Learning Counterfactual Explanation of Graph Neural Networks via Generative Flow Network

Kangjia He[a,b,c], Li Liu[a,b,c,*], Youmin Zhang[a,b,c], Ye Wang[a,b,c], Qun Liu[a,b,c], Guoyin Wang[a,b,c]

[a] *Chongqing Key Laboratory of Computational Intelligence, Chongqing, China*
[b] *Key Laboratory of Cyberspace Big Data Intelligent Security, Ministry of Education. Chongqing, China*
[c] *School of Computer Science and Technology, Chongqing University of Posts and Telecommunications,Chongqing, 400065, China*

Abstract

Counterfactual subgraphs explain Graph Neural Networks (GNNs) by answering the question: "How would the prediction change if a certain subgraph were absent in the input instance?" The differentiable proxy adjacency matrix is prevalent in current counterfactual subgraph discovery studies due to its ability to avoid exhaustive edge searching. However, a prediction gap exists when feeding the proxy matrix with continuous values and the thresholded discrete adjacency matrix to GNNs, compromising the optimization of the subgraph generator. Furthermore, the end-to-end learning schema adopted in the subgraph generator limits the diversity of counterfactual subgraphs. To this end, we propose CF-GFNExplainer, a flow-based approach for learning counterfactual subgraphs. CF-GFNExplainer employs a policy network with a discrete edge removal schema to construct counterfactual subgraph generation trajectories. Additionally, we introduce a reward and loss function designed to guide CF-GFNExplainer's optimization. The discrete adjacency matrix generated in each trajectory eliminates the prediction gap, enhancing the validity of the learned subgraphs. Furthermore, the multi-trajectories sampling strategy adopted in CF-GFNExplainer results in diverse counterfactual subgraphs. Extensive ex-

*Corresponding author
Email address: liliu@cqupt.edu.cn (Li Liu)

periments conducted on synthetic and real-world datasets demonstrate the effectiveness of the proposed method in terms of validity and diversity.

Keywords: Graph neural networks, Model Explanation, Counterfactual Subgraph, Generative Flow Network.

1. Introduction

Graph Neural Networks (GNNs) [1, 2] have demonstrated remarkable performance in various fields, such as social networks [3, 4], drug discovery [5], and financial systems [6]. Despite their significant success, the architecture of GNNs, characterized by the stacking of multiple non-linear layers and the end-to-end training processes, results in lacking explainability of its prediction behaviors. This lack of explainability hinders the application of GNNs in fields such as finance and healthcare. Consequently, explaining the predictive behavior of GNNs has become one of the key challenges in the current development of GNNs.

Currently, a mainstream approach of explaining GNNs is to identify subgraph patterns that contribute most to the predictions. According to the purpose of the explainer, the identified patterns can be categorized into two types: factual subgraphs and counterfactual subgraphs. The factual subgraph is defined as a subgraph pattern that produces the same prediction label as the input graph. While the counterfactual subgraph is defined as subgraph pattern that, when the counterfactual subgraph is absent, leads to a change in the prediction label. From a causal standpoint, counterfactual subgraph defines the "*necessity*" [7, 8] of the model's predictions. Counterfactual is considered the highest level [9] in the causal hierarchy as they address both association and intervention questions [9, 10]. Therefore, counterfactual explanations possess richer semantics, making the explanation more concise and easily understandable for the explained object [11, 12], and providing effective support for refining model behavior.

Specifically, the counterfactual explanation approach for GNNs aims to understand the predictive behavior by answering the question: "*How would the prediction change if a certain subgraph were absent in the input instance?*" For instance, in the context of drug design [13, 14], counterfactual subgraphs help identify the necessary modifications that can change the desired property of a molecular structure. This not only aids in designing molecules with specific properties but also facilitates the understanding of molecular structures associated with these properties, which can be instrumental in the effective treatment of particular diseases [15, 16, 17].

In practice, counterfactual subgraph discovery of GNNs involves identifying subgraphs obtained by perturbing the input graph's structure to induce changes in the original prediction. However, this task presents a challenging combinatorial optimization problem with large graphs due to the exponential growth in the number of potential perturbations. To this end, some recent studies [8, 18, 19] learn counterfactual subgraph generators using differentiable proxy adjacency matrices. Although these methods avoid the combinatorial explosion issue during counterfactual subgraph generation, they still suffer from the following challenges:

1) Prediction gap exists when feeding the proxy matrix with continuous values and the thresholded discrete adjacency matrix of the subgraph. To avoid the exhaustive search for a large number of edge perturbations, prior studies [8, 18, 19] have adopted differentiable proxy adjacency matrices to facilitate the training of counterfactual generators. These approaches leverage a continuous-valued proxy matrix and subsequently apply thresholding to obtain the adjacency matrix of the learned subgraph for loss calculation and counterfactual graph generation. However, as illustrated in Figure 1, it cannot be guaranteed that the proxy adjacency matrix G^{pCF} and the thresholded adjacency matrix G^{tCF} yield identical prediction labels, resulting in a prediction gap. This prediction gap negatively impacts the loss calculation and parameter optimization of the generator, thereby compromising the validity of the generated counterfactual subgraph discovery.

2) The end-to-end optimization process limits the diversity of dis-

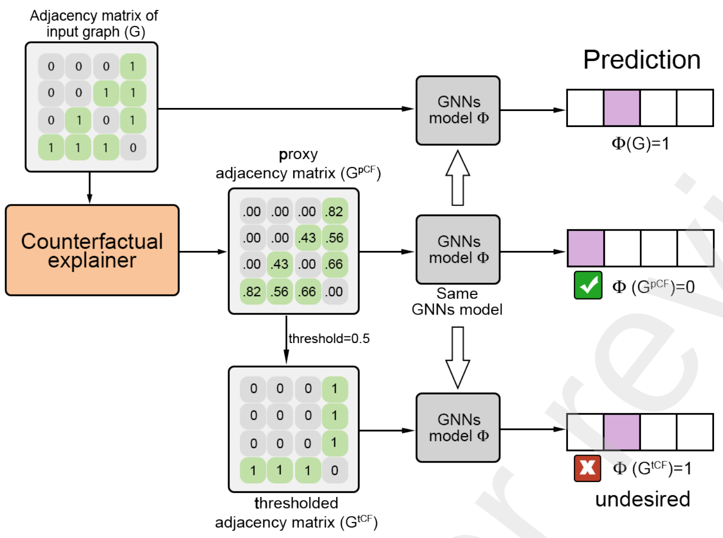

Figure 1: A toy example between differentiable proxy and thresholded adjacency matrix generated by threshold=0.5.

covered counterfactual subgraphs. In practice, a given graph structure typically contain multiple counterfactual subgraphs. These subgraphs represent different combinations of removed edges that lead to predictions distinct from the original ones, as illustrated by S_6 and S_7 in Figure 2. However, existing counterfactual discovery approaches [8, 18, 19] that employ end-to-end learning frameworks that directly optimize the input proxy adjacency matrix, ignoring the intermediate trajectories that may produce multiple counterfactual subgraphs. Thus, the lack of step-wise learning consideration in these approaches limits the diversity of discovered counterfactual subgraphs.

To address the challenges mentioned above, we propose a flow-based counterfactual subgraph generator, namely CF-GFNExplainer. Unlike previous counterfactual subgraph discovery that use end-to-end learning schema, CF-GFNExplainer generates counterfactual subgraphs through a step-wise process of sampling discrete edges. To effectively model the edge removal process, we propose a policy network that can step-wise generate a removing edge trajectory with probabilities proportional to a sophisticatedly designed reward function for counterfac-

