Explaining Temporal Graph Neural Network via Quantum-Inspired Evolutionary Algorithm

Masahiro Mitani[1], **Yuya Sasaki**[1]

[1]The University of Osaka

Abstract

Temporal Graph Neural Network (TGNN) explanation has attracted increasing attention due to its applicability in dynamic scenarios such as recommendation systems. However, existing explanation methods for TGNNs face two key limitations: (1) computational inefficiency and (2) a restricted focus on either factual or counterfactual explanations, but not both. In this paper, we propose QIEA-TGX, an efficient and unified explanation algorithm based on a quantum-inspired evolutionary algorithm. QIEA-TGX effectively generates explanatory subgraphs that significantly influence TGNN predictions, without requiring additional model training or extensive inference. Experimental results on real-world datasets demonstrate that QIEA-TGX improves explanation fidelity by up to 31% while reducing computation time by up to 92% compared to state-of-the-art baselines.

Code — https://github.com/yuya-s/QIEA-TGX

Introduction

A temporal graph is a data structure in which nodes and edges evolve over time (Holme and Saramäki 2012). Each edge can be viewed as an *event*, representing an interaction between two nodes at a continuous time. Temporal graphs are widely used to model dynamic interactions in domains such as social networks (Cook and Holder 2006; Deng, Rangwala, and Ning 2019), transportation systems (Li et al. 2017; Jiang and Luo 2022), communication networks (Joshi and Hadi 2015), and recommendation systems (Wu et al. 2022; Gao et al. 2022), enabling the analysis of time-varying relational structures.

Temporal Graph Neural Networks (TGNNs) are deep learning models tailored to capture these evolving relationships (Wu et al. 2022; Rossi et al. 2020). They have shown strong performance in event prediction tasks (i.e., link prediction) on temporal graphs. However, like other neural network models, TGNNs operate as black boxes, obscuring the reasoning behind their predictions and limiting trust and transparency (He et al. 2022). To address this issue, Explainable AI (XAI) has gained attention for TGNN (Shneiderman 2020), which aims to reveal the internal decision-making mechanisms and identify the key temporal patterns driving predictions.

Explainability of TGNNs. A variety of methods have been proposed to explain TGNN predictions (Kakkad et al. 2023). One prominent approach is perturbation-based explanation, which constructs an *explanation graph* by selecting a subset of events from the input temporal graph (Yuan et al. 2022; Kakkad et al. 2023). These methods can be applied in both factual and counterfactual settings (Ying et al. 2019; Luo et al. 2020; Lucic et al. 2022; Tan et al. 2022):

- Factual explanations identify minimal subgraphs that preserve the model's original prediction (Xia et al. 2023; Chen and Ying 2023),
- Counterfactual explanations identify minimal changes that would lead to a different prediction (Qu, Gomm, and Färber 2024).

Both tasks are fundamentally combinatorial optimization problems, requiring efficient exploration of a large event space (Xia et al. 2023).

Motivation. While several perturbation-based explainers have been developed for TGNNs such as T-GNNExplainer (Xia et al. 2023), GreeDy (Qu, Gomm, and Färber 2024), and TempME (Chen and Ying 2023), these methods suffer from major limitations. First, they are often computationally expensive, relying on additional model training and/or a large number of inferences. Second, most methods are tailored to either factual or counterfactual explanations, making it unclear how well they generalize across other explanation types.

For instance, T-GNNExplainer employs Monte Carlo tree search over TGNN predictions, which incurs high runtime and is only evaluated for factual cases. TempME focuses on discovering temporal motifs to preserve predictions but cannot be easily adapted for counterfactual reasoning.

Consequently, there is a clear need for a unified and efficient approach that can generate both factual and counterfactual explanations across both existing and non-existing events without excessive computation.

Contributions. In this paper, we propose a novel and efficient explanation algorithm for TGNNs, based on a quantum-inspired evolutionary algorithm (QIEA) (Zhang 2011), which we call QIEA-TGX. To the best of our knowledge, this is the first TGNN explainer to leverage evolution-

ary search methods. We begin by formulating a unified problem definition that covers four explanation scenarios: factual and counterfactual explanations for both existing and non-existing events. Given a TGNN model, a temporal graph, and a target event, the goal is to identify key events that lead to either preserving or changing the model's prediction.

The core of QIEA-TGX is its use of quantum-bit-based population representations, allowing multiple solution candidates to be modeled in superposition. Explanation graphs are generated by observing quantum bits, evaluated using the TGNN model, and then fed back to update the quantum state. This iterative process enables efficient exploration of the solution space while avoiding local optima. Furthermore, we incorporate graph structure-aware optimizations into the observation process to reduce unnecessary exploration and improve fidelity. We evaluate QIEA-TGX on two real-world temporal graph datasets (Wikipedia and Reddit) and show that it consistently outperforms state-of-the-art explainers. Our method improves fidelity by up to 31% and reduces runtime by up to 92% compared to baselines.

Our key contributions are as follows:

- **Unified problem definition**. We present a unified formulation that encompasses factual and counterfactual explanations for both existing and non-existing events.
- **Novel algorithm**. We propose QIEA-TGX, the first TGNN explainer based on a quantum-inspired evolutionary algorithm, capable of efficiently generating high-fidelity explanation graphs.
- **Extensive evaluation**. We demonstrate the effectiveness, efficiency, and interpretability of QIEA-TGX through experiments on two real-world temporal graphs, outperforming existing approaches in both fidelity and runtime.

Related Work

Temporal graph neural networks. Discrete-time dynamic graphs (DTDG) and continuous-time dynamic graphs (CTDG) are two types of temporal graphs (Kazemi et al. 2020). A DTDG is a collection of static graph snapshots saved at fixed time intervals. On the other hand, a CTDG consists of a set of timestamped events, including the addition, deletion, and feature transformation of edges and nodes. In this paper, we model temporal graphs by CTDGs.

