message passing.

Explicit attention flow with graph networks. The interpretability benefit of explicit attention flow can be gained even when enjoying the expressivity of graph networks. We present the attention flow mechanism by introducing the node-level attention, called *focused attention* $\mathbf{a}$, and the edge-level attention, called *flowing attention* $\tilde{\mathbf{a}}$. With the superscript step t, the dynamics are driven by:

$$\tilde{a}_{ij}^t = T_{ij}^t a_i^t, \quad a_j^{t+1} = \sum_i \tilde{a}_{ij}^t, \quad \text{s.t.} \sum_i a_i^t = 1, \quad \sum_{ij} \tilde{a}_{ij}^t = 1, \tag{2}$$

where transition $\mathbf{T}^t$ relies on the rich context carried by the underlying message passing in GNs, thus called *information flow*, in addition to attention flow. See the two-flow model architecture in Figure 1.

It is obvious that the information flow determines the attention flow, but we are more curious about how the attention flow can affect back the information flow. We have seen the node-level backward acting of the focused attention on the sum of node states as above. Here, we study on the edge level how the flowing attention acts on the information flow by defining the message-attending function $\tilde{\mathbf{m}}_{ij}^t = \mathcal{A}(\tilde{a}_{ij}^t, \mathbf{m}_{ij}^t)$ to produce attended message $\tilde{\mathbf{m}}_{ij}^t$ to replace original message $\mathbf{m}_{ij}^t$:

- No acting: $\mathcal{A}(\tilde{a}_{ij}^t, \mathbf{m}_{ij}^t) := \mathbf{m}_{ij}^t$
- Multiplying: $\mathcal{A}(\tilde{a}_{ij}^t, \mathbf{m}_{ij}^t) := \tilde{a}_{ij}^t \cdot \mathbf{m}_{ij}^t$
- Non-linearly acting after multiplying: $\mathcal{A}(\tilde{a}_{ij}^t, \mathbf{m}_{ij}^t) := \mathrm{MLP}(\tilde{a}_{ij}^t \cdot \mathbf{m}_{ij}^t)$

To design a meaningful $\mathcal{A}(\tilde{a}_{ij}^t, \mathbf{m}_{ij}^t)$, when $\tilde{a}_{ij}^t = 0$ we make $\mathcal{A}(0, \mathbf{m}_{ij}^t)$ independent from $\mathbf{m}_{ij}^t$, implying no attention paid to this piece of message but not necessarily being 0. We find $\mathrm{MLP}(\tilde{a}_{ij}^t \cdot \mathbf{m}_{ij}^t)$ performs the best in most cases, revealing not only the importance of backward acting but also the necessity of keeping information flow even if not being attended.

Connections to reinforcement learning and probabilistic latent models. If we inject noises and then pick the top-1 attended node each step, the process becomes similar to reinforcement learning in some way. If we apply noises but keep it soft in the Gumbel-Softmax [31] or Concrete [32] distribution, it turns into a probabilistic latent model. Attention flow can be viewed as graph-level computation operating directly andnumerically in a probability space rather than in a discrete sample space.

4 Experiments

4.1 Experimental Procedure

Due to space limitation, we put all the details of our experimental procedure in the appendix, including dataset generation and statistics, models in comparison, training and evaluation details, and evaluation metrics. We also show all the tables and figures about experimental comparison and visualization results in the appendix.

4.2 Experimental Results

Objectives of comparison. We list our objectives of comparative evaluation from three aspects.

- To test how well the explicit attention flow modeling can leverage rich context carried by message passing in graph networks, compared to the modeling purely based on random walks.
- To test whether the backward acting of attention flow on message passing is useful, and which way can be the most effective.
- To test whether the explicit attention flow can improve the prediction accuracy.

Discussion on comparison results. First, we compare the models that explicitly model attention flow, between the random walk-based and the graph network-based. From Table 1 and 2, we see in most cases the models favored by graph networks surpass the random walk-based models, often by a large margin. Although there are exceptions that *RW-Stationary* performs strong in the location-dependent cases, probably due to little context needed other than current location information, the best of the graph network-based models, such as *FullGN-MulMlp*, can still beat it. Second, we compare the backward acting mechanism between no acting, the multiplying acting, and the non-linearly acting. The non-linearly acting after multiplying performs the best in almost all cases. What surprises us is that simply doing multiplying may degrade the performance, making it worse than no acting. How to design an effective backward acting mechanism is worth further study in future work. Last, we compare our remodeled graph networks with explicit attention flow against the regular graph networks. For the 32×32 datasets, *{FullGN,GGNN,GAT}-MulMlp* exceed their regular graph network counterparts respectively except for dataset groups *SINE-SZ32-STP16-**. For the larger 64×64 datasets, we test *GGNN* and *GGNN-MulMlp*, and find that *GGNN-MulMlp* performs significantly much better than *GGNN* on every evaluation metric as shown in Table 3.

Discussion on visualization results. We visualize the learned attention flow in a 16×16 corrupted grid map, compared with the true trajectories and latent directions by taking one example for each direction setting as shown in Figure 2. At first glance, the drawn attention flow over the T steps makes up a belt linking from the source node to the destination node, almost matching the true trajectories, especially as shown in the first and second rows in Figure 2. On closer inspection, we find that the attention flow might not necessarily follow the one-path pattern but instead branch to enlarge the exploring area that is more likely to contain a destination node especially near gaps, as shown in the last two rows.

5 Conclusion

In this paper, we introduce the attention flow mechanism to explicitly model the reasoning process on graphs, leveraging the rich context carried by message passing in graph networks. We treat this mechanism as a way to offer accurate predictions as well as clear interpretations. In addition, we study the backward acting of attention flow on information flow implemented by message passing, and show some interesting findings from experimental results. The interaction between the two flows, one favoring the fitting capacity and one offering the interpretability, may be worth further study in future work.

References

[1] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 729–734. IEEE, 2005.

[2] 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, 2009.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018.

[5] Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

[6] William W Cohen. Tensorlog: A differentiable deductive database. *arXiv preprint arXiv:1605.06523*, 2016.

[7] William W Cohen, Fan Yang, and Kathryn Rivard Mazaitis. Tensorlog: Deep learning meets probabilistic dbs. *arXiv preprint arXiv:1707.05390*, 2017.

[8] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2319–2328, 2017.

[9] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[10] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[11] David Raposo, Adam Santoro, David Barrett, Razvan Pascanu, Timothy Lillicrap, and Peter Battaglia. Discovering objects and their relations from entangled scene representations. *arXiv preprint arXiv:1702.05068*, 2017.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[13] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[14] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[16] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[17] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[18] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[19] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[20] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[21] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[22] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, page 3, 2017.

[23] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[25] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.

[26] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

[27] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[28] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[30] WWM Kool and M Welling. Attention solves your tsp. *arXiv preprint arXiv:1803.08475*, 2018.

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

[32] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Appendix

1 Experiments (cont'd)

1.1 Experimental Procedure

Dataset generation and statistics. We generate a number of dataset groups each representing a randomized grid world with a specific version of trajectories, consisting of sequences of nodes, driven by a specific setting of latent dynamics applied. More specifically, for each dataset group, we build a corrupted $N \times N$ grid world and then apply a latent direction function $\vec{d}_{t,x,y}$ to draw multiple trajectories starting from each node. The data generation parameters include:

- N: The size of a grid map. Without nodes and edges dropped, we have N^2 nodes and $2(2N-1)(2N-2)$ directed edges at most, where each internal node is connected to 8 incoming edges as well as 8 outgoing edges. In the experiments, we test models in two sizes: 32×32 and 64×64.
- $p_{\text{node\_drp}}$ and $p_{\text{edge\_drp}}$: The dropping probabilities of randomly removing a node or an edge. If a node is removed, all its connected edges should be gone; if a node is left with no edges, that is, a single isolated point, we remove it. When we remove an edge connected by a pair of nodes v_1 and v_2, we drop edges (v_1, v_2) and (v_2, v_1) in both directions. In the experiments, we try two settings: dropping nodes only with $p_{\text{node\_drp}} = 0.1$, and dropping edges only with $p_{\text{edge\_drp}} = 0.2$.
- T: The maximal steps of a trajectory. Without being blocked, a trajectory would end with a maximal length of T. In the experiments, we set $T = 16$ in 32×32 and $T = 32$ in 64×64. Therefore, the total problem scale depends on N, $p_{\text{node\_drp}}$ and $p_{\text{edge\_drp}}$, and T.
- σ: The standard deviation of sampling an edge around latent direction $\vec{d}_{t,x,y}$. Larger σ increases the chance to bypass gaps caused by removed nodes and edges, and also brings a larger deviation of positions of the destination nodes given the same source, leading to a larger exploring area and more uncertainty for prediction. In the experiments, we pick two values, $\sigma = 0.2$ and $\sigma = 0.5$.
- n_{rollout}: The number of rollouts to draw a trajectory starting from a specific node. With each node as a source, we try n_{rollout} times and then remove the duplicated source-destination pairs. In the experiments, we use $n_{\text{rollout}} = 10$.
- $\vec{d}_{t,x,y}$: The latent direction function. In the experiments, we try four settings as shown in Section 2: (1) a straight line with a constant sloped direction; (2) a sine curve with time-dependent varying directions; (3) a curve with location-dependent varying directions; (4) a curve with location history-dependent varying directions.
- $seed$: The datasets in each group share the same generation parameters listed above except $seed$. The purpose is to make experimental results less impacted by accidental factors. We use five random seeds in the experiments.