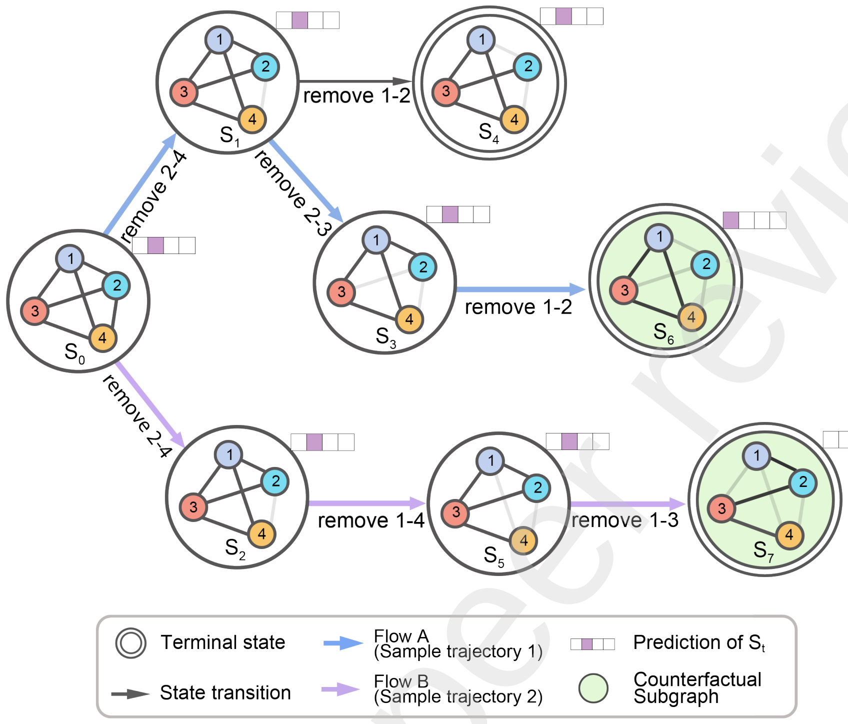

Figure 2: This is a step-wise sampling process. The initial state S_0 corresponds to the original input graph. State transition from one state to another corresponds to action removing an edge from previous state. The found counterfactual graphs are in the *green* box, i.e., S_6 and S_7. In the top right of each state is the prediction of the GNNs. Flow A and Flow B correspond to trajectories of generating the counterfactual subgraphs S_6 and S_7 by removing an edge.

tual subgraphs. Furthermore, the multi-trajectories sampling strategy adopted in CF-GFNExplainer ensures the generation of diverse counterfactual subgraphs during this process.

To summarize, this paper targets explaining GNNs from a counterfactual perspective. We address two challenges that affect the validity and diversity of counterfactual subgraph generation: the prediction gap between proxy continuous and thresholded discrete adjacency matrices, and the end-to-end learning schema. We propose CF-GFNExplainer, a flow-based method that leverages a suitable policy network with a discrete edge removal schema to generate counterfactual subgraphs. We further introduce a sophisticatedly designed reward and loss function to guide the optimization of CF-GFNExplainer. Extensive experiments on synthetic and real-world datasets demonstrate that the proposed method achieves superior performance in terms of validity and diversity.

2. Related works

2.1. Explanation of Graph Neural Networks

The prior studies of explaining GNNs can be categorized into model-level explanation and instance-level explanation [20]. Model-level explanation methods [21, 22, 23, 24, 25] aim to provide a high-level decision-making process of the GNN model. Among them, XGNN [21] finds graph patterns that maximize a specific prediction using reinforcement learning. GNNInterpreter [22] learns a graph distribution to explore the discriminative and representative patterns towards the target class. Model-level explanation methods may generate graph patterns that do not exist in the input instances. In contrast, instance-level explanation methods [26, 27, 28, 29, 30] explain the model by identifying the most important input subgraphs or sub-features for the model's predictions. GNNExplainer [26] learns continuous values mask by maximizing the mutual information between the predictions of the original graph and the subgraph/sub-features. PGExplainer [27] parameterizes the generation process of explanations using deep neural networks. Instance-level explanation methods provide explanations for individual instances based on real input instances, making them easier to understand.

These explanations primarily focused on explaining a trained GNN, known as post-hoc explanations. In addition to the post-hoc explanations, self-explainable models aim to provide the explanations along with their prediction. For instance, ProtGNN [31] learns some prototypes representations for each label, and its predictions are obtained by comparing the input graph with the learned prototype. Nevertheless, self-explainable models typically depend heavily on model architecture, resulting in weak generalization. Furthermore, the optimization of an explainer affects the model's training process, which is why we tend to favor post-hoc explanation methods.

Both of the aforementioned methods are factual explanations, while counterfactual explanations are also attracting increasing attention. A counterfactual explanation is typically described in the form of a causal statement: "If X had not occurred, Y would not have occurred [32]." It addresses both association and intervention questions [10]. Therefore, counterfactual explanations possess richer semantics. Recently, several studies [8, 18, 19, 33] attempt to explain GNNs using counterfactual reasoning. CF-GNNExplainer [18] trains a proxy matrix and sets a threshold to remove edges, causing a change in node classification. RCExplainer [19] aims to find explanations within the decision boundary to generate robust counterfactuals. CFF [8] generates factual explanations by balancing factual and counterfactual reasoning. These methods optimize a differentiable proxy matrix to obtain a continuous-valued adjacency matrix and then derive the counterfactual subgraph by thresholding it, the prediction gap between the proxy matrix and the thresholded matrix may result in the low validity of the learned counterfactual subgraphs. This issue motivates us to construct counterfactual graph generator by discrete edge sampling.

2.2. Generative Flow Networks

GFlowNet [34, 35] is a discrete and compositional object generation model, initially applied in the field of drug design, where it utilizes the characteristics of GFlowNet to generate new molecules with desired drug properties. The GFlowNet is initially inspired by temporal-difference algorithms [36] in reinforcement learning (RL). It involves three components: state, action (state transition), and reward. The key idea is to adopt a policy network that samples the next action to generate a sampled trajectory through a series of trainable steps. This sampled trajectory can be considered as a pipeline, where flow is going into an initial state and going out from the terminal state. The flow of the terminal state is the total flow and is equivalent to the reward function at that state. In contrast to RL an optimal policy that maximizes the reward, GFlowNet aims to sample combinatorial objects with probabilities proportional to the reward.

GFlowNet provides a unified framework [37] for generative models by adopting suitable policy networks for downstream tasks. Its objective is to train a policy network that can sample combinatorial objects with probabilities proportional to a given reward function. Recently, GFlowNet is also applied in Bayesian structure learning with the objective of generating a directed acyclic graph (DAG) [38].

Our proposed CF-GFNExplainer is an extension of GFlowNet. We have customized GFlowNet to explain graph neural networks by generating counterfactual subgraphs. Unlike the commonly adopted approach of adding nodes from an initial empty state in other works based on GFlowNet [34, 38, 39, 40], we employ an edge removal scheme to facilitate the discovery of counterfactual graphs. Additionally, we introduce a sophisticatedly designed reward and loss function for training a policy network to model each trainable step.

3. Preliminary and Methodology

3.1. Preliminary

Graph and Subgraph. Let $G = (\mathcal{E}, \mathcal{V})$ denotes a graph with edges $\mathcal{E}$ and nodes $\mathcal{V}$ that are associated with d-dimensional node features. A graph corresponds to the initial state S_0 in Figure 2. A subgraph $G^i = (\mathcal{E}^i, \mathcal{V}^i) \in G$ corresponding to a specific state S_i.

Graph Neural Networks. Let $\Phi(A, X; W) = Y$ be a GNN model, where Y is the predicted label, A is an $n \times n$ adjacency matrix. And $X = \{x_1, ..., x_n\}$, where $x_i \in \mathbb{R}^d$ is feature vector corresponding to each node, and W is the learnable parameters of Φ.

Counterfactual Subgraph Generator. A counterfactual subgraph G^{CF} is a subgraph within an input graph G such that the prediction $\Phi(G) \neq \Phi(G \backslash G^{CF})$. We propose to train a flow-based generator $g : g(G) = G \backslash G^{CF}$, such that the

G^{CF} removed in the generation process can be leveraged to explain GNNs from the counterfactual perspective.

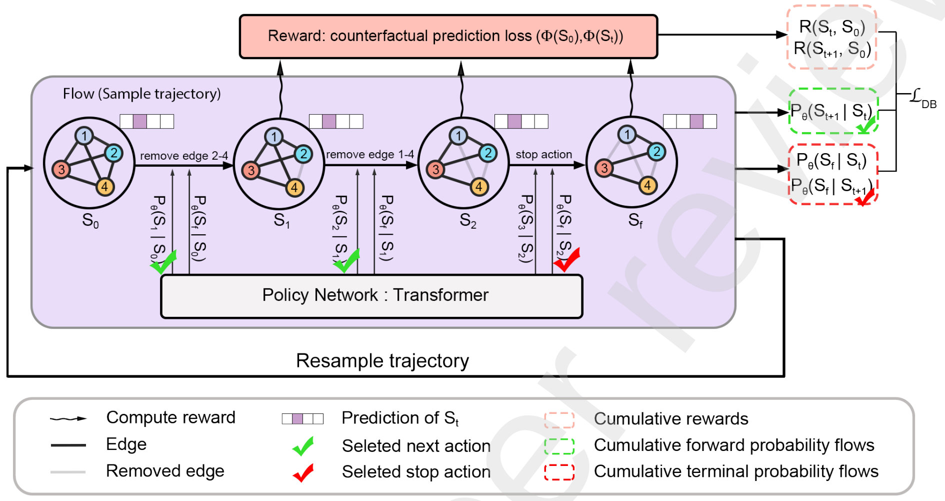

Figure 3: Counterfactual explainer Generator g. A trajectory of generating a counterfactual subgraph is guided by the policy network. The initial state S_0 is the input graph. Resampling another trajectory while sampling terminal state S_f. GFlowNet cumulates probability flows and rewards for multiple sampling trajectories in the dashed box, which is used to compute the $detailed - balance\ loss$.

