Limitations of Perturbation-based Explanation Methods for Temporal Graph Neural Networks

Minh N. Vu and My T. Thai
Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611, USA
Email: {minhvu, mythai}@cise.ufl.edu

2023 IEEE International Conference on Data Mining (ICDM) | 979-8-3503-0788-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDM58522.2023.00071

***Abstract*—Recently, there has been significant interest in Temporal Graph Neural Networks (TGNN) because of their capability to learn from time-evolving graph-related data. However, similar to Graph Neural Network (GNN), explaining the predictions of TGNN is non-trivial due to its black-box and complex nature. A major approach for this problem in GNNs is by analyzing the model's responses to some perturbations of the model's inputs, called perturbation-based explanation methods. These methods are convenient and flexible as they do not require access to the model's internals. However, a question arises: Does the lack of internal access limit these methods from uncovering crucial information about the predictions? Motivated by the question, this study explores the limitations of some popular classes of perturbation-based explanation methods. By constructing specific instances of TGNNs, we show (i) Node-perturbation is not reliable for identifying the paths that carry out the prediction, (ii) Edge-perturbation cannot reliably determine all the nodes contributing to the prediction and (iii) perturbing both nodes and edges does not consistently help identify the graph components responsible for the temporal aggregation in TGNNs. Our experimental results further demonstrate situations for failures of explanations can occur frequently in both synthetic and real-world scenarios. Thus, they emphasize the importance of perturbation choices and the internal information of the explained model in determining faithful explanations of the model's predictions.**

I. Introduction

Graph Neural Networks have been achieving successful performance in many practical graph-related problems including social networks, citation networks, and biological networks [1], [2], [3]. Various architectures with elegant designs and competitive performance have been introduced in recent years [4], [5], [6]. Along those works, a notable branch of GNNs is developed to integrate temporal information into the graph structure, called Temporal Graph Neural Networks [7], [8], [9]. This variant has shown promising outcomes in domains where the data has strong correlations with time such as transportation and weather forecast.

Since GNNs and TGNNs inherit the black-box nature of neural networks, interpreting their predictions remains non-trivial as internal information about the models is not available. In response, many explanation methods, called explainers, have been introduced to explain local predictions of GNNs [10], [11], [12], [13]. These methods generally rely on the model's responses to some perturbations of the input to find the explanations. While the approach has shown many heuristic successes, there is little theoretical result that follows. In particular, is there any information on the model's internal behavior that a given method of perturbation cannot uncover? Addressing this question will help us design better explainers for variants of GNNs, such as TGNNs and many other architectures to come. More importantly, analyzing the limits of explaining methods also helps prevent false claims and incorrect inferences from the explanations.

Our work focuses on the limit of perturbation-based explanation methods when applying to TGNN, i.e. what information cannot be revealed by some given class of popular perturbations. The classes are categorized based on the input's features that they perturb: *node-only*, *edge-only*, and *node-and-edge*. We introduce a proof structure, called *Unidentifiable Proof*, through which the limit of perturbation methods can be formalized and examined. For each class of explainers, we identify a training task (Fig. 1) and construct some models such that there is no method in the class that can identify the internal dynamics of those models when generating predictions. Specifically, given a constant K depending on the model's parameters, we show:

- Node-perturbation methods bounded by K cannot identify the path carrying out the message passing. (Fig. 1a).
- Edge-perturbation methods cannot identify all nodes contributing to a max aggregation. (Fig. 1b).
- Node-and-Edge-perturbation methods bounded by K cannot identify which nodes carry out the temporal aggregation. (Fig. 1c).

In practice, K is the result of training and can be arbitrarily large. Thus, our analysis is relevant and applicable to many practical scenarios as large perturbations are often weighted lightly due to the notion of locality [14]. While most of our results are applicable to GNN, we focus on TGNN due to its lack of study. Another reason is, as TGNNs add the temporal dimension to GNNs, it introduces a corresponding temporal dimension to the explaining problem. We find studying this temporal aspect novel and interesting by itself.

The outline of this manuscript is as follows. Sect. II and Sect. III discuss the related works and preliminaries, respectively. Our proposed Unidentifiable Proof and some related notions are introduced in Sect. IV. Sects. V, VI and VII formally describe and prove the type of information that Node-perturbation, Edge-perturbation, and Node-and-Edge-perturbation cannot identify. Sect. VIII provides synthetic and real-world experiments showing the impact of perturbation schemes on the explaining tasks. Sect. IX concludes the paper

This work is partially supported by the National Science Foundation under Grant No. FAI-1939725 and SCH-2123809.

979-8-3503-0788-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDM58522.2023.00071

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

with discussions on the theoretical implications and practical aspects of our results.

II. Related Works

To our knowledge, there is currently no work theoretically studying the limits of explanations for GNNs or TGNNs. Even though many experiments evaluating explanation methods have been conducted [15], [16], there exist many pitfalls and challenges in those evaluations as the ground-truth explanations are often unavailable [17]. Furthermore, with the increasing number of datasets, model architectures, and explanation methods, conducting comprehensive evaluations is becoming much more challenging, especially for black-box methods of which the computation complexity is significantly higher than that of white-box methods [16].

Our work is directly related to black-box perturbation-based explaining methods for GNNs, including GNNExplainer [10], PGExplainer [18], GraphLIME [13], and some others [12], [19], [11], [20]. Table I summarizes the target of perturbation conducted by those explainers and the scope of results in each section of this paper.

TABLE I: Summary of perturbation methods used by explainers and the scope of our results.

	Node	Edge	Sect.V	Sect.VI	Sect.VII
GNNExplainer [10]	*	*			*
PGExplainer [18]		*		*	*
GraphLIME [13]	*		*		*
PGMExplainer [12]	*		*		*
RelEx [19]		*		*	*
GraphSVX [11]	*		*		*
ZORRO [20]	*		*		*

III. Preliminaries

We now introduce some preliminaries and notations that are commonly used in the research of GNNs and the explaining problem. We also briefly introduce Dynamic Bayesian Networks, which we use in our Unidentifiable Proofs.

Notation. For all models studied in this work, their inputs are defined on a graph $G = (V, E)$, where V is the set of nodes and E is the set edges. The inputs of TGNN are a sequence of feature vectors $\boldsymbol{X}_{t_s,t_e} \coloneqq [X^{(t_s)}, \cdots, X^{(t_e)}]$ and an adjacency matrix $A \in \mathcal{A} \coloneqq \{0,1\}^{|V|\times|V|}$. Here, t_s, t_e and $X^{(t)} \in \mathbb{R}^{|V|\times F}$ denote the starting time, the ending time, and the input feature sequence. The model is referred to by its forwarding function $\Phi : \mathcal{X} \times \mathcal{A} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the spaces of the input feature sequence and the output.

Graph Neural Networks. We use the general formulation of GNNs based on the message passing mechanism [5], which involves 3 computations: propagation, aggregation, and update:

$$m_{ij}^{(l)} = \text{MSG}\left(h_i^{(l-1)}, h_j^{(l-1)}\right),$$
$$a_i^{(l)} = \text{AGG}\left(\left\{m_{ji}^{(l)}\right\}_{j\in\mathcal{N}_i}\right), h_i^{(l)} = \text{UPD}\left(a_i^{(l)}, h_i^{(l-1)}\right)$$

where m_{ij} is the message from node i to node j, $h_i^{(l)}$ is the hidden representations of node i at layer l and $\mathcal{N}_i$ is node i's neighbors. The final representation at the last layer L, $h_i^{(L)}$, is commonly used to generate a prediction, i.e. $Y = \text{READOUT}(h_i^{(L)})$. Typically, the MSG, UPD, and READOUT functions consist of trainable weights and biases followed by an activation function. Some common choices for the AGG are max, mean, and concatenating operations.

Temporal Graph Neural Networks. The forwarding function $\Phi : \mathcal{X} \times \mathcal{A} \to \mathcal{Y}$ of a TGNN can be formulated based on its sequential implementation [7]:

$$H^{(t_s)} = \bar{\Phi}(X^{(t_s)}, A)$$
$$H^{(t)} = \bar{\Phi}(H^{(t-1)}, X^{(t)}, A),\ t = t_s + 1, ...t_e \quad (1)$$

where $\bar{\Phi}$ is the forwarding function of a GNN and $H^{(t)}$ is the temporal messages. $T \coloneqq t_e - t_s + 1$ is the input's length.