We generate 24 dataset groups with their names and generation parameters listed in Table 4 in the appendix. Each dataset group contains five datasets with different $seed$s. The observed part of each dataset includes a grid map containing all edge information and a list of source-destination pairs used for training, validation and test. We make the training, validation and test sets by an $8:1:1$ splitting on source nodes, so that we can assess models based on their performances of handling pairs with unseen source nodes. The statistics of datasets are given in Table 5 in the appendix.

Models in comparison. To fully evaluate the attention flow mechanism, we choose three types of graph networks plus two random walk-based models to set the benchmark. Graph networks include a full Graph Network (FullGN) [3], a Gated Graph Neural Network (GGNN) [9], and a Graph Attention Network (GAT) [13]. Note that the regular versions of these graph networks are incapable to explicitly model attention flow and fulfill the trajectory reasoning purpose, though able to make predictions. We remodel them with the attention flow mechanism imposed respectively, which is implemented in three different ways on how the flowing attention acts back on the passed messages.

Regular graph networks. Node states $\mathbf{h}_i^t = [\dot{\mathbf{h}}_i^t : \ddot{\mathbf{h}}_i^t] \in \mathbb{R}^d$ have $d = 40$ channels (or dimensions) where the number of attention channels is $d' = 8$ and the rest channels are left for carrying auxiliary messages. We tried several combinations for the pair of channel numbers and found 8 attention channels performed the best. To initialize $\dot{\mathbf{h}}_i^0$, we set $\dot{\mathbf{h}}_{src}^0 = \tilde{\mathbf{1}} = \mathbf{1}/\sqrt{d'}$ for the source node and $\dot{\mathbf{h}}_i^0 = \mathbf{0}$ for the rest. We also initialize $\ddot{\mathbf{h}}_i^0 = f_{\text{init}}(\mathbf{u}_i)$ where $\mathbf{u}_i \in \mathbb{R}^d$ represents stationary node embeddings and f_{init} is a single-layer feedforward network with the $\tanh$ activation function. The loss is defined in the cross entropy between the one-hot true destination labels and the predicted probability distribution by $\text{softmax}(\langle \dot{\mathbf{h}}_i^T, \tilde{\mathbf{1}} \rangle_{i=1}^n)$.

- *FullGN*: This model has global state $\mathbf{g}^t$, and both the message function and the node update function take $\mathbf{g}^t$ as one of their inputs. Here, each function uses a single-layer feedforward network.
  - Message function: $\mathbf{m}_{ij}^t = \tanh(\mathbf{W}_{e_{ij}}[\mathbf{h}_i^t : \mathbf{h}_j^t : \mathbf{g}^t] + \mathbf{b}_{e_{ij}})$.
  - Node update function: $\mathbf{h}_i^{t+1} = \tanh(\mathbf{W}_{\text{node}}[\mathbf{h}_i^t : \bar{\mathbf{m}}_i^t : \mathbf{u}_i : \mathbf{g}^t] + \mathbf{b}_{\text{node}})$.
  - Global update function: $\mathbf{g}^{t+1} = \tanh(\mathbf{W}_{\text{global}}[\mathbf{g}^t : \bar{\mathbf{h}}^t : \bar{\mathbf{m}}^t] + \mathbf{b}_{\text{global}})$.
- *GGNN*: This model computes messages in a non-pairwise linear manner that depends on sending nodes and edge types. It ignores the global state and changes the node update function into a gated recurrent unit (GRU).
  - Message function: $\mathbf{m}_{ij}^t = \mathbf{W}_{e_{ij}}\mathbf{h}_i^t + \mathbf{b}_{e_{ij}}$.
  - Node update function: $\mathbf{h}_i^{t+1} = \text{GRU}(\mathbf{h}_i^t,\ [\bar{\mathbf{m}}_i^t : \mathbf{u}_i])$.
- *GAT*: This model uses multi-head self-attention layers. Here, we take edge types into account to define weights. We also apply a GRU to the node update function, taking the concatenation of all multi-head aggregated messages $\bar{\mathbf{m}}_i^{t,k}$ and node embedding $\mathbf{u}_i$ as the input. We use $K = 5$ heads each with an 8-dimensional self-attention so that the concatenated message still has 40 dimensions.
  - Multi-head self-attention: $\alpha_{ij}^{t,k} = \text{softmax}_j(\text{LeakyRelu}(\mathbf{a}^T[\mathbf{W}_{e_{i*}}^k\mathbf{h}_i^t : \mathbf{W}_{e_{i*}}^k\mathbf{h}_*^t]))$.
  - Message function: $\mathbf{m}_{ij}^{t,k} = \alpha_{ij}^{t,k}\mathbf{W}_{e_{ij}}^k\mathbf{h}_i^t$.
  - Node update function: $\mathbf{h}_i^{t+1} = \text{GRU}(\mathbf{h}_i^t,\ [\bar{\mathbf{m}}_i^{t,1} : \ldots : \bar{\mathbf{m}}_i^{t,K} : \mathbf{u}_i])$.

Remodeled graph networks with explicit attention flow. We add our attention flow module onto the computation framework of graph networks as shown in Figure 1. At the initialization phase, we define $\mathbf{a}^0$ by $a_{src}^0 = 1$ and $a_i^0 = 0$ for the rest; at the output phase, we take $\mathbf{a}^T$ to compute the loss:

$$\sum_{(src,dst)} -\log a_{dst}^T$$

For the propagation phase, we need to compute two more functions:

- Transition logits function: $\tau_{ij}^t = \mathbf{w}_{e_{ij}}^T[\mathbf{h}_i^t : \mathbf{h}_j^t : (\mathbf{h}_i^t \otimes \mathbf{h}_j^t)] + b_{e_{ij}}$ in order to compute $\mathbf{T}^t$.
- Message-attending function: $\tilde{\mathbf{m}}_{ij}^t = \mathcal{A}(\tilde{a}_{ij}^t, \mathbf{m}_{ij}^t)$ to produce attended message $\tilde{\mathbf{m}}_{ij}^t$ in place of original message $\mathbf{m}_{ij}^t$. We study three ways to implement it: (1) no acting, (2) multiplying, (3) non-linearly acting after multiplying. For (3), we simply use a single-layer feedforward network. Finally, we derive three explicit attention flow models based on each graph network.
  - *{FullGN, GGNN, GAT}-NoACT*.
  - *{FullGN, GGNN, GAT}-Mul*.
  - *{FullGN, GGNN, GAT}-MulMlp*.

Random walk-based models. If we model the attention flow without considering message passing conducted in graph networks, the method falls into the family of differentiable random walk models with a learned transition matrix. Here, we try two types of transition, stationary and dynamic.

- *RW-Stationary*:
  - Transition logits function: $\tau_{ij}^t = \mathbf{w}_{e_{ij}}^T[\mathbf{u}_i : \mathbf{u}_j : (\mathbf{u}_i \otimes \mathbf{u}_j)] + b_{e_{ij}}$
- *RW-Dynamic*:
  - Transition logits function: $\tau_{ij}^t = \mathbf{w}_{e_{ij}}^T[\mathbf{h}_i^t : \mathbf{h}_j^t : (\mathbf{h}_i^t \otimes \mathbf{h}_j^t)] + b_{e_{ij}}$.
  - Node update function: $\mathbf{h}_i^{t+1} = \tanh(\mathbf{W}_{\text{node}}[\mathbf{h}_i^t : \mathbf{u}_i : \mathbf{g}^t] + \mathbf{b}_{\text{node}})$.
  - Global update function: $\mathbf{g}^{t+1} = \tanh(\mathbf{W}_{\text{global}}[\mathbf{g}^t : \bar{\mathbf{h}}^t] + \mathbf{b}_{\text{global}})$ where $\bar{\mathbf{h}}^t = \sum_i a_i^t\mathbf{h}_i^t$.

Training and evaluation details. Considering that trajectories might terminate before reaching the maximal steps, we add a selfloop edge onto each node so that we can treat all trajectories as ones with a fixed number of steps. Thus, there would be 9 types of edges during training. Our training hyperparameters include the batch size of 16, the representation dimensions of 40, the weight decay on node embeddings of 0.00001, the decayed learning rates from 0.0005 to 0.0001 diminished by 0.0001 every 10 epochs, and a total number of training epochs of 50. We use the Adam SGD optimizer for all models. When dealing with larger datasets of 64×64, we reduce the batch size to 4 and the representation dimensions to 30. During experiments, for each model we conducted 10 runs on each dataset group by five different generation seeds and two different input shufflings. We saved one model snapshot every epoch and chose the best three according to their validation performance, and then computed the mean and standard deviation of their evaluation metrics on the test set.

Evaluation metrics. We use Hits@1, Hits@5, Hits@10, the mean rank (MR), and the mean reciprocal rank (MRR) to evaluate these models. Hits@k means the proportion of test source-destination pairs for which the target destination is ranked in the top-k predictions, and thus Hits@1 is the prediction accuracy. Compared to Hits@k, MR and MRR can evaluate prediction results even when the target destination is ranked out of the

Table 1: Comparison results on dataset groups *{LINE,SINE,LOCATION,HISTORY}-SZ32-STP16-NDRP-STD0.2*. This table focuses on comparative evaluation between all the explicit attention flow models that offer clear interpretations as well as prediction results, so that we gray the results from the implicit attention flow models, that is, regular graph networks. Each column indicates one comparison in a specific metric based on the same dataset group. $*$ represents the highest metric score acquired by random walk-based models. ✓ represents the graph network-based explicit attention flow models that beat the best random walk-based models. Futhermore, we compare the three message-attending approaches between the explicit attention flow models based on the same graph network, and then highlight the best in bold.