3.2. *Methodology: CF-GFNExplainer*

As illustrated in Figure 3, our proposed CF-GFNExplainer consists of three main components. The first is a flow that generates terminal state S_f (i.e., $G \backslash G^{CF}$). The second is a policy network that parameterizes the state transition probability in the flow. The last is tailored reward and loss function that guides the optimization of policy network.

3.2.1. *Learning a Flow*

In the proposed CF-GFNExplainer, a flow is defined as a non-negative function $F(\cdot)$, which helps to measure the state transition probability on a trajectory. As illustrated in Figure 3, a sequence of states $\tau = (S_0, S_1, ..., S_n, S_f)$ forms a trajectory, where S_0 represents the initial state corresponding to the original input graph G, the terminal state S_f corresponds to a counterfactual subgraph by removing edges from G: $S_f = G \backslash G^{CF}$.

A flow network is defined as a triple ($\mathcal{S}$, $\mathcal{A}$, F) with a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, and multiple flows F. For the definition of $F(\cdot)$, $F(S_t \rightarrow S_{t+1})$ corresponds to edge flow, and the state flow $F(S_{t+1})$ is sum of all edge flows going into that state, denoted as $F(S_{t+1}) = \sum_{S_t \in Pa(S_{t+1})} F(S_t \rightarrow S_{t+1})$, $Pa(S_{t+1})$ is all parent states of state S_{t+1}.

Based on this flow network, the correlation of state transition probability with flow is defined as:

$$P_\theta(S_{t+1}|S_t) = \frac{F_\theta(S_t \rightarrow S_{t+1})}{F_\theta(S_t)} \tag{1}$$

and when subsequent state S_{t+1} equals a terminal state S_f, i.e. $S_{t+1} = S_f$, we have:

$$\begin{aligned} P_\theta\left(S_f \mid S_t\right) &= \frac{F_\theta\left(S_t \rightarrow S_f\right)}{F_\theta(S_t)} = \frac{R\left(S_t\right)}{F_\theta\left(S_t\right)} \\ \Leftrightarrow \quad F_\theta\left(S_t\right) &= \frac{R\left(S_t\right)}{P_\theta\left(S_f \mid S_t\right)} \end{aligned} \tag{2}$$

Then a flow with a terminal state $S_f = G \backslash G^{CF}$ is called a terminal flow. The goal of the proposed CF-GFNExplainer is to learn a flow to match terminal flow. Generally, it has been pointed out that the matching can be achieved when *flow-matching condition* is satisfied [34, 35], shown in Eq. (3).

$$\sum_{S_t \in Pa(S_{t+1})} F_\theta(S_t \rightarrow S_{t+1}) = \sum_{S_{t+2} \in Ch(S_{t+1})} F_\theta(S_{t+1} \rightarrow S_{t+2}) + R(S_{t+1}) \tag{3}$$

where $F_\theta(S_t \rightarrow S_{t+1})$ represents edge flow from state $S_t \rightarrow S_{t+1}$, $Pa(S_t)$ represents all parent states of state S_t, and $Ch(S_{t+1})$ represents the child states of state S_{t+1}. In other words, the total flow going into state S_{t+1} is equal to the flow going out S_{t+1} and residual reward $R(S_{t+1})$ (refer to Eq. 6 for details).

However, as the size of the input graph increases, the number of $Pa(S_t)$ also increases, and the flow becomes larger[34]. It poses challenges to find the terminal flow and estimate F_θ. To this end, we utilize *detailed-balance condition* comes from the Markov chain as an alternative solution. Specifically, we utilize parameterized transition probability $P_\theta(S_t|S_{t+1})$ and backward transition probability $P_B(S_{t+1}|S_t)$ to match the terminal flow, where P_B represents the prior distribution of all parent states of S_{t+1}. The *detailed-balance condition* is defined as Eq. (4):

$$F_\theta(S_t)P_\theta(S_{t+1}|S_t) = F_\theta(S_{t+1})P_B(S_t|S_{t+1}) \tag{4}$$

When Eq. (4) is satisfied for all transition states $S_t \to S_{t+1}$, we can generate a sampling trajectory with a probability of $P_\theta(S_{t+1}|S_t)$ to sample the next action. Then the transition probability $P_\theta(S_{t+1}|S_t) \propto R(S_{t+1})$. We fix P_B to a uniform distribution, making P_θ as the only parameter to optimize.

3.2.2. Policy Network to Parameterize State Transition Probability

We train a policy network to parameterize the state transition probability P_θ. As shown in Figure 4, the policy network consists of two neural networks: $P_\theta(S_{t+1}|S_t, \neg S_f)$ and $P_\theta(S_f|S_t)$.

- The first neural network is to model the transition probabilities $P_\theta(S_{t+1}|S_t, \neg S_f)$ from state S_t to the next state S_{t+1}. This network is to remove an edge from S_t and obtain a new subgraph S_{t+1}.
- The second neural network is to model the transition probability $P_\theta(S_f|S_t)$ from state S_t to the terminal state S_f. This network focuses on the probabilities of transitioning toward terminal state.

As depicted in Figure 4, we construct policy network using the *Linear Transformer* [41] due to two reasons. The first is that it can well model structural information. The second is that it is a pre-trained model that can facilitate multiple outputs via fine-tuning. Specifically, the input of policy network consists of the adjacency matrix A and feature matrix X of the input graph, and the output is two state probabilities $P_\theta(S_{t+1}|S_t, \neg S_f)$ and $P_\theta(S_f|S_t)$. The linear

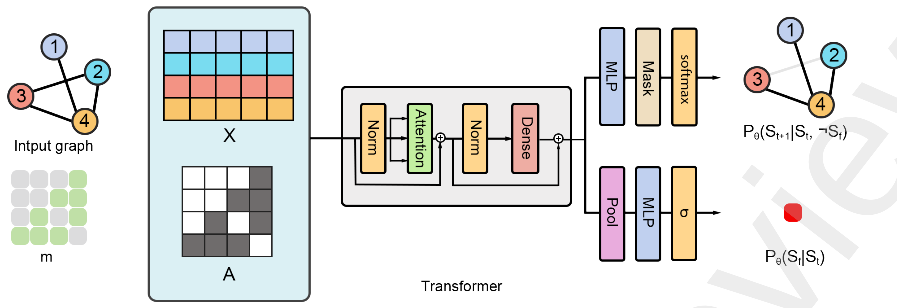

Figure 4: The policy network architecture of the state transition probability $P_\theta(S_{t+1}|S_t)$. The input is adjacency matrix of an example and is fed into a Linear Transformer, with two separate output heads. The first head (above) gives the probability to remove an edge $P_\theta(S_{t+1}|S_t, \neg S_f)$, using the mask m to filter out invalid actions. The second head (below) gives the probability to terminate the trajectory $P_\theta(S_f|S_t)$.

attention mechanism in the transformer is defined as:

$$Q = xW_Q \quad K = xW_K \quad V = xW_V$$
$$\text{LinearAtten}_k(x) = \frac{\sum_{j=1}^{J} \left(\phi\left(Q_k\right)^\top \phi\left(K_j\right)\right) V_j}{\sum_{j=1}^{J} \phi\left(Q_k\right)^\top \phi\left(K_j\right)} \tag{5}$$

where x denotes the input of the linear attention layer, while ϕ denotes a non-linear feature mapping, J denotes the size of the input, and Q, K, and V refer to linear transformations of x, each corresponding respectively to the queries, keys, and values.

In the counterfactual subgraph generation, we train the policy network to generate multi-trajectories. Each trajectory starts from the initial state S_0, and the next action is guided by transition probability $P_\theta(S_{t+1}|S_t, \neg S_f)$ to obtain a new subgraph. When the policy network samples stop action (e.g., $P_\theta(S_f|S_2)$ in Figure 3), we resample another trajectory from the initial state S_0.

