Discovering Counterfactual Temporal Explanations for Temporal Graph Neural Networks

***Abstract*—Temporal graph neural networks (TGNNs) extend graph neural networks for dynamic network analysis, and have shown promising performance. Nevertheless, the explainability of TGNNs remains much less investigated than their GNN counterparts for static networks. This paper introduces TEMporal Graph eXplainer (TemGX), a post-hoc, query-able framework that help users interpret and understand TGNN behavior by discovering temporal subgraphs and their evolution that are responsible for TGNN output of interests. We introduce a two-level explanation structure: instance-level explanations, as a sequence of temporal subgraphs with high temporal explainability, and a global explanation, to capture the temporal dependencies over instance-level counterparts over time. We introduce a class of explainability measures that extends influence maximization and resistant distance to characterize temporal influence. We formulate a bi-level optimization problem, and propose fast algorithms to discover explanations with guarantees on their temporal explainability. Our experimental study verifies the effectiveness and efficiency of TemGX for TGNN explanation, compared with state-of-the-art explainers. We also showcase how TemGX supports inference queries for dynamic network analysis.**

I. Introduction

Temporal graphs provide a powerful abstraction to represent complex dynamic systems, where nodes specify evolving entities, and edges denote temporal relationship. Temporal Graph Neural Networks (TGNNs) [1] model temporal graphs by incorporating temporal features, relations, and their dependencies, making them capable of predicting evolving interactions. TGNNs have demonstrated their effectiveness in various tasks, such as link prediction and node classification, in various applications *e.g.,* traffic analysis, transaction networks.

In a nutshell, a TGNN $\mathcal{M}$ takes as input a temporal graph (a sequence of graph snapshots) $\mathcal{G}_t = \{G^1, \dots G^{t-1}\}$, where each snapshot G^i is represented by a pair (X_i, A_i) of a node feature matrix X_i and an adjacency matrix A_i at time i ($i \in [1, t]$). It is trained to predict the representation (X_t, A_t) of a next graph G^t, which can then be transformed to task-specific output, such as class labels for node classification, features for regression analysis, or new links (events) for link prediction.

Despite their powerful predictive capabilities, TGNNs are often considered lack of interpretability. The decision-making process of TGNNs is deeply embedded within multiple layers of nonlinear transformations and complex temporal mechanisms, which results in opaque predictions [2]. Such "black-box" behavior raises significant concerns in applications requiring transparent, trustworthy, and accountable temporal graph analysis, such as financial transactions or health-related predictions. Consider the following example.

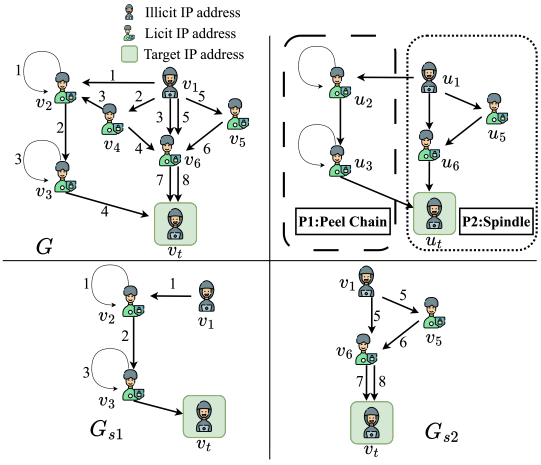


Fig. 1: A temporal bitcoin transaction network G that depicts money laundering activities (nodes: IP addresses; edges: transactions).

Example 1: In Bitcoin blockchain transactions, money launderers employ a variety of techniques to conceal illicit cryptocurrency activities and evade detection by law enforcement agencies and AI-based monitoring systems [3], [4]. As shown in Figure 1, an input temporal graph $\mathcal{G}$ includes account IP addresses and Bitcoin transactions among the accounts at different timestamps. An edge $(v_1, v_2, `2')$ means IP address v_1 committed a transaction to address v_2 at timestamp 2. Each IP address is associated with a set of transaction features, *e.g.,* the number of blockchain transactions, the number of transacted bitcoins, and the amount of fees in bitcoins.

There are two illicit IP addresses: v_1 and v_t. They utilize money laundering patterns known as *Peel Chain* [3] (P_1) and *Spindle* [4] (P_2) to obscure their illegal transactions. The perpetrators take *Spindle* strategy to generate multiple shadow addresses to transfer illicit assets along lengthy paths that converge at a specific destination. Each path typically involves small amounts to avoid detection. *Peel Chain* launders cryptocurrency through a series of small transactions that are 'peeled' from the original address and sent to exchanges for conversion into fiat currency or other assets to minimize risks.

A TGNN-based classifier has detected both accounts as "illicit", at timestamps 4 and 9, respectively. Law enforcement agencies want to uncover evidence to help validate the results. A temporal subgraph such as G_{s_1} is helpful to explain "why" v_t is "illicit" by the TGNN. Indeed, it contains a sequence of historical transactions during a period $[1, 4]$, such that if *not* captured in history, TGNN *fails* to detect v_t as "illicit" (known as counterfactual explanations [5], [6]).

There may also exist many such structures as time goes by,

which remain daunting to inspect. Hence, a "global" model that fits these structures over their distribution is desired (as illustrated at top right in Fig. 1). Such a summary can be conveniently validated by known patterns (such as "Peel Chain"), but also provide insights on how the driving strategies "evolved" over time. For example, the summary can be readily grounded by observing a "Peel Chain" pattern, which traces back to the transaction level facts that: "v_1 *has temporal dependency with* v_t *via a sequence of suspicious transactions*", and "*their temporal dependencies involve into a new pattern during period* $[5, 8]$", indicating a switching of money laundering strategy from "Peel Chain" to "Spindle". □

The above example calls for effective method that can (1) generate "instance-level" explanations that expose activities as temporal subgraphs that are responsible for TGNN outputs in terms of counterfactual analysis; (2) summarize such structures over time as "global" statistical models, and (3) be conveniently queried to understand their evolving interactions. Several methods have been proposed for TGNN explanation *e.g.,* [2], [2]. These methods aim to trace the model's behavior by finding important node features *e.g.,* [2], which may not suffice to capture complex temporal dependencies between nodes and temporal edges; or sampling cohesive motifs following Information Bottleneck principle [7], which may contain irrelevant, non-counterfactual structures for TGNN inferences. How to make the explanations accessible and responsive to users' queries is also not addressed by prior work.

Overview. We introduce an explanatory framework to discovery explanations for TGNN outputs.

(1) We introduce temporal graph explanations (TemGX), a bi-level structure for TGNN output (**Section IV**). A TemGX integrates (a) a set of counterfactual, cohesive temporal subgraphs, to provide "instance-level" interpretation for TGNN output; and (b) a queryable, "global" probabilistic graph model adapting dynamic Bayesian networks (DBN), to summarize the temporal dependencies among the instance-level explanatory subgraphs. We extend counterfactual analysis [5] to a sliding window, to justify historical critical structures as those whose removal alter the output of TGNNs if not observed. The bi-level structure captures critical structures not only at specific time, but also how they evolve over the period.

(2) We propose a class of temporal explanability measures to quantify the quality of TemGX. Our idea extends information cascading models and resistant distance analysis, to capture the temporal influence in terms of temporal topology and node embedding similarity as an adaptive measure more suitable for evolving graphs. Based on the quality measure, we formulate a bi-level optimization problem to compute optimal TemGX for TGNN output. We show that while it is in PTIME to verify the counterfactual properties for temporal subgraphs, the generation of explanations remains NP-hard.

(3) We develop efficient algorithms to discover TemGX for TGNN output (**Section VI**). The algorithm assembles instance-level temporal subgraphs by mining a set of promising explanatory nodes, and then learns an upper-level dynamic Beyasian network (DBN) to best capture their distribution in an interpretable domain via Markov Blanket optimization.

(4) Moreover, we demonstrate how TemGX can be directly queried, to retrieve both instance-level structures to interpret TGNN output of users' interests, and track their changes via temporal probabilistic inference analysis (Section VI-B).

(5) Using real-world datasets, we experimentally verify the efficiency and explanability of TemGX-based TGNN explanation (**Section VII**). We show that TemGX efficiently generates both instance-level and high-level probabilistic explanations that can be justified by real events, and outperform state-of-the-art TGNN explainers in terms of common measures such as fidelity. Our case study further verifies the applications of TemGX for *e.g.,* interpretable event detection.

II. Related Work

Temporal Graph Neural Networks. TGNNs extends graph neural networks by enriching their architectural design, which often integrates graph convolutions with temporal dependency modeling of dynamic patterns over time. By combining these features, TGNNs can effectively capture both spatial dependencies and temporal sequences in a dynamic network, leading to more accurate predictions of future graph states [8]. The temporal inference process operates on a sequence of temporal features to produce a prediction of the expected output at the subsequent times. Notable TGNNs include TGCNs [9], DCRNNs [10], among others [11].

Graph Neural Network Explanation. Explanation methods have been studied for GNNs. *Post-hoc* methods [12]–[16], create a separate model to provide explanations for a GNN. For example, PGExplainer [13] extract important features and substructures that influence GNN predictions. [14] generates a Bayesian network that best fit a set of subgraphs for GNN output. [15] generate motifs as explanations. [16] generates explanatory views that contain graph patterns and matching subgraphs as explanatory structures.

Several methods have been developed to explain TGNNs. [17] extends [14] to learn optimal Bayesian networks over sliding windows. TGNNExplainer [2] pretrains a navigator (a feed-forward neural network) that infers importance scores among events, and an explorer that uses Monte Carlo Tree Search to infer events that minimize a cross entropy loss to target events. TempME [7] samples temporal motifs and learns to infer importance scores at motifs-level to explain targeted events. Events with high information bottleneck scores are then extracted as explanations. This work is among the first to unify counterfactual analysis, instance-level explainability and continuous temporal evolution into an efficient queryable framework for TGNN interpretation.