TGNNs effectively model graph structures that evolve over time, enabling their application to various tasks. To capture temporal dependencies and structural features effectively, various TGNN methods have been proposed, which can be classified into Message Passing (MP)-based methods (Trivedi et al. 2019; Wang et al. 2021a,b; Zhou et al. 2022; Yu et al. 2023; Xu et al. 2020; Rossi et al. 2020), Walk Aggregation (WA)-based methods (Wang et al. 2021c; Jin, Li, and Pan 2022), and hybrid methods integrating both approaches (Souza et al. 2022; Luo and Li 2022). We use two standard MP-based methods, TGAT (Xu et al. 2020) and TGN (Rossi et al. 2020), in experimental studies.

Explainability of GNN. Both factual and counterfactual explanation methods for GNNs are still evolving. These methods enhance the interpretability and usability of GNN models in real-world applications (Kakkad et al. 2023).

Factual explanation methods for GNNs aim to identify the input features or substructures that most significantly influence the model's predictions. These methods can be divided into two approaches: self-interpretable explanations and post-hoc explanations (Kakkad et al. 2023). Self-interpretable explanations are built into the architecture of the predictive model itself, enabling the generation of explanations alongside predictions (Miao, Liu, and Li 2022). Post-hoc explanation methods, in contrast, operate independently of the predictive model (Schlichtkrull, De Cao, and Titov 2020). Post-hoc explanation methods utilize separate models to explain the predictions of pre-trained GNNs, making them applicable to a wide range of GNN architectures. There are several types, such as decomposition, gradient-based, perturbation-based, generation, and surrogate explanations (Ying et al. 2019; Yuan et al. 2020; Shan et al. 2021; Huang et al. 2022; Duval and Malliaros 2021). Our study focuses on a perturbation-based explanation, which is a post-hoc explanation method.

We review existing perturbation-based methods here. GNNExplainer (Ying et al. 2019) identifies key nodes and edges by observing changes in output when parts of the input static graph are perturbed. PGExplainer (Luo et al. 2020) estimates the importance of events and generates explanation graphs by sampling events according to their importance. T-GNNExplainer (Xia et al. 2023) adapts PGExplainer (Luo et al. 2020) for temporal graphs using Monte-Carlo tree search to capture temporal dependencies. TempMe (Chen and Ying 2023) identifies key temporal motifs. TempMe is hard to apply counterfactual explanations because motifs not included in the graphs might need to change predictions.

Counterfactual explanations for GNNs aim to identify minimal changes to the input graph that would result in a different prediction. These methods help users understand how changes in input affect predictions, offering actionable insights in real-world applications. In TGNNs, CoDy and GreeDy (Qu, Gomm, and Färber 2024) provide counterfactual explanations for link prediction tasks. CoDy uses Monte-Carlo tree search, and GreeDy iteratively removes the event with the highest impact on the TGNN prediction.

Preliminaries

Continuous-Time Dynamic Graph (CTDG) is denoted by $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ and $\mathcal{E} = \{e_1, e_2, ..., e_T\}$ denote the sets of nodes and events, respectively. Each event $e_i = (u_i, v_i, t_i)$ represents an interaction from a source node u_i to a target node v_i that occurs at timestamp t_i. We define that e_i happens earlier than e_{i+1} (i.e., t_i is smaller than t_{i+1}), and all nodes in $\mathcal{N}$ are associated with at least a single event.

A link prediction task in TGNNs aims to accurately predict the occurrence/absence of a given link $e_k = (u_k, v_k, t_k)$ from graph $\mathcal{G}^k = (\mathcal{N}^k, \mathcal{E}^k)$ where $\mathcal{E}^k = \{e_1, e_2, ..., e_{k-1}\}$. Since we handle links as events, it predicts the occurrence or absence of *target event*. In TGNNs, it is well-known that it is effective only to use events close in time and/or distance from the target event instead of using the whole graph $\mathcal{G}^k$ (Xia et al. 2023; Rossi et al. 2020; Xu et al. 2020). We call a subset of $\mathcal{G}^k$ as neighboring graph $\mathcal{G}^c = (\mathcal{N}^c, \mathcal{E}^c)$.

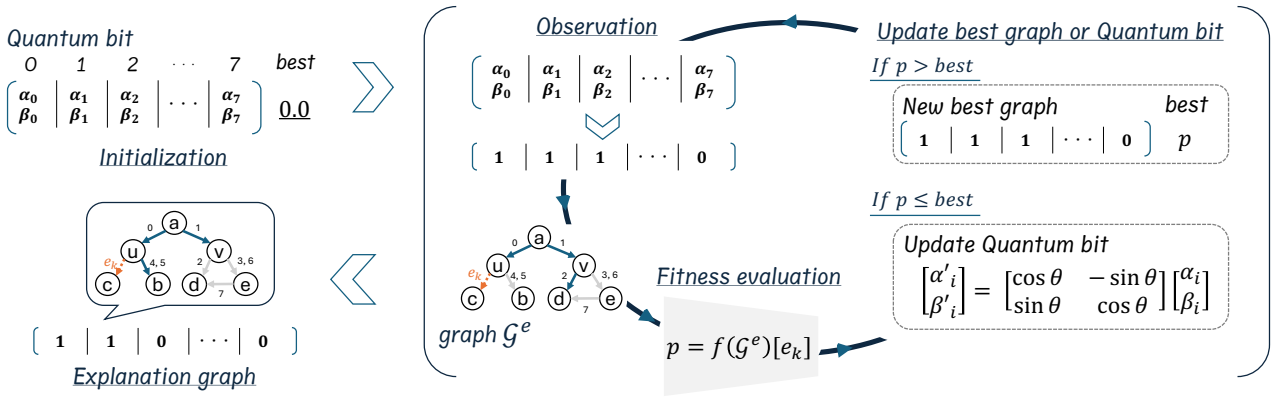


Figure 1: Overview of QIEA-TGX. This flow shows a factual explanation for existing events.