The *base* GNN $\bar{\Phi}$ typically consists of some graph layers followed by a readout. The output Y is computed either by applying a readout on the temporal message at the last layer $H^{(t_e)}$ or from the node's final hidden features. In this manuscript, capital letters, e.g. X, Y and H, refer to external signals of the GNN blocks, while small letters, e.g. m, a and h, are for internal signals.

The Class of Explainers. This work studies black-box explainers of GNNs and TGNNs based on the type of perturbations that the explainers use:

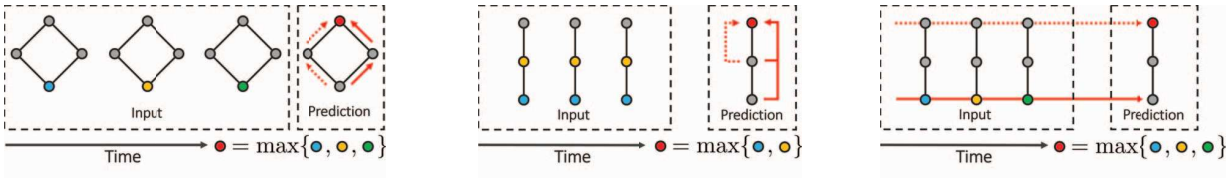
- Node-perturbation class $\mathcal{G}_v$: the explainer can perturb the entries of the feature matrices in $\boldsymbol{X}_{t_s,t_e}$.
- Edge-perturbation class $\mathcal{G}_e$: the explainer can remove some edges from the input adjacency matrix A.
- Node-and-Edge-perturbation class $\mathcal{G}_a$: the explainer can perturb both the feature matrices in $\boldsymbol{X}_{t_s,t_e}$ and remove some edges in the input adjacency matrix A.

Dynamic Bayesian Networks (DBNs). The usage of DBNs in this work is to model internal computations of TGNNs so that theoretical analysis can be conducted. A DBN [21] can be considered as an extension of Bayesian networks (BNs) [22] to model the temporal dependency of systems' variables. Temporal information is integrated via edges between adjacent time steps. Figs. 2a shows an example of a DBN: the Two-Timeslice Bayesian Network (2TBN) [23]. Its equivalent BN in the form of unrolled 4-time-step BN is shown in Figs. 2b. Readers can find more details of DBNs in [23].

IV. Unidentifiable Proof for Neural Networks

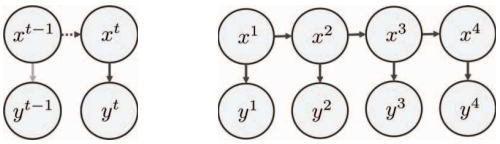
Given a black-box model Φ and a class of explanation methods, the Unidentifiable Proof formalizes the idea that certain information of Φ cannot be identified and used as the explanation by a class of explainers. Before describing the Unidentifiable Proof, we need to formalize the *ground-truth explanation*. The first two subsections discussing about the *interpretable domain* and the *Transparent Model* serve that purpose. Intuitively, the interpretable domain is the domain of all available explanations and the Transparent Model is a domain's member that can faithfully capture the model. The latter part of this section describes the Unidentifiable Proof. The general idea of the Unidentifiable Proof is by construction: it constructs two instances of the model whose Transparent

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.



a Multi-path aggregation task. b Multi-node aggregation task. c Temporal aggregation task

Fig. 1: Tasks for the class of (a) Node-perturbation, (b) Edge-perturbation, and (c) Node-and-Edge-perturbation explanation methods. The dash arrows and the dotted arrows show different internal computations that the model can carry out. Our Unidentifiable Proofs show that the corresponding explanation methods cannot differentiate the computations; thus, cannot identify/explain those internal dynamics.



a A 2TBN. b The unroll BNs.

Fig. 2: An example of a DBN and its unroll BN. The intra-slice connections are solid and the inter-slice connections are dashed. The brighter line can be omitted as it can be inferred from other edges.

Models are different; however, the information extracted from them by a given class of explainers is exactly the same. This means no explainer of that class can identify the information differentiating the two Transparent Models. This gives us formal notions of unidentifiable information.

A. The Interpretable Domain

Given a black-box model Φ and an input $\boldsymbol{X}$, the explainers solve for an interpretable representation of the prediction $\Phi(\boldsymbol{X})$, denoted as $g(\Phi(\boldsymbol{X}))$. For the sake of explaining, $g(\Phi(\boldsymbol{X}))$ should be intuitive and interpretable; therefore, we call the space of $g(\Phi(\boldsymbol{X}))$ the *interpretable domain*. For example, the interpretable domains for GNNs have been chosen to be a set of scores on some nodes/edges' features, the set of linear functions, and the set of probabilistic models on the input's nodes [11], [12], [13]. Intuitively, a good interpretable domain should balance its representative power and interpretability. In this work, we consider it to be the set of DBNs. We describe how DBNs can help explain TGNNs in the next subsection.

B. The Transparent Model

Given an interpretable domain and a black-box model, there is no guarantee that there exists an interpretable representation that correctly explains Φ. Nevertheless, in some specific contexts, an interpretable representation that can fully describe the black-box model exists. Particularly, the work [24] embeds a linear function inside a black-box model, which means that a linear function can faithfully describe and explain that black-box. This implies, for a given interpretable domain and for some Φ, an explanation that can fully explain Φ exists. We denote it by *the Transparent Model* $\mathcal{I}$. In some cases, the Transparent Model only exists for a subset of inputs $\mathcal{S} \subseteq \mathcal{X}$. We write the Transparent Model in those cases as $\mathcal{I}(\Phi(\boldsymbol{X})), \boldsymbol{X} \in \mathcal{S}$. Furthermore, we call the assumption $\mathcal{I}(\Phi(\boldsymbol{X}))$ exists the *Existence assumption*.

We now discuss the Transparent Model $\mathcal{I}(\Phi)$ in terms of DBNs. For all our Unidentifiable Proofs, the target of explanation will be the prediction on a node of the input graph. The explanation will be in the form of a DBN $\mathcal{B}$, whose variables are associated with the corresponding nodes in the input graph. As each node of the input graph is physically associated with a distinct set of neurons in the graph layers of the TGNN, we associate each variable of $\mathcal{B}$ to the sending messages of the neurons corresponding to that node in the TGNN. Note that the sending messages from a node consist of not only internal messages in the graph layers but also temporal messages and output messages. These associations allow us to capture the dynamics of the TGNN via DBN. Fig. 3 provides an illustration of these associations.

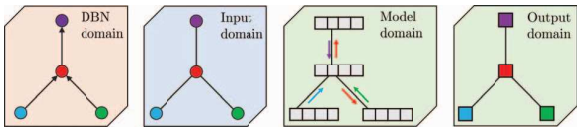


Fig. 3: The association among variables of the explanation DBN, the input nodes and the messages in the TGNN: Components of the same color are associated with each other.

We say a DBN $\mathcal{B}$ is the Transparent Model of a model Φ if (i) all independence claims of $\mathcal{B}$ about its variables are consistent with the messages sent from the corresponding neurons in the model and (ii) $\mathcal{B}$ is minimal. Condition (i) is obvious since incorrect claims from the explanation are undesirable. Note that, this condition implies the DBN $\mathcal{B}$ can represent all communicating messages during the forward computation of Φ. Condition (ii) enforces the explaining DBN to be as informative as possible, i.e. it should remove unnecessary edges when they do not help explain the model's computations.

C. The Unidentifiable Proof

Under the Existence assumption, i.e. $\mathcal{I}(\Phi(\boldsymbol{X}))$ exists, a good explanation method g is expected to return $g(\Phi(\boldsymbol{X}))$ to be similar to $\mathcal{I}(\Phi(\boldsymbol{X}))$. This provides us a necessary condition to theoretically analyze the limits of explanation methods: given two models Φ_1 and Φ_2 with distinctively different transparent models $\mathcal{I}(\Phi_1)$ and $\mathcal{I}(\Phi_2)$, a good explainer must return different explanations, i.e. $g(\Phi_1(\boldsymbol{X})) \neq g(\Phi_2(\boldsymbol{X}))$. This necessary condition is illustrated via Fig. 4.