3.2.3. Reward and Loss Function

We define cross-entropy loss as the reward. It calculates the distance between the generated counterfactual subgraph S_f and the original input graph S_0, as shown in Eq. (6).

$$R(S_f, S_0) = exp\left(\mathcal{L}\left(\Phi\left(S_f\right), \Phi\left(S_0\right)\right)\right) \tag{6}$$

where $\mathcal{L}$ is the cross-entropy loss function, S_f is the counterfactual subgraph generated for an instance and S_0 is the initial state of that counterfactual subgraph. Φ is the trained GNNs. We use the exponential term here to avoid negative reward.

After defining the reward function, we combine Eq. (2) and Eq. (4) together. Then the *detailed-balance condition* can be rewritten as Eq. (7).

$$R\left(S_t, S_0\right) P_\theta\left(S_{t+1} \mid S_t\right) P_\theta\left(S_f \mid S_{t+1}\right) = R\left(S_{t+1}, S_0\right) P_B\left(S_t \mid S_{t+1}\right) P_\theta\left(S_f \mid S_t\right) \tag{7}$$

when condition Eq. (7) is satisfied, g can generate a sampling trajectory with a probability $P_\theta(S_{t+1}|S_t) \propto R(S_{t+1})$ parameterized by P_θ.

To fit the parameters θ of the state transition probability, we can minimize the non-linear least squares objective for all the transitions $S_t \rightarrow S_{t+1}$, shown in Eq. (8):

$$\mathcal{L}_{DB}(\theta) = \sum_{S_t \rightarrow S_{t+1}}^{N} \left[\log \frac{R\left(S_{t+1}, S_0\right) P_B\left(S_t \mid S_{t+1}\right) P_\theta\left(S_f \mid S_t\right)}{R(S_t, S_0) P_\theta\left(S_{t+1} \mid S_t\right) P_\theta\left(S_f \mid S_{t+1}\right)}\right]^2 \tag{8}$$

where N is the number of transition states $S_t \rightarrow S_{t+1}$ during the sampling trajectory, which is a hyperparameter. $R(S_t, S_0)$ is the reward function of the subgraph generated from the current state S_t, calculated by Eq. (6). P_B is fixed in a uniform distribution. This loss function cumulates the losses of all sampled transition states. Minimizing this loss function ensures that the reward of the generated counterfactual subgraphs is proportional to probabilities of the next action, facilitating the learning of the proposed CF-GFNExplainer. The overall algorithm is shown in Algorithm 1.

Algorithm 1 CF-GFNExplainer

```
Input: $\Phi$: trained GNN model; $g(\cdot)$: counterfactual subgraph generator; $\mathcal{V} = (A, X)$: a valid instance, adjacency matrix A and node features X; $N$: the number of transition states; $\mathcal{B}$: the number of batch size of transformer; $P$: the number of prefill; $\mathcal{H}$: ReplayBuffer; $\mathcal{A}$: the set of removing edge
1:  Initialize $S_0 = mask = A, \mathcal{H}$
2:  for $epoch$ in $N$ do
3:      $P_\theta(S_{t+1}|S_t), P_\theta(S_f|S_t) \leftarrow g(S_t, X; \theta)$
4:      $a_t = [a_{t,next}, a_{t,stop}] \sim P_\theta$
5:      if $a_t = a_{t,next}$ then
6:          Obtain the new Graph $S_{t+1}$ from $S_t$ remove $a_t$
7:      else
8:          $S_{t+1} = S_0$
9:      end if
10:     $\mathcal{A} = \mathcal{A} \cup \{a_t\}$
11:     if $\Phi(S_{t+1}) \neq \Phi(S_0)$ then
12:         $G^{CF} = \mathcal{A}$
13:     end if
14:     $P^{'}_\theta(S_{t+2}|S_{t+1})$, $P^{'}_\theta(S_f|S_{t+1}) \leftarrow g(S_{t+1}, X; \theta)$
15:     Calculate reward $R(S_t)$ and $R(S_{t+1})$ based on Eq. (6)
16:     Make a state transition: $S_t = S_{t+1}$
17:     Push $P_\theta, P^{'}_\theta, R$ into $\mathcal{H}$, update $\mathcal{H}$
18:     if $epoch \geq P$ then
19:         Sample $\mathcal{B}$ trajectories from $\mathcal{H}$
20:         Calculate $\nabla\mathcal{L}_{DB}$ based on Eq. (7)
21:         Update the parameters $\theta$
22:     end if
23: end for
Output: $G^{CF}$
```

4. Experiments

4.1. Datasets, Experimental Setup, Baselines and Metrics

The proposed CF-GFNExplainer aims to address two challenges that affect the validity and diversity of counterfactual explanation. To evaluate its effectiveness, we compare it with three state-of-the-art counterfactual explainers on six datasets.

4.1.1. Datasets

We evaluate our method on three synthetic datasets and three real-world datasets. The three synthetic datasets including BA-Shapes, Tree-Cycles, and Tree-Grids, which are introduced by GNNExplainer [26] and specifically designed for explaining node classification task. Each of them has ground-truth motif, which are houses, cycles, and grids structures. In addition, we randomly add 300 edges to the adjacency matrices of the Tree-Cycles and Tree-Grids datasets to evaluate the validity and diversity of our model. The three real-world datasets are Cora[42], Citeseer[43], and MUTAG [44]. Cora[42] and Citeseer[43] are citation networks for node classification, where ground-truth motifs are absent. MUTAG [44] is a chemical compounds dataset for graph classification, where nodes are different types of atoms, and edges are chemical bonds. It points out that molecules with NO_2 structures are more susceptible to ”mutagenesis” and are treated as the ground-truth explanations for the mutagenic compounds.

4.1.2. Experimental Setup

The experimental setup consists of two phases: a) Training a based GNNs model to be explained. b) Generating valid instances for validation.

Training GNNs to be explained . we train two GNN models for node classification and graph classification, respectively.

- **Node Classification.** We use the same GNN model experimental setup as CF-GNNExplainer. The GNN model trains a 3-layer GCN. For synthetic datasets, we split these datasets into training set (80%), testing set (20%). For Cora dataset, we use 140 nodes as train set and 1000 nodes as test set. For Citeseer dataset, we exclude isolated nodes from the Citeseer dataset during preprocessing, and we use 1333 nodes as train set and 415 nodes as test set. The testing accuracy is 97.86% , 72.00% , 77.33% , 82.10% and 79.09% on BA-Shapes, Tree-Cycles, Tree-Grids, Cora, and Citeseer datasets.

- **Graph Classification.** This model follows a similar architecture as GNN for node classification, with difference being the application of an additional max-pooling and mean-pooling layer as the readout function. The testing accuracy is 84.21% on MUTAG dataset.

Generating valid instance for validation. It is worth noticing that, given a trained GNN and testing instances, there existing instances can not yield counterfactual subgraph even exhaustively enumerate all edge combinations. Therefore, we filter the instances without counterfactual subgraphs to obtain a set of *valid* samples. Specifically, from node classification, we obtain 140, 175, 247, 424, and 371 valid nodes for BA-Shapes, Tree-Cycles, Tree-Grids, Cora dataset, and Citeseer datasets, respectively. From graph classification, we obtain 188 valid graphs for the MUTAG dataset[1]. Table 1 illustrates the details of the used datasets.

4.1.3. Baselines

Here we provide a brief description of the baselines[2] used in our evaluation.

[1] Due to the small number of testing instances, we add training instances for evaluation.

[2] The codes for these baselines can be found at https://github.com/a-lucic/cf-gnnexplainer, https://developer.huaweicloud.com/develop/aigallery/notebook/detail?id=e41f63d3-e346-4891-bf6a-40e64b4a3278, https://github.com/chrisjtan/gnn_cff.

Dataset	#of class	#of node/graph	#edge	#testing	#valid instance
BA-Shapes	4	700	2055	140	140
Tree-Cycles	2	871	1270	175	175
Tree-Grids	2	1231	2004	247	247
Cora	7	2708	5278	1000	424
Citeseer	6	3264	4536	415	371
MUTAG	2	188	3721	38	188

Table 1: Detailed statistics of the datasets.