	LINE		SINE		LOCATION		HISTORY	
Model	H@1(%)	MRR	H@1(%)	MRR	H@1(%)	MRR	H@1(%)	MRR
RW-Stationary	15.80	0.3409	8.56	0.2177	50.06$*$	0.6625$*$	16.44	0.3215
RW-Dynamic	16.64$*$	0.3562$*$	19.15$*$	0.3418$*$	45.94	0.6320	20.41$*$	0.3656$*$
FullGN	*15.13*	*0.3451*	*51.71*	*0.6665*	*25.67*	*0.4393*	*16.61*	*0.3095*
FullGN-NoAct	16.65✓	0.3574✓	30.10✓	0.4476✓	46.47	0.6337	20.80✓	0.3729✓
FullGN-Mul	16.69✓	0.3636✓	37.49✓	0.4915✓	43.44	0.6029	21.35✓	0.3618
FullGN-MulMlp	**16.99**✓	**0.3662**✓	**39.91**✓	**0.5195**✓	**50.93**✓	**0.6598**	**23.94**✓	**0.3850**✓
GGNN	*15.49*	*0.3493*	*51.02*	*0.6611*	*29.20*	*0.4699*	*22.56*	*0.3677*
GGNN-NoAct	16.64	0.3570✓	25.14✓	0.3918✓	49.50	0.6621	21.69✓	0.3818✓
GGNN-Mul	16.95✓	0.3610✓	23.62✓	0.3776✓	45.22	0.6185	23.81✓	0.3824✓
GGNN-MulMlp	**17.08**✓	**0.3673**✓	**34.75**✓	**0.4699**✓	**50.28**✓	**0.6637**✓	**26.06**✓	**0.4001**✓
GAT	*16.01*	*0.3469*	*43.19*	*0.5566*	*18.18*	*0.3583*	*12.11*	*0.2333*
GAT-NoAct	16.02	0.3536	15.77	0.3221	46.10	0.6356	**23.17**✓	**0.3818**✓
GAT-Mul	15.86	0.3501	20.14✓	0.3429✓	45.83	0.6208	22.70✓	0.3762✓
GAT-MulMlp	**17.07**✓	**0.3646**✓	**30.64**✓	**0.4390**✓	**47.52**	**0.6371**	20.71✓	0.3655

Table 2: Comparison results on dataset groups *{LINE,SINE,LOCATION,HISTORY}-SZ32-STP16-NDRP-STD0.5*. The marks in this table have the same meanings as Table 1.

	LINE		SINE		LOCATION		HISTORY	
Model	H@1(%)	MRR	H@1(%)	MRR	H@1(%)	MRR	H@1(%)	MRR
RW-Stationary	15.74$*$	0.3348	9.07	0.2267	19.40$*$	0.3860$*$	11.72	0.2547
RW-Dynamic	15.48	0.3429$*$	13.38$*$	0.2905$*$	17.91	0.3722	12.25$*$	0.2820$*$
FullGN	*14.61*	*0.3355*	*17.10*	*0.3525*	*16.09*	*0.3476*	*13.79*	*0.3051*
FullGN-NoAct	15.50	0.3410	16.60✓	0.3360✓	18.83	0.3816	13.90✓	0.3059✓
FullGN-Mul	16.21✓	0.3498✓	16.93✓	0.3283✓	18.50	0.3787	12.93✓	0.2807
FullGN-MulMlp	**16.07**✓	**0.3502**✓	**17.31**✓	**0.3389**✓	**19.64**✓	**0.3991**✓	**14.89**✓	**0.3145**✓
GGNN	*14.53*	*0.3344*	*17.45*	*0.3555*	*17.11*	*0.3689*	*14.83*	*0.3217*
GGNN-NoAct	15.58	0.3415	16.51✓	0.3262✓	**19.66**✓	**0.3912**✓	13.84✓	0.2957✓
GGNN-Mul	15.79✓	0.3448✓	16.03✓	0.3226✓	17.84	0.3723	14.16✓	0.2971✓
GGNN-MulMlp	**15.99**✓	**0.3497**✓	**17.31**✓	**0.3370**✓	19.39	0.3911✓	**14.80**✓	**0.3053**✓
GAT	*14.79*	*0.3300*	*16.48*	*0.3338*	*14.51*	*0.3227*	*10.80*	*0.2538*
GAT-NoAct	15.83✓	0.3414	15.15✓	0.3161✓	17.82	0.3702	12.60✓	0.2829✓
GAT-Mul	15.01✓	0.3351	14.85✓	0.3070✓	18.27	0.3749	13.49✓	0.2885✓
GAT-MulMlp	**16.25**✓	**0.3493**✓	**16.45**✓	**0.3292**✓	**18.93**	**0.3843**	**13.75**✓	**0.2933**✓

top-k. MR provides a more intuitive sense about how many are ranked before the target on average, but often suffers from its instability susceptible to the worst example and becomes very large. MRR scores always range from 0.0 to 1.0. For MR lower score reflects better prediction, whereas for MRR higher score means better.

Table 3: Comparison results on larger datasets of 64×64 that are *{LINE,SINE,LOCATION,HISTORY}-SZ64-STP32-NDRP-STD{0.2,0.5}*.

Std	Model	LINE		SINE		LOCATION		HISTORY	
		H@1(%)	MRR	H@1(%)	MRR	H@1(%)	MRR	H@1(%)	MRR
0.2	GGNN	12.36	0.2918	27.08	0.4047	20.98	0.4323	11.47	0.2785
	GGNN-MulMlp	**15.56**	**0.3335**	**39.98**	**0.5317**	**47.56**	**0.6534**	**23.49**	**0.3888**
0.5	GGNN	11.04	0.2759	10.53	0.2616	9.74	0.2347	8.67	0.2222
	GGNN-MulMlp	**14.53**	**0.3179**	**16.08**	**0.3137**	**18.39**	**0.3782**	**13.60**	**0.2902**

Table 4: Parameters of generating the datasets

Dataset Group	$\vec{d}_{t,x,y}$	N	T	σ	$p_{\text{node\_drp}}$	$p_{\text{edge\_drp}}$
LINE-SZ32-STP16-NDRP-STD0.2	Line	32	16	0.2	0.1	0.0
LINE-SZ32-STP16-NDRP-STD0.5	Line	32	16	0.5	0.1	0.0
SINE-SZ32-STP16-NDRP-STD0.2	Sine	32	16	0.2	0.1	0.0
SINE-SZ32-STP16-NDRP-STD0.5	Sine	32	16	0.5	0.1	0.0
LOCATION-SZ32-STP16-NDRP-STD0.2	Location	32	16	0.2	0.1	0.0
LOCATION-SZ32-STP16-NDRP-STD0.5	Location	32	16	0.5	0.1	0.0
HISTORY-SZ32-STP16-NDRP-STD0.2	History	32	16	0.2	0.1	0.0
HISTORY-SZ32-STP16-NDRP-STD0.5	History	32	16	0.5	0.1	0.0
LINE-SZ32-STP16-EDRP-STD0.2	Line	32	16	0.2	0.0	0.2
LINE-SZ32-STP16-EDRP-STD0.5	Line	32	16	0.5	0.0	0.2
SINE-SZ32-STP16-EDRP-STD0.2	Sine	32	16	0.2	0.0	0.2
SINE-SZ32-STP16-EDRP-STD0.5	Sine	32	16	0.5	0.0	0.2
LOCATION-SZ32-STP16-EDRP-STD0.2	Location	32	16	0.2	0.0	0.2
LOCATION-SZ32-STP16-EDRP-STD0.5	Location	32	16	0.5	0.0	0.2
HISTORY-SZ32-STP16-EDRP-STD0.2	History	32	16	0.2	0.0	0.2
HISTORY-SZ32-STP16-EDRP-STD0.5	History	32	16	0.5	0.0	0.2
LINE-SZ64-STP32-NDRP-STD0.2	Line	64	32	0.2	0.1	0.0
LINE-SZ64-STP32-NDRP-STD0.5	Line	64	32	0.5	0.1	0.0
SINE-SZ64-STP32-NDRP-STD0.2	Sine	64	32	0.2	0.1	0.0
SINE-SZ64-STP32-NDRP-STD0.5	Sine	64	32	0.5	0.1	0.0
LOCATION-SZ64-STP32-NDRP-STD0.2	Location	64	32	0.2	0.1	0.0
LOCATION-SZ64-STP32-NDRP-STD0.5	Location	64	32	0.5	0.1	0.0
HISTORY-SZ64-STP32-NDRP-STD0.2	History	64	32	0.2	0.1	0.0
HISTORY-SZ64-STP32-NDRP-STD0.5	History	64	32	0.5	0.1	0.0