III. Temporal Graphs and TGNNs

Temporal Graph. A temporal graph $\mathcal{G}$ is a pair $(V, \mathcal{E})$, where V is a finite set of nodes, and $\mathcal{E}$ is a set of temporal edges. Each temporal edge e is a triple (u, v, t) representing a directed interaction from a node u to a node v $(u, v \in V)$ at time t

(indexed with timestamp t). Each node $v \in V$ carries (a) a label $v.L$, and (b) a feature set X_v, where each feature $X_v^t \in \mathbb{R}^d$ is a d-ary feature vector at time t.

We shall use the following notations.

(1) A temporal graph $\mathcal{G} = (V, \mathcal{E})$ can be represented by a discrete-time dynamic graph $\{G_1, \dots G_n\}$ [1], which is a sequence of snapshots. Each snapshot $G_i = (V, \mathcal{E}_i)$ has a node set V, and an edge set $\mathcal{E}^i \subseteq \mathcal{E}$ that contains all the temporal edges at timestamp i in $\mathcal{E}$ ($i \in [1, n]$). The *size* of $\mathcal{G}$ refers to the total number of nodes and temporal edges, *i.e.,* $|V|+|\mathcal{E}|$.

(2) Given a temporal graph $\mathcal{G}$ and a time window W with length t, an temporal graph induced by W is a temporal graph $\mathcal{G}_t = \{G^1, \dots, G^{t-1}\}$ of $t-1$ consecutive snapshots (a length $t-1$ subsequence of the snapshots) from $\mathcal{G}$, where G^j refers to the j-th snapshot in the samples with a counterpart G_i at timestamp i in $\mathcal{G}$ (i may not be the same as j).

(3) Given $\mathcal{G}$, a *temporal path* ρ_δ within *duration* δ between a node v_s to another node v is a time-ordered sequence of temporal edges $\{(v_s, v_1, t_0), \dots (v_i, v_{i+1}, t_i), \dots (v_m, v, t_m)\}$, where $[1 \leq t_0 \leq t_m \leq t]$, and $t_m - t_0 \leq \delta$. We say v_s δ-*reaches* v if such a path ρ_δ exists.

(4) We say a temporal edge (v, v', t') can be "induced" by a node set V_ϵ in $\mathcal{G}_t$, if both $v, v' \in V_\epsilon$, and $t' \leq t$. A temporal graph $\mathcal{G}_\epsilon = (V, \mathcal{E}')$ is a *temporal subgraph* of $\mathcal{G}_t = (V, \mathcal{E})$, if it has the same set of nodes V, and $\mathcal{E}' \subseteq \mathcal{E}$.

Temporal Graph Neural Networks. Following the prior study of TGNN interpretation [1], [2], [8], we consider a TGNN $\mathcal{M}$ as a composition of an encoder function f_E and a decoder function f_D. The encoder f_E takes an input temporal graph $\mathcal{G}_t = \{G^1, \dots G^{t-1}\}$ with $t-1$ snapshots (typically induced by a time window sliding over an original temporal graph $\mathcal{G}$), and obtains Z^{t-1}, where Z_v^i is the embedding of the node $v \in V$ at time i. The decoder f_D takes as input Z^{t-1} and output logits Z^t at a next timestamp.

$$f_E(\mathcal{G}_t) = Z^{t-1}; \quad f_D(Z^{t-1}) = Z^t$$

Here the logits $Z^t(v)$ is the *output embedding* of a node v at time t, and $\mathcal{M}(\mathcal{G}_t, v)$ (resp. $\mathcal{M}(\mathcal{G}_t, e)$) refers to an *output* of $\mathcal{M}$ at node v (resp. edge e). The logits Z^t can be post-processed to task-specific "output" such as a probabilistic matrix or discrete values. For example, $\mathcal{M}(\mathcal{G}_t, v)$ for node regression (resp. node classification) refers to a predicted values (resp. a class label), and $\mathcal{M}(\mathcal{G}_t, e)$ is a binary label for link prediction ("true" if e exists in $\mathcal{G}$ at time t).

Example 2: Revisiting Example 1, the original temporal graph G contains 8 snapshots $\{G_1, \dots G_8\}$. An TGCN $\mathcal{M}$ takes as input $\mathcal{G}_t$ with 4 consecutive snapshots induced by a length-4 time window, *e.g.,* $\mathcal{G}_t = \{G_1, \dots, G_4\}$ and infers the label $\mathcal{M}(\mathcal{G}_t, v_t)$ ='illicit' at time 5. □

IV. Explanatory Structure

As remarked earlier, a desirable explanatory structure shall be (1) responsible for TGNN output in terms of counterfactual analysis, (2) "influential" to TGNN output [16], and (3) be queryable to retrieve temporal dependencies of the explanations over time. We next introduce a temporal structure to address all the three requirements.

A. Instance-level Explanation

Given an input temporal graph $\mathcal{G}_t$=$(V, \mathcal{E})$ of a TGNN $\mathcal{M}$, and an output $\mathcal{M}(\mathcal{G}_t, v)$ to be explained at a target node v,

- a temporal edge $e = (u, v, t)$ is a *counterfactual edge* of $\mathcal{M}(\mathcal{G}_t, v)$, if $\mathcal{M}(\mathcal{G}_t \setminus \{e\}, v) \neq \mathcal{M}(\mathcal{G}_t, v)$; here $\mathcal{G}_t \setminus \{e\}$ is the temporal graph obtained by removing e from $\mathcal{G}_t$;
- a node $v_s \in V$ is an *explanatory node* of $\mathcal{M}(\mathcal{G}_t, v)$, if there exists at least one of its adjacent temporal edge e in $\mathcal{G}_t$ as a counterfactual edge of $\mathcal{M}(\mathcal{G}_t, v)$.

Explanatory temporal subgraphs. Given $\mathcal{M}(\mathcal{G}_t, v)$, and a duration constraint δ, an *explanatory temporal subgraph* $\mathcal{G}_\epsilon = (V_\epsilon, \mathcal{E}_\epsilon)$ of $\mathcal{M}(\mathcal{G}_t, v)$ is a temporal subgraph of $\mathcal{G}_t$ induced by a time window of duration δ, where

- $V_\epsilon = V_s \cup V_c$, where (a) V_s is a set of explanatory nodes of $\mathcal{M}(\mathcal{G}_t, v)$, and (b) V_c is a set of *temporal connection nodes*, such that for each $v_s \in V_s$, v_s δ-reaches v in $\mathcal{G}_\epsilon$ via a set of connection nodes from V_c; here V_s and V_c are not necessarily disjoint, *i.e.,* an explanatory node may also be a connection node;
- $\mathcal{E}_\epsilon$ refers to the temporal edges induced by V_ϵ in $\mathcal{G}_t$.

The *counterfactual set* of explanatory node set V_s that contains all the counterfactual edges of the nodes in V_s.

We justify the above structure by highlighting the following. (1) An explanatory temporal subgraph is "temporally" cohesive, a desirable property [7], [18] that ensures the temporal reachability from any explanatory nodes to the target node v, and (2) retains useful temporal structures that conform to counterfactual analysis to TGNN output. Unlike prior work, we distinguish "explanatory nodes" V_s and "connecting nodes" V_c, based on a natural observation that not all nodes that (temporally) reach v are necessarily "impacting" the decision of TGNNs. Those which may be responsible for TGNN output can be "remote", non-adjacent with v.

Example 3: Recall Example 1 with TGNN $\mathcal{M}$ in Example 2. Given $\mathcal{M}(\mathcal{G}_t, v_t)$ = 'illicit' with input $\mathcal{G}_t = \{G_1, G_2, G_3, G_4\}$, Fig. 2 illustrates a temporal explanatory subgraph (extending G_{s_1} in Fig. 1) of $\mathcal{M}(\mathcal{G}_t, v_t)$ in $\mathcal{G}_t$. We observe the following.

(1) The nodes v_1 and v_2 are verified to be two explanatory nodes, which induces edge set $\mathcal{E}_1 = \{(v_1, v_2, 1), (v_1, v_4, 2)\}$, and $\mathcal{E}_2 = \{(v_2, v_3, 2)\}$ as two counterfactual sets, respectively. The connecting set V_c is $\{v_3, v_4\}$. It tells us that "removing" v_1 and its relevant counterfactual sets $\mathcal{E}_1$, or v_2 with $\mathcal{E}_2$, lead to the change of the label from "illicit" to "licit" if an inference is processed on the remaining temporal graph. This indicates an impact from critical illicit IPs (v_1) and "bridging" IPs (v_2). Removing v_4 or v_3 does not change the output of $\mathcal{M}$ at v_t at timestamp 5, hence they are not explanatory nodes.