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

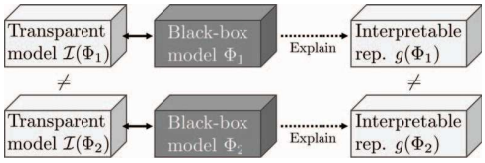


Fig. 4: Necessary explaining conditions in the Existence assumption.

The above necessary conditions can be formalized as follows. Given two neural networks Φ_1 and Φ_2, the Unidentifiable Proof holds if their Transparent Models exist and:

$$g(\Phi_1(\boldsymbol{X})) = g(\Phi_2(\boldsymbol{X})),\ \forall g \in \mathcal{G}, \forall \boldsymbol{X} \in \mathcal{S} \subseteq \mathcal{X} \quad (2)$$

$$\exists \boldsymbol{X} \in \mathcal{S} \subseteq \mathcal{X} \text{ s.t } \mathcal{I}(\Phi_1(\boldsymbol{X})) \neq \mathcal{I}(\Phi_2(\boldsymbol{X})) \quad (3)$$

The first condition says the explanations of the two models provided by all explainers in $\mathcal{G}$ are the same. The condition can be shown by examining the forwarding computations of the two models. The second condition states the existence of some inputs such that their Transparent Models are different. The main challenge in proving that condition is in concretely determining $\mathcal{I}(\Phi_1(\boldsymbol{X}))$ and $\mathcal{I}(\Phi_2(\boldsymbol{X}))$. The two conditions then imply the explainer cannot learn the Transparent Model of at least one of the two models. More importantly, as g outputs the same information in explaining both models, any information that can be used to differentiate the two models cannot be inferred from g. Thus, all information differentiating $\mathcal{I}(\Phi_1(\boldsymbol{X}))$ and $\mathcal{I}(\Phi_2(\boldsymbol{X}))$ cannot be inferred from the explainer either. The arguments, therefore, establish the unidentifiable information for the class of explainers $\mathcal{G}$.

V. Unidentifiable Proof for Node-perturbation

We now provide the Unidentifiable Proof for the Node-perturbation class $\mathcal{G}_v$. We show that for a simple max computation conducted by the TGNNs, Node-perturbation cannot identify the messages' propagating paths carrying out the predictions in the model. We also elaborate on how the result can be applied to GNNs in Sect. IX.

The Training Task. In this construction, the TGNNs operate on a graph of 4 nodes forming a square, with the following adjacency matrix:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

The training task is to recognize the maximum positive inputs of node 3 and return the result at node 1:

$$Y_1^{(t)} = \max\{0 \text{ and } X_3^{(t')}, 0 \le t' \le t\}$$
$$Y_2^{(t)} = Y_3^{(t)} = Y_4^{(t)} = 0 \quad (4)$$

The model's input and output at each time-step t are both in $\mathbb{R}^4$. Fig. 1a provides an illustration of this training task.

The Models. We construct two TGNNs, Φ_1^v and Φ_2^v, with the same architecture but different parameters. We consider the input's length $T = 2$ for the sake of brevity. Each node i is associated with a hidden feature vector $\boldsymbol{h}_i = [hr_i, ht_i, hs_i, hz_i, ho_i^+, ho_i^-] \in \mathbb{R}^6$, whose features mean:

- hr_i: message that node i receives.
- ht_i: temporal message that node i receives.
- hs_i: feature determining if node i sends message.
- hz_i: feature determining if node i outputs zero.
- ho_i^+ and ho_i^-: features determining the output of node i.

During the all computations, h_s and h_z are constant. In practice, they can be the results of a zero weight combined with a constant bias. Their values in the two constructed models are shown in Table II. The upcoming construction will ensure that, if $hs_i = k_s$, node i does not send any message, and if $hz_i = k_z$, the output of node i will be zero.

TABLE II: The constant features of the TGNNs in G_v's proof.

Node	**1**		**2**		**3**		**4**	
Hidden features	hs_1	hz_1	hs_2	hz_2	hs_3	hz_3	hs_4	hz_4
TGNN Φ_1^v	k_s	0	0	k_z	0	k_z	k_s	k_z
TGNN Φ_2^v	k_s	0	k_s	k_z	0	k_z	0	k_z

Our proposed TGNN architecture has 2 graph layers followed by a READOUT layer (Fig. 5). By conventions, we use $l \in \{0, 1, 2\}$ to indicates the model's graph layers with $h_i^{(t,l=0)}$ refers to the model's input:

$$\boldsymbol{h}_i^{(t,l=0)} = \left[X_i^{(t)}, H_i^{(t-1)}, *, *, 0, 0\right] \quad (5)$$

where $*$ means the features are determined by the model's weights and biases, i.e. by hs and hz. The temporal signal $H^{(t-1)}$ has the same dimension as the model's output Y.

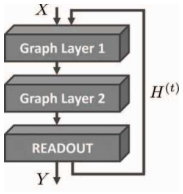


Fig. 5: The model.

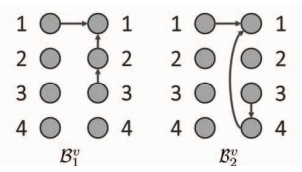


Fig. 6: The DBNs explaining the two TGNNs Φ_1^v and Φ_2^v.

The two models have trainable weights and biases such that the MSG and AGG functions work as follows:

$$m_{ji}^{(t,l)} = \text{ReLU}\left(hr_j^{(t,l-1)} - hs_j^{(t,l-1)}\right) \quad (6)$$

$$a_i^{(t,l)} = \sum_{j \in \mathcal{N}_i} m_{ji}^{(t,l)} \quad (7)$$

for $l \in \{1, 2\}$. This means, if $hs_j = k_s$ is large and unchanged, there is no message coming out of node j. Thus, the a_i consists of messages only from node j with $hs_j = 0$.

Meanwhile, the UPD returns $\boldsymbol{h}_i^{(t,l)}$, which is $\text{ReLU}\left(\boldsymbol{w}_h^\top \boldsymbol{h}_i^{(t,l-1)} + w_a a_i^{(t,l)}\right)$. The parameters are chosen such that:

$$hr_i^{(t,l)} = \text{ReLU}\left(a_i^{(t,l)}\right), ht_i^{(t,l)} = \text{ReLU}\left(ht_i^{(t,l-1)}\right) \quad (8)$$

$$hs_i^{(t,l)} = \text{ReLU}\left(hs_i^{(t,l-1)}\right), hz_i^{(t,l)} = \text{ReLU}\left(hz_i^{(t,l-1)}\right)$$

$$ho_i^{\pm(t,l)} = \text{ReLU}\left(\pm a_i^{(t,l)} \mp ht_i^{(t,l-1)}\right) \quad (9)$$

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

The READOUT is chosen as:

$$H_i^{(t)} = \text{ReLU}\left((hr_i + ht_i + ho_i^+ + ho_i^-)/2 - hz_i\right) \quad (10)$$

Finally, the prediction of the model Y is assigned to $H^{(t=2)}$. We can check that h_t, h_s, and h_z are unchanged during the forwarding computation and hr_i is indeed the received signal at node i with the above specification.

The Forwarding computation of Φ_1^v and Φ_2^v. To show that the forwarding computations of the two constructed models satisfy the training tasks, we first pay attention to ho_i^+ and ho_i^-, whose summation is the difference between $a_i^{(t,l)}$ and $ht_i^{(t,l-1)}$. Thus, the maximum signal that node i received is:

$$\begin{aligned}
&\frac{1}{2}\left(hr_i^{(t,L)} + ht_i^{(t,L)} + ho_i^{+(t,L)} + ho_i^{-(t,L)}\right)\\
=&\frac{1}{2}\left(a_i^{(t,L)} + H_i^{(t-1)} + \left|a_i^{(t,l)} - H_i^{(t-1)}\right|\right)\\
=&\max\left\{a_i^{(t,L)}, H_i^{(t-1)}\right\} = \max\left\{hr_i^{(t,L)}, H_i^{(t-1)}\right\} \quad (11)
\end{aligned}$$

Due to the READOUT in (10), the outputs of nodes with large hz_i are zero. Thus, from Table II, we have:

$$H_i^{(t)} = \begin{cases} \max\left\{hr_i^{(t,L)}, H_i^{(t-1)}\right\} & \text{for } i = 1 \\ 0 & \text{for } i \in \{2,3,4\} \end{cases} \quad (12)$$

As the prediction of the model Y is set to $H^{(t=2)}$, to show that the models work as specified in the training tasks, we need to verify that $hr_1^{(t,L)} = X_3^{(t)}$ for all t. We now show the claim for Φ_1^v.