Definition 1 (TGNN)**.** *Given target event e_k and neighboring graph $\mathcal{G}^c \subseteq \mathcal{G}^k$, temporal graph neural network model f outputs the probability $f(\mathcal{G}^c)[e_k]$ of existence of e_k.*

Problem Definition

TGNNs are commonly black box models. Since it is challenging to determine which events affect the prediction, temporal graph neural network explainers aim to identify a subgraph $\mathcal{G}^e = (\mathcal{N}^c, \mathcal{E}^e)$, called an *explanation graph*, that includes essential events.

We have two types of explanations: factual and counterfactual explanations. Factual explanation aims to identify a *small* subgraph that keeps the same prediction, whereas counterfactual explanation aims to identify a *large* subgraph that changes the prediction (i.e., larger subgraphs are more similar to the original, as fewer edges are removed). In addition, target events can be either existing or non-existing. For example, a counterfactual explanation for the non-existing target event outputs a subgraph that would lead to the occurrence of the event that actually does not happen.

Therefore, there are four types of explainer problems. We define our problem as follows:

Problem 2 (Unified Temporal Graph Neural Network Explanation Problem)**.** *Given temporal graph $\mathcal{G}^c \subseteq \mathcal{G}^k$, target event e_k, TGNN model f, and sparsity $\phi \in (0, 1)$, the problem aims to identify $\mathcal{G}^e$ that*

1. *in factual explanation for existing events: maximizes $f(\mathcal{G}^e)[e_k]$ subject to $\frac{|\mathcal{E}^e|}{|\mathcal{E}^c|} \leq \phi$,*
2. *in factual explanation for non-existing events: minimizes $f(\mathcal{G}^e)[e_k]$ subject to $\frac{|\mathcal{E}^e|}{|\mathcal{E}^c|} \leq \phi$,*
3. *in counterfactual explanation for existing events: minimizes $f(\mathcal{G}^c \backslash \mathcal{G}^e)[e_k]$ subject to $\frac{|\mathcal{E}^c \backslash \mathcal{E}^e|}{|\mathcal{E}^c|} \leq \phi$, and*
4. *in counterfactual explanation for non-existing events: maximizes $f(\mathcal{G}^c \backslash \mathcal{G}^e)[e_k]$ subject to $\frac{|\mathcal{E}^c \backslash \mathcal{E}^e|}{|\mathcal{E}^c|} \leq \phi$,*

where $\mathcal{G}^c \backslash \mathcal{G}^e = (\mathcal{N}^c, \mathcal{E}^c \backslash \mathcal{E}^e)$.

Following this problem definition, we can modify algorithms for factual explanations to counterfactual explanations and vice versa. Notably, this is a combinatorial optimization problem (Xia et al. 2023). To obtain the optimal explanation graph, we need to evaluate all patterns of explanation graphs that are not solvable in a reasonable time.

Proposed method: QIEA-TGX

QIEA-TGX leverages the Quantum-Inspired Evolutionary Algorithm (QIEA) (Zhang 2011), an evolutionary algorithm with high search capabilities based on quantum mechanics concepts. Since QIEA represents the population by quantum bits, it essentially includes any patterns of individuals with different probabilities. This nature of QIEA is helpful in generating diverse explanation graphs.

Overview of QIEA-TGX. Figure 1 illustrates the overall framework of QIEA-TGX. This method utilizes quantum bits as the population and generates explanation graphs by observing quantum bits. After generating and evaluating $\mathcal{G}^e$, it updates quantum bits according to the newly generated explanation graphs.

Algorithm 1 shows pseudocode of QIEA-TGX consisting of four steps: initialization, observation, fitness evaluation, and update. The input of ζ indicates a given explanation type, such as a factual explanation for existing events. The initialization step initializes the quantum bit $\mathcal{Q}$, which is the basis for generating explanation graphs (line 1). The observation step generates an explanation graph from $\mathcal{Q}$ (lines 2 and 5). The generated explanation graph must satisfy the sparsity condition. The fitness evaluation step measures the fitness of explanation graphs, indicating how good they are (lines 3 and 6). In our study, fitness is the probability of the target event's existence, calculated through TGNN inference. If $\mathcal{G}^e$ is better than the current best $\mathcal{G}^e_{best}$, it keeps $\mathcal{G}^e$ (line 7). Here, $\succ_\zeta$ indicates an operator to compare the probability depending on the explanation type. The update step updates $\mathcal{Q}$ to generate better individuals based on the generated explanation graph (line 10). These processes are repeated until L explanation graphs are generated.

Quantum Bit Representation and Initialization

QIEA represents the population as sequences of quantum bits. A quantum bit is defined as $|q\rangle = \alpha|0\rangle + \beta|1\rangle$, satisfying $\alpha^2 + \beta^2 = 1$. α^2 and β^2 represent the probabilities of observing the quantum bit as 0 and 1, respectively. This

Algorithm 1: Procedures of QIEA-TGX

input : TGNN model f, target event e_k, neighboring graph $\mathcal{G}^c$, sparsity ϕ, explanation type ζ, # of generations L
output: Explanation graph $\mathcal{G}^e$

$\mathcal{Q} \leftarrow$ INITIALIZE($\mathcal{E}^c$);
$\mathcal{G}^e_{best} \leftarrow$ OBSERVE($\mathcal{Q}, \mathcal{G}^c, \phi$);
$p_{best} \leftarrow$ EVALUATE($\mathcal{G}^e_{best}, f, e_k, \mathcal{G}^c, \zeta$);
repeat
  $\mathcal{G}^e \leftarrow$ OBSERVE($\mathcal{Q}, \mathcal{G}^c, \phi$) ;
  $p \leftarrow$ EVALUATE($\mathcal{G}^e, f, e_k, \mathcal{G}^c, \zeta$);
  if $p \succ_\zeta p_{best}$ **then**
    $\mathcal{G}^e_{best} \leftarrow \mathcal{G}^e$; $p_{best} \leftarrow p$;
  else
    $\mathcal{Q} \leftarrow$ UPDATE($\mathcal{Q}, \mathcal{G}^e, p$);
until L *times*
return $\mathcal{G}^e_{best}$

representation enables simultaneous handling of a large solution space, improving both search efficiency and diversity. Each quantum bit corresponds to each event in $\mathcal{E}^c$.