(2) Although v_6 is in the induced input graph $\mathcal{G}_t$, it is not an explanatory node, due to that its removal poses no changes to

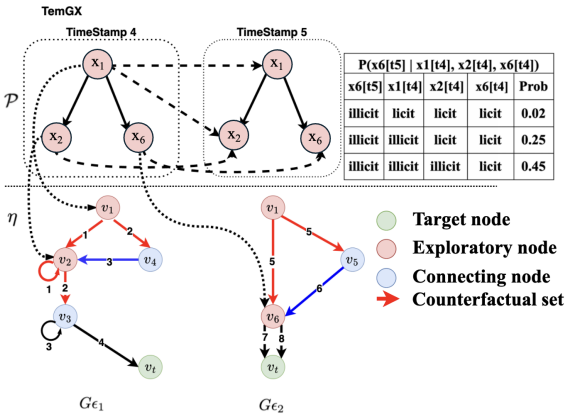


Fig. 2: A TemGX for the output $\mathcal{M}(\mathcal{G}_t, v_t)$ with label 'illicit', with a duration constraint 4. The instance-level explanation η contains $\mathcal{G}_{\epsilon_1}$ and $\mathcal{G}_{\epsilon_2}$, induced from the windows $[1, 4]$, and $[5, 8]$, respectively. Global explanation $\mathcal{P}$ dense dashed lines: the mapping from DBN random variables to instance-level nodes; sparser dashed lines: inter-slice edges; solid lines: intra-slice edges.

the results for the predicted label at timestamp 5. Moreover, although edge $(v_1, v_6, 3)$ is an edge in $\mathcal{G}_t$, it's neither in a counterfactual set induced by the explanatory nodes $\{v_1, v_2\}$, nor induced by $V_s \cup V_c$ as a "connecting" edge to v_t, hence, $(v_1, v_6, 3)$ should not be included in the temporal explanatory subgraph G_{s_1}. Indeed, this transaction did not contribute to delivering the influence of v_1 or v_2's embedding via TGNN inference up to time 4 to its decision at time 5. □

Temporal Explainability. We next quantify the explanability of an explanatory temporal subgraph G_ϵ in terms of its temporal influence to a targeted output $\mathcal{M}(\mathcal{G}_t, v)$, considering both temporal propagation and the node embedding similarity.

(1) Our first formulation for temporal influence is inspired by network influence analysis under the Independent Cascade Model (ICM). ICM captures influence propagation through independent edge activations, where each edge (v, v') has a propagation probability $p(v, v', t)$ at time t. An analogy between influence propagation and TGNNs inference is that both quantify the contribution of a local "message" (node embedding) via message passing following temporal paths.

Given G_ϵ and $\mathcal{M}(\mathcal{G}_t, v)$ at a target node v, for each temporal edge e in G_ϵ, the temporal influence of e (to v) is defined as:

$$p_e = |Pr(\mathcal{G}_t, v) - Pr(\mathcal{G}_t \setminus \{e\}, v)|$$

where $Pr(\mathcal{G}_t, v)$ represents the inferred task-specific probability of the target output given $Z^t(v)$ (output logits) at timestamp i; and $Pr(\mathcal{G}_t \setminus \{e\}, v)$ denotes a counterpart if edge e is removed[1]. The probability Pr can be inferred by a task-specific post-processing (*e.g.,* MLP or sigmoid function) given the output embedding Z^t. The influence of an explanatory node v_s to v is recursively defined as:

$$\mathsf{inf}(v_s, v, t) = 1 - \prod_{\substack{(u,v,t') \in E; \\ t - t' \le \delta; t' \le t}} (1 - p_{(u,v,t')} \cdot \mathsf{inf}(v_s, u, t'))$$

[1]We take the absolute value as edge removal may increase or decrease the output probability.

where $\mathsf{inf}(v_s, v, t)$ accumulates the influence of explanatory node v_s on v recursively over all $\delta-$reachable temporal paths via a connection node u. We set $\mathsf{inf}(v_s, v_s, t) = 1$ to ensure that an explanatory node has full influence to itself.

(2) Given the nature of a TGNN inference that propagates node embeddings via temporal edges to output for decision making, we extend *effective resistance distance*, which has an emerging application in (temporal) graph embedding [19], GNNs layer-wise behavior analysis [20], and model optimization by link rewiring [21]. Given a pair (v_s, v) of an explanatory node v_s and a targeted node v, the temporal resistance distance at time t' is defined as:

$$\mathsf{trd}(v_s, v, t') = (Z^{t'}(v_s) - Z^t(v))^T L^I_{t'} (Z^{t'}(v_s) - Z^t(v))$$

where $Z^{t'}(v_s)$ (resp. $Z^t(v)$) refers to the embedding of v_s at time $t' \le t$ (resp. the output embedding of v at time t). Let $L_{t'}$ be the Laplacian matrix of the snapshot $G_{t'} \in \mathcal{G}_t$ (*i.e.,* $L_{t'} = D_{t'} - A_{t'}$; where $D_{t'}$ and $A_{t'}$ refers to the degree matrix and the adjacency matrix of snapshot $G_{t'}$ with eigenvalues $\lambda_1, \ldots \lambda_n$ and corresponding eigenvectors $u_1, \ldots, u_n$, the Laplacian pseudo inverse $L^I_{t'}$ at time t' is computed as $\sum_{l=2}^{n} \frac{1}{\lambda_l} u_l \; u_l^T$. We adopt efficient algorithms such as [22] to approximate $\mathsf{trd}(v_s, v, t)$ in low polynomial time.

The accumulated *temporal influence* of a set of explanatory nodes V_s over time period $[t', t]$ is consistently defined as:

$$\Phi(V_s, v) = \frac{1}{\delta} \sum_{v_s \in V_s} \sum_{t'=t-\delta+1}^{t} [\mathsf{inf}(v_s, v, t') - h(\mathsf{trd}(v_s, v, t'))]_+$$

where $h : \mathbb{R}_{\ge 0} \to [0, 1]$ is a normalization function that converts the (unbounded) trd to $[0, 1]$ (*e.g.,* $h(x) = x/(x + \alpha)$ with α set to the mean or a high quantile of trd), and $[x]_+ = \max\{0, x\}$ denotes the positive part operator. Consequently, $\Phi(V_s, v) \in [0, 1]$. A larger Φ indicates high influence with low temporal resistance, subject to δ-reachability. Note that the "cohesiveness" does not enforce a temporal path to be consecutive temporal edges.

B. Global-level Explanation

Temporal dependencies arise from the interactions among explanatory nodes and sequences of explanatory temporal subgraphs. Probabilistic graphical models such as dynamic Bayesian networks (DBN) [23] has been verified to be a concise and expressive model to capture temporal dependencies. To summarize and track instance-level explanations over time, we introduce a DBN-based, global explanatory structure.

Explanation structure. Given a set of temporal explanatory subgraphs $\eta = \{\mathcal{G}^1_\epsilon, \ldots \mathcal{G}^n_\epsilon\}$ over time $[1, t-1]$ as an *interpretable domain* of $\mathcal{M}(\mathcal{G}_t, v)$, a *global explanation* $\mathcal{P}$ is a directed acyclic graph (DAG) which contains a set of variables $\mathcal{X}^i$ for each timestamp i ($i \in [1, t]$), where:

(1) Each variable x^i_s in $\mathcal{P}$ corresponds to an explanatory node v^i_s at time i from a subgraph in η, and is associated with a discrete, temporal *state* from a finite state set S via a state encoding function from v^i_s to x^i_s. The state set at timestamp i is denoted as X_i, which may encode node label in node classification, or edge labels for link prediction. Each variable

x_s^i carries a conditional probability table $x_s^i.T$ to bookkeep its temporal dependencies with its neighboring variables.

(2) There are two types of edges: (a) An "intra-slice" edge outlines the conditional dependencies between two variables from a subgraph in η at time i, and (b) a set of "inter-slice" edges, connecting variables across time stamps, with their directions aligned with the progression of time. These edges help track temporal state changes as the variables evolve, capturing the dynamic nature of the temporal subgraphs.

Quality of global explanation. Constructing global explanation $\mathcal{P}$ over a given interpretable domain of temporal explanatory subgraphs can be consistently formulated as learning an optimal dependency structure of a DBN [24]. The learning objective of $\mathcal{P}$ can be formulated as optimizing a unified Bayesian Information Criterion (BIC) that optimizes both intra-slice and inter-slice dependencies. Given η and a DBN $\mathcal{P}$, the BIC score is computed as follows:

$$\psi(\mathcal{P}, \eta.S) = \mathsf{LL}(\eta.S|\mathcal{P}) - \frac{|\mathcal{P}|}{2} \log |\eta.S|$$

where $\mathsf{LL}(\eta.S|\mathcal{P})$ is the log-likelihood of the states $\eta.S$ given η and the learned structure of $\mathcal{P}$, $|\mathcal{P}|$ is the number of the free parameters in the learned structure of $\mathcal{P}$, and $|\eta.S|$ is the number of observed states associated to the nodes in η. Given η, TemGX learns a global explanation $\mathcal{P}$ with a state encoding and structure such that ψ is optimized.

TemGX: Bi-level Structure. We next introduce our temporal explanation structure. Given an output $\mathcal{M}(\mathcal{G}_T, v)$ to be explained, and a user-defined duration bound δ, a *temporal graph explanation*, denoted as TemGX, is a bi-level structure $\mathcal{T}_\epsilon = (\mathcal{P}, \eta)$, where (1) η is a set of instance-level temporal explanatory subgraphs, and (2) $\mathcal{P}$ is a set of DBNs that best fits the temporal distribution of η, as a "global-level" explanation.