For $l = 1$, we have $m_{32}^{(t,1)} = hr_3^{(t,0)} = X_3^{(t)}$ from (6). Thus, $hr_2^{(t,1)} = a_2^{(t,1)} = m_{32}^{(t,1)} = X_3^{(t)}$ ((7) and (8)). For $l = 2$, $m_{21}^{(t,2)} = hr_2^{(t,1)} = X_3^{(t)}$ and $hr_1^{(t,2)} = a_1^{(t,2)} = m_{21}^{(t,2)} = X_3^{(t)}$. Therefore, Φ_1^v fulfills the training task. Note that we have $a_2^{(t,1)} = m_{32}^{(t,1)}$ and $a_1^{(t,2)} = m_{21}^{(t,2)}$ because there is no message sending from node 1 and node 4. The claim for Φ_2^v trivially follows by swapping node 2 with node 4.

The Transparent Models of Φ_1^v and Φ_2^v. We now examine the Transparent models $\mathcal{I}(\Phi_1^v(\boldsymbol{X}))$ and $\mathcal{I}(\Phi_2^v(\boldsymbol{X}))$. Fig. 6 shows two DBNs whose variables represent the messages coming out of the model's nodes. Particularly, the variable of node i at time t, denoted by $\mathcal{V}_i^t$, represents $m_{ij}^{(t,l)}$ $(\forall l, \forall j \in \mathcal{N}_i)$ and the $H_i^{(t)}$. Our claim is the two DBNs can faithfully explain the two models when their inputs are bounded by $K := \min\{k_s, k_z\}$:

Lemma 1. *The DBN $\mathcal{B}_1^v$ ($\mathcal{B}_2^v$) in Fig. 6 can embed all information of the hidden features of TGNN Φ_1^v (Φ_2^v) without any loss when the input signal is bounded by $K := \min\{k_s, k_z\}$. Furthermore, the DBN is minimal.*

Proof. To show that the DBN $\mathcal{B}_1^v$ can represent Φ_1^v without any loss, we show:

- $\mathcal{B}_1^v$ can express how the predictions $H_i^{(t)}$ are generated.
- $\mathcal{B}_1^v$ can express how the messages propagate in the model.

The first claim only involves node $i = 1$ (as $H_i^{(t)} = 0$ for all other nodes). Because $H_1^{(t)} = \max\left\{X_3^{(t)}, H_1^{(t-1)}\right\}$ (shown below (12)), the paths from $\mathcal{V}_3^t$ to $\mathcal{V}_1^t$ and from $\mathcal{V}_1^{t-1}$ to $\mathcal{V}_1^t$ are sufficient to represent how the predictions are generated.

From Table II, we observe that, as long as the inputs are bounded by K, only nodes 2 and 3 send messages. Thus, to show the second claim, we only need to consider messages from those nodes. It is clear from (6), (7) and (8) that:

$$\begin{aligned}
&X_2^{(t)} \to m_{21}^{(t,1)} \to a_1^{(t,1)} \to hr_1^{(t,1)} \to \varnothing\\
&X_2^{(t)} \to m_{23}^{(t,1)} \to a_3^{(t,1)} \to hr_3^{(t,1)} \to m_{32}^{(t,2)} \text{ and } m_{34}^{(t,2)}\\
&X_3^{(t)} \to m_{32}^{(t,1)} \to a_2^{(t,1)} \to hr_2^{(t,1)} \to m_{23}^{(t,2)} \text{ and } m_{21}^{(t,2)}\\
&X_3^{(t)} \to m_{34}^{(t,1)} \to a_4^{(t,1)} \to hr_4^{(t,1)} \to \varnothing
\end{aligned}$$

where the arrow means *determining* and $\to \varnothing$ means the signals result in no other messages. We can see that the messages sent out from nodes 2 and 3 are only dependent on the signals of those nodes. As those dependencies can be captured by an edge between $\mathcal{V}_2^t$ and $\mathcal{V}_3^t$, we have the claim.

The above arguments also show that $\mathcal{B}_1^v$ is minimal. Specifically, the edges $(\mathcal{V}_1^{t-1}, \mathcal{V}_1^t)$ and $(\mathcal{V}_2^t, \mathcal{V}_3^t)$ are necessary because of the temporal dependency $H_1^{(t-1)} \to H_1^{(t)}$ and the messages' dependency between nodes 2 and 3. We then require a path from $\mathcal{V}_3^t$ to $\mathcal{V}_1^t$ to capture the dependency $X_3^{(t)} \to H_1^{(t)}$ when $X_3^{(t)} > H_1^{(t-1)}$. Thus, at least another edge is needed. Since $\mathcal{B}_1^v$ has 3 edges, it is minimal. □

From Lemma 1, we write $\mathcal{B}_1^v = \mathcal{I}(\Phi_1^v(\boldsymbol{X}))$ and $\mathcal{B}_2^v = \mathcal{I}(\Phi_2^v(\boldsymbol{X}))$ for all $\boldsymbol{X}$ whose entries are bounded by K.

Unidentifiable Proof. As the two DBNs contain distinct information regarding $\mathcal{V}_2$ and $\mathcal{V}_4$, for an explainer that is capable of explaining the two corresponding models Φ_1^v and Φ_2^v, it must be able to differentiate the two DBNs. Unfortunately, in the next Lemma 2, we show that the outputs of the two TGNNs are the same under node-perturbation; hence, explainers of the class cannot explain them:

Lemma 2. *For all $\boldsymbol{X}$ such that $X_i^{(t)} \le \min\{k_s, k_z\}$, we have $\Phi_1^v(\boldsymbol{X}) = \Phi_2^v(\boldsymbol{X})$.*

Proof. From the examination of the forwarding computations, we know that both models satisfy (4) for all $\boldsymbol{X}$ bounded by $\min\{k_s, k_z\}$. Thus, their outputs on such $\boldsymbol{X}$ are the same. Thus, we have the Lemma. □

We are now ready for the unidentifiable result of Node-perturbation:

Theorem 1. *For a TGNN Φ, denote $\mathcal{P} := \{(X, A, \Phi(X, A))|X_i \le K\}_X$, i.e. the set of Node-perturbation-response of Φ when the perturbations are bounded by K. Denote g an algorithm accepting $\mathcal{P}$ as inputs. For any $K > 0$ and g, there exists a Φ such that:*

1) *For the interpretable domain of DBNs, the Transparent Model of Φ exists for all inputs in $\mathcal{P}$.*
2) *g cannot determine the Transparent Model of Φ.*

Proof. We first set k_s and k_z (Table II) to K. We then construct Φ_1^v and Φ_2^v as described from (5) to (10). Denote $\mathcal{P}_1$ and $\mathcal{P}_2$ the sets of Node perturbation-response of Φ_1^v and Φ_2^v,

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

respectively. Note that, from Lemma 1, we have $\mathcal{B}_1^v$ and $\mathcal{B}_2^v$ are the Transparent Models of Φ_1^v and Φ_2^v.

Given a Node-perturbation-response, suppose g returns either $\mathcal{B}_1^v$ or $\mathcal{B}_2^v$ (or equivalents claims on which DBN is fitter). As $\mathcal{P}_1$ is the same as $\mathcal{P}_2$ (Lemma 2), the outputs of g on the 2 perturbation-response sets must be the same. If, for example, $g(\mathcal{P}_1) = \mathcal{B}_1^v$, g cannot determine that $\mathcal{B}_2^v$ is the Transparent Model for Φ_2^v as $g(\mathcal{P}_2) = g(\mathcal{P}_1) = \mathcal{B}_1^v$. Thus, selecting Φ_2^v as Φ proves the Theorem. □

From the proof of Theorem 1, we see that, even though Φ_1^v and Φ_2^v operate on different paths (reflected in the difference between $\mathcal{B}_1^v$ and $\mathcal{B}_2^v$), all explanations produced by methods in $\mathcal{G}_v$ cannot differentiate Φ_1^v and Φ_2^v. Therefore, we can conclude that Node-perturbation methods are not able to reliably identify which paths carry out the model's predictions.