- **CF-GNNExplainer** [18] is a counterfactual explainer that learns a differentiable continuous proxy matrix. It obtains the adjacency matrix of counterfactual subgraph via removing the edges with the corresponding elements less than or equal to 0.5 and setting the remains as 1.
- **RCExplainer** [19] learns robust counterfactual explanation by extracting decision boundaries of GNNs. It first trains a differentiable proxy matrix to ensure the prediction is faithfulness to the input graph. Then it removes the edges whose corresponding element in proxy matrix is less than 0.35 to produce the adjacency matrix of counterfactual subgraph.
- **CFF** [8] learns a GNN explainer by balancing factual and counterfactual reasoning. It first trains a differentiable proxy matrix that is consistent with the condition for factual reasoning. Then it removes the edges whose corresponding element in proxy matrix is less than 0.5 to produce the adjacency matrix of counterfactual subgraph.

4.1.4. Metrics

We use three metrics to evaluate our method as follows:

- **Validity** is defined as the proportion of instances with found counterfactual subgraphs out of the total number of instances:

$$Validity = \frac{1}{M}\sum_{i=1}^{M} \mid \mathbb{I}(\Phi(S_f^i) \neq \Phi(S_0^i)) \mid \tag{9}$$

where M is the number of instances, S_f^i is the counterfactual subgraph of the i-th instance, S_0^i is the input instance. Φ is a trained GNNs, and $\mathbb{I}$ is an indicator function that outputs 1 when the input is true, and 0 otherwise.

- **Diversity** is the number of counterfactual subgraphs obtained, denoted as $G^{CF} = \{G_1^{CF}, G_2^{CF}, ..., G_n^{CF}\}$. We filter out the *overlapped* subgraphs during the evaluation, where *overlapped* is defined as an inclusion relationship between the two subgraphs, i.e. $G_1^{CF} \subseteq G_2^{CF}$ or $G_2^{CF} \subseteq G_1^{CF}$. Then the retained subgraphs are non-redundant for the counting. Note that baseline models typically learn only one counterfactual subgraph during inference. To conduct a comparison of diversity, we record all intermediate subgraphs obtained during the optimization of the baseline as the learned subgraphs.

- **Sparsity** measures the proportion of edges in S_0 that are removed [20]. A value of 0 indicates edges are all removed in an instance.

4.2. Quantitative Analysis

We evaluate the performance of our method and baseline models in terms of counterfactual validity and diversity. Table 2 and Table 3 illustrate the comparison results for synthetic data and real-world data, respectively. According to the results, we have the following observations.

Method	BA-Shapes			Tree-Cycles			Tree-Grids		
	Valid.% ▲	Spars.% ▲	Diver. ▲	Valid.% ▲	Spars.% ▲	Diver. ▲	Valid.% ▲	Spars.% ▲	Diver. ▲
CF-GNNExplainer	64.29	**99.77**	1.00	53.14	<u>96.47</u>	1.00	<u>97.98</u>	**98.16**	1.00
RCExplainer	<u>85.00</u>	14.73	1.15	76.57	84.11	1.00	78.14	70.89	1.00
CFF	80.71	56.75	<u>1.27</u>	<u>82.29</u>	20.89	<u>1.06</u>	93.52	23.35	<u>1.02</u>
CF-GFNExplainer	**100.00**	<u>91.76</u>	**33.84**	**100.00**	**96.86**	**44.27**	**100.00**	<u>98.05</u>	**46.43**

Table 2: For the synthetic datasets, the results of comparing our method with CF-GNNExplainer, RCExplainer and CFF. Below each metric, ▲ indicates a higher value is desirable.

In terms of validity, it is observed that our proposed CF-GFNExplainer achieves 100% validity on four datasets (BA-Shapes, Tree-Cycles, Tree-Grids, and MUTAG). Particularly, our proposed CF-GFNExplainer outperforms the CFF by 70.21% in the validity for MUTAG. An exception is the Citeseer dataset, where CF-GFNExplainer falls short by 0.54% compared to CFF. Nevertheless, it still ranks as the second-best model among all models. This observation indicates that the binarily removing edge schema adopted in CF-GFNExplainer significantly benefits the precise identification of counterfactual subgraphs.

Furthermore, the proposed CF-GFNExplainer far exceeds other baselines on both the synthetic and the real-world datasets in terms of diversity. We believe it benefits from the multi-trajectories sampling strategy adopted in CF-GFNExplainer.

Method	Cora			Citeseer			MUTAG		
	Valid.% ▲	Spars.% ▲	Diver. ▲	Valid.% ▲	Spars.% ▲	Diver. ▲	Valid.% ▲	Spars.% ▲	Diver. ▲
CF-GNNExplainer	85.61	**97.92**	1.39	77.36	**97.85**	1.28	38.30	69.73	1.00
RCExplainer	82.55	35.77	1.03	55.26	21.92	1.01	73.40	90.45	4.26
CFF	89.62	57.66	1.31	**94.61**	67.20	1.52	29.79	40.17	1.00
CF-GFNExplainer	**98.58**	83.33	**37.20**	94.07	76.69	**25.30**	**100.00**	**92.88**	**48.55**

Table 3: For the real-world datasets, the results of comparing our method with CF-GNNExplainer, RCExplainer and CFF. Below each metric, ▲ indicates a higher value is desirable.

In terms of sparsity, our proposed CF-GFNExplainer is ranked either first or second. This implies that the size of the removed edge does not impact the discovery of counterfactuals, which further demonstrates that the differentiable matrix with thresholding learned by the baseline methods influences the generation of counterfactual subgraphs.

Overall, our approach demonstrates strong performance in terms of validity and diversity.

4.3. Study of Hyperparameter N

The hyperparameter N in Eq. (8) controls the number of states that can be sampled and also affects the validity and diversity of finding counterfactual subgraphs. Figure 5 illustrates the influence of N on CF-GFNExplainer when generating counterfactual subgraphs on both real-world and synthetic datasets, and the validity and diversity of the generated counterfactual subgraphs are positively correlated with N.

In terms of diversity, as shown in Figure 5 (b) and (c), the diversity score exceeds 70 on Tree-Cycles and Tree-Grids, which is significantly higher than other baselines. Moreover, as the number of state transitions increases, there is a corresponding escalation in diversity. This phenomenon illustrates the beneficial impact of adopting a multi-trajectories sampling strategy and highlights its positive contribution to enriching counterfactual diversity.

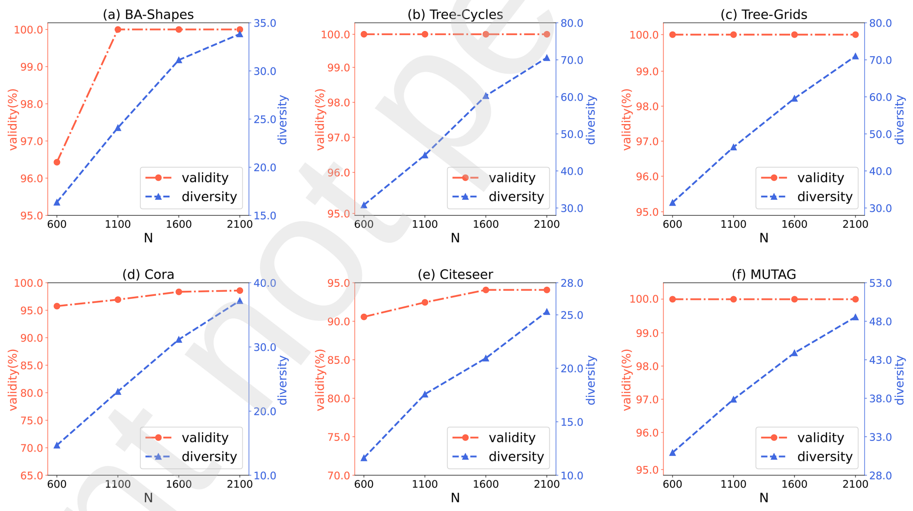

Figure 5: Analysis of Hyperparameter N

4.4. Case Study

In addition to quantitative analysis, we provide visual case studies to demonstrate the effectiveness of the proposed CF-GFNExplainer in terms of validity and diversity.