Example 4: Continuing with Example 3, we illustrate a TemGX $\mathcal{T}_\epsilon = (\eta, \mathcal{P})$ for $\mathcal{M}(\mathcal{G}_t, v_t)$ at timestamp 9, with duration constraint 4 (Fig. 2). As a sliding window of length 4 over $\mathcal{G}$, we observe the following. (1) η contains two instance-level explanatory temporal subgraphs $\mathcal{G}_{\epsilon_1}$ and $\mathcal{G}_{\epsilon_2}$, induced by time window at $[1, 4]$ and $[5, 8]$, respectively, and involves explanatory nodes v_1, v_2, and v_6. (2) $\mathcal{P}$ is a global-level DBN explanation that outlines the temporal dependencies among the three explanatory nodes. For example, v_1 and v_2 are temporally correlated only at time period $[1, 4]$ but not at period $[5, 8]$, yet new temporal dependencies occur among explanatory nodes v_1 and v_6, indicating a change of strategy, as justified by the money laundry patterns in Example 1. □

V. Explanation Generation for TGNNs

Based on the explanability measures, we formulate TGNN explanation problem as a bilevel optimization problem. Given a set of targeted nodes V_T, with target outputs $\mathcal{M}(\mathcal{G}_t, v)$ $(v \in V_T)$, and a duration bound δ, we aim to find an optimal TemGX $\mathcal{T}^* = (\mathcal{P}^*, \eta^*)$, for each node $v \in V_T$, such that

$$\mathcal{P}^* = \arg\max_{\mathcal{P}} \psi(\mathcal{P}, \eta^*.S)$$

$$\text{subject to: } V_s^* = \arg\max_{|V_s| \le k} \sum_{v \in V_T} \sum_{v_s \in V_s} \Phi(V_s, v);$$

where v ranges over V_T, η^* is the interpretable domain induced by the explanatory nodes $V_s^* = \cup_{v \in V_T} V_s^*$, and V_s^* is the set of all the explanatory nodes for a target output $\mathcal{M}(\mathcal{G}_t, v)$. In addition, a size bound k constraints the total number of explanatory nodes, to allow users specify concise, size-bounded instance-level explanations.

The above problem aims to compute instance-level explanations by solving a "low-level" optimization problem that selects a set of explanatory nodes $\mathcal{V}_s^*$ with maximized importance score to the nodes in V_T. Treating the output embeddings of $\mathcal{V}_s^*$ as a space of "state values" the variables in $\mathcal{P}$ can take, it then aims to learn an optimal dynamic Bayesian network $\mathcal{P}$, subject to user-specified duration constraint δ.

We next present the hardness analysis of the problem. We start with a *verification* problem as a "building block" task for explanation discovery. Given an output $\mathcal{M}(\mathcal{G}_t, v)$ to be explained, a duration bound δ, and a set of nodes V_s, it is to verify if V_s induces an explanatory temporal subgraph with temporal explanability above a given threshold. Successful verification of V_s indicates the existence of instance-level explanations and a proper interpretable domain η.

Lemma 1: *The verification problem is in* PTIME. □

We prove the above result by constructing a PTIME procedure (Verify). It computes influence scores of a given temporal subgraph G_s to be verified, and invokes TGNN inference process to verify if its counterfactual set is empty.

Despite the tractability of verification, the overall explanation discovery problem is nontrivial.

Lemma 2: TGNN *explanation problem is* NP*-hard. In particular, (1) the low-level problem is* NP*-hard alone, and (2) the upper-level optimization remains* NP*-hard, even when the input sets V_s and η are fixed.* □

The hardness of the low-level optimization can be verified by a reduction from the maximum coverage problem (MCP), a known NP-hard problem. The hardness of the upper-level optimization is due to the inherent difficulty of learning an optimal Bayesian network from sampled data, which is known to be NP-hard [24]. We present the detailed proof in [25].

Despite the hardness, we next introduce efficient algorithms to generate TemGX with guarantees in temporal explanability.

VI. Discovering Temporal Graph Explanations

A. Discovery Algorithms

We start with a setting for a single output $\mathcal{M}(\mathcal{G}_t, v)$. Our algorithm, simply denoted as TemGX, follows a two-phase process. (1) It first invokes a greedy-based node selection procedure genInstanceX to discover a set of exploratory $\mathcal{V}_s$ for the lower-level optimization problem. This process induces a set of instance-level verified explanatory temporal subgraphs as an interpretable domain η. (2) It consistently constructs variables for the global explanation (the "nodes" in a DBN), and perform an incremental independent tests between pairs

Procedure genInstanceX $(\mathcal{G}_t, \mathcal{M}, \mathcal{M}(\mathcal{G}_t, v), k)$

1. set $\eta := \emptyset$; set $\mathcal{C} := \emptyset$;
2. **for each** window W of size δ over $\mathcal{G}_t$ **do**
3. set $V_s := \{v\}$; set $\mathcal{C} := \emptyset$;
4. induce a temporal graph G_W with current window W;
5. **for each** time step t_i in W **do**
6. induce L-hop temporal neighbor set $N_L(v)$ in G_W;
7. **for each** $v' \in N_L(v)$**do**
8. update $v'.\Phi := \Phi(v', v)$; $\mathcal{C} := \mathcal{C} \cup \{v_s\}$;

/ select and verify explanatory nodes */*

9. **while** $\mathcal{C} \neq \emptyset$ **do**
10. select node $v_s^* = \arg\max_{v_s \in \mathcal{C}} \Phi(v_s, v)$; updatePhi($\mathcal{C}$);
11. **if** Verify$(\mathcal{M}, \mathcal{M}(\mathcal{G}_t, v), V_s, v)$ = false **then**
12. $\mathcal{C} := \mathcal{C} \setminus \{v_s^*\}$; **continue**;
13. **if** $|V_s| < k$ **then**
14. $V_s := V_s \cup \{v\}$; $\mathcal{C}^t := \mathcal{C}^t \setminus \{v_s\}$; **break**;
15. **else** set Rset := V_s; */* replacement policy */*
16. **while** Rset $\neq \emptyset$ **do**
17. select node $v_s^{*'}$ as $\arg\max_{v_s' \in \text{Rset}} \Phi(V_s \setminus \{v_s^{*'}\} \cup \{v_s^*\})$;
18. **if** Verify$(\mathcal{M}, \mathcal{M}(\mathcal{G}_t, v), V_s \setminus \{v_s^{*'}\} \cup \{v_s^*\}, v)$ = false
19. **then** Rset:=Rset $\setminus \{v_s^{*'}\}$; **continue**;
20. **else** $V_s := V_s \setminus \{v_s^{*'}\} \cup \{v_s^*\}$; $\mathcal{C} := \mathcal{C} \setminus \{v_s\}$; **break**;

/ construct a temporal explanatory subgraph */*

21. induces $\mathcal{G}_\epsilon$ with V_s, (V_c (connect nodes) and $\mathcal{E}_\epsilon$;
22. $\eta := \eta \cup \{\mathcal{G}_\epsilon\}$;
23. **return** η;

Fig. 3: Procedure genInstanceX

of variable nodes to initialize states and structures. It invokes a learning procedure genGlobalX to refine the edges in the global explanation $\mathcal{P}$ by optimizing the BIC score, given η.

Procedure genInstanceX **(Fig. 3):** works with a sliding window of length δ over $\mathcal{G}_t$, to generate a set of instance-level explanations η. For each window W, it generates a temporal explanatory subgraph with at most k explanatory nodes.

(1) *Initialization* (lines 3-8). Procedure genInstanceX maintains a set V_s to store selected explanatory nodes, and a set $\mathcal{C}$ of potential nodes that may serve as explanatory nodes. It induces a temporal subgraph G_W with the current time window W (line 4). It also sets a set $\mathcal{C}$ to be the nodes from the L-hop temporal neighbors of v, denoted as $N_L(v)$, which includes all the temporal nodes and edges that are δ-reachable to v, for a TGNN $\mathcal{M}$ with L spatial-temporal layers. This step exploits the spatiotemporal data locality to necessarily constrain the search space to the nodes that may participate in TGNN inference, under duration constraint δ. For all such nodes, the influence scores are computed (lines 7-8).

(2) *Explanatory node discovery* (lines 9-20). genInstanceX then adopts a greedy selection and replacement policy to maintain V_s by a "one-pass" processing of $\mathcal{C}^t$. When $|V_s|$ is smaller than k, it fills the set V_s with the best node v_s^* that (1) maximizes the temporal influence scores, and (2) passes the verification of counterfactual property. The verification is determined by invoking a procedure Verifier, a PTIME process following the proof of Lemma 1 (line 11). For $|V_s| = k$, it switches to a greedy-based replacement strategy, that favors to replace a node $v_s^{*'}$ with v_s^* that (1) maximizes the overall influence $\Phi(V_s \setminus \{v_s^{*'}\} \cup \{v_s^*\})$, and (2) with adjacent edges that is a counterfactual set (determined by Verifier) (line 17). Whenever a node is processed or verified, it is removed from $\mathcal{C}$. This process repeats until $\mathcal{C}$ is completely verified ($\mathcal{C} = \emptyset$).

(3) *Explanation construction*. Once a set V_s is selected, a temporal explanatory subgraph G_ϵ is constructed as follows. (a) It induces, for each node $v_s \in V_s$ and the node v, a connecting set $V_c(v_s)$ that consists of all the nodes in G_W that is on a path via which v_s δ-reaches v. It then sets $V_\epsilon = V_s \cup V_c$, where $V_c = \bigcup_{v_s \in V_s} V_c(v_s)$. (b) It then induces connection nodes V_c with V_s, and accordingly, induces $\mathcal{E}_\epsilon$ with $V_s \cup V_c$, to yield the complete structure of a temporal explanatory subgraph G_ϵ. It adds G_ϵ to η, and continues to process the next window. Once V_s are selected, TemGX deterministically induces the connection nodes and counterfactual set by enforcing δ-reachability. The overall process terminates when the sliding window consumes all the snapshots in $\mathcal{G}_t$.

Explanability Guarantees. Denote the the optimal set of explanatory nodes as V_s^O, we present the following result.