VI. Unidentifiable Proof for Edge-perturbation

This section is about the Unidentifiable Proof for Edge-perturbation class $\mathcal{G}_e$. We show that removing edges from input graphs is not enough to identify all nodes contributing to a max operation conducted by the TGNNs. The intuition is, if the messages are gated by the features, edge perturbation does not reveal the sources of those messages.

The Training Task and the Models. Our proof considers a graph of 3 nodes forming a line. The task (Fig. 1b) is to recognize the maximum positive inputs observed in nodes 2 and 3, and return results at node 1:

$$Y_1^{(t)} = \max\left\{0, X_2^{(t')} \text{ and } X_3^{(t')}, 0 \le t' \le t\right\} \quad (13)$$

The outputs on other nodes are zeros.

We use the same architecture as in Sect. V (Fig. 5) to construct two TGNNs named Φ_1^e and Φ_2^e. The hidden vector of each node has 5 main features, i.e. $\boldsymbol{h}_i = [hr_i, ht_i, hs_i, hz_i, hl_i]$, and 6 additional features just for output purposes, denoted by $\boldsymbol{ha}_i = [hrl_i^+, hrl_i^-, hrt_i^+, hrt_i^-, hlt_i^+, hlt_i^-]$. The only new feature in $\boldsymbol{h}_i$ compared to the previous construction in Node-perturbation is hl_i, which is the lag version of hr_i: $hl_i^{(t,l)} = \text{ReLU}\left(hr_i^{(t,l-1)}\right)$. Φ_1^e and Φ_2^e use the same MSG, AGG, and UPD functions as described from (6) to (9). The difference between Φ_1^e and Φ_2^e is only in node 3: while it sends a message in Φ_1^e (as $hs_3 = 0$), it does not in Φ_2^e (as $hs_3 = k_s$).

Regarding the 6 additional features $\boldsymbol{ha}_i$, they are zeros at initialization. Their updates are:

$$\begin{aligned}
hrl_i^{\pm(t,l)} &= \text{ReLU}\left(\pm\left(a_i^{(t,l)} - hr_i^{(t,l-1)}\right)\right)\\
hrt_i^{\pm(t,l)} &= \text{ReLU}\left(\pm\left(a_i^{(t,l)} - ht_i^{(t,l-1)}\right)\right)\\
hlt_i^{\pm(t,l)} &= \text{ReLU}\left(\pm\left(hr_i^{(t,l-1)} - ht_i^{(t,l-1)}\right)\right)
\end{aligned}$$

Additionally, the READOUT and the prediction are:

$$H_i^{(t)} = \text{ReLU}\left(\tau_i^{(t,l=2)} - hz_i^{(t,l=2)}\right), \quad Y = H^{(t=2)} \quad (14)$$

where $\tau_i^{(t,l)} := 1/3(\mathbf{1}^\top \boldsymbol{ha}_i^{(t,l)} + hr_i^{(t,l)} + hl_i^{(t,l)} + ht_i^{(t,l)})$. The goal of the above setting is to make

$$H_1^{(t)} = \max\left\{X_3^{(t)}, X_2^{(t)}, H_1^{(t-1)}\right\}$$

and $H_i^{(t)} = 0$ for $i \in \{2,3\}$. In other words, it makes ϕ_1^e satisfy the training task (13) as $Y = H^{(t=2)}$.

Since $hs_3 = k_s$ in Φ_2^e, node 3 does not send messages. This makes $hr_1^{(t,l=2)} = 0$ as there is no message coming to node 1 at $l = 2$. This makes the output of Φ_2^e at node 1 equal to:

$$H_1^{(t)} = \max\left\{hl_1^{(t,L)}, H_1^{(t-1)}\right\} = \max\left\{X_2^{(t)}, H_1^{(t-1)}\right\}$$

Therefore, by assigning the output Y to $H^{(t)}$, we make the output of Φ_2^e on node 1 equal:

$$Y_1^{(t)} = \max\left\{0 \text{ and } X_2^{(t')}, 0 \le t' \le t\right\} \quad (15)$$

By comparing (13) to (15), it is clear that Φ_1^e and Φ_2^e are different. However, when $X_2^{(t)} > X_3^{(t)}$, the responses of Φ_1^e and Φ_2^e are the same even when some edges are removed from the input graph. We state that observation below:

Lemma 3. *For the task in Fig 1b, denote $\bar{A}$ the adjacency matrix obtained by either keeping the input adjacency matrix A unchanged or by removing some edges. For any given $\boldsymbol{X}$ such that $X_i^{(t)} \le \min\{k_s, k_z\}$ and $X_2^{(t)} > X_3^{(t)}$, we have $\Phi_1^e(\boldsymbol{X}, \bar{A}) = \Phi_2^e(\boldsymbol{X}, \bar{A})$.*

Proof. We only need to consider the output at node 1 since the outputs of all other nodes are 0 (as $hz_i = k_z$ for $i \in \{2,3\}$. If no edge is removed, from the analysis of the forwarding computation (below (14)), we know that both models return the maximum of $X_2^{(t)}$ at node 1 as $X_2^{(t)} > X_3^{(t)}$. If the edge between node 1 and node 2 is removed, there is no message coming to node 1 and $hr_1^{(t,l=2)}$ in both models will be 0. The remained case is when only the edge between nodes 2 and 3 is removed. In this case, Φ_1^e simply becomes Φ_2^e and their outputs must be the same. We then have the Lemma. □

The Transparent Models and Unidentifiable Proof. As Φ_1^e is different with Φ_1^v only in node 4 and the additional content in the propagating messages, it follows that $\mathcal{B}_1^e$ (Fig. 7) is the Transparent Model of Φ_1^e for $\boldsymbol{X}$ bounded by $\min\{k_s, k_z\}$. We write $\mathcal{B}_1^e = \mathcal{I}(\Phi_1^e)$. Note that even when $X_2^{(t)} > X_3^{(t)}$, $m_{21}^{(t,l=1)}$ is determined by $X_3^{(t)}$. Thus, the edge between $\mathcal{V}_3^t$ and $\mathcal{V}_2^t$ in $\mathcal{B}_1^e$ is necessary. Regarding Φ_2^e, as it is just Φ_1^e with node 3 disconnected, $\mathcal{B}_2^e$ (Fig. 7) is the Transparent Model of Φ_2^e, i.e. $\mathcal{B}_2^e = \mathcal{I}(\Phi_2^e)$. The above arguments combined with Lemma 3

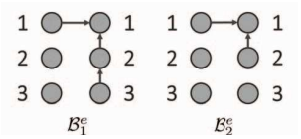


Fig. 7: The DBNs for Unidentifiable Proof of $\mathcal{G}_e$.

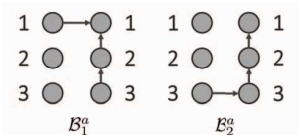


Fig. 8: The DBNs for Unidentifiable Proof of $\mathcal{G}_a$.

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

give us the following Theorem about the Unidentifiable Proof for Edge-perturbation:

Theorem 2. *For a TGNN Φ, denote $\mathcal{P} := \{(X, \bar{A}, \Phi(X, \bar{A}))|X_i \leq K\}_{\bar{A}}$, i.e. the set of Edge-perturbation-response of Φ where $\boldsymbol{X}$ are fixed and bounded by K, and $\bar{A}$ is defined as in Lemma 3. Denote g an algorithm accepting $\mathcal{P}$ as inputs. For any $K > 0$ and g, there exists a Φ satisfying the two conditions in Theorem 1.*

Proof. We first choose k_s and k_z in the node's features to K in the Theorem. We then construct Φ_1^e and Φ_2^e as described above. Denote $\mathcal{P}_1$ and $\mathcal{P}_2$ the sets of Edge-perturbation-response of Φ_1^e and Φ_2^e, respectively. From the discussion of Transparent Models, we have $\mathcal{B}_1^e$ and $\mathcal{B}_2^e$ are the Transparent Models of the two TGNNs.