In QIEA-TGX, each quantum bit is initialized to a superposition state with equal probabilities for "0" and "1": all α and β are $\frac{1}{\sqrt{2}}$, satisfying the normalization condition.

Observation and Fitness Evaluation

Observation transforms the quantum bit sequence into a sequence of binary bits. This sequence of binary bits is then used for fitness evaluation. Each quantum bit $|q_i\rangle$ collapses to "0" or "1" based on probabilities α_i^2 and β_i^2. Through the following Q-gate update, values of α that correspond to important events become larger values, on the other hand, β corresponding less important events become larger values. Therefore, this observation can generate promising explanation graphs while maintaining diversity.

Fitness evaluation uses TGNN to obtain the probability of occurrence of the target event. Based on the observed explanation graph $\mathcal{G}^e$ and the explanation type, the fitness of $\mathcal{G}^e$ is determined as the output $f(\mathcal{G}^e)[e_k]$ and $f(\mathcal{G}^e \backslash \mathcal{G}^c)[e_k]$ for factual and counterfactual explanations, respectively.

Update

QIEA-TGX updates quantum bits based on the fitness evaluation. The operations consist of Q-gate and H-gate updates.

Q-gate update. The update of quantum bits is through Q-gate that uses a rotation matrix in the following:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} \cos(\theta_Q) & -\sin(\theta_Q) \\ \sin(\theta_Q) & \cos(\theta_Q) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix},$$

where θ_Q is a hyper parameter.

These updates are applied only if the binary solution from observation has a lower fitness than the current best solution and differs in the corresponding bit. In this way, we can efficiently update the quantum bit states, and their values converge on promising solutions within the search space.

H-gate update. If α or β are close to 1, the observation step outputs explanation graphs that the corresponding events are always unused or used, leading to reduced search diversity and may lead to local optima. The H-gate transforms a quantum bit to avoid local optima (Nielsen and Chuang 2010). After Q-gate updates, if α_i (resp. β_i) is larger than $\sqrt{1-\theta_H}$, H-gate updates such quantum bit α_i (resp. β_i) to $\sqrt{1-\theta_H}$ and β_i (resp. α_i) to $\sqrt{\theta_H}$. This update is simple but effective in maintaining diverse explanation graphs.

Graph Structure-aware Optimization

We present an optimization technique to effectively generate explanation graphs to reduce unnecessary explorations aligning with the structure of the temporal graph. The observation step in conventional QIEA does not consider the characteristics of TGNNs and the structures of temporal graphs. Events that are not connected to a target event do not affect the prediction of TGNNs.

To address this, the observation step leverages the structural features of the temporal graph. Specifically, the optimization first observes only 1-hop events from the target event. Then, for 2-hop events, it determines whether to observe or not based on their connectivity with the target event through 1-hop events and proceeds with observations accordingly. It repeats the same procedures for further events. In addition, it ensures that the number of events in an explanation graph does not exceed the allowable size by stopping the observation. This optimization can exclude unnecessary events, thereby improving efficiency.

Time Complexity

The time complexity of QIEA-TGX is $O(L \cdot (|\mathcal{E}^c| + T_f))$, where T_f is the inference time of the TGNN model f. The initialization, observation, and update steps take $O(|\mathcal{E}^c|)$, as they iterate over the neighboring events to compute or update the quantum states. The fitness evaluation step requires T_f, corresponding to a forward pass of f to evaluate the probability of the target event. Therefore, the total time complexity of QIEA-TGX is linear in the number of generated explanation graphs L, the size of the neighboring event set $|\mathcal{E}_c|$, and the TGNN inference time T_f.

Experimental Study

In our experimental study, we (1) validate the performance of our algorithms in both factual and counterfactual explanations against the state-of-the-art baselines, (2) analyze our methods, and (3) show examples of explanation graphs.

The experiments were conducted on a server with an NVIDIA Quadro RTX 6000 GPU. All methods were implemented in Python3.

Datasets. We use two real-world temporal graph datasets, Wikipedia[1] and Reddit[2], following existing studies (Xia et al. 2023; Chen and Ying 2023; Qu, Gomm, and Färber 2024; Seo et al. 2024; Fang et al. 2024). The Wikipedia dataset consists of approximately 9,300 nodes with around

[1] http://snap.stanford.edu/jodie/wikipedia.csv
[2] http://snap.stanford.edu/jodie/reddit.csv

	Factual				Counterfactual			
	Wikipedia		Reddit		Wikipedia		Reddit	
	TGAT	TGN	TGAT	TGN	TGAT	TGN	TGAT	TGN
Random	$0.046_{\pm 0.13}$	$0.078_{\pm 0.11}$	$0.053_{\pm 0.12}$	$0.063_{\pm 0.11}$	$0.335_{\pm 0.26}$	$0.293_{\pm 0.21}$	$0.257_{\pm 0.24}$	$0.121_{\pm 0.13}$
GreeDy	$\underline{0.060}_{\pm 0.11}$	$\underline{0.082}_{\pm 0.10}$	$\underline{0.064}_{\pm 0.12}$	$\underline{0.081}_{\pm 0.11}$	$\mathbf{0.568}_{\pm 0.33}$	$\underline{0.352}_{\pm 0.23}$	$\underline{0.823}_{\pm 0.23}$	$\underline{0.509}_{\pm 0.32}$
PGExplainer	$-0.102_{\pm 0.20}$	$-0.110_{\pm 0.20}$	$-0.143_{\pm 0.32}$	$-0.252_{\pm 0.32}$	$0.078_{\pm 0.15}$	$0.074_{\pm 0.14}$	$0.079_{\pm 0.14}$	$0.029_{\pm 0.06}$
T-GNNExplainer	$0.051_{\pm 0.10}$	$0.070_{\pm 0.09}$	$0.047_{\pm 0.10}$	$0.045_{\pm 0.08}$	$0.286_{\pm 0.24}$	$0.298_{\pm 0.21}$	$0.184_{\pm 0.21}$	$0.076_{\pm 0.09}$
GA-TGX	$\underline{0.060}_{\pm 0.10}$	$0.075_{\pm 0.10}$	$0.058_{\pm 0.12}$	$0.061_{\pm 0.11}$	$0.324_{\pm 0.26}$	$0.272_{\pm 0.21}$	$0.224_{\pm 0.22}$	$0.122_{\pm 0.13}$
QIEA-TGX	$\mathbf{0.066}_{\pm 0.11}$	$\mathbf{0.094}_{\pm 0.11}$	$\mathbf{0.084}_{\pm 0.12}$	$\mathbf{0.087}_{\pm 0.11}$	$\underline{0.560}_{\pm 0.33}$	$\mathbf{0.443}_{\pm 0.25}$	$\mathbf{0.825}_{\pm 0.20}$	$\mathbf{0.597}_{\pm 0.27}$

Table 1: Fidelity. Bold and underlined indicate the best and second-best results, respectively.