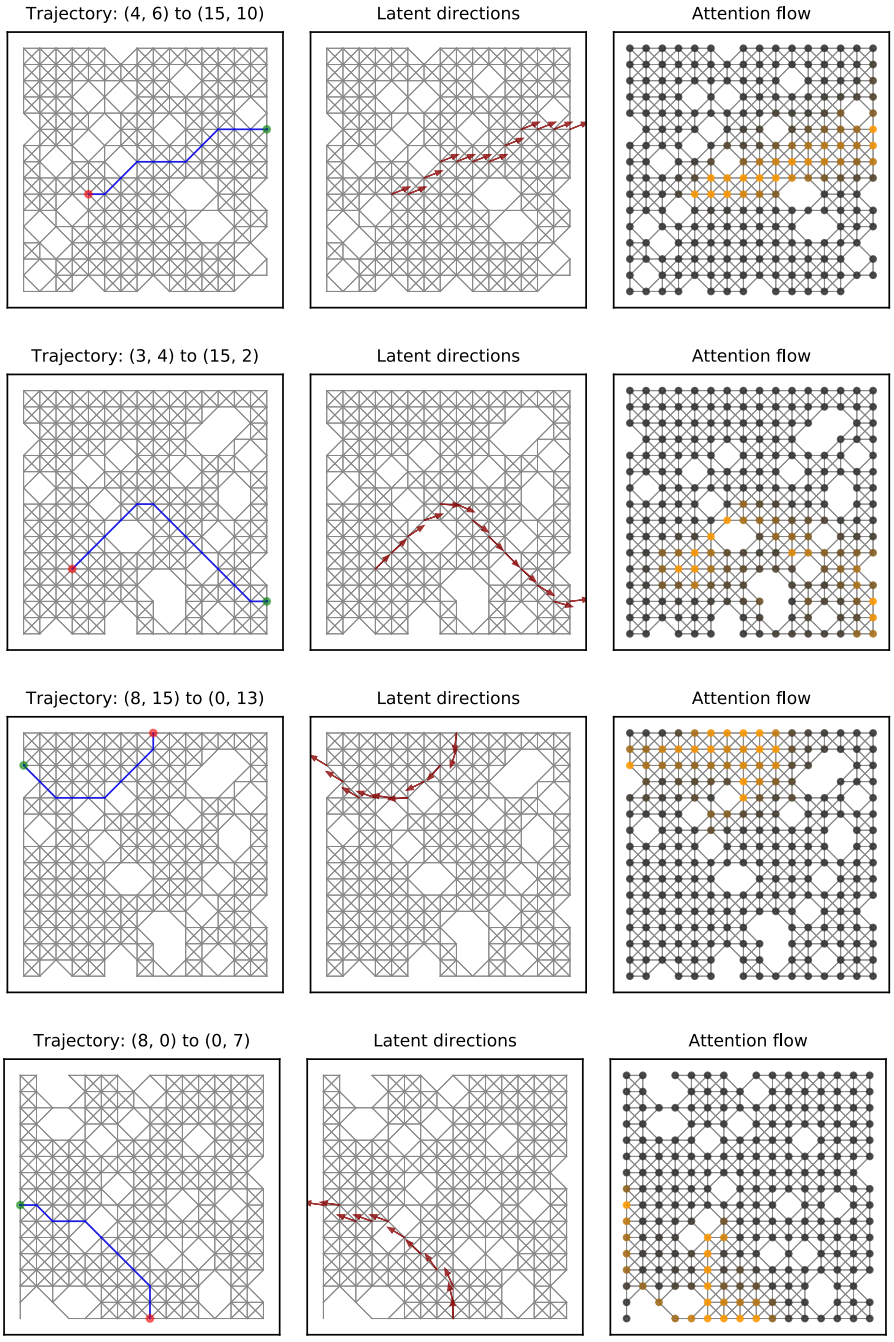

Figure 2: The true trajectory v.s. the latent directions in arrows v.s. the learned attention flow. The first row reflects a constant sloped direction, the second row time-dependent directions, the third row location-dependent directions, and the last row history-dependent directions. The drawn attention flow is based on the max aggregation of normalized attention distributions over the T steps.

1.2 More Discussion about Results

During the experiments, we found some results in our expectation as discussed in the model section, as well as some unexpected results that surprise us, probably worth further study in future work. Now we summarize them as follows:

- ***Backward acting of attention flow on information flow is useful, better than no acting in most cases.***

Table 5: Dataset statistics (All numbers represent average results. Note that dataset groups *LINE-*-STD0.2* produce more trajectories per node than their counterparts, because the slop we choose makes the latent direction equally between two candidate edges, introducing more randomness to generate trajectories.)

Dataset Group	#Nodes	#Edges	#Trajs	#Trajs-per-node	Traj-length
LINE-SZ32-STP16-NDRP-STD0.2	921	6319	4829	5.2	9.1
LINE-SZ32-STP16-NDRP-STD0.5	921	6319	5029	5.5	9.0
SINE-SZ32-STP16-NDRP-STD0.2	921	6319	1555	1.7	9.0
SINE-SZ32-STP16-NDRP-STD0.5	921	6319	4380	4.8	9.4
LOCATION-SZ32-STP16-NDRP-STD0.2	921	6319	1440	1.6	7.9
LOCATION-SZ32-STP16-NDRP-STD0.5	921	6319	4098	4.4	8.8
HISTORY-SZ32-STP16-NDRP-STD0.2	921	6319	1541	1.7	8.5
HISTORY-SZ32-STP16-NDRP-STD0.5	921	6319	4418	4.8	9.2
LINE-SZ32-STP16-EDRP-STD0.2	1023	6248	4828	4.7	6.8
LINE-SZ32-STP16-EDRP-STD0.5	1023	6248	5051	4.9	6.7
SINE-SZ32-STP16-EDRP-STD0.2	1023	6248	1441	1.4	6.3
SINE-SZ32-STP16-EDRP-STD0.5	1023	6248	4238	4.1	6.8
LOCATION-SZ32-STP16-EDRP-STD0.2	1023	6248	1576	1.5	6.0
LOCATION-SZ32-STP16-EDRP-STD0.5	1023	6248	4327	4.2	6.6
HISTORY-SZ32-STP16-EDRP-STD0.	1023	6248	1586	1.5	6.0
HISTORY-SZ32-STP16-EDRP-STD0.	1023	6248	4441	4.3	6.6
LINE-SZ64-STP32-NDRP-STD0.2	3686	25891	21390	5.8	11.4
LINE-SZ64-STP32-NDRP-STD0.5	3686	25891	22106	6.0	11.1
SINE-SZ64-STP32-NDRP-STD0.2	3686	25891	6691	1.8	11.4
SINE-SZ64-STP32-NDRP-STD0.5	3686	25891	19233	5.2	11.7
LOCATION-SZ64-STP32-NDRP-STD0.2	3686	25891	6522	1.8	10.1
LOCATION-SZ64-STP32-NDRP-STD0.5	3686	25891	17482	4.7	10.8
HISTORY-SZ64-STP32-NDRP-STD0.2	3686	25891	6987	1.9	11.0
HISTORY-SZ64-STP32-NDRP-STD0.5	3686	25891	19303	5.2	11.7

  - *FullGN-{Mul,MulMlp}* both perform better than *FullGN-NoAct* on Hits@1, Hits@5, Hits@10, MR and MRR for dataset groups *LINE-SZ32-STP16-**, on Hits@1, Hits@5, Hits@10 and MRR for dataset groups *SINE-SZ32-STP16-NDRP-STD0.2, SINE-SZ32-STP16-EDRP-*, LOCATION-SZ32-STP16-EDRP-STD0.5*, on Hits@1 for dataset groups *SINE-SZ32-STP16-NDRP-STD0.5, HISTORY-SZ32-STP16-*-STD0.2*.
  - *GGNN-{Mul,MulMlp}* both perform better than *GGNN-NoAct* on Hits@1, Hits@5, Hits@10, MR and MRR for dataset group *LINE-SZ32-STP16-NDRP-STD0.2*, on Hits@1, MR and MRR for dataset group *LINE-SZ32-STP16-NDRP-STD0.5*, on Hits@1 and MRR for dataset groups *HISTORY-SZ32-STP16-NDRP-**, on Hits@1 for dataset group *SINE-SZ32-STP16-NDRP-STD0.5*.
  - *GAT-{Mul,MulMlp}* both perform better than *GAT-NoAct* on Hits@1, Hits@5 and MRR for dataset groups *SINE-SZ32-STP16-NDRP-STD0.2, LOCATION-SZ32-STP16-NDRP-STD0.5, HISTORY-SZ32-STP16-NDRP-STD0.5*.

- ***Simply applying multiplying to backward acting might cause degradation.***
  - *{FullGN,GGNN,GAT}-Mul* perform poorly on dataset groups *LOCATION-SZ32-STP16-**.
  - *{GGNN,GAT}-Mul* perform poorly on dataset group *SINE-SZ32-STP16-NDRP-STD0.5*.
  - *GGNN-Mul* performs poorly on dataset group *SINE-SZ32-STP16-NDRP-STD0.2*.

- ***Non-linear backward acting after multiplying can work consistently well, always performing the best among the backward acting of NoAct, Mul and MulMlp, and even often the best among all the models.***
  - *{FullGN,GGNN,GAT}-MulMlp* perform the best on Hits@1, Hits@5, MR and MR for dataset groups *LINE-SZ32-STP16-*-STD0.2*, on Hits@1 and MRR for dataset groups *LOCATION-SZ32-STP16-*-STD0.2*
  - *{GGNN,GAT}-MulMlp* perform the best on Hits@1, Hits@5 and MR for dataset groups *LINE-SZ32-STP16-NDRP-STD0.5, HISTORY-SZ32-STP16-NDRP-STD0.5*, on MR for dataset group *SINE-SZ32-STP16-NDRP-STD0.2*
  - *{FullGN,GAT}-MulMlp* perform the best on Hits@1, Hits@5, Hits@10 and MR for dataset group *LOCATION-SZ32-STP16-NDRP-STD0.5*, on MR for dataset group *SINE-SZ32-STP16-NDRP-STD0.2*, on MR and MRR for dataset group *LINE-SZ32-STP16-NDRP-STD0.5*

– *{FullGN,GGNN}-MulMlp* perform the best on Hits@1 and MRR for dataset group *HISTORY-SZ32-STP16-NDRP-STD0.2*.