Figure 6 illustrates counterfactual subgraphs for two selected instances (id 935 in Tree-Grids and id 31 in MUTAG) that are generated by CF-GNNExplainer, RCExplainer, CFF, and the proposed CF-GFNExplainer. We notice that the CFF on the MUTAG dataset cannot find any counterfactual subgraph for the given instance; therefore, it is not presented. In contrast, our proposed CF-GFNExplainer successfully finds counterfactuals on both datasets. For the Tree-Grids dataset, we observe that the counterfactual subgraphs detected by RCExplainer and CFF are redundant. On the other hand, the subgraphs obtained by the proposed CF-GFNExplainer are all contained within the ground-truth motif and are sparser. We believe that GNNs can adopt motifs as the rationale for predictions, given that the patterns in synthetic datasets are often more distinct than the complex patterns found in real-world datasets. In the real-world dataset MUTAG, we observe that the subgraphs identified by CF-GFNExplainer better fit the ground-truth motif, such as NO_2, which has been shown to have a high likelihood of being mutagenic [45]. This indicates the precision of CF-GFNExplainer in identifying subgraphs accurately. Additionally, it is worth noticing that the CF-GFNExplainer identifies 40 counterfactual subgraphs for the prediction where the existing subgraph pattern differs from NO_2. This also takes into consideration the possibility that GNNs might not predict according to human-intended rationales.

We further present a case to analyze the reasons behind the limited performance of the baselines. Figure 7 illustrates a real case learned by CFF in the Tree-Grids dataset. We observe that a prediction gap exists between the proxy matrix G^{pCF} and the thresholded adjacency matrix G^{tCF}, leading to different prediction labels. We believe that the schema of thresholding the differentiable proxy matrix introduces a negative impact on loss gradients and optimization, thereby affecting the validity of counterfactual discovery. Note that this gap is not specific to this case; it also exists in CF-GNNExplainer and RCExplainer. Here, we provide the case of CFF for illustrative purposes.

In terms of diversity, we observe that CF-GNNExplainer, RCExplainer, and

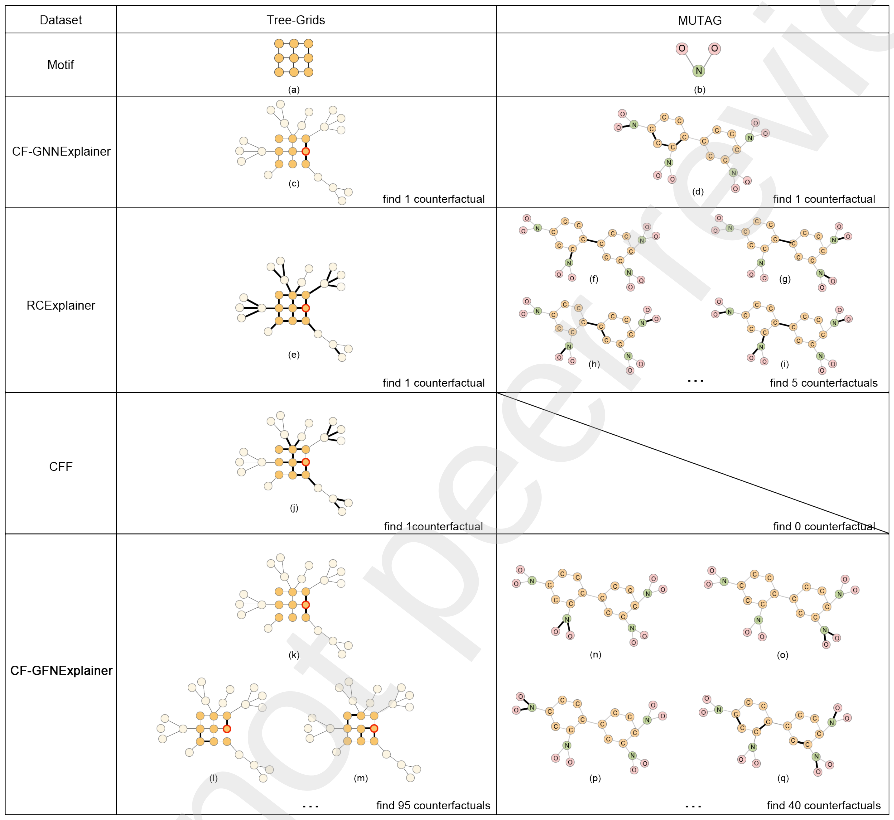

Figure 6: Explanation of CF-GNNExplainer, RCExplainer, CFF, and CF-GFNExplainer on Tree-Grids and MUTAG datasets. The counterfactual explanations are highlighted in black and the node highlighted in red is being explained. Left column corresponds to node classification and the right column corresponds to graph classification. CFF on MUTAG doesn't yield any counterfactual explanation and is not presented. The number in the bottom-right corner is the number of counterfactuals removing overlapping.

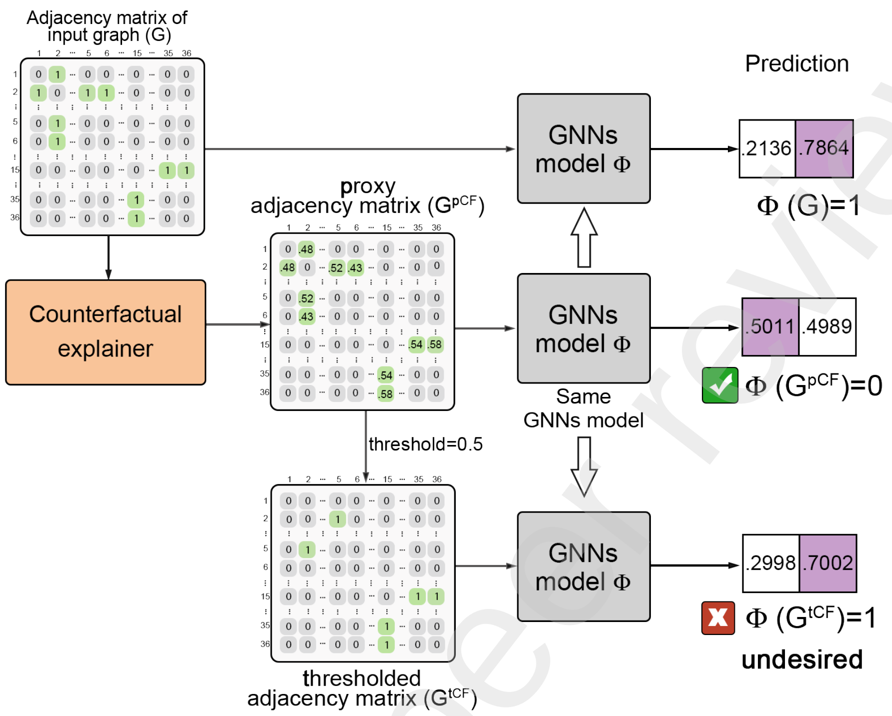

Figure 7: A real case study about the prediction gap in Tree-Grids dataset. The counterfactual explainer is CFF, and Φ is trained GNNs.

CFF generate only one counterfactual subgraph on the Tree-Grids dataset. In contrast, our proposed CF-GFNExplainer generates 95 counterfactual subgraphs. Here, we present only a selection of representative subgraphs due to space constraints. Similarly, on the MUTAG dataset, CF-GNNExplainer generates only one counterfactual, while RCExplainer generates 5 counterfactuals, and our method generates 40 counterfactuals. We present four selected counterfactual subgraphs due to space limitations. Overall, our method generates a significantly greater number of counterfactuals compared to these baselines.

Next, we provide a detailed study to analyze the reason for generating diverse subgraphs, i.e., the multi-trajectories sampling strategy. In Figure 8 and Figure 9, we showcase the counterfactual subgraphs generated by our CF-

GFNExplainer on the Tree-Grids and MUTAG datasets using the multi-trajectories sampling strategy. Each trajectory represents an ordered edge sequence obtained through step-wise edge removal guided by the policy network. We visualize the changing trend of the GNN's predicted probabilities during the counterfactual generation process. Specifically, for each step of edge sampling, we illustrate the corresponding predicted probabilities. Based on Figure 8 and Figure 9, we have the following observations.

In Figure 8, it is observed that three trajectories generate distinct subgraph patterns that induce a different predicted label compared to the original ones. This phenomenon provides evidence of the effectiveness of the multi-trajectories sampling strategy. Furthermore, according to the states $S_0 \to S_4$ in Trajectory 3, we observe that removing edges on the motif significantly affects the prediction probabilities. This observation further confirms the significance of motifs on synthetic datasets, aligning with the viewpoint that motifs play a crucial role in the prediction of GNNs.