160,000 temporal edges. The Reddit dataset consists of approximately 11,000 nodes with 700,000 temporal events. In both datasets, each edge is associated with a 172-dimensional feature vector.

Models. We use two representative TGNN models, TGAT (Xu et al. 2020) and TGN (Rossi et al. 2020), following existing works (Xia et al. 2023; Chen and Ying 2023; Seo et al. 2024; Fang et al. 2024). These models are pre-trained in each dataset. We input events within one and two hops from u and v of target events. The events from a node are the closest 10 events, following existing study (Xia et al. 2023). The maximum number of events in $\mathcal{G}^c$ is 220 (i.e., at most 20 one-hop and 200 two-hop events).

Baselines. We use the following five baselines:

- **Random** is the most naive approach to sample events in uniform random and evaluates event sets.
- **PGExplainer** (Luo et al. 2020) estimates the importance of each event and adds them to the explanation graph in descending order of importance, which is for GNNs on static graphs.
- **GreeDy** (Qu, Gomm, and Färber 2024) iteratively adds or removes an event and applies the change that most improves the result, which is for counterfactual explanations in TGNNs.
- **T-GNNExplainer** (Xia et al. 2023) explores the graph using Monte Carlo Tree Search guided by an NN-based navigator, which is for factual explanations in TGNNs.
- **GA-TGX** employs a genetic algorithm with selection, crossover, and mutation, to search for event sets. We use our optimization technique for evolutionary operations.

In QIEA-TGX, we set θ_Q and θ_H to 0.01π and 0.1 as default values, respectively. We terminate generations of explanation graphs if generating 500 explanation graphs (i.e., $L = 500$) or exceeding 10 minutes.

Target events. We generate four cases of target events: existing and non-existing events for factual and counterfactual explanations. Each target event is correctly predicted by TGNNs (e.g., if target events exist in temporal graphs, TGNN correctly predicts their occurrence). We generate 100 pairs of sources and targets with timestamps for each case (i.e., a total of 400 pairs) in each dataset.

Evaluation. We evaluate fidelity and runtime to generate explanation graphs. Fidelity for factual and counterfactual explanations are defined as $\mathbb{1}(Y_k = 1)(f(\mathcal{G}^e)[e_k] - f(\mathcal{G}^c)[e_k] + \mathbb{1}(Y_k = 0)(f(\mathcal{G}^c)[e_k] - f(\mathcal{G}^e)[e_k]$, and $\mathbb{1}(Y_k = 1)(f(\mathcal{G}^c)[e_k] - f(\mathcal{G}^c \backslash \mathcal{G}^e)[e_k] + \mathbb{1}(Y_k = 0)(f(\mathcal{G}^c \backslash \mathcal{G}^e)[e_k] - f(\mathcal{G}^c)[e_k])$, respectively. $\mathbb{1}(\cdot)$ denotes the indicator function, and Y_k is the original prediction, indicating existing events if $Y_k = 1$ and non-existing evens if $Y_k = 0$. Higher fidelity indicates that the explanation graph can more accurately account for the TGNN's output.

We report average fidelity for factual and counterfactual explanations. We vary sparsity ϕ from 0.1 to 0.5; 0.2 is a default value.

Effectiveness and Efficiency

Table 1 shows the fidelity of each method in factual and counterfactual explanations. From these results, in most cases, QIEA-TGX and GreeDy achieve high fidelity. Since PGExplainer is for GNNs on static graphs, it does not work well for TGNNs. T-GNNExplainer and GA-TGX also do not work well in many cases; in particular, T-GNNExplainer often performs worse than random.