– *FullGN-MulMlp* performs the best on Hits@1, Hits@5 and MRR for dataset groups *LINE-SZ32-STP16-EDRP-STD0.2, LOCATION-SZ32-STP16-EDRP-**, on Hits@1 on dataset groups *LINE-SZ32-STP16-EDRP-*, LOCATION-SZ32-STP16-EDRP-*, HISTORY-SZ32-STP16-EDRP-STD0.2, SINE-SZ32-STP16-EDRP-STD0.5*, on MRR for dataset groups *LINE-SZ32-STP16-EDRP-STD0.5, SINE-SZ32-STP16-EDRP-STD0.5, LOCATION-SZ32-STP16-EDRP-STD0.5*

• ***Reasoning purely based on random walks with no message passing may be the worst in most cases, but in a few cases it can work surprisingly well.***

– *RW-Stationary* performs very poorly on dataset groups *LINE-SZ32-STP16-NDRP-*, SINE-SZ32-STP16-NDRP-*, HISTORY-SZ32-STP16-NDRP-** but surprisingly well on dataset groups *LOCATION-SZ32-STP16-NDRP-**. It is probably because under the location-dependent latent directions it requires little context except current location information to do the trajectory reasoning.

– When considering global context information, *RW-Dynamic* works better than *RW-Stationary* in the cases where *RW-Stationary* performs poorly, but it obtains lower scores than *RW-Stationary* on dataset groups *LOCATION-SZ32-STP16-NDRP-**, demonstrating again that little context is needed here.

• ***Regular graph networks work extremely well in the cases with the time-dependent latent directions, but they might not be suitable for other cases, such as the location-dependent and the history-dependent latent directions.***

– *FullGN, GGNN, GAT* obtain the highest scores and exceed the second by a large margin on dataset groups *SINE-SZ32-STP16-** but still get a large MR score.

– *FullGN, GGNN, GAT* perform very poorly on dataset groups *LOCATION-SZ32-STP16-*, HISTORY-SZ32-STP16-** except *GGNN* on *HISTORY-SZ32-STP16-NDRP-STD0.5*.

• ***Models with attention flow taking non-linear backward acting might perform significantly better on a larger scale than those without***.

– Due to the limitation of computation resource, we have not taken multiple runs on the dataset groups of 64×64 but only run *GGNN, GGNN-MulMlp* once on dataset groups *{LINE,SINE,LOCATION,HISTORY}-SZ64-STP32-NDRP-STD{0.2,0.5}*. From the results, we can see *GGNN-MulMlp* surpasses *GGNN* by a large amount on every evaluation metric.

Table 6: Comparison results on dataset group *LINE-SZ32-STP16-NDRP-STD0.2*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	15.80 ±0.56	56.95 ±1.96	78.25 ±2.31	9.857 ±1.167	0.3409 ±0.0098
RW-Dynamic	16.64 ±0.65	59.65 ±1.89	82.13 ±1.77	7.061 ±0.594	0.3562 ±0.0084
FullGN	15.13 ±0.74	59.75 ±2.24	**83.83** ±1.99	6.800 ±0.993	0.3451 ±0.0082
FullGN-NoAct	16.65 ±0.96	59.35 ±2.08	82.50 ±1.82	6.628 ±0.392	0.3574 ±0.0102
FullGN-Mul	16.69 ±0.86	61.20 ±1.66	83.58 ±2.24	6.399 ±0.609	0.3636 ±0.0108
FullGN-MulMlp	**16.99** ±0.91	**61.59** ±2.05	83.73 ±1.73	**6.296** ±0.615	**0.3662** ±0.0109
GGNN	15.49 ±0.98	60.30 ±1.80	**83.60** ±1.96	6.605 ±0.580	0.3493 ±0.0100
GGNN-NoAct	16.64 ±0.86	59.62 ±2.02	82.52 ±1.53	6.760 ±0.429	0.3570 ±0.0095
GGNN-Mul	16.95 ±0.70	59.80 ±1.65	82.61 ±2.05	6.477 ±0.532	0.3610 ±0.0089
GGNN-MulMlp	**17.08** ±0.77	**61.45** ±1.82	83.54 ±1.89	**6.316** ±0.684	**0.3673** ±0.0111
GAT	16.01 ±1.06	58.86 ±1.82	80.78 ±1.91	18.306 ±16.927	0.3469 ±0.0121
GAT-NoAct	16.02 ±0.65	59.42 ±1.95	82.29 ±1.70	6.707 ±0.397	0.3536 ±0.0091
GAT-Mul	15.86 ±0.99	58.77 ±2.64	81.67 ±1.98	6.518 ±0.474	0.3501 ±0.0122
GAT-MulMlp	**17.07** ±0.79	**60.60** ±1.70	**83.22** ±2.18	**6.488** ±0.670	**0.3646** ±0.0100

Table 7: Comparison results on dataset group *LINE-SZ32-STP16-NDRP-STD0.5*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	15.74 ±0.54	55.28 ±1.15	76.40 ±1.24	10.087 ±0.610	0.3348 ±0.0040
RW-Dynamic	15.48 ±0.82	57.07 ±1.30	79.19 ±1.04	7.735 ±0.345	0.3429 ±0.0071
FullGN	14.61 ±0.90	57.97 ±1.20	**80.90** ±1.02	8.472 ±1.331	0.3355 ±0.0090
FullGN-NoAct	15.50 ±0.48	57.09 ±1.31	79.55 ±1.10	7.761 ±0.416	0.3410 ±0.0064
FullGN-Mul	**16.21** ±0.35	**58.18** ±1.45	80.73 ±0.93	7.463 ±0.284	0.3498 ±0.0047
FullGN-MulMlp	16.07 ±0.56	58.16 ±1.02	80.59 ±1.29	**7.431** ±0.283	**0.3502** ±0.0046
GGNN	14.53 ±0.72	57.33 ±1.01	80.34 ±1.25	7.856 ±0.669	0.3344 ±0.0073
GGNN-NoAct	15.58 ±0.53	57.44 ±1.00	79.62 ±1.17	7.789 ±0.349	0.3415 ±0.0055
GGNN-Mul	15.79 ±0.63	57.36 ±1.92	80.13 ±1.16	**7.387** ±0.303	0.3448 ±0.0060
GGNN-MulMlp	**15.99** ±0.59	**58.17** ±1.26	**80.79** ±0.98	7.391 ±0.325	**0.3497** ±0.0052
GAT	14.79 ±1.17	56.51 ±2.26	79.20 ±1.79	16.323 ±12.684	0.3300 ±0.0147
GAT-NoAct	15.83 ±0.39	56.95 ±1.25	79.28 ±1.31	7.631 ±0.309	0.3414 ±0.0047
GAT-Mul	15.01 ±0.84	55.95 ±1.22	78.55 ±1.62	7.793 ±0.530	0.3351 ±0.0071
GAT-MulMlp	**16.25** ±0.57	**57.61** ±1.35	**80.60** ±0.80	**7.292** ±0.190	**0.3493** ±0.0056

Table 8: Comparison results on dataset group *SINE-SZ32-STP16-NDRP-STD0.2*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	8.56 ±1.76	35.81 ±1.81	51.33 ±2.47	23.883 ±2.757	0.2177 ±0.0103
RW-Dynamic	19.15 ±3.02	50.69 ±4.20	67.13 ±3.58	13.554 ±2.560	0.3418 ±0.0266
FullGN	**51.71** ±3.46	**87.46** ±3.93	**93.20** ±4.78	17.588 ±19.805	**0.6665** ±0.0348
FullGN-NoAct	30.10 ±6.69	61.67 ±7.22	76.88 ±5.48	**9.329** ±2.224	0.4476 ±0.0597
FullGN-Mul	37.49 ±1.97	63.60 ±2.72	76.91 ±2.31	9.800 ±1.077	0.4915 ±0.0212
FullGN-MulMlp	39.91 ±2.80	66.15 ±3.01	78.19 ±2.33	9.377 ±0.824	0.5195 ±0.0242
GGNN	**51.02** ±2.10	**87.15** ±3.44	**93.51** ±2.92	12.796 ±18.578	**0.6611** ±0.0208
GGNN-NoAct	25.14 ±6.53	54.30 ±6.68	69.91 ±4.57	12.509 ±2.212	0.3918 ±0.0604
GGNN-Mul	23.62 ±2.30	53.62 ±2.64	67.37 ±3.12	15.002 ±1.354	0.3776 ±0.0168
GGNN-MulMlp	34.75 ±2.93	60.76 ±4.26	73.72 ±3.33	**12.392** ±1.525	0.4699 ±0.0264
GAT	**43.19** ±3.24	**73.32** ±3.67	**79.30** ±3.32	62.170 ±12.886	**0.5566** ±0.0294
GAT-NoAct	15.77 ±3.97	50.17 ±4.34	67.36 ±3.93	13.988 ±1.933	0.3221 ±0.0336
GAT-Mul	20.14 ±2.10	50.70 ±2.94	66.65 ±2.81	15.725 ±1.922	0.3429 ±0.0193
GAT-MulMlp	30.64 ±2.37	58.59 ±3.91	73.46 ±3.34	**12.022** ±1.639	0.4390 ±0.0216