Lemma 3: *Procedure* genInstanceX *is a* $(1 - \frac{1}{e})$*-approximation for the low-level optimization,* i.e., $\Phi(V_s, v) \geq (1 - \frac{1}{e})\Phi(V_s^O, v)$. □

Proof sketch: We show this by (1) formulating an equivalent monotone submodular maximization problem with a cardinality constraint, and (2) proving that genInstanceX is a greedy algorithm that ensures a $(1-\frac{1}{e})$-approximation for the problem in (1) [26]. We present the detailed proofs in [25]. □ .

Procedure genGlobalX. For each explanatory node $v_s \in V_s$ at timestamp i, genGlobalX extracts feature embeddings from temporal snapshots and discretizes the values using uniform binning to ensure a finite state representation. This yields a structured discrete state set $\eta.S$ to encode the temporal distribution of V_s in a probabilistic space. genGlobalX then runs (procedure structureLearn) to learn a set of intra-slice edges and inter-slice edges, for V_s and v, using BIC score evaluated on aggregated data from all consecutive timestamp pairs $[i, i+1]$ for $i \in [1, t–2]$. To this end, it invokes fast heuristics [27] that ensures a competitive local optima. It finally performs a Maximum Likelihood Estimation to optimizes the conditional probability distributions of the state nodes, to ensure $\mathcal{P}$ accurately fits the observed η.

Time cost. For instance-level explanation generation, it takes genInstanceX at most t rounds of node selection, each in $O(|N_L(v)|T_I)$ time to compute influence score, where T_I is the polynomial-time inference cost of the TGNN $\mathcal{M}$. The global-level generation takes a (output-sensitive) cost in $O\big(t \cdot k^2 \cdot m \cdot r\big)$, where m is the number of observed states per node, and r denotes the maximum iteration of Hill Climbing. Hence, the total time cost is in $O(|N_L(v)|T_I + t \cdot k^2 \cdot m \cdot r)$.

TemGX can be readily extended to explain multiple outputs, where the quality guarantee remains intact. We present the details, with time cost and quality analysis in [25].

B. Inference Analysis of TemGX

We next show how TemGX support queryable explanation at both instance-level and global level.

Instance-level Search. TemGX provides a natural query access point with semantics of temporal pattern matching [7], [15]. A temporal pattern $Q_\eta = (V_Q, E_Q, \sigma, \delta')$ specifies a set of query nodes V_Q and query edges E_Q. Each query node $v_q \in V_Q$ may carry a label $v_q.L$. Here σ is a temporal order posed on E_Q. Given a set of temporal explanatory subgraphs η with a node set V_η, and a query $Q_\eta = (V_Q, E_Q, \sigma, \delta')$, there exists a *match* of Q in η, if there is a mapping $h : V_Q \to V_\eta$, such that for each edge $e_q^i = (u_i, u_{i+1})$ in Q, (1) $h(u_i) = v_i$, $u_i.L = v_i.L$; (2) $h(u_{i+1}) = v_{i+1}$, $u_{i+1}.L = v_{i+1}.L$; and (3) v_i δ'-reaches v_i'. Such queries can be efficiently processed by temporal pattern matching algorithms *e.g.,* [28], [29].

Global-level Inference. Beyond instance-level search, TemGX naturally supports inference analysis with the learned DBN structure $\mathcal{P}$. One type of useful inference analysis is to request the likelihood of a value of a variable x_u of interests in η at a particular time $t' \in [1, t]$ (where the value may not be seen in $\mathcal{G}_t$), given a confirmed occurrence of the value of another node variable x_u' at another time $t'' \in [1, t]$. This allows efficient "What if" questions. For example, a "What-if" question may ask "*How likely v_t's label becomes 'licit' (indicating that* TGNN *makes a mistake) if the label of v_1 is enforced to be 'licit', given interpretable domain η?*" The inference analysis can be performed by efficient DBN inference process [23], with time cost in $O(|\eta|t)$ time, where $|\eta|$ is the total number of nodes and edges of the temporal subgraphs in η, and t refers to the total period length (see more query examples in [25]).

VII. Experimental Study

We evaluate the explainability, efficiency, and scalability of TemGX. Our code and datasets are available[2].

A. Experimental Setup

Datasets. We use five real-world temporal graphs below (summarized in Table I). (1) METR-LA [10] and PEMS-BAY [10] are transportation networks with traffic speeds in Los Angeles and Bay Areas, respectively, where nodes are locations and edges represent road segments. (2) Wiki [30] includes one month of Wikipedia edits, with nodes as editors and pages and edges as timestamped posting requests. Edge features are 172-dimensional vectors from edited texts. (3) MimicIII [31] is a clinical database containing de-identified health records, where nodes represent patients, medical procedures, and diagnoses, and edges capture temporal medical events and treatments. (4) Multihost [32] is a cybersecurity dataset capturing multi-step attack scenarios, where nodes represent network hosts, processes, and files, and edges denote attack propagation events including process executions and network connections.

TGNNs. We have trained six types of TGNNs. (1) TGCN [9] employs graph convolution to capture spatial dependencies and temporal convolution to model temporal patterns. (2) DCRNN [10] integrates graph diffusion processes into a recurrent neural network, and excels in capturing the spatial dependencies via graph diffusion and temporal dependencies via recurrent units, making it suitable for time-series forecasting. (3) TGN [33] uses memory modules and attention mechanisms to capture the temporal evolution of node states. (4) TGAT [11] combines self-attention to capture temporal dependencies and neighborhood aggregation to learn spatial dependencies. (5) ROLAND [34] employs a recurrent architecture with learnable attention mechanisms for dynamic graph representation learning. (6) WinGNN [35] uses random gradient aggregation windows for dynamic graph neural networks, enabling efficient temporal pattern learning through stochastic window sampling. We trained TGCN, DCRNN, ROLAND, and WinGNN for METR-LA, PEMS-BAY, and MimicIII; and TGN and TGAT for Wiki and Multihost, uniformly with 70% of the dataset for training, 15% for validating, and 15% for testing.

TABLE I: Summary of datasets. snapshot: the average snapshot size in (# nodes, # edges); # feature: the average number of features per node/edge; period: the total time t, *i.e.,* number of snapshots in $\mathcal{G}_t$; δ: window size; task: the task a TGNN is trained for.

Dataset	snapshot	#feature	(period,δ)	task
METR-LA	(207,1515)	1	(34272,12)	regression
PEMS-BAY	(325,2369)	1	(52128,12)	regression
Wiki	(9227,157474)	172	(731,14)	prediction
MIMIC-III	(26109,785586)	8	(3500,12)	classification
Multihost	(6457,252457)	8	(66,4)	classification

TGNN Explainers. We have implemented the following. (1) TemGX explainers; (2) TGVex, an extension of GVex [16] by applying it over "static" snapshots induced by a sliding window to generate a sequence of explanatory subgraphs; (3) TGNNExplainer [2], an explorer-navigator framework to explain TGNN-based link (event) inference; (4) TempME [7], a temporal motif-based TGNN explainer that finds events in terms of IB scores. Among these, TempME are learning-based explainers; TGNNExplainer, TemGX use both search and learning; As TGNNExplainer and TempME are designed for event explanation, they generate critical edges as explanations.

Evaluation Metrics. For a fair comparison with [2], [7], we use three common measures below. (1) Fidelity. The fidelity Fid measures the difference between the original and new prediction probabilities. Consider an induced explanatory temporal subgraph $\mathcal{G}_\epsilon$ with temporal edges $\mathcal{E}_\epsilon$, of an explainer. For TGNNExplainer and TempME, $\mathcal{E}_s$ refers to a set of critical events (edges) as explanations. (1) For link prediction, we compare TemGX, TGNNExplainer, and TempME. Follow the convention [2] that measure the difference over the learned logits $Z_\epsilon = f_D(f_E(\mathcal{G}_\epsilon))$, we define $\text{Fid}(\mathcal{G}_\epsilon, V_T) = \mathbb{I}(Y_T = 1)\cdot(Z_\epsilon[V_T] - (Z[V_T]) + \mathbb{I}(Y_T = 0)\cdot(Z_\epsilon[V_T] - Z[V_T])$, where $\mathbb{I}(\cdot)$ is the indicator function, Y_T is the true label of V_T, $Z[V_T]$ and $Z_\epsilon[V_T]$ represent the logits of V_T on input graph $\mathcal{G}_t$ and $\mathcal{G}_\epsilon$, respectively. (2) For node regression tasks, TGNNs output predicted values rather than links, hence with their source codes, TempME and TGNNExplainer are not directly

[2]https://anonymous.4open.science/r/TemGX-9D77/

applicable. We compare TemGX and TGVex, and define fidelity as: Fid($\mathcal{M}(\mathcal{G}_s)[V_T]$)= 1 - $\left(\frac{|\mathcal{M}(\mathcal{G}_s)[V_T]-\mathcal{M}(\mathcal{G}^t)[V_T]|}{|\mathcal{M}(\mathcal{G}^t)[V_T]|}\right)$ where $\mathcal{M}(\mathcal{G}_s)[V_T]$ and $\mathcal{M}(\mathcal{G}^t)[V_T]$ are the model predictions for node V_T on the explanatory temporal subgraph $\mathcal{G}_\epsilon$ and the full graph $\mathcal{G}^t$, respectively. (3) The sparsity assesses the conciseness of the explanations. We normalize the sparsity as $\frac{\mathcal{E}_\epsilon}{|\mathcal{E}^L(V_T)|}$, where $\mathcal{E}^L(V_T)$ refers to the temporal edges within L-hop neighbors of the nodes in targeted nodes V_T in the temporal graph $\mathcal{G}_T$, for L-layered TGNNs. The higher, the better [2], [7]. (4) We evaluate performance using the fidelity-sparsity curve and calculate the Area Under the Fidelity-Sparsity Curve (AUFSC). The higher, the better [2], [7].