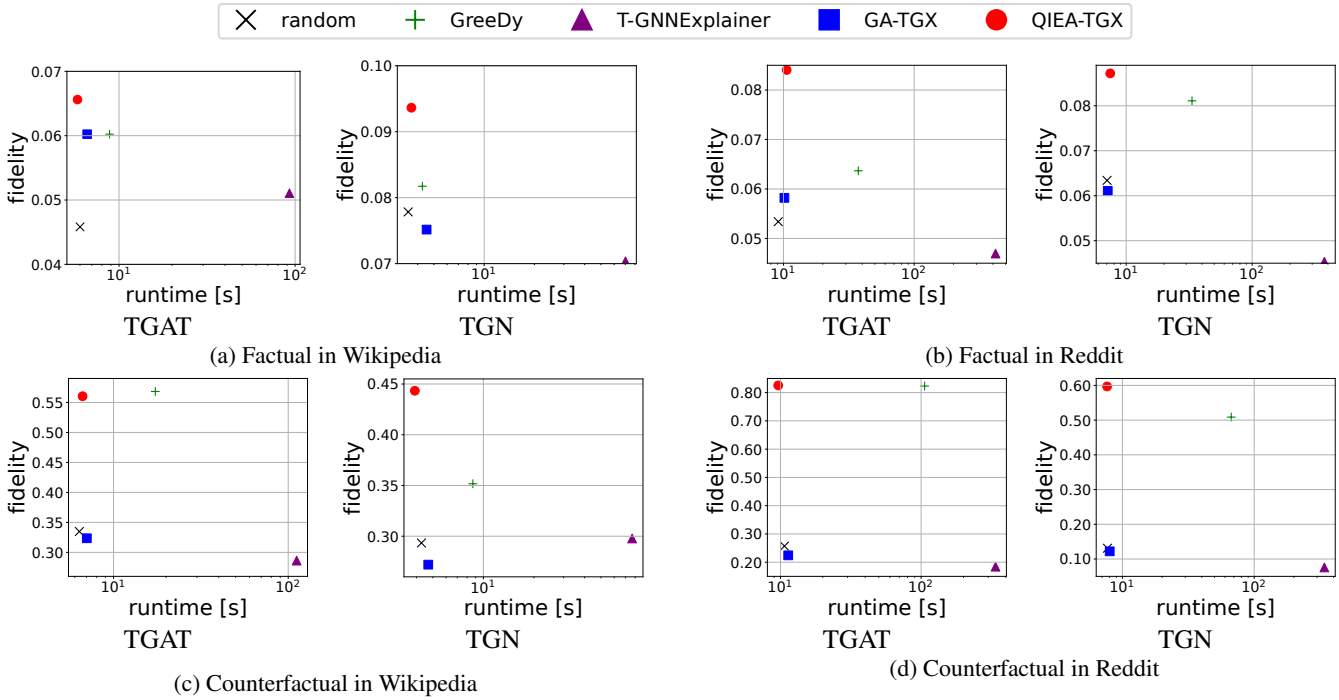
Figure 2 shows the trade-off between fidelity and runtime. This result shows QIEA-TGX achieves high fidelity (i.e., top) and efficiency (i.e., left). GA-TGX also achieves high efficiency but low fidelity. GreeDy and T-GNNExplainer are inefficient because these algorithms require a large number of TGNN predictions.

Figure 3 shows the fidelity and runtime as varying sparsity ϕ from 0.1 to 0.5. As the sparsity increases, the fidelity increases because the number of events within explanation graphs increases, leading to the expressive power of explanation graphs that change/keep the predictions. QIEA-TGX achieves the highest fidelity for all sparsity except for $\phi = 0.2$ of counterfactual explanations. In terms of runtime, Random, GA-TGX, and QIEA-TGX have much shorter runtime than GreeDy and T-GNNExplainer, and do not increase even for large sparsity and large temporal graphs. Consequently, QIEA-TGX achieves effectiveness and efficiency in any datasets, TGNN models, sparsity, and explanation types.

Analysis of our Algorithm

We analyze the effectiveness of the optimization, convergence, and hyper-parameter sensitivity in QIEA-TGX.

Table 2 shows the performance improvements for the graph structure-aware optimization in QIEA-TGX and GA-TGX. The optimization technique works better in QIEA-TGX than GA-TGX. Since the quantum bit of QIEA-TGX represents a large solution space, it is effective to restrict the generation of explanation graphs by graph structures. It



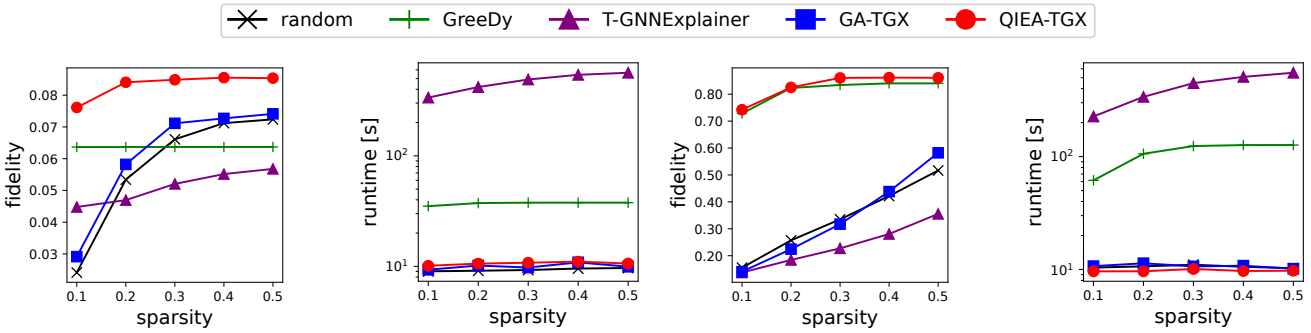
(a) Factual in Wikipedia

(b) Factual in Reddit

(c) Counterfactual in Wikipedia

(d) Counterfactual in Reddit

Figure 2: Fidelity and runtime. The x- and y-axes indicate (log scale) runtime and fidelity, respectively. Methods located in the upper-left corner indicate higher efficiency and accuracy.



(a) Factual

(b) Counterfactual

Figure 3: Impact on sparsity in TGAT and Reddit

works well for counterfactual explanations on Reddit for two reasons: (1) generating an explanation graph is more critical for large graphs, and (2) counterfactual explanations have more impact on events with one hop than factual explanations because, after removing one hop events, they can remove all two-hop events connecting to the one hop events.

Figure 4 shows the fidelity in the number of generated explanation graphs. QIEA-TGX with the optimization works well to increase the fidelity in a small number of counts, helping to quickly find high-quality explanation graphs.

Figure 5 shows the performance varying θ_Q and θ_H to validate hyperparameter sensitivity in QIEA-TGX. From these results, QIEA-TGX is not very sensitive to θ_Q and θ_H, but a very small θ_H causes local optima.

	Factual				Counterfactual			
	Wikipedia		Reddit		Wikipedia		Reddit	
	TGAT	TGN	TGAT	TGN	TGAT	TGN	TGAT	TGN
GA-TGX	-1.7%	-1.3%	1.7%	1.6%	0%	1.8%	5.8%	4.1%
QIEA-TGX	0%	0%	2.4%	0%	5.2%	11.9%	24.4%	25.2%

Table 2: Improvement of optimization

Examples of Explanation Graphs

We present examples of explanation graphs generated by Random, GreeDy, GA-TGX, and QIEA-TGX. The results of PGExplainer and T-GNNExplainer are omitted due to their low fidelity. For this analysis, we use the Wikipedia dataset and the TGAT model.