1.3 More Visualization Results

Table 9: Comparison results on dataset group *SINE-SZ32-STP16-NDRP-STD0.5*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	9.07 ±0.60	34.71 ±1.38	50.39 ±3.15	19.937 ±1.535	0.2267 ±0.0090
RW-Dynamic	13.38 ±1.06	46.26 ±1.98	64.54 ±2.05	12.380 ±1.032	0.2905 ±0.0103
FullGN	17.10 ±0.67	**57.91** ±1.90	**76.37** ±1.58	**9.517** ±1.175	**0.3525** ±0.0069
FullGN-NoAct	16.60 ±1.13	52.81 ±1.95	71.91 ±1.58	9.542 ±0.790	0.3360 ±0.0106
FullGN-Mul	16.93 ±0.99	50.41 ±1.98	67.74 ±1.89	12.163 ±0.526	0.3283 ±0.0133
FullGN-MulMlp	**17.31** ±0.79	52.74 ±1.68	70.69 ±2.67	11.328 ±1.042	0.3389 ±0.0072
GGNN	**17.45** ±1.03	**57.80** ±1.59	**76.72** ±2.14	**10.011** ±1.730	**0.3555** ±0.0098
GGNN-NoAct	16.51 ±1.51	50.63 ±3.06	69.49 ±2.48	10.611 ±1.031	0.3262 ±0.0163
GGNN-Mul	16.03 ±1.21	50.61 ±1.80	69.17 ±2.34	11.253 ±0.514	0.3226 ±0.0156
GGNN-MulMlp	17.31 ±1.55	53.07 ±2.75	70.65 ±1.08	11.500 ±0.705	0.3370 ±0.0177
GAT	**16.48** ±0.91	**53.47** ±1.60	**71.89** ±1.70	21.869 ±12.869	**0.3338** ±0.0070
GAT-NoAct	15.15 ±1.34	49.65 ±3.17	68.24 ±2.86	**11.026** ±1.152	0.3161 ±0.0153
GAT-Mul	14.85 ±0.90	48.34 ±2.27	67.67 ±2.24	11.681 ±0.363	0.3070 ±0.0097
GAT-MulMlp	16.45 ±1.18	51.23 ±2.81	68.86 ±2.24	11.701 ±0.526	0.3292 ±0.0169

Table 10: Comparison results on dataset group *LOCATION-SZ32-STP16-NDRP-STD0.2*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	50.06 ±3.23	84.70 ±5.23	91.99 ±3.76	5.997 ±2.266	0.6625 ±0.0403
RW-Dynamic	45.94 ±6.63	85.81 ±5.03	93.05 ±2.62	6.036 ±2.205	0.6320 ±0.0544
FullGN	25.67 ±8.66	69.33 ±10.43	80.63 ±7.18	18.589 ±9.211	0.4393 ±0.0884
FullGN-NoAct	46.47 ±3.86	84.30 ±3.88	**92.22** ±2.64	**5.797** ±2.316	0.6337 ±0.0346
FullGN-Mul	43.44 ±6.02	82.88 ±5.66	89.49 ±4.43	7.395 ±2.546	0.6029 ±0.0511
FullGN-MulMlp	**50.93** ±3.21	**85.46** ±4.93	91.03 ±3.72	7.649 ±3.921	**0.6598** ±0.0362
GGNN	29.20 ±7.85	70.59 ±8.83	79.83 ±5.75	16.471 ±6.033	0.4699 ±0.0742
GGNN-NoAct	49.50 ±3.91	**87.84** ±4.27	**93.48** ±3.09	**5.850** ±2.498	0.6621 ±0.0383
GGNN-Mul	45.22 ±4.67	83.48 ±5.20	90.46 ±3.57	8.098 ±3.410	0.6185 ±0.0437
GGNN-MulMlp	**50.28** ±3.36	86.41 ±4.16	91.55 ±3.27	6.634 ±2.951	**0.6637** ±0.0342
GAT	18.18 ±4.96	59.84 ±6.92	73.31 ±5.37	21.609 ±3.249	0.3583 ±0.0579
GAT-NoAct	46.10 ±4.34	85.67 ±4.78	92.54 ±3.61	**5.332** ±1.829	0.6356 ±0.0400
GAT-Mul	45.83 ±2.74	82.99 ±3.29	89.72 ±2.19	7.059 ±1.654	0.6208 ±0.0232
GAT-MulMlp	**47.52** ±7.39	**85.68** ±4.04	**92.97** ±2.80	6.276 ±2.278	**0.6371** ±0.0597

Table 11: Comparison results on dataset group *LOCATION-SZ32-STP16-NDRP-STD0.5*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	19.40 ±1.03	64.37 ±1.68	80.30 ±1.63	9.569 ±1.020	0.3860 ±0.0113
RW-Dynamic	17.91 ±0.75	62.47 ±1.60	80.55 ±1.87	9.379 ±1.392	0.3722 ±0.0101
FullGN	16.09 ±1.51	58.99 ±4.85	78.30 ±4.49	16.608 ±6.832	0.3476 ±0.0256
FullGN-NoAct	18.83 ±1.17	63.27 ±1.71	82.63 ±1.61	**8.140** ±0.986	0.3816 ±0.0111
FullGN-Mul	18.50 ±1.10	62.79 ±1.96	81.34 ±1.55	8.928 ±1.070	0.3787 ±0.0099
FullGN-MulMlp	**19.64** ±1.02	**66.33** ±1.75	**83.87** ±1.57	8.235 ±1.025	**0.3991** ±0.0100
GGNN	17.11 ±1.07	63.48 ±2.29	81.83 ±1.95	14.964 ±3.551	0.3689 ±0.0100
GGNN-NoAct	**19.66** ±0.81	64.74 ±1.20	82.97 ±0.98	**8.118** ±0.762	**0.3912** ±0.0086
GGNN-Mul	17.84 ±1.11	62.36 ±1.50	81.60 ±2.31	9.220 ±1.276	0.3723 ±0.0114
GGNN-MulMlp	19.39 ±0.60	**65.81** ±1.65	**83.57** ±1.40	8.375 ±0.746	0.3911 ±0.0061
GAT	14.51 ±1.17	55.19 ±2.14	72.87 ±2.42	28.074 ±4.756	0.3227 ±0.0110
GAT-NoAct	17.82 ±0.96	61.62 ±1.47	80.28 ±2.40	9.147 ±1.037	0.3702 ±0.0082
GAT-Mul	18.27 ±0.73	62.42 ±2.34	80.82 ±1.26	9.260 ±1.066	0.3749 ±0.0093
GAT-MulMlp	**18.93** ±1.18	**64.53** ±1.90	**83.06** ±1.92	**8.283** ±0.988	**0.3843** ±0.0114

Table 12: Comparison results on dataset group *HISTORY-SZ32-STP16-NDRP-STD0.2*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	16.44 ±3.08	47.22 ±3.67	59.40 ±4.04	34.573 ±4.230	0.3215 ±0.0319
RW-Dynamic	20.41 ±2.07	55.07 ±2.87	68.41 ±2.77	24.968 ±3.194	0.3656 ±0.0206
FullGN	16.61 ±4.79	48.68 ±12.52	59.51 ±14.14	63.999 ±22.762	0.3095 ±0.0765
FullGN-NoAct	20.80 ±2.37	**57.16** ±3.94	**70.07** ±4.00	**21.744** ±3.576	0.3729 ±0.0243
FullGN-Mul	21.35 ±3.31	52.83 ±3.56	64.58 ±3.05	30.538 ±3.866	0.3618 ±0.0286
FullGN-MulMlp	**23.94** ±1.75	54.98 ±3.86	65.76 ±3.01	30.251 ±5.933	**0.3850** ±0.0230
GGNN	22.56 ±2.43	53.52 ±3.10	62.88 ±2.97	70.509 ±11.580	0.3677 ±0.0179
GGNN-NoAct	21.69 ±2.26	**57.50** ±4.18	**69.54** ±3.70	**24.676** ±2.777	0.3818 ±0.0186
GGNN-Mul	23.81 ±2.21	54.66 ±2.18	65.62 ±2.62	29.538 ±3.694	0.3824 ±0.0175
GGNN-MulMlp	**26.06** ±1.51	56.04 ±2.94	66.18 ±3.54	32.089 ±5.656	**0.4001** ±0.0173
GAT	12.11 ±1.71	36.94 ±3.05	47.97 ±4.17	94.705 ±8.713	0.2333 ±0.0184
GAT-NoAct	**23.17** ±2.02	56.06 ±3.09	**68.00** ±2.89	**27.490** ±3.953	**0.3818** ±0.0186
GAT-Mul	22.70 ±2.58	**56.12** ±3.22	67.75 ±3.05	30.113 ±3.024	0.3762 ±0.0188
GAT-MulMlp	20.71 ±2.98	54.86 ±2.76	66.01 ±3.23	29.533 ±5.556	0.3655 ±0.0225