Given an Edge-perturbation-response, suppose g returns either $\mathcal{B}_1^e$ or $\mathcal{B}_2^e$. Due to Lemma 3, the Edge-perturbation-response $\mathcal{P}_1$ is the same as $\mathcal{P}_2$; therefore, the outputs of g on the 2 perturbation-response sets must be the same. Hence, similar arguments as in the proof of Theorem 1 give us Theorem 2. □

From the proof of Theorem 2, we can see that when the propagating path of TGNN's computations is gated by an intermediate node, the whole path cannot be identified by $\mathcal{G}_e$. This means Edge-perturbation might not be faithful in detecting all graph features contributing to TGNN's predictions.

VII. Unidentifiable Proof for Node-and-Edge Perturbation in TGNN

This section provides the Unidentifiable Proof for the Node-and-Edge-perturbation. The proof shows perturbing both nodes and edges is not sufficient to identify which nodes carry out the temporal aggregation in TGNNs.

The Training Task. The TGNNs operate on a line graph (Fig. 1c). The task is to record the maximum positive inputs observed in node 3 and return the result at node 1:

$$Y_1^{(t)} = \max\{0 \text{ and } X_3^{(t')}, 0 \leq t' \leq t\} \tag{16}$$

The outputs on other nodes are zeros. This proof constructs 2 TGNNs whose internal behaviors are described by the DBNs shown in Fig. 8. The main difference of this proof compared to the previous is that the models involve temporal messages.

The Models. We use the same architecture as in Fig. 5. The hidden feature vectors have 7 features, i.e. $\boldsymbol{h}_i = [hr_i, ht_i, hs_i, hz_i, ho, ho_i^+, ho_i^-]$. Except for the newly introduced ho_i, all features have the same meaning as described in Sect. V. The two constructed models, called Φ_1^a and Φ_2^a, have different MSG functions, READOUT functions, and hidden constant features, i.e. hs_i and ho_i. The constant features for the two models are shown in Table III.

The MSG, AGG, UPD, and READOUT of Φ_1^a are as specified from (6) to (10). Since ho_i in Φ_1^a is just a dummy variable, Φ_1^a satisfies (16) since Φ_1^a is Φ_1^v without node 4.

In Φ_2^a, we use hz_i to control the temporal messages $H^{(t)}$ and the newly introduced ho_i is to control the output Y. The model uses the same AGG and UPD functions as specified from (7) to (9). The update rule of ho_i is $ho_i^{(t,l)} = \text{ReLU}(ho_i^{(t,l-1)})$. The MSG function has the trainable weight $\boldsymbol{w}_m$ such that:

$$m_{ij}^{(t,l)} = \text{ReLU}\left(hr_i + ht_i + ho_i^+ + ho_i^-)/2 - hs_i\right) \tag{17}$$

Here, all variables on the RHS have temporal index t and layer index l. The final difference in Φ_2 compared to Φ_1 is its READOUT as we set $H_i^{(t)} = \text{ReLU}\left(hr_i - hz_i\right)$ and $Y_i = \text{ReLU}\left(hr_i - ho_i\right)$.

TABLE III: The constant features of the TGNNs in G_a's proof.

Node	1			2			3		
Features	hs_1	hz_1	ho_1	hs_2	hz_2	ho_2	hs_3	hz_3	ho_3
TGNN Φ_1^a	k_s	0	0	0	k_z	0	0	k_z	0
TGNN Φ_2^a	k_s	0	0	0	k_z	k_z	k_s	k_z	k_z

Forwarding computation of Φ_2^a. We now show that the forwarding computation of Φ_2^a fulfils (16). We start with the following observations:

- The message m_{ij} (17) is the maximum of hr_i and ht_i (similar to arguments at (11)) when $hs_i = 0$.
- Because of the READOUT (specified below (17)), $H_2^{(2)}$ and ht_2 are always zeros. Thus, m_{2j} is always hr_2.
- Since node 2 only receives messages from node 3 (as $hs_1 = 0$) and the input message, $hr_2^{(t,l)}$ is always the message sent from 3 for l odd. Combining with the above point, $m_{21}^{(t,l=2)}$ is always $m_{32}^{(t,l=1)}$.
- As node 1 only connects to node 2, $hr_1^{(t,l=2)}$ is always $m_{21}^{(t,l=2)}$, which is $m_{32}^{(t,l=1)}$.

From those observations, we have $hr_1^{(t,l=2)}$ is the maximum of $hr_3^{(t,l=0)}$ and ht_3. If ht_3 is always the maximum input in the past and $hr_3^{(t,l=0)}$ is the current input of node 3, then we have the model fulfill the training task (Fig. 1c).

To see that ht_3 is always the maximum input in the past, we refer to Table IV tracking the received message hr_i and the outgoing message m_{i*} for arbitrary inputs $X^{(t=1)} = [\alpha_1, \alpha_2, \alpha_3]$ and $X^{(t=2)} = [\beta_1, \beta_2, \beta_3]$. We first consider the claim for the input's length $T = 2$. For the first time-step $t = 1$, the sending and receiving messages can be deduced in the same manner as in Φ_1^v (Sect. V). For $t = 2$, we have $ht_3 = \alpha_3$ (because only $hz_3 = 0$), which is indeed the maximum signal in the past of node 3. For larger T, we can further examine the Table IV and deduce that claim: at $l = 0$, node 3 sends out $\max\{X_3^{(t)}, H_3^{(t-1)}\}$. At $l = 1$, this message is received at node 2. Finally, this message is sent back to node 3 at $l = 2$.

TABLE IV: Hidden features of the TGNN Φ_2^a for input $X^{(t=1)} = [\alpha_1, \alpha_2, \alpha_3]$ and $X^{(t=2)} = [\beta_1, \beta_2, \beta_3]$. $\gamma_3 = \max\{\alpha_3, \beta_3\}$ and $\gamma_2 = \max\{\gamma_3, \beta_2\}$

Variable		Input	Layer 1	Layer 2
$t = 1$	m_{i*}		$0, \alpha_2, \alpha_3$	$0, \alpha_3, \alpha_2$
$H^{(t)} = [0, 0, 0]$	hr_i	$\alpha_1, \alpha_2, \alpha_3$	$\alpha_2, \alpha_3, \alpha_2$	$\alpha_3, \alpha_2, \alpha_3$
$t = 2$	m_{i*}		$0, \beta_2, \gamma_3$	$0, \gamma_3, \gamma_2$
$H^{(t)} = [0, 0, \alpha_3]$	hr_i	$\beta_1, \beta_2, \beta_3$	$\beta_2, \gamma_3, \beta_2$	$\gamma_3, \gamma_2, \gamma_3$

The Transparent Models of Φ_1^a and Φ_2^a. As Φ_1^a is Φ_1^v without node 4, we have the $\mathcal{B}_1^a$, which is $\mathcal{B}_1^v$ without variables for node 4, is the Transparent Model of Φ_1^a. We write $\mathcal{B}_1^a = \mathcal{I}(\Phi_1^a(\boldsymbol{X}))$ for $\boldsymbol{X}$ bounded by $\min\{k_s, k_z\}$.

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

On the other hand, the transparent model of Φ_2^a can be shown to be the DBN $\mathcal{B}_2^a$ in Fig. 8 by the following Lemma:

Lemma 4. *The DBN $\mathcal{B}_2^a$ in Fig. 8 can embed all information of the hidden features of TGNN Φ_2^a without any loss when the input signal is bounded by $K := \min\{k_s, k_z\}$. Furthermore, the DBN is minimal.*

Proof. The proof has the same structure as in Lemma 1, in which we show the DBN can express the predictions and the messages. For the prediction, in the forwarding computation, we have shown $hr_1^{(t,l=2)}$ is the maximum of $hr_3^{(t,l=0)}$ and ht_3. Thus, the path $\mathcal{V}_3^{t-1} - \mathcal{V}_3^t - \mathcal{V}_2^t - \mathcal{V}_1^t$ is sufficient to express the prediction. For the messages, since only node 2 and 3 are sending out messages, we only need the edge $\mathcal{V}_3^t - \mathcal{V}_2^t$ and edge $\mathcal{V}_2^t - \mathcal{V}_1^t$ to represent them. We can also track the signal via Table IV to verify this.

