Published: 19 August 2025
Online AM: 01 October 2024
Accepted: 02 September 2024
Revised: 30 June 2024
Received: 28 December 2023

RESEARCH-ARTICLE

GCFExplainer: Global Counterfactual Explainer for Graph Neural Networks

MERT KOSAN, University of California, Santa Barbara, Santa Barbara, CA, United States

ZEXI HUANG, University of California, Santa Barbara, Santa Barbara, CA, United States

SOURAV MEDYA, University of Illinois at Chicago, Chicago, IL, United States

SAYAN RANU, Indian Institute of Technology Delhi, New Delhi, DL, India

AMBUJ KUMAR SINGH, University of California, Santa Barbara, Santa Barbara, CA, United States

GCFExplainer: Global Counterfactual Explainer for Graph Neural Networks

MERT KOSAN and ZEXI HUANG, University of California, Santa Barbara, CA, USA
SOURAV MEDYA, University of Illinois, Chicago, IL, USA
SAYAN RANU, Indian Institute of Technology, Delhi, India
AMBUJ SINGH, University of California, Santa Barbara, CA, USA

Graph neural networks (GNNs) find applications in various domains such as computational biology, natural language processing, and computer security. Owing to their popularity, there is an increasing need to explain GNN predictions since GNNs are black-box machine learning models. One way to address this issue involves using *counterfactual* reasoning where the objective is to alter the GNN prediction by minimal changes in the input graph. Existing methods for counterfactual explanation of GNNs are limited to instance-specific *local* reasoning. This approach has two major limitations of not being able to offer global recourse policies and overloading human cognitive ability with too much information. In this work, we study the *global* explainability of GNNs through global counterfactual reasoning. Specifically, we want to find a *small* set of representative counterfactual graphs that explains *all* input graphs. Toward this goal, we propose GCFExplainer, a novel algorithm powered by *vertex-reinforced random walks* on an *edit map* of graphs with a *greedy summary*. Extensive experiments on real graph datasets show that the global explanation from GCFExplainer provides important high-level insights of the model behavior and achieves a *46.9%* gain in recourse coverage, a *9.5%* reduction in recourse cost compared to the state-of-the-art local counterfactual explainers. We also demonstrate that GCFExplainer generates explanations that are more consistent with input dataset characteristics, and is robust under adversarial attacks. In addition, K-GCFExplainer, which incorporates a graph clustering component into GCFExplainer, is introduced as a more competitive extension for datasets with a clustering structure, leading to superior performance in three out of four datasets in the experiments and better scalability.

CCS Concepts: • **Computing methodologies** → *Causal reasoning and diagnostics*; • **Theory of computation** → *Graph algorithms analysis*;

Additional Key Words and Phrases: Counterfactual explanation; Graph neural networks