Table 13: Comparison results on dataset group *HISTORY-SZ32-STP16-NDRP-STD0.5*

Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
RW-Stationary	11.72 ±2.02	41.07 ±3.66	54.65 ±4.07	34.876 ±1.670	0.2547 ±0.0231
RW-Dynamic	12.25 ±1.39	46.04 ±3.81	64.03 ±3.78	20.600 ±2.735	0.2820 ±0.0191
FullGN	13.79 ±1.65	51.04 ±3.10	**69.30** ±2.87	31.874 ±4.526	0.3051 ±0.0207
FullGN-NoAct	13.90 ±1.29	49.87 ±2.94	67.39 ±2.87	**15.716** ±1.772	0.3059 ±0.0161
FullGN-Mul	12.93 ±0.90	45.60 ±2.75	61.88 ±2.12	21.881 ±2.319	0.2807 ±0.0120
FullGN-MulMlp	**14.89** ±0.96	**51.20** ±2.36	68.48 ±2.88	17.917 ±2.680	**0.3145** ±0.0103
GGNN	**14.83** ±1.27	**53.28** ±3.22	**70.76** ±2.63	33.821 ±7.460	**0.3217** ±0.0153
GGNN-NoAct	13.84 ±1.21	48.51 ±2.39	65.86 ±2.34	**20.799** ±1.546	0.2957 ±0.0126
GGNN-Mul	14.16 ±1.06	47.27 ±3.53	63.62 ±3.58	23.457 ±2.224	0.2971 ±0.0168
GGNN-MulMlp	14.80 ±1.26	49.47 ±2.74	66.35 ±3.00	22.230 ±2.410	0.3053 ±0.0142
GAT	10.80 ±1.14	43.44 ±2.81	60.12 ±3.54	73.118 ±7.201	0.2538 ±0.0105
GAT-NoAct	12.60 ±1.48	45.84 ±3.56	62.84 ±2.95	**21.903** ±1.581	0.2829 ±0.0179
GAT-Mul	13.49 ±1.28	46.23 ±3.15	62.68 ±2.85	23.887 ±2.012	0.2885 ±0.0149
GAT-MulMlp	**13.75** ±1.19	**47.93** ±2.72	**64.23** ±2.23	22.535 ±1.538	**0.2933** ±0.0132

Table 14: Comparison results on dataset groups *{LINE,SINE,LOCATION,HISTORY}-SZ32-STP16-EDRP-STD0.2*

	Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
LINE	FullGN	17.59 ±1.17	65.27 ±2.71	**85.47** ±1.96	8.239 ±1.769	0.3780 ±0.0141
	-NoAct	19.39 ±1.14	64.42 ±2.06	83.84 ±1.67	7.189 ±0.458	0.3887 ±0.0132
	-Mul	20.20 ±1.28	65.51 ±2.77	84.56 ±2.30	6.260 ±0.499	0.3985 ±0.0145
	-MulMlp	**20.26** ±1.26	**66.35** ±2.87	84.88 ±2.40	**6.185** ±0.501	**0.3999** ±0.0161
SINE	FullGN	**63.08** ±2.03	**90.32** ±2.38	**93.16** ±2.21	18.052 ±5.502	**0.7464** ±0.0187
	-NoAct	52.87 ±2.48	72.77 ±4.75	81.82 ±3.51	10.496 ±2.357	0.6193 ±0.0247
	-Mul	58.74 ±2.14	75.56 ±4.72	83.33 ±3.38	**7.518** ±1.472	0.6681 ±0.0281
	-MulMlp	56.94 ±1.79	76.97 ±2.38	83.37 ±2.73	7.710 ±1.647	0.6606 ±0.0162
LOCA	FullGN	22.74 ±5.22	68.86 ±7.39	81.88 ±4.49	20.447 ±6.473	0.4213 ±0.0554
	-NoAct	41.18 ±2.98	82.45 ±1.74	**91.31** ±1.61	**5.390** ±0.849	0.5881 ±0.0249
	-Mul	37.69 ±4.07	77.94 ±4.28	87.52 ±2.72	8.093 ±1.841	0.5501 ±0.0401
	-MulMlp	**43.39** ±3.26	**84.32** ±3.50	90.11 ±3.15	6.249 ±1.747	**0.6081** ±0.0297
HIST	FullGN	18.62 ±2.96	61.12 ±3.74	74.94 ±3.95	32.250 ±12.497	0.3648 ±0.0264
	-NoAct	27.96 ±2.04	**65.73** ±3.95	**76.76** ±3.44	**13.564** ±3.033	**0.4494** ±0.0200
	-Mul	28.40 ±3.76	61.72 ±5.35	72.94 ±3.07	19.180 ±3.192	0.4332 ±0.0338
	-MulMlp	**29.13** ±2.91	62.26 ±3.71	72.31 ±3.65	18.469 ±2.971	0.4416 ±0.0297

Table 15: Comparison results on dataset groups *{LINE,SINE,LOCATION,HISTORY}-SZ32-STP16-EDRP-STD0.5*

	Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
LINE	FullGN	16.87 ±1.85	**63.11** ±2.29	**82.59** ±1.53	9.296 ±1.374	0.3669 ±0.0177
	-NoAct	18.75 ±1.67	62.33 ±2.36	81.39 ±2.00	8.564 ±0.822	0.3764 ±0.0189
	-Mul	19.42 ±1.79	63.04 ±2.25	81.63 ±1.75	**7.811** ±0.684	0.3857 ±0.0193
	-MulMlp	**19.63** ±1.85	63.03 ±2.29	81.56 ±1.59	8.029 ±0.526	**0.3867** ±0.0198
SINE	FullGN	19.08 ±1.11	**62.90** ±1.61	**80.16** ±1.61	18.303 ±9.662	0.3759 ±0.0117
	-NoAct	20.41 ±1.61	56.18 ±2.94	73.19 ±3.01	11.914 ±1.874	0.3656 ±0.0160
	-Mul	21.53 ±1.42	59.52 ±2.30	75.77 ±1.73	**11.893** ±1.124	0.3829 ±0.0132
	-MulMlp	**22.12** ±1.21	60.09 ±2.66	75.74 ±1.96	12.262 ±1.532	**0.3884** ±0.0134
LOCA	FullGN	15.95 ±1.08	56.61 ±2.39	76.61 ±1.77	15.732 ±4.803	0.3392 ±0.0100
	-NoAct	18.55 ±1.54	59.81 ±2.67	78.41 ±2.44	9.751 ±1.260	0.3703 ±0.0183
	-Mul	19.32 ±1.14	60.34 ±1.82	79.69 ±1.92	10.382 ±1.586	0.3759 ±0.0136
	-MulMlp	**19.36** ±1.21	**60.92** ±2.14	**80.31** ±1.84	**9.352** ±1.572	**0.3783** ±0.0132
HIST	FullGN	14.00 ±1.52	51.60 ±2.51	70.76 ±3.17	25.511 ±6.818	0.3091 ±0.0145
	-NoAct	16.61 ±1.49	**54.18** ±2.26	**70.87** ±2.95	**13.968** ±1.772	**0.3343** ±0.0149
	-Mul	**16.97** ±0.94	50.46 ±2.38	67.69 ±1.77	16.933 ±2.255	0.3252 ±0.0123
	-MulMlp	16.27 ±1.31	51.63 ±3.22	68.57 ±3.77	16.647 ±2.403	0.3258 ±0.0158

Table 16: Comparison results on dataset groups *{LINE,SINE,LOCATION,HISTORY}-SZ64-STP32-NDRP-STD0.2*

	Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
LINE	GGNN	12.36	48.34	71.44	15.348	0.2918
	GGNN-MulMlp	**15.56**	**55.01**	**75.34**	**9.050**	**0.3335**
SINE	GGNN	27.08	53.67	76.19	73.990	0.4047
	GGNN-MulMlp	**39.98**	**69.10**	**78.52**	**12.078**	**0.5317**
LOCATION	GGNN	20.98	74.25	92.26	19.753	0.4323
	GGNN-MulMlp	**47.56**	**91.17**	**97.20**	**3.471**	**0.6534**
HISTORY	GGNN	11.47	46.08	**71.66**	67.990	0.2785
	GGNN-MulMlp	**23.49**	**58.55**	71.43	**36.734**	**0.3888**

Table 17: Comparison results on dataset groups *{LINE,SINE,LOCATION,HISTORY}-SZ64-STP32-NDRP-STD0.5*

	Model	Hits@1 (%)	Hits@5 (%)	Hits@10 (%)	MR	MRR
LINE	GGNN	11.04	46.54	71.39	15.953	0.2759
	GGNN-MulMlp	**14.53**	**52.56**	**72.32**	**11.489**	**0.3179**
SINE	GGNN	10.53	43.77	65.82	36.048	0.2616
	GGNN-MulMlp	**16.08**	**49.77**	**66.53**	**20.024**	**0.3137**
LOCATION	GGNN	9.74	37.14	58.94	42.255	0.2347
	GGNN-MulMlp	**18.39**	**62.45**	**82.35**	**7.931**	**0.3782**
HISTORY	GGNN	8.67	35.42	55.62	65.156	0.2222
	GGNN-MulMlp	**13.60**	**47.06**	**64.77**	**33.193**	**0.2902**