Figure 6 illustrates both the original graph and the ex-

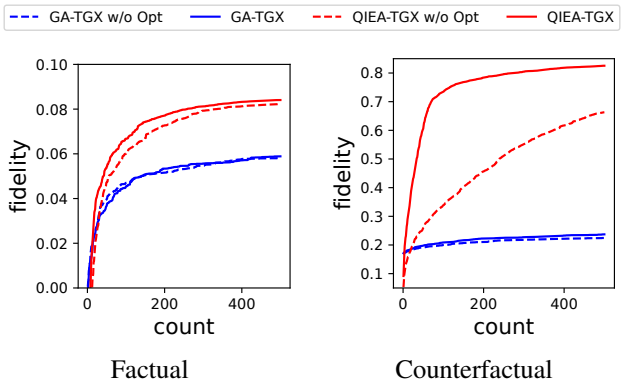
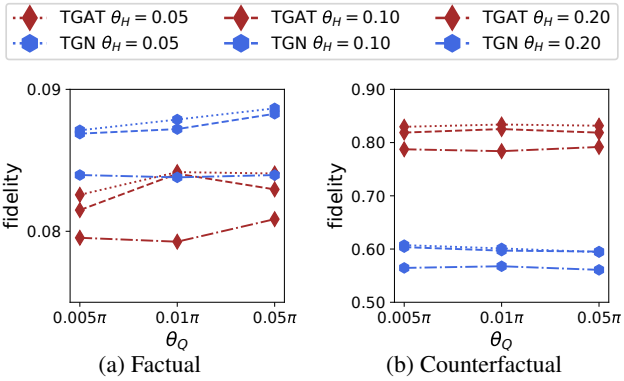


Figure 4: Convergence in TGAT and Reddit. "count" indicates the number of generated graphs.



(a) Factual (b) Counterfactual

Figure 5: Hyperparameter sensitivity in Reddit

planation graphs generated by each method in factual and counterfactual explanations for a non-existing event. In this example, the methods aim to explain why the user in the top-left does not edit the page in the top-right.

First, in the factual explanation, the graphs generated by GreeDy, GA-TGX, and QIEA-TGX are highly similar. The fidelity scores of GreeDy and QIEA-TGX are high, while GA-TGX exhibits slightly lower fidelity. These results suggest that the reason the top-left user does not edit the target page is likely due to two factors: (i) the user edits a different page (the third top-right) around the same time, and (ii) another user (the second top-left) edits the target page.

Second, in the counterfactual explanation, among the methods, QIEA-TGX and GreeDy achieve high fidelity. In contrast, GA-TGX performs less effectively in this case, possibly because it fails to capture the key events required to alter the model's prediction. GreeDy removes eight events occurring between the same pair of nodes (from the second top-left user to the top-right page), indicating that these interactions are highly influential. However, due to its lower fidelity compared to QIEA-TGX, the result may represent a local optimum. In contrast, QIEA-TGX removes both the important events identified by GreeDy and events that overlap with those selected by GA-TGX. Consistent with the insights from the factual explanation, the counterfactual explanation suggests that the top-left user would edit the target page if (i) they had not edited the third top-right page at a similar time, and (ii) the second top-left user had not edited the target page. In addition, it would edit the target page if the second top-right pages have not been edited. These results imply that the top-left and second top-left users exhibit similar editing behavior.

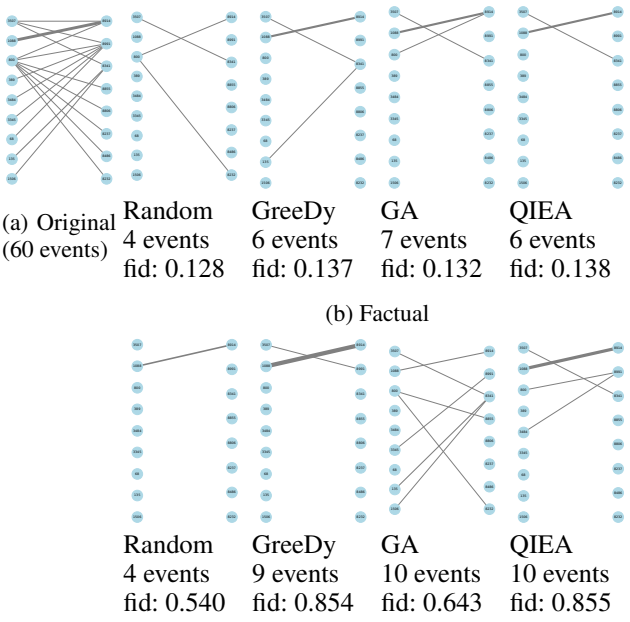


Figure 6: Examples of explanation graphs for non-existing events. Target event is an event between top-left and top-right nodes. Line thickness of links indicates the number of links between nodes (bolder is a larger number).

Overall, these examples demonstrate that QIEA-TGX provides interpretable and faithful explanation graphs while maintaining low inference time.

Conclusion

This paper is the first study that employs evolutionary algorithms for temporal graph neural network explainer. We proposed QIEA-TGX, using a quantum-inspired evolutionary algorithm. Our experiments showed that QIEA-TGX outperforms the state-of-the-art methods in terms of both effectiveness and efficiency.

Limitation and future work. Since this is the first study to use evolutionary algorithms to explain TGNNs, we have several directions to extend this work. First, QIEA works very well, but we may use other evolutionary algorithms. Second, we do not consider adding events and changing attributes as explanations, but they may be helpful for explanations. Third, we can extend it for specific domains, such as chemical reactions and recommendation systems.

Acknowledgments

This work was supported by Japan Science and Technology Agency (JST) as part of Adopting Sustainable Partnerships for Innovative Research Ecosystem (ASPIRE), Grant Number JPMJAP2328, and JSPS KAKENHI Grant Numbers JP23K28096 and JP25H01117.

References

Chen, J.; and Ying, R. 2023. Tempme: Towards the explainability of temporal graph neural networks via motif discovery. *NeurIPS*.

Cook, D. J.; and Holder, L. B. 2006. *Mining graph data*. John Wiley & Sons.