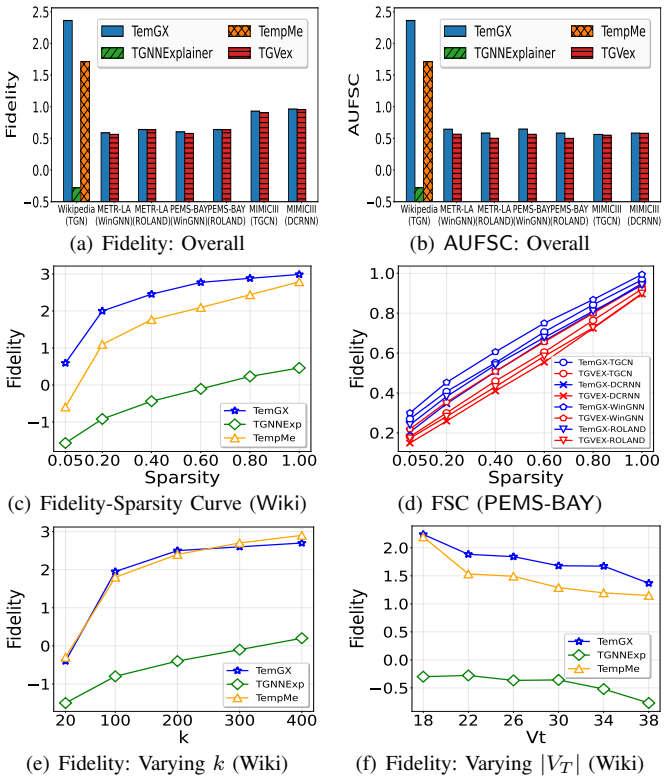
(Hyper)parameter Configuration. Our tests tunes the following parameters: explanatory node budget $k \in \{20, 100, 200, 300, 400\}$; duration constraint $\delta \in \{4, 12, 14\}$ based on temporal data characteristics; L-hop neighborhood, with $L = 2$ by default; DBN state discretization with 10 uniform bins; Hill climbing iterations $r = 100$. Parameters are chosen according to the best performance for fair comparison.

Set up. All methods and tests are implemented in Python 3.10.12 by PyTorch Geometric. Tests are deployed on a HPC cluster with Tesla V100-SXM2-32GB GPU (CUDA ver. 12.2), an NVIDIA driver (ver. 535.54.03), and a dual Intel Xeon Gold 6226 CPU with 24 cores at 3.7 GHz, managed by Slurm. All tests were conducted 5 independent times using different random seeds, and the average results are reported.

B. Experimental Results

Exp-1: Explanability. We first report the fidelity and sparsity of the TGNN explaners over the real-world datasets. *Fidelity: Overall*. We report the fidelity of all the TGNN explainers in Fig. 4(a). For Wiki, we set k = 2000 and $|V_T|$ = 100. We set a total input test data covering a period of 12766 minutes. For METR-LA and PEMS-BAY, we used WinGNN and ROLAND for the node regression, for Wiki, we used TGN for the link prediction, and for MimicIII, we used TGCN and DCRNN for the node classification. (1) TemGX achieves best fidelity among all the explainers due to its temporal counterfactual property; this justifies the choice of temporal influence scores for instance-level explanations, which aligns with the fidelity. For example, it outperforms TempME by 35.3% on Wiki and is 8.67 times better than TGNNExplainer. TemGX outperforms TGVex by 20% on METR-LA (WinGNN), 20% on METR-LA (ROLAND), and maintains similar improvements on PEMS-BAY variants. (2) TGVex is not performing well due to its snapshot-based approach, which ignores δ-reachability and dynamic dependencies that may contribute to TGNN inference. TGNNExplainer is not performing well, in particular on complex datasets like Wiki, as its edge-centric Monte Carlo search cannot effectively identify nodes with accumulated temporal influence. TempME performs better at specifying motif-based explanations, yet we found it mainly missed non-frequent yet influential temporal patterns.

AUFSC: *Overall*. Using the same setting, we report AUFSC in Fig. 4(b). TemGX performs consistently well and achieves



(a) Fidelity: Overall (b) AUFSC: Overall

(c) Fidelity-Sparsity Curve (Wiki) (d) FSC (PEMS-BAY)

(e) Fidelity: Varying k (Wiki) (f) Fidelity: Varying $|V_T|$ (Wiki)

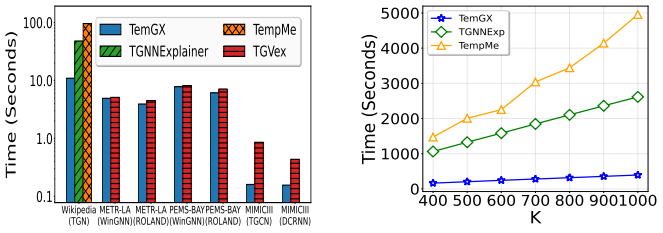
Fig. 4: Explanability: Fidelity & Sparsity

best results for all datasets and models. TGVex is not able to maintain high fidelity, despite that it often reports small structures. This is due to that TGVex is constrained to small connected subgraphs over static snapshots, and falls short at capturing temporally cohesive structures as other explainers.

Fidelity-sparsity curves. Using Wiki with the same setting as in Fig. 4(a), we report the Fidelity-Sparsity curves of TemGX and baseline TGNN explainers in Fig. 4(c). (1) TemGX can achieve high fidelity at low sparsity due to its guarantee on factual and counterfactual explanations. (2) TemGX "converges" to high fidelity faster. This verifies that it can exploit small budget on node selection to effectively induce highly influential structures for multiple outputs.

Performance over TGNNs. Using PEMS-BAY, we report the Fidelity-Sparsity curves of TemGX and TGVex for TGCN, DCRNN, WinGNN, and ROLAND in Fig. 4(d). TemGX outperforms TGVex consistently across all four models. TemGX excels because it captures spatial and temporal dependencies more effectively, aligning with the models' strengths in dynamic graph representations. In contrast, TGVex struggles due to its static snapshot approach, which fails to capture the temporal cohesiveness critical for high fidelity in these models.

Impact of k. Using Wiki, we evaluate the impact of the size constraint k to fidelity on the explanatory temporal subgraphs in terms of its edge size $|\mathcal{E}_s|$. Fixing $|V_T|$=20, we varied k from 20 to 400. As shown in Fig. 4(e), all the explainers can perform better as k increases, due to that more temporal edges can be explored to generate explanations with higher fidelity.



(a) Efficiency: Overall (logscale) (b) Varying k

Fig. 5: Efficiency of TGNN Explainers

Impact of $|V_T|$. To avoid skewed distribution of neighborhood features, we sampled test nodes with similar topological properties such as average degrees. TemGX is more robust to test nodes with different sizes, and consistently outperform other baselines in achieving high fidelity. TempME is more sensitive as V_T grows. This is possibly due to that more test nodes introduce larger variance in embedding distribution and importance scores, which leads to inductive bias in TempME.

Exp-2: Efficiency. We report the cost as the total learning overhead for TempME (as learning-based explainer), the search overhead for TGVex (as search-only explainer), and a total cost of search and learning for TGNNExplainer and TemGX. Here, $|\mathcal{G}_t|$ refers to the number of snapshots.

Overall Efficiency. Using the same setting as in Fig. 4(a), we report the time cost of all the methods in Fig. 5(a). (1) TemGX is feasible: in all cases, it takes no more than 11 seconds to discover explanations. (2) It is 4.39 (resp. 8.79) times faster than TGNNExplainer (resp. TempME) on Wiki, and a 24.7x speedup compared with TGVex on MimicIII. This is attributed to its fast node selection and the learning of size-bounded DBN structure, without exhaustive search. (2) TGVex is comparable with TemGX on sparser networks METR-LA and PEMS-BAY. Such cases favor TGVex as it benefits better from batched processing of sparser snapshots, yet at a cost of lower fidelity (see Fig 4(a)). We found TemGX improves TGVex better over denser networks. (3) TempME and TGNNExplainer incur higher costs on denser networks *e.g.,*Wiki, due to expensive random walks and Monte Carlo Tree Search, respectively.

Varying Impact Factors. We also evaluate the impact of k, $|V_T|$ and $|\mathcal{G}_t|$ (as the number of snapshots). While all algorithms take more time for larger k, $|V_T|$ and $|\mathcal{G}_t|$, TemGX is least sensitive as it benefits better from fast node selection, batched processing of V_T, and spatial-temporal locality of TGNN inference. Other methods are more sensitive to the fast-growing space due to "edges-driven" computation. We report the result of varying k in Fig. 5(b), and more details in [25].

Exp-3: Case Analysis. We next showcase how TemGX supports queryable explanation for real-world TGNN use cases.

Clarifying Suspicious Transactions. Fig. 6(a) (i) shows a fraction of the Bitcoin network around a node 140 (an "illicit" IP detected by a TGN). Fig. 6(a) (ii) illustrates three temporal explanatory subgraphs $\mathcal{G}_{\epsilon_1}$-$\mathcal{G}_{\epsilon_3}$ generated by TemGX for node 140. These explanations are well grounded by patterns (highlighted with blue dashed circles, indicating matches for "Spindle"; or red dashed circles, for matches of "Peel Chain"). For instance, IP addresses 418, 2819, 2411, and 140 indicate a Spindle strategy at the first timestamp. In the next timestamp, 418's role is replaced by a new account 782, which also made use of 2558, as new Spindle matches. Next, a new strategy "Peel Chain" involving 2216 was captured by TemGX. These temporal subgraphs can be readily queried to return suspicious matches of "Peel Chain" or "Spindle" for further investigation.