It is easy to verify that the DBN $\mathcal{B}_2^a$ is indeed minimal: simply from the fact that the prediction at node 1 depends on the past signal ht_3, which means a path between $\mathcal{V}_3^{t-1}$ and $\mathcal{V}_1^t$ must be maintained. Thus, we cannot remove any edges from $\mathcal{B}_2^a$ while keeping it consistent with Φ_2^a. □

As $\mathcal{B}_1^a$ and $\mathcal{B}_2^a$ contain different information, e.g. different set of independent variables, Lemma 4 allows us to claim $\mathcal{I}(\Phi_1^a(\boldsymbol{X})) \neq \mathcal{I}(\Phi_2^a(\boldsymbol{X}))$ for some $\boldsymbol{X}$ bounded by K.

Unidentifiable Proof. Similar to previous proofs, we show that, for all $\boldsymbol{X}$ bounded by K and a valid adjacency matrix, the outputs of the two constructed models are the same, which is stated in the following Lemma:

Lemma 5. *For the training task in Fig 1c, denote $\bar{A}$ the adjacency matrix obtained by either keeping the input adjacency matrix A unchanged or by removing some edges from A. For all $\boldsymbol{X}$ such that $X_i^{(t)} \leq \min\{k_s, k_z\}$, we have $\Phi_1^a(\boldsymbol{X}, \bar{A}) = \Phi_2^a(\boldsymbol{X}, \bar{A})$.*

Proof. First, if A is fixed, from the forwarding computation, we know that both models satisfy (16) for all $\boldsymbol{X}$ bounded by $\min\{k_s, k_z\}$. Thus, we have the Lemma for A fixed.

If the edge between nodes 1 and 2 is removed, there is no message coming to node 1 and the models' outputs will always be zeros. The remained case is only the edge between nodes 2 and 3 is removed. In that situation, at $l = 1$, there is no incoming message to node 2 (because node 1 does not send and node 3 is disconnected) and $hr_2^{(t,l=1)} = 0$. This means at $l = 2$, there is no incoming message to node 1 because $m_{21}^{(t,l=2)} = hr_2^{(t,l=1)} = 0$. As a result, the models' outputs will also always be zeros. We then have the Lemma. □

We are now ready to state the Unidentifiable Proof for the class of Node-and-Edge-perturbation:

Theorem 3. *For a TGNN Φ, denote $\mathcal{P} := \{(X, \bar{A}, \Phi(X, \bar{A}))|X_i \leq K\}_{X \in \mathcal{X}, \bar{A}}$, i.e. the set of Node-and-Edge-perturbation-response of Φ where $\boldsymbol{X}$ are fixed and bounded by K, and $\bar{A}$ is defined as in Lemma 5. Denote g an algorithm accepting $\mathcal{P}$ as inputs. For any $K > 0$ and g, there exists a Φ satisfying the two conditions in Theorem 1.*

Proof. We first choose k_s and k_z in Table III to K in the Theorem. We then construct Φ_1^a and Φ_2^a as described in Sect. VII. Denote $\mathcal{P}_1$ and $\mathcal{P}_2$ the sets of Node-and-Edge-perturbation-response of Φ_1^a and Φ_2^a, respectively. Note that, from the discussion of Transparent Models in that section, we have $\mathcal{B}_1^a$ and $\mathcal{B}_2^a$ are the Transparent Models the two models.

Given an Node-and-Edge-perturbation-response, suppose g returns either $\mathcal{B}_1^a$ or $\mathcal{B}_2^a$. From Lemma 5, $\mathcal{P}_1$ is the same as $\mathcal{P}_2$; therefore, the outputs of g on the 2 perturbation-response sets must be the same. Following similar arguments as in the proof of Theorem 1, we have Theorem 3. □

As in previous proofs, the proof of Theorem 3 also shows Node-and-Edge-perturbation cannot differentiate Φ_1^a with Φ_2^a, whose Transparent Models are two DBNs with different temporal information. Therefore, we can conclude Node-and-Edge-perturbation cannot identify the model's components conducting the temporal messaging and aggregations.

VIII. Experiments

This section reports our experiments demonstrating the impact of unidentifiable information in the explanation tasks for GNNs and TGNNs, in both synthetic and real-world settings. In particular, our experiments aim to elaborate on the failures of Node-perturbation in identifying the correct propagating paths (Sect. V) and Node-and-Edge-perturbation in capturing the temporal information carried out by the models (Sect. VII). Our code is anonymously available at [25].

Synthetic Experiment on Node-perturbation. To demonstrate the unidentifiable result of Node-perturbation, we construct a synthetic node-classification task as shown in Fig. 9. The input is a 6-node-circle graph with one activated node. The task is to transmit that activation to the 2-hop away nodes from those activated nodes using 2 graph layers. Due to the restriction of 2 graph layers, the ground-truth explanation must be as indicated in Fig. 9 **(iv)**.

TABLE V: GNNExplainer fails to detect the ground-truth explanations of the synthetic task (Fig. 9) when using Node-perturbation (Node) and Node-and-Edge perturbation (All).

Method	Return Edge		Return Node	
	TPR	FPR	TPR	FPR
Naive Edge	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
GNNExplainer (Node)	1.00 ± 0.00	0.50 ± 0.00	0.61 ± 0.20	0.39 ± 0.20
GNNExplainer (All)	0.50 ± 0.18	0.25 ± 0.09	0.58 ± 0.20	0.42 ± 0.20

Node-perturbation methods are expected to fail since they cannot differentiate the contributions of the two 1-hop neighbors of the target node. We consider the state-of-the-art explanation method, the GNNExplainer [10], to elaborate that claim. Table V reports the True-positive-rate (TPR) and False-positive-rate (FPR) of the method in this synthetic task. The result not only supports our claim in the case of Node-perturbation but also in Node-and-Edge perturbation. Interestingly, a naive greedy Edge-perturbation method in which the edges are selected based on their sensitivity to the predictions can match the ground truth perfectly. These results emphasize clearly the importance of the choice of perturbations for explanations.

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

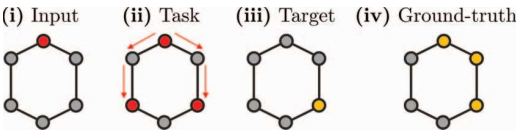


Fig. 9: The synthetic static task demonstrating the failure of node-perturbation explanation methods.

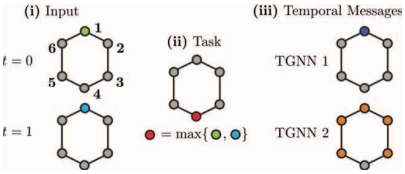


Fig. 10: The synthetic temporal task demonstrating the failure of node-and-edge perturbation-based explanation methods.

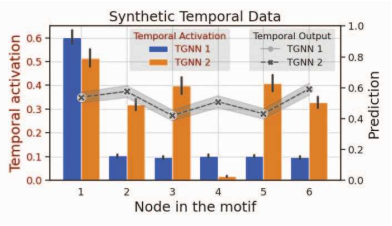


Fig. 11: The temporal activations of TGNNs in synthetic temporal task demonstrating the failure of Node-and-edge perturbation-based explanation methods.

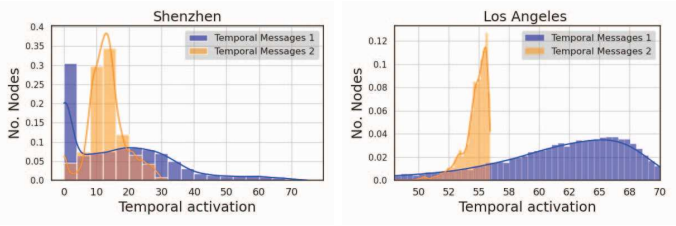


Fig. 12: The real-world temporal messages demonstrating the failure of Node-and-edge perturbation-based explanation methods. Note that the outputs of the two TGNNs in both experiments are identical regardless of the perturbations.