Deng, S.; Rangwala, H.; and Ning, Y. 2019. Learning dynamic context graphs for predicting social events. In *SIGKDD*, 1007–1016.

Duval, A.; and Malliaros, F. D. 2021. GraphSVX: Shapley Value Explanations for Graph Neural Networks. In *ECMLPKDD*, 302–318.

Fang, L.; Yang, Y.; Wang, K.; Feng, S.; Feng, K.; Gui, J.; Wang, S.; and Ong, Y.-S. 2024. SIG: Efficient Self-Interpretable Graph Neural Network for Continuous-time Dynamic Graphs. *arXiv preprint arXiv:2405.19062*.

Gao, C.; Wang, X.; He, X.; and Li, Y. 2022. Graph neural networks for recommender system. In *WSDM*, 1623–1625.

He, W.; Vu, M. N.; Jiang, Z.; and Thai, M. T. 2022. An explainer for temporal graph neural networks. In *GLOBECOM*, 6384–6389.

Holme, P.; and Saramäki, J. 2012. Temporal networks. *Physics reports*, 519(3): 97–125.

Huang, Q.; Yamada, M.; Tian, Y.; Singh, D.; and Chang, Y. 2022. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(7): 6968–6972.

Jiang, W.; and Luo, J. 2022. Graph neural network for traffic forecasting: A survey. *Expert systems with applications*, 207: 117921.

Jin, M.; Li, Y.-F.; and Pan, S. 2022. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. *NeurIPS*.

Joshi, M.; and Hadi, T. H. 2015. A review of network traffic analysis and prediction techniques. *arXiv preprint arXiv:1507.05722*.

Kakkad, J.; Jannu, J.; Sharma, K.; Aggarwal, C.; and Medya, S. 2023. A survey on explainability of graph neural networks. *arXiv preprint arXiv:2306.01958*.

Kazemi, S. M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; and Poupart, P. 2020. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70): 1–73.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Lucic, A.; Ter Hoeve, M. A.; Tolomei, G.; De Rijke, M.; and Silvestri, F. 2022. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *AISTATS*, 4499–4511.

Luo, D.; Cheng, W.; Xu, D.; Yu, W.; Zong, B.; Chen, H.; and Zhang, X. 2020. Parameterized explainer for graph neural network. *NeurIPS*.

Luo, Y.; and Li, P. 2022. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*.

Miao, S.; Liu, M.; and Li, P. 2022. Interpretable and generalizable graph learning via stochastic attention mechanism. In *ICML*, 15524–15543.

Nielsen, M. A.; and Chuang, I. L. 2010. *Quantum computation and quantum information*. Cambridge university press.

Qu, Z.; Gomm, D.; and Färber, M. 2024. GreeDy and CoDy: Counterfactual Explainers for Dynamic Graphs. *arXiv preprint arXiv:2403.16846*.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

Schlichtkrull, M. S.; De Cao, N.; and Titov, I. 2020. Interpreting graph neural networks for NLP with differentiable edge masking. *arXiv preprint arXiv:2010.00577*.

Seo, S.; Kim, S.; Jung, J.; Lee, Y.; and Park, C. 2024. Self-Explainable Temporal Graph Networks based on Graph Information Bottleneck. In *SIGKDD*, 2572–2583.

Shan, C.; Shen, Y.; Zhang, Y.; Li, X.; and Li, D. 2021. Reinforcement learning enhanced explainer for graph neural networks. *NeurIPS*.

Shneiderman, B. 2020. Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems. *TiiS*, 10(4): 1–31.

Souza, A.; Mesquita, D.; Kaski, S.; and Garg, V. 2022. Provably expressive temporal graph networks. *NeurIPS*.

Tan, J.; Geng, S.; Fu, Z.; Ge, Y.; Xu, S.; Li, Y.; and Zhang, Y. 2022. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *WWW*, 1018–1027.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. Dyrep: Learning representations over dynamic graphs. In *ICLR*.

Wang, L.; Chang, X.; Li, S.; Chu, Y.; Li, H.; Zhang, W.; He, X.; Song, L.; Zhou, J.; and Yang, H. 2021a. Tcl: Transformer-based dynamic graph modelling via contrastive learning. *arXiv preprint arXiv:2105.07944*.

Wang, X.; Lyu, D.; Li, M.; Xia, Y.; Yang, Q.; Wang, X.; Wang, X.; Cui, P.; Yang, Y.; Sun, B.; et al. 2021b. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *SIGMOD*, 2628–2638.

Wang, Y.; Chang, Y.-Y.; Liu, Y.; Leskovec, J.; and Li, P. 2021c. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *ICLR*.

Wu, S.; Sun, F.; Zhang, W.; Xie, X.; and Cui, B. 2022. Graph neural networks in recommender systems: a survey. *CSUR*, 55(5): 1–37.

Xia, W.; Lai, M.; Shan, C.; Zhang, Y.; Dai, X.; Li, X.; and Li, D. 2023. Explaining temporal graph models through an explorer-navigator framework. In *ICLR*.

Xu, D.; Ruan, C.; Körpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. In *ICLR*.

Ying, Z.; Bourgeois, D.; You, J.; Zitnik, M.; and Leskovec, J. 2019. Gnnexplainer: Generating explanations for graph neural networks. *NeurIPS*.

Yu, L.; Sun, L.; Du, B.; and Lv, W. 2023. Towards better dynamic graph learning: New architecture and unified library. *NeurIPS*.

Yuan, H.; Tang, J.; Hu, X.; and Ji, S. 2020. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 430–438.

Yuan, H.; Yu, H.; Gui, S.; and Ji, S. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5): 5782–5799.

Zhang, G. 2011. Quantum-inspired evolutionary algorithms: a survey and empirical study. *Journal of Heuristics*, 17(3): 303–351.

Zhou, H.; Zheng, D.; Nisa, I.; Ioannidis, V.; Song, X.; and Karypis, G. 2022. TGL: a general framework for temporal GNN training on billion-scale graphs. *PVLDB*, 15(8): 1572–1580.