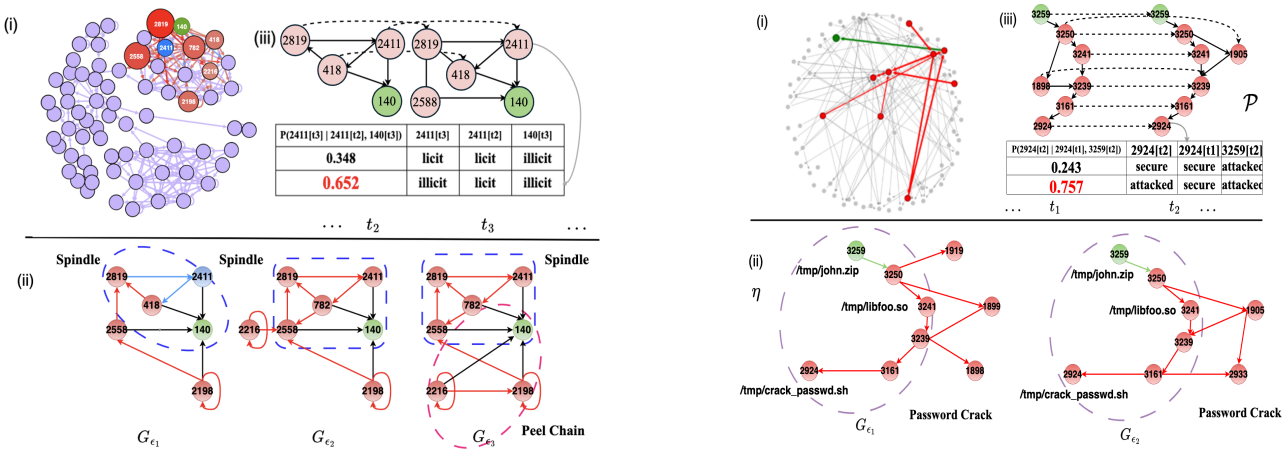
Fig. 6(a) (iii) illustrates a fraction of global explanation $\mathcal{P}$, referring to explanatory IPs and their temporal states over time, which allows analysts to query cross-time dependencies that standard static models would miss. For example, an inference query that asks "*what is the likelihood of an 'licit' account 2411 to be 'illicit'?*" returns that 2411 has a 63.10% chance to remain licit and 36.90% chance to become illicit at time t_2, yet a higher chance 65.20% at time t_3. Indeed, 2411 is under more substantial influence from illicit IP 140, grounded by Spindle and Peel Chain matches in G_{ϵ_3} at instance level.

Multi-stage Cyber Attack Analysis. Fig. 6(b) (i) illustrates a multi-stage cyber attack scenario from the Multihost dataset, where TemGX explains malicious activities across network hosts. The ground-truth attack involves three stages: initial compromise via Shellshock vulnerability (*e.g.,* at host1, executes the script gather_password.sh, identifies host2 and sends malicious script crack_passwd.sh to it), password cracking (*e.g.,* initialized by crack_passwd.sh on host2 by downloading a malicious payload, "libfoo.so"), and data exfiltration (*e.g.,* by using an MD5 hash to extract an embedded file "john.zip".). Two temporal explanatory subgraphs discovered by TemGX are shown in Fig. 6(b) (ii). As verified by security analysts, they are well grounded by correctly covering critical attack paths from host1 to host2 with explanatory nodes and connections involving *e.g.,*gather_password.sh and crack_passwd.sh.

Fig. 6(b) (iii) illustrates a fraction of global explanation $\mathcal{P}$. An inference query that asks "*what is the likelihood of host 2924 being attacked if host 3259 is attacked?*" posed on TemGX returns that 2924 has only a 5% risk when host 3259 is secure, yet when host 3259 is attacked, this probability increases to 32.1% at the second timestamp and further escalates to 75.7% during the lateral movement phase. This demonstrates TemGX's ability to provide grounded explanations for complex multi-step cyber attack analysis.

VIII. Conclusions

We have presented TemGX, a novel framework for TGNN output explanation. We introduced a bi-level structure that integrates instance-level temporal explanation subgraphs and a dynamic Bayesian network-based global explanation, to capture their temporal dependencies over time. We have developed fast algorithms to generate TemGX. Our experimental study has verified that TemGX outperforms state-of-the-art TGNN explainers in explainability and efficiency. A future topic is to optimize TemGX to scale TGNN interpretation over large distributed networks and real-time graph stream analysis.



(a) Bitcoin Transaction Analysis

(b) Multi-stage Attack Detection in Provenance Graphs

Fig. 6: Case Analysis

REFERENCES

[1] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey," *Journal of Machine Learning Research*, vol. 21, no. 70, pp. 1–73, 2020.

[2] W. Xia, M. Lai, C. Shan, Y. Zhang, X. Dai, X. Li, and D. Li, "Explaining temporal graph models through an explorer-navigator framework," in *ICLR*, 2022.

[3] C. Bellei, M. Xu, R. Phillips, T. Robinson, M. Weber, T. Kaler, C. E. Leiserson, Arvind, and J. Chen, "The shape of money laundering: Subgraph representation learning on the blockchain with the elliptic2 dataset," in *KDD Workshop on Machine Learning in Finance*, 2024.

[4] L. Cheng, F. Zhu, Y. Wang, R. Liang, and H. Liu, "Evolve path tracer: Early detection of malicious addresses in cryptocurrency," in *KDD*, 2023.

[5] A. Kuratomi, Z. Lee, P. Tsaparas, G. D. Junior, E. Pitoura, T. Lindgren, and P. Papapetrou, "Counterfair: Group counterfactuals for bias detection, mitigation and subgroup identification," in *ICDM*, 2024.

[6] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, "Cf-gnnexplainer: Counterfactual explanations for graph neural networks," in *AISTATS*, 2022, pp. 4499–4511.

[7] J. Chen and R. Ying, "Tempme: Towards the explainability of temporal graph neural networks via motif discovery," *Advances in Neural Information Processing Systems*, vol. 36, pp. 29 005–29 028, 2023.

[8] A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini, "Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities," *arXiv preprint arXiv:2302.01018*, 2023.

[9] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *IJCAI*. International Joint Conferences on Artificial Intelligence Organization, 2018.

[10] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*, 2018.

[11] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," 2020. [Online]. Available: https://arxiv.org/abs/2002.07962

[12] T. Funke, M. Khosla, M. Rathee, and A. Anand, "Zorro: Valid, Sparse, and Stable Explanations in Graph Neural Networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8687–8698, 2023.

[13] 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, pp. 19 620–19 631, 2020.

[14] M. N. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," 2020.

[15] F. Ding, N. Luo, S. Yu, T. Wang, and F. Xia, "Mega: Explaining graph neural networks with network motifs," in *2023 International Joint Conference on Neural Networks (IJCNN)*, 2023, pp. 1–9.

[16] T. Chen, D. Qiu, Y. Wu, A. Khan, X. Ke, and Y. Gao, "View-based explanations for graph neural networks," *Proceedings of the ACM on Management of Data*, vol. 2, no. 1, pp. 1–27, 2024.

[17] W. He, M. N. Vu, Z. Jiang, and M. T. Thai, "An explainer for temporal graph neural networks," in *GLOBECOM*, 2022, pp. 6384–6389.

[18] X. Chen, K. Wang, X. Lin, W. Zhang, L. Qin, and Y. Zhang, "Efficiently answering reachability and path queries on temporal bipartite graphs," *Proceedings of the VLDB Endowment*, 2021.

[19] M. Zhu, L. Zhu, H. Li, W. Li, and Z. Zhang, "Resistance distances in directed graphs: Definitions, properties, and applications," *Theoretical Computer Science*, vol. 1009, p. 114700, 2024.

[20] M. Black, Z. Wan, A. Nayyeri, and Y. Wang, "Understanding over-squashing in gnns through the lens of effective resistance," in *ICML*, 2023.

[21] X. Shen, P. Lio, L. Yang, R. Yuan, Y. Zhang, and C. Peng, "Graph rewiring and preprocessing for graph neural networks based on effective resistance," *TKDE*, 2024.

[22] M. Liao, R.-H. Li, Q. Dai, H. Chen, H. Qin, and G. Wang, "Efficient resistance distance computation: The power of landmark-based approaches," *SIGMOD*, vol. 1, no. 1, pp. 1–27, 2023.

[23] K. P. Murphy, *Dynamic bayesian networks: representation, inference and learning*. University of California, Berkeley, 2002.

[24] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[25] "Temgx-full version," 2025. [Online]. Available: https://anonymous.4open.science/r/TemGx-full-version-D66E

[26] C. Qian, J.-C. Shi, Y. Yu, and K. Tang, "On subset selection with general cost constraints." in *IJCAI*, vol. 17, 2017, pp. 2613–2619.

[27] J. A. Gámez, J. L. Mateo, and J. M. Puerta, "Learning bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood," *Data Mining and Knowledge Discovery*, 2011.

[28] F. Li and Z. Zou, "Subgraph matching on temporal graphs," *Information Sciences*, vol. 578, pp. 539–558, 2021.

[29] A. Aghasadeghi, J. Van den Bussche, and J. Stoyanovich, "Temporal graph patterns by timed automata," *The VLDB Journal*, vol. 33, no. 1, pp. 25–47, 2024.

[30] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *KDD*, ser. KDD '19, 2019.

[31] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[32] P. Gao, F. Shao, X. Liu, X. Xiao, Z. Qin, F. Xu, P. Mittal, S. R. Kulkarni, and D. Song, "Enabling efficient cyber threat hunting with cyber threat intelligence," in *ICDE*, 2021, pp. 193–204.