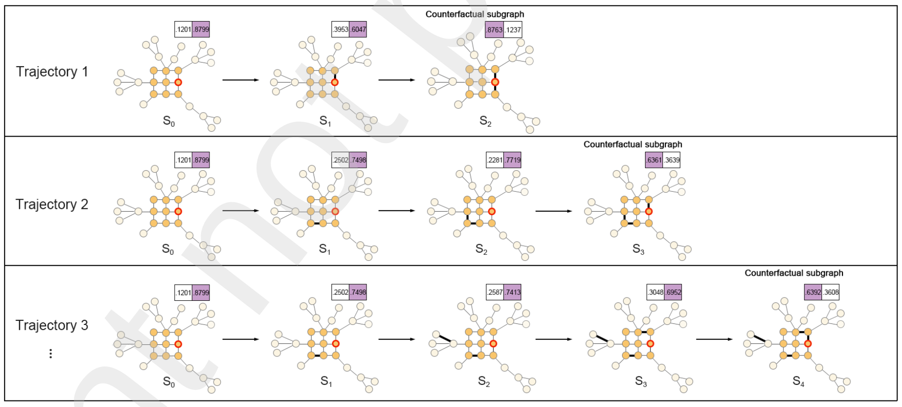

Figure 8: Multi-trajectories sampling strategy in CF-GFNExplainer on Tree-Grids dataset. The counterfactual explanations are highlighted in black. The found counterfactual subgraphs are at the end of each trajectory (i.e. S_2, S_3 and S_4). In the top right of each state is the prediction of the GNNs, along with the displayed changing trend of probabilities.

In Figure 9, we make similar observations as compared to the synthetic dataset Tree-Grids; that is, CF-GFNExplainer can still generate distinct subgraph patterns in the real-world dataset. Additionally, we observe that removing a single edge within a motif does not significantly impact the GNN's prediction. However, removing the entire NO_2 group leads to a drastic change in the predicted probability. Apart from perturbing edges within motifs, we also observe that perturbing edges outside of motifs can also lead to prediction changes in Trajectory 2 and Trajectory 3. The reason is that treating the sub-structure (NO_2) as ground-truth explanations is not rigorous, as it is reported that only 76% of the compounds with NO_2 are mutagenic [46].

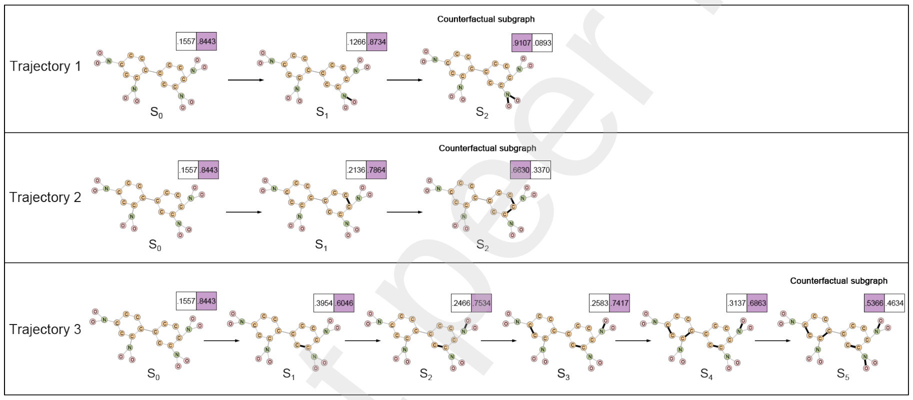

Figure 9: Multi-trajectories sampling strategy CF-GFNExplainer on MUTAG dataset. The counterfactual explanations are highlighted in black. The found counterfactual subgraphs are at the end of each trajectory (i.e. S_2 and S_5). In the top right of each state is the prediction of the GNNs, along with the displayed changing trend of probabilities.

To summarize, the observations of case studies are consistent with the *quantitative analysis* provided in subsection 4.2. The proposed CF-GFNExplainer demonstrates its superior ability to generate valid and diverse counterfactual subgraphs, addressing the issues of the prediction gap and limited diversity that exist in current counterfactual GNN explainers.

5. Conclusion

In this paper, we investigate explaining GNN predictions from a counterfactual perspective. We discuss the limitations on the validity and diversity of counterfactual subgraphs discovery, which arise from the adoption of proxy matrices and the end-to-end learning approach. To address these issues, we propose CF-GFNExplainer, a flow-based generator that generates counterfactual explanations through a step-wise process of removing edges and a multi-trajectories sampling strategy. Specifically, we design an edge removal scheme and a sophisticated reward and loss function to guide the explainer's learning. We conduct experiments on both node classification and graph classification tasks. Comparison results on both synthetic and real-world datasets demonstrate the effectiveness of the proposed CF-GFNExplainer in terms of validity and diversity. For future work, we are interested in enhancing the generation efficiency of CF-GFNExplainer.

Acknowledgment

The work is partially supported by National Natural Science Foundation of China (61936001, 62221005, 61806031), and in part by the Natural Science Foundation of Chongqing (cstc2019jcyj-cxttX0002, cstc2020jcyj-msxmX0943), and in part by the key cooperation project of Chongqing Municipal Education Commission (HZ2021008), and in part by Doctoral Innovation Talent Program of Chongqing University of Posts and Telecommunications (BYJS202118).

References

[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018, pp. 1–12.
URL https://openreview.net/forum?id=rJXMpikCZ

[2] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017, pp. 1–10.
URL https://openreview.net/forum?id=SJU4ayYgl

[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems 32 (1) (2021) 4–24. doi:10.1109/TNNLS.2020.2978386.

[4] E. Cho, S. A. Myers, J. Leskovec, Friendship and mobility: User movement in location-based social networks, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA, 2011, p. 1082–1090. doi:10.1145/2020408.2020579.
URL https://doi.org/10.1145/2020408.2020579

[5] A. Rassil, H. Chougrad, H. Zouaki, Deep multi-agent fusion q-network for graph generation, Knowledge-Based Systems 269 (2023) 110509. doi:https://doi.org/10.1016/j.knosys.2023.110509.
URL https://www.sciencedirect.com/science/article/pii/S0950705123002599

[6] D. V. Carvalho, E. M. Pereira, J. S. Cardoso, Machine learning interpretability: A survey on methods and metrics, Electronics 8 (8). doi:10.3390/electronics8080832.
URL https://www.mdpi.com/2079-9292/8/8/832

[7] J. Pearl, Causality: Models, Reasoning and Inference, 2nd Edition, Cambridge University Press, USA, 2009.

[8] J. Tan, S. Geng, Z. Fu, Y. Ge, S. Xu, Y. Li, Y. Zhang, Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning, in: Proceedings of the ACM Web Conference 2022, Association for Computing Machinery, New York, NY, USA, 2022, p. 1018–1027.

doi:10.1145/3485447.3511948.
URL https://doi.org/10.1145/3485447.3511948

[9] J. Pearl, Theoretical impediments to machine learning with seven sparks from the causal revolution, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 3. doi:10.1145/3159652.3176182.
URL https://doi.org/10.1145/3159652.3176182

[10] J. Pearl, D. Mackenzie, The Book of Why: The New Science of Cause and Effect, 1st Edition, Basic Books, Inc., USA, 2018.

[11] R. Moraffah, M. Karami, R. Guo, A. Raglin, H. Liu, Causal interpretability for machine learning-problems, methods and evaluation, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Vol. 22, Association for Computing Machinery, New York, NY, USA, 2020, p. 18–33. doi:10.1145/3400051.3400058.
URL https://doi.org/10.1145/3400051.3400058

[12] K. Sokol, P. A. Flach, Counterfactual explanations of machine learning predictions: Opportunities and challenges for AI safety, in: Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019, Vol. 2301 of CEUR Workshop Proceedings, CEUR-WS.org, 2019, pp. 1–4.
URL https://ceur-ws.org/Vol-2301/paper_20.pdf

[13] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, et al., A deep learning approach to antibiotic discovery, Cell 180 (4) (2020) 688–702. doi:https://doi.org/10.1016/j.cell.2020.01.021.
URL https://www.sciencedirect.com/science/article/pii/S0092867420301021

[14] Y. Xie, C. Shi, H. Zhou, Y. Yang, W. Zhang, Y. Yu, L. Li, Mars: Markov molecular sampling for multi-objective drug discovery, in: International Conference on Learning Representations, 2021, pp. 1–12.
URL https://openreview.net/forum?id=kHSu4ebxFXY

[15] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, T. Langer, A compact review of molecular property prediction with graph neural networks, Drug Discovery Today: Technologies 37 (2020) 1–12. doi:https://doi.org/10.1016/j.ddtec.2020.11.009.
URL https://www.sciencedirect.com/science/article/pii/S1740674920300305