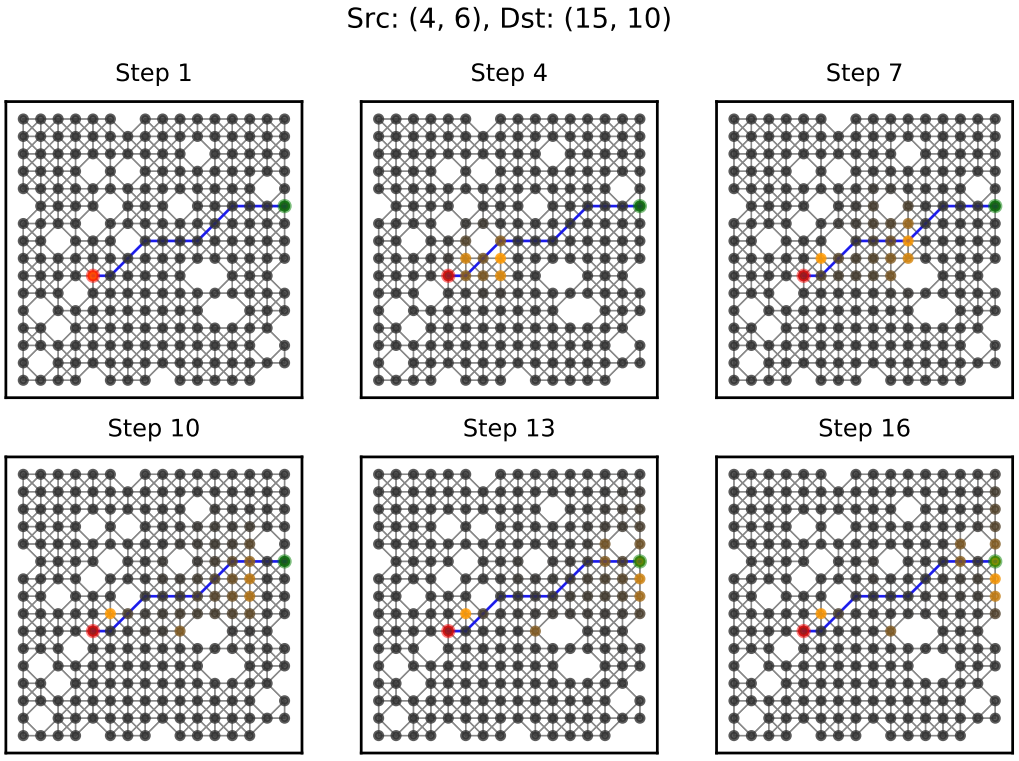

Figure 3: Attention distributions at different steps. The latent directions follow a straight line.

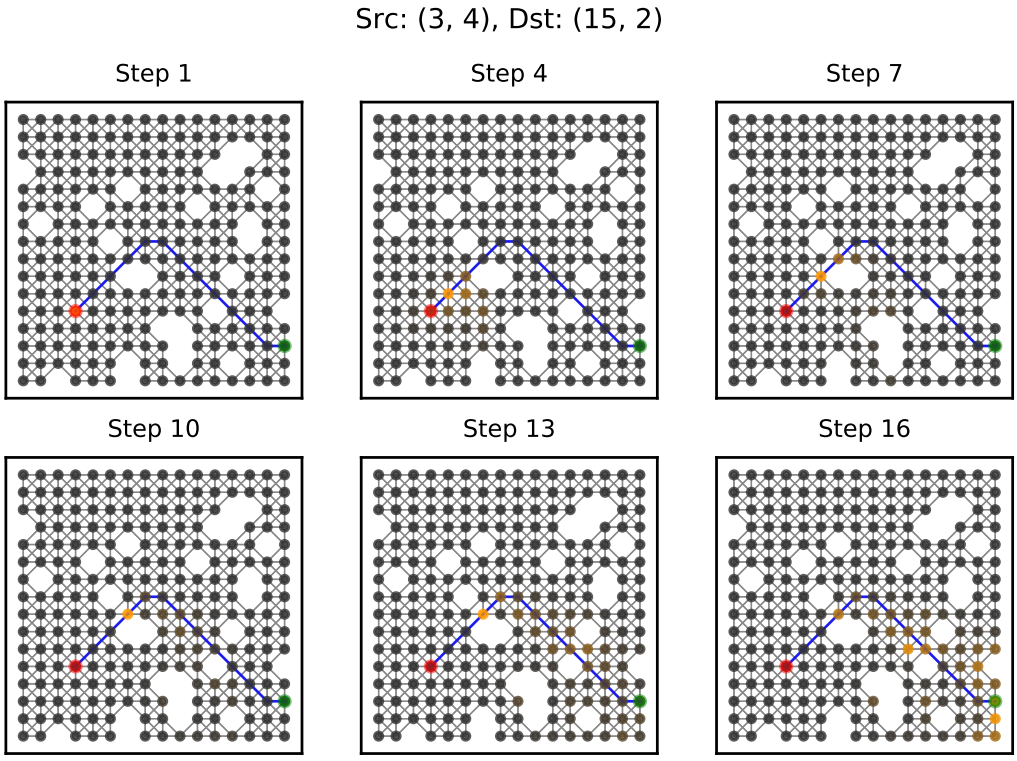

Figure 4: Attention distributions at different steps. The latent directions follow a sine curve, depending on time.

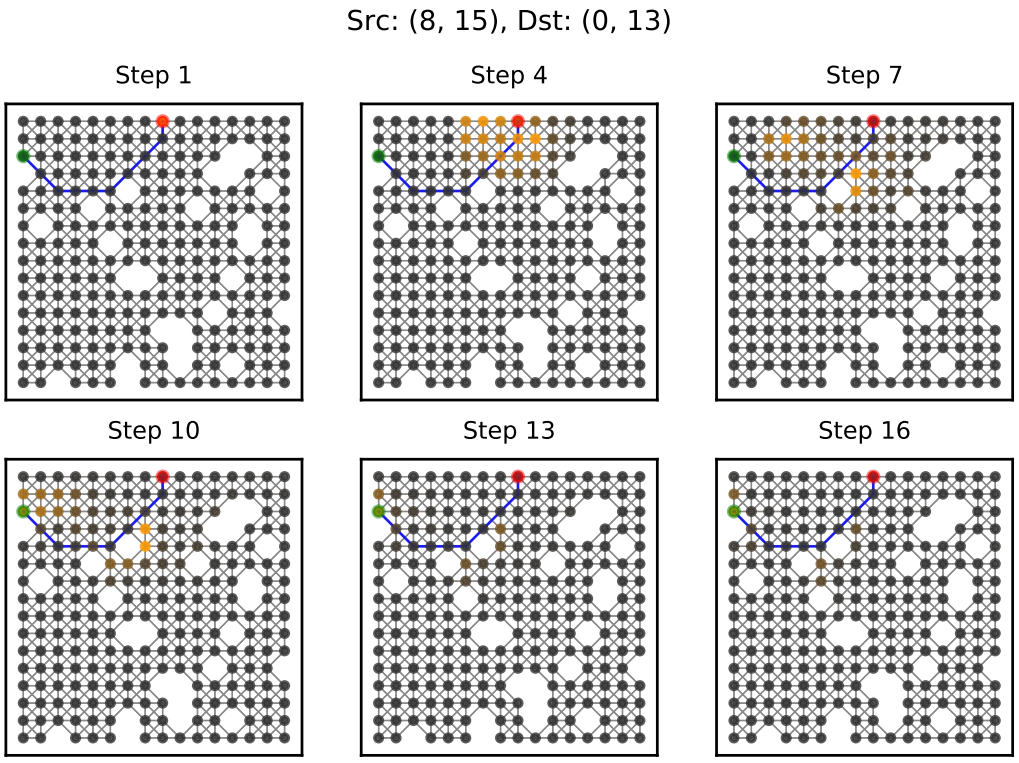

Figure 5: Attention distributions at different steps. The latent directions depend on current locations.

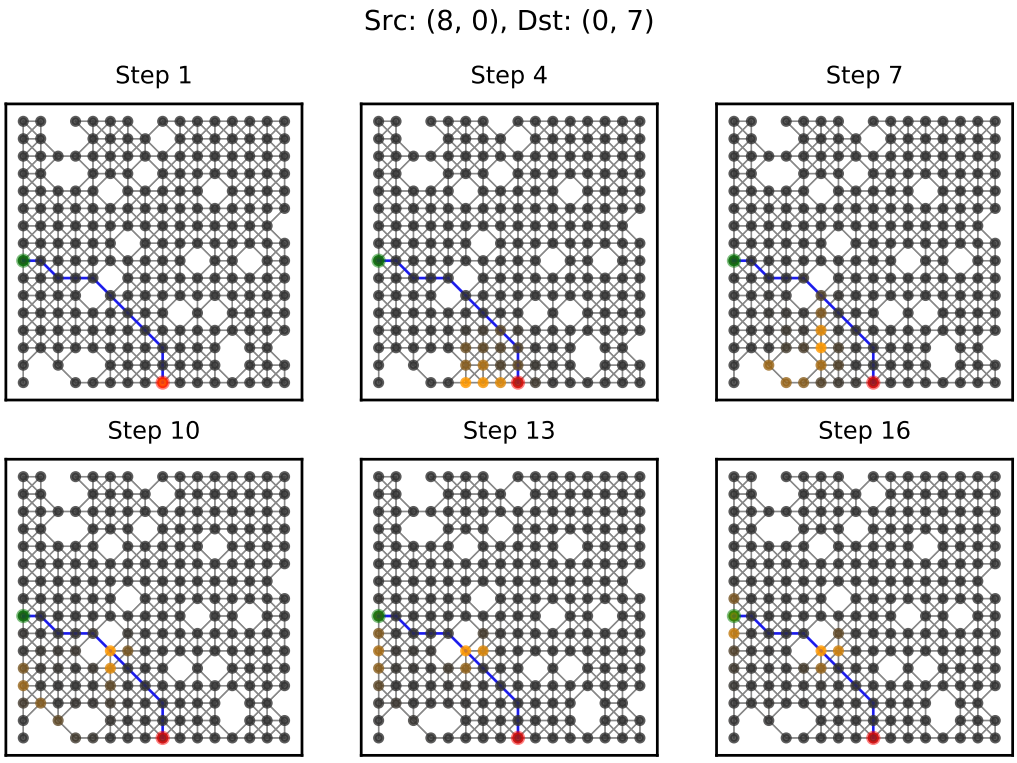

Figure 6: Attention distributions at different steps. The latent directions depend on location history.