[33] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," 2020. [Online]. Available: https://arxiv.org/abs/2006.10637

[34] J. You, T. Du, and J. Leskovec, "Roland: graph learning framework for dynamic graphs," 2022.

[35] Y. Zhu, F. Cong, D. Zhang, W. Gong, Q. Lin, W. Feng, Y. Dong, and J. Tang, "Wingnn: dynamic graph neural networks with random gradient aggregation window," 2023.

[36] V. V. Vazirani, *Approximation algorithms*. Springer, 2013.
[37] B. C. Levy, A. Benveniste, and R. Nikoukhah, "High-level primitives for recursive maximum likelihood estimation," *IEEE Transactions on Automatic Control*, vol. 41, no. 8, pp. 1125–1145, 1996.

APPENDIX

Proof of Lemma 3. Procedure genInstanceX is a $(1-\frac{1}{e})$-approximation for the low-level optimization problem, *i.e.,* $\Phi(V_s, v) \geq (1-\frac{1}{e})\Phi(V_s^O, v)$.

Proof. We prove this result by showing that our problem can be formulated as a monotone submodular function maximization under a cardinality constraint, a well-known setting where a greedy algorithm guarantees a $(1-\frac{1}{e})$-approximation [26].

Specifically, consider the set of candidate explanatory nodes $\mathcal{C}$ for the target output $\mathcal{M}(G_t, v)$, and let $\Phi(V_s, v)$ be the temporal influence of any $V_s \subseteq \mathcal{C}$ on $\mathcal{M}(G_t, v)$. Define V_s^O as an optimal subset of size at most k, i.e.,

$$V_s^O \;=\; \arg \max_{U \subseteq \mathcal{C},\, |U| \leq k} \Phi(U, v).$$

We claim that $\Phi(\cdot)$ is *monotone submodular* on subsets of $\mathcal{C}$.

a) Monotonicity: Monotonicity holds because adding an explanatory node cannot reduce $\Phi(V_s, v)$ *i.e.,* for any $V_s \subseteq \mathcal{C}$ and $v_s \in \mathcal{C} \backslash V_s$, $\Phi(V_s \bigcup v_s, v) \geq \Phi(V_s, v)$, this follows directly from the definition:

$$\Phi(V_s, v) = \frac{1}{\delta} \sum_{v_s \in V_s} \sum_{t'=t-\delta+1}^{t} [\inf(v_s, v, t') - h(\mathsf{trd}(v_s, v, t'))]_+$$

where $[\cdot]_+$ denotes $\max\{0, \cdot\}$. Since each summation term is non-negative, adding an additional explanatory node either introduces new positive terms or leaves the existing sum unchanged, thus maintaining monotonicity.

b) Submodularity (Diminishing Returns): To establish submodularity, we need to show that for any sets $A \subseteq B \subseteq \mathcal{C}$ and node $v_s \in \mathcal{C} \setminus B$, the marginal gain satisfies:

$$\Phi(A \cup \{v_s\}, v) - \Phi(A, v) \;\geq\; \Phi(B \cup \{v_s\}, v) - \Phi(B, v).$$

Notice from the definition that the marginal gain introduced by adding node v_s is independent of the nodes already present in the set. Specifically, the contribution of node v_s is given explicitly by $\frac{1}{\delta} \sum_{t'=t-\delta+1}^{t} \big[\inf(v_s, v, t') - h(\mathsf{trd}(v_s, v, t'))\big]_+$, which is unaffected by the presence or absence of other nodes in the selected set. Consequently, the marginal gain is constant with respect to the subset context, trivially fulfilling the diminishing returns condition required for submodularity. Therefore, $\Phi(\cdot)$ is submodular over subsets of $\mathcal{C}$.

Since we have demonstrated that $\Phi(\cdot)$ is monotone submodular, applying the standard greedy algorithm under the cardinality constraint $|V_s| \leq k$ yields a $(1-\frac{1}{e})$-approximation guarantee. Formally, the solution V_s returned by genInstanceX satisfies:

$$\Phi(V_s, v) \geq (1-\tfrac{1}{e})\, \Phi(V_s^O, v).$$

This completes the proof. □

□

Performance analysis of genInstanceX **for multiple target outputs**. By reducing the problem to maximum coverage problem [36], genInstanceX ensures the same guarantee of $1-\frac{1}{e}$ as its counterpart for single targeted TGNN output, and can be implemented with the same worst-case time cost.

The maximum weighted coverage problem takes as input a collection $\mathcal{S}$ of sets $\{S_1, \ldots S_n\}$, and aims to find a subset $\mathcal{S}' \subset \mathcal{S}$ of size k, such that the total weight of the elements in $\bigcup_{S \in \mathcal{S}'} S$ is maximized. Given graph G as a single snapshot with a set of target nodes V_T and their outputs, we construct the following reduction. (1) We set $\mathcal{S}$ to be V_T. (2) For each node v in G, we invoke the verification procedure to check if any of its adjacent edge is a counterfactual edge for any target node $v_j \in V_T$; if so, add the node v to a set of explanatory nodes V_{ϵ_j} corresponding to the target node v_j. We say v "covers" v_j in this case. This process takes at most $|V|$ verifications. (3) We then construct a universal explanatory node set V_ϵ, which contains explanatory nodes that at least cover one target node in V_T. For each explanatory node $v_\epsilon \in V_\epsilon$, We initialize a set of subsets of nodes $\mathcal{S}$, such that each subset $S \in \mathcal{S}$ contains the test nodes covered by v_ϵ. (4) We set the weights of each element as the temporal influence score $\Phi(v_\epsilon, v)$, for each pair of explanatory node v_ϵ and the target node v it covers. We may then verify that the solution of the maximum weighted coverage problem, as a subset $S' \subset \mathcal{S}$, can be converted to a set of selected explanatory nodes that cover exactly a set S of targeted nodes in S'. This provides a solution for node selection with multiple outputs.

The optimality guarantee follows from the greedy strategy applied for solving maximum weighted coverage problem, which is ensured by solving the problem of maximization of submodular functions with a cardinality constraint. As the sum of submodular functions remain to be a monotone submodular function, we only need to extend procedure genInstanceX for V_T, which (1) for each node v in G, verifies if each of its edges is in a counterfactual set, for every node in V_T, in a single batch, to decide the many-to-many coverage relation; and (2) initializes an instance of a maximum weighted coverage problem, and apply consistently the greedy selection strategy to obtain the results.

The overall cost is in $O(|V_T| t \times k \log k + |N_L(V_T)| T_I)$. The worst case occurs when each node in V_T has a distinct set of explanatory nodes that are disjoint with any counterpart of the rest of the nodes in V_T.

Global-level Inference: Inference of DBN. The inference analysis over DBN is supported by efficient algorithms [23], and can be processed in low polynomial time *e.g.,* $O(|\eta| t)$, where $|\eta|$ is the total number of nodes and edges of the explanatory temporal subgraphs in η, and t refers to the total period length [23]. We present such an inference algorithm over TemGX as follows.

The **interface** algorithm provides an efficient implementation of the forward/backward(FB) operators [37] for exact inference in DBNs, leveraging the junction tree algorithm to perform message passing. Organizing the inference

process around interface variables, it localizes computations within each time slices and facilitates efficient propagation of information across slices. The forward operator computes the belief about the hidden states given observations up to time t. It recursively updates the forward message as: $f_{t|t} = \sum_{X_{t-1}} P(X_t|X_{t-1}) f_{t-1|t-1} P(y_t|X_t)$, where $P(X_t|X_{t-1})$ and $P(y_t|X_t)$ represent the state transition and observation likelihood, respectively. The associated conditional log-likelihood is: $L_t = \log P(y_t|y_{1:t-1})$. The backward operator propagates information from future observations to earlier time steps. It updates recursively as: $b_{t|T} = \sum_{X_{t+1}} P(X_{t+1}|X_t) P(y_{t+1}|X_{t+1}) b_{t+1|T}$, where $b_{t|T} = P(y_{t+1:T}|X_t)$ represents the likelihood of future observations given the current state. Once the forward and backward messages are computed, the smoothed marginal distribution of the hidden states at each time step can be obtained as: $P(X_t|y_{1:T}) \propto f_{t|t} \cdot b_{t|T}$, this allows the incorporation of information from both past and future observations to refine the estimate of the hidden states. To efficiently handle the computational complexity of exact inference in DBNs, the interface algorithm utilizes the junction tree framework, the key steps are:

1) Treat each time slice of DBN as a separate Bayesian network, the dependencies within slices are represented as a clique tree(junction tree), enabling efficient local computations.
2) Only connecting consecutive time slices are used for inter-slice message passing, as this reduces the computational overhead by focusing on the variables that encode temporal dependencies.
3) Within each time slice, belief propagation is performed on the junction tree to compute the local marginals of variables, the results are then used to update the interface variables for the next time slices.

The computational complexity of the interface algorithm lies in $O(K^{I+D})$ per time step, where: I is the size of the forward interface, D is the number of hidden nodes per slice, and K is the maximum number of values each discrete hidden node can take. For t total time steps, the overall complexity of the algorithm becomes $O(t \cdot K^{I+D})$, where t refers to the total period length. The interface algorithm lies in pseudo-polynomial time due to its linear dependence on t, and the structural complexity of DBN is determined by $|\eta|$, which directly affects I, D, K, specifically: $|\eta|$ governs the size of the junction tree, and the largest clique size $w \approx I + D$ determines the cost of local computations, K reflects the influence of hidden node cardinality and structural dependencies. This complexity ensures the algorithm scales well with the size of the DBN structure $|\eta|$ and the temporal length t, making it efficient and practical for real-world applications. The interface algorithm provides a computationally efficient solution for performing exact inference in DBNs.