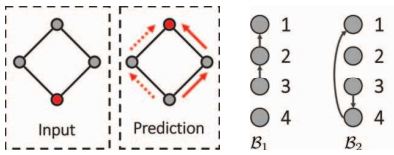
Experiments on Node-and-Edge-perturbation. We now describe our synthetic experiment showing the issue of explaining TGNNs' temporal dynamic using the conventional Node-and-Edge perturbation. The learning task is the prediction of the maximum signals observed in the opposite node in a 6-node-circle motif as depicted in Fig. 10. Two distinct 2-time-step TGNNs are obtained by the combinations of some separately trained neural layers so that their outputs are always identical on the same inputs. Furthermore, the constructions also make the two models have very different temporal dynamics: while the first model transmits temporal information via the source node, the other relies on the whole graph for temporal transmission (Fig. 11). The existence of those 2 models implies that the unidentifiable situations pointed out in our theoretical analysis can occur as a result of the neural network's training process.

We further strengthen that claim with our experiments on two real-world Shenzhen and Los Angeles traffic datasets [7]. The training task is to predict the traffic speed using past data. For each dataset, we use the same construction and training strategy as in the synthetic experiment to obtain 2 instances of TGNNs whose temporal activations are shown in Fig. 12. Since the outputs of the two TGNNs are also identical, we do not report the predictions in these cases. It can be observed from both datasets that the temporal dynamics learned by TGNNs can be varied not only in terms of the number of nodes contributing to the temporal messaging but also in the range and meaning of activation values. The results further point out that faithful explanations for temporal models cannot be obtained by simply perturbing node and edge features of the input graph.

IX. Discussion and Conclusion

This work studies the fundamental limit of different perturbation explanation methods in explaining black-box TGNNs. We have shown that there is key information on how the TGNNs generate their predictions that cannot be identified by some given classes of explanation methods. We now further point out several interesting implications of our theoretical results.

Theorem 1 and 2 for GNNs. The Unidentifiable Proofs for Node-perturbation and Edge-perturbation explanation methods can be applied directly to GNNs by dropping the feedback loop of $H^{(t)}$ (Fig. 5). This modification will just set $H_i^{(t-1)}$ in (5) to zeros. Note that in GNNs, we do not have the temporal dimension in the interpretable domains, i.e. they are BNs instead of DBNs. As illustrations, Fig. 13a and 14 show the components of the proofs for Node-perturbation and Edge-perturbation in GNNs.



a The training task. b The BNs.

Fig. 13: The components for Node-perturbation Unidentifiable Proof of GNNs.

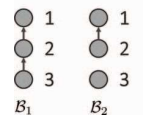


Fig. 14: The BNs for Edge-perturbation Unidentifiable Proof of GNNs.

Theorem 3 for GNNs? Our proof cannot readily apply to the case of GNNs because the two constructions will have the same Transparent Model, i.e. $\mathcal{I}(\Phi_1^a) = \mathcal{I}(\Phi_2^a)$.

What practical models are applicable to our Unidentifiable Proofs? As our analysis uses the most basic constructions of the TGNNs, our Unidentifiable Proofs in Sects. V, VI and VII are applicable to all versions of the TGNNs found in [7], [9], [26]. As the base GNN can be considered as TGNN with zero temporal feedback (see Fig. 5), Our results in Sects. V and VI are also applicable to many modern variants of GNNs including GCN [4], GraphSage [5] and GAT [6].

Usage of other interpretable domains and Transparent Models (not DBNs). Unidentifiable results can be obtained by other interpretable domains as long as (i) the Transparent

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.

Models of the constructed models can be identified (similar to Lemma 1 and 4) and (ii) they contain meaningful information that helps establish the unidentifiable information. While condition (i) requires the domain to have strong expressive power, condition (ii) requires the domain's members to be somewhat interpretable. We find DBN is a balanced choice for the analysis of TGNN.

What do Theorem 1 and 2 imply about the reliability of existing explanation methods for GNNs? Existing explanation methods have been successfully identifying many important features contributing to the predictions; however, the results are still limited. Our results establish a fundamental limit of perturbation-based explanation methods.

For example, Theorem 1 implies explanations obtained by only perturbing nodes cannot reliably inform us of the paths determining the predictions. For the case of the two constructed Φ_1^v and Φ_2^v, the contributions of node 2 and node 4 will always be considered equal by all Node-perturbation methods. This means both will be included or discarded by the explainers, even when the actual messages are only transmitted through one of them. Thus, Node-perturbation methods are bound to commit false positives or false negatives. This claim is further supported by our experiments in Fig. 9.

What does Theorem 3 imply about the design of explanation methods for TGNNs? Even though the Theorem states that the Node-and-Edge perturbation methods cannot identify the temporal component of the model, it does not mean there is nothing we can do to tackle this challenging problem. Careful readers might realize that one key aspect of our proof is based on the fact that removing an edge in the input graph will disconnect that connection at all rounds of temporal computations. If there is a mechanism to remove edge only at some temporal computations, it is possible to differentiate Φ_1^a from Φ_2^a, which is crucial to identify whether node 1 or node 3 conducts the temporal aggregation. In other words, *temporal perturbation* might be something we need to explain TGNNs more faithfully.

REFERENCES

[1] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 6412–6422.

[2] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[3] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, p. 457–466, 2018.

[4] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[7] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.

[8] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu, "Traffic flow prediction via spatial temporal graph neural network," in *Proceedings of The Web Conference 2020*, 2020, pp. 1082–1092.

[9] S. Min, Z. Gao, J. Peng, L. Wang, K. Qin, and B. Fang, "Stgsn—a spatial–temporal graph neural network framework for time-evolving social networks," *Knowledge-Based Systems*, vol. 214, p. 106746, 2021.

[10] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf

[11] A. Duval and F. Malliaros, "Graphsvx: Shapley value explanations for graph neural networks," in *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2021.

[12] M. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 12 225–12 235.

[13] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *CoRR*, vol. abs/2001.06216, 2020. [Online]. Available: https://arxiv.org/abs/2001.06216

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, p. 1135–1144.

[15] B. Sanchez-Lengeling, J. Wei, B. Lee, E. Reif, P. Wang, W. Qian, K. McCloskey, L. Colwell, and A. Wiltschko, "Evaluating attribution for graph neural networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 5898–5910. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/417fbbf2e9d5a28a855a11894b2e795a-Paper.pdf

[16] K. Amara, R. Ying, Z. Zhang, Z. Han, Y. Shan, U. Brandes, S. Schemm, and C. Zhang, "Graphframex: Towards systematic evaluation of explainability methods for graph neural networks," *arXiv preprint arXiv:2206.09677*, 2022.

[17] L. Faber, A. K. Moghaddam, and R. Wattenhofer, "When comparing to ground truth is wrong: On evaluating gnn explanation methods," *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[18] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[19] Y. Zhang, D. DeFazio, and A. Ramesh, "Relex: A model-agnostic relational model explainer," *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021.

[20] T. Funke, M. Khosla, and A. Anand, "Zorro: Valid, sparse, and stable explanations in graph neural networks," *arXiv preprint arXiv:2105.08621*, 2021.

[21] P. Dagum, A. Galper, and E. Horvitz, "Dynamic network models for forecasting," in *Uncertainty in Artificial Intelligence*, D. Dubois, M. P. Wellman, B. D'Ambrosio, and P. Smets, Eds. Morgan Kaufmann, 1992, pp. 41–48. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781483282879500104

[22] J. Pearl, "Chapter 3 - markov and bayesian networks: Two graphical representations of probabilistic knowledge," in *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988, pp. 77 – 141.

[23] K. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, University of California, 01 2002.

[24] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 2020.

[25] Anonymous, *Source Code for this paper*, https://drive.google.com/file/d/11iK8ASQWIk272IgS2e5Fs_WN7DMe8n1n/view?usp=sharing.

[26] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 424–14 432.

Authorized licensed use limited to: University of Florida. Downloaded on August 30,2024 at 02:09:11 UTC from IEEE Xplore. Restrictions apply.