[16] Z. Guo, C. Zhang, W. Yu, J. Herr, O. Wiest, M. Jiang, N. V. Chawla, Few-shot graph learning for molecular property prediction, in: Proceedings of the Web Conference 2021, Association for Computing Machinery, New York, NY, USA, 2021, p. 2559–2567. doi:10.1145/3442381.3450112.
URL https://doi.org/10.1145/3442381.3450112

[17] C. Q. Nguyen, C. Kreatsoulas, K. M. Branson, Meta-learning gnn initializations for low-resource molecular property prediction, in: 4th Lifelong Machine Learning Workshop at ICML 2020, 2020, pp. 1–6.
URL https://openreview.net/forum?id=MQ_t7LRvsW

[18] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, F. Silvestri, Cf-gnnexplainer: Counterfactual explanations for graph neural networks, in: Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, Vol. 151 of Proceedings of Machine Learning Research, PMLR, 2022, pp. 4499–4511.
URL https://proceedings.mlr.press/v151/lucic22a.html

[19] M. Bajaj, L. Chu, Z. Y. Xue, J. Pei, L. Wang, P. C.-H. Lam, Y. Zhang, Robust counterfactual explanations on graph neural networks, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, Vol. 34, Curran

Associates, Inc., 2021, pp. 5644–5655.
URL https://proceedings.neurips.cc/paper_files/paper/2021/file/2c8c3a57383c63caef6724343eb62257-Paper.pdf

[20] H. Yuan, H. Yu, S. Gui, S. Ji, Explainability in graph neural networks: A taxonomic survey, IEEE Transactions on Pattern Analysis & Machine Intelligence 45 (05) (2023) 5782–5799. doi:10.1109/TPAMI.2022.3204236.

[21] H. Yuan, J. Tang, X. Hu, S. Ji, Xgnn: Towards model-level explanations of graph neural networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA, 2020, p. 430–438. doi:10.1145/3394486.3403085.
URL https://doi.org/10.1145/3394486.3403085

[22] X. Wang, H. W. Shen, GNNInterpreter: A probabilistic generative model-level explanation for graph neural networks, in: The Eleventh International Conference on Learning Representations, 2023, pp. 1–11.
URL https://openreview.net/forum?id=rqq6Dh8t4d

[23] Y.-M. Shin, S.-W. Kim, E.-B. Yoon, W.-Y. Shin, Prototype-based explanations for graph neural networks (student abstract), Proceedings of the AAAI Conference on Artificial Intelligence 36 (11) (2022) 13047–13048. doi:10.1609/aaai.v36i11.21660.
URL https://ojs.aaai.org/index.php/AAAI/article/view/21660

[24] X. Kang, D. Liang, Q. Li, Ganexplainer: Explainability method for graph neural network with generative adversarial nets, in: Proceedings of the 2022 11th International Conference on Computing and Pattern Recognition, ICCPR '22, Association for Computing Machinery, New York, NY, USA, 2023, p. 297–302. doi:10.1145/3581807.3581850.
URL https://doi.org/10.1145/3581807.3581850

[25] H. Xuanyuan, P. Barbiero, D. Georgiev, L. C. Magister, P. Lió, Global

concept-based interpretability for graph neural networks via neuron analysis, arXiv preprint arXiv:2208.10609.

[26] Z. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, pp. 1–12.
URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf`

[27] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, X. Zhang, Parameterized explainer for graph neural network, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 19620–19631.
URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/e37b08dd3015330dcbb5d6663667b8b8-Paper.pdf`

[28] M. Vu, M. T. Thai, Pgm-explainer: Probabilistic graphical model explanations for graph neural networks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 12225–12235.
URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/8fb134f258b1f7865a6ab2d935a897c9-Paper.pdf`

[29] M. S. Schlichtkrull, N. D. Cao, I. Titov, Interpreting graph neural networks for {nlp} with differentiable edge masking, in: International Conference on Learning Representations, 2021, pp. 1–14.
URL `https://openreview.net/forum?id=WznmQa42ZAx`

[30] H. Yuan, H. Yu, J. Wang, K. Li, S. Ji, On explainability of graph neural networks via subgraph explorations, in: Proceedings of the 38th International Conference on Machine Learning, Vol. 139 of Proceedings of Machine

Learning Research, PMLR, 2021, pp. 12241–12252.
URL `https://proceedings.mlr.press/v139/yuan21c.html`

[31] Z. Zhang, Q. Liu, H. Wang, C. Lu, C. Lee, Protgnn: Towards self-explaining graph neural networks, in: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, AAAI Press, 2022, pp. 9127–9135.
URL `https://ojs.aaai.org/index.php/AAAI/article/view/20898`

[32] L. Schut, O. Key, R. Mc Grath, L. Costabello, B. Sacaleanu, M. Corcoran, Y. Gal, Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties, in: Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, Vol. 130 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 1756–1764.
URL `http://proceedings.mlr.press/v130/schut21a.html`

[33] D. Numeroso, D. Bacciu, Meg: Generating molecular counterfactual explanations for deep graph networks, in: 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–8. `doi:10.1109/IJCNN52387.2021.9534266`.

[34] E. Bengio, M. Jain, M. Korablyov, D. Precup, Y. Bengio, Flow network based generative models for non-iterative diverse candidate generation, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 27381–27394.
URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/e614f646836aaed9f89ce58e837e2310-Paper.pdf`

[35] Y. Bengio, S. Lahlou, T. Deleu, E. Hu, M. Tiwari, E. Bengio, Gflownet

foundations, in: arXiv preprint 2111.09266, 2021, 2021, pp. 1–76.
URL https://arxiv.org/abs/2111.09266

[36] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[37] D. Zhang, R. T. Q. Chen, N. Malkin, Y. Bengio, Unifying generative models with gflownets and beyond (2022). arXiv:2209.02606.

[38] T. Deleu, A. Góis, C. C. Emezue, M. Rankawat, S. Lacoste-Julien, S. Bauer, Y. Bengio, Bayesian structure learning with generative flow networks, in: The 38th Conference on Uncertainty in Artificial Intelligence, 2022, pp. 1–11.
URL https://openreview.net/forum?id=HElfed8j9g9

[39] W. Li, Y. Li, Z. Li, J. HAO, Y. Pang, DAG matters! GFlownets enhanced explainer for graph neural networks, in: The Eleventh International Conference on Learning Representations, 2023, pp. 1–12.
URL https://openreview.net/forum?id=jgmuRzM-sb6

[40] E. Laird, A. Madushanka, E. Kraka, C. Clark, Xinsight: Revealing model insights for gnns with flow-based explanations, in: The World Conference on eXplainable Artificial Intelligence, 2023, pp. 1–18.
URL https://arxiv.org/pdf/2306.04791.pdf

[41] A. Katharopoulos, A. Vyas, N. Pappas, F. Fleuret, Transformers are rnns: Fast autoregressive transformers with linear attention, in: Proceedings of the 37th International Conference on Machine Learning, ICML'20, JMLR.org, 2020, pp. 1–10.

[42] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Magazine 29 (3) (2008) 93. doi:10.1609/aimag.v29i3.2157.
URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2157

[43] C. L. Giles, K. D. Bollacker, S. Lawrence, Citeseer: An automatic citation indexing system, in: Proceedings of the Third ACM Conference on Digital Libraries, DL '98, Association for Computing Machinery, New York, NY, USA, 1998, p. 89–98. doi:10.1145/276675.276685.
URL https://doi.org/10.1145/276675.276685

[44] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann, Tudataset: A collection of benchmark datasets for learning with graphs, in: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020), 2020, pp. 1–11.
URL www.graphlearning.io

[45] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, C. Hansch, Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity, Journal of Medicinal Chemistry 34 (2) (1991) 786–797. arXiv:https://doi.org/10.1021/jm00106a046, doi:10.1021/jm00106a046.
URL https://doi.org/10.1021/jm00106a046

[46] E. Zeiger, J. Ashby, G. Bakale, K. Enslein, G. Klopman, H. Rosenkranz, Prediction of Salmonella mutagenicity, Mutagenesis 11 (5) (1996) 471–484. arXiv:https://academic.oup.com/mutage/article-pdf/11/5/471/3541870/11-5-471.pdf, doi:10.1093/mutage/11.5.471.
URL https://doi.org/10.1093/mutage/11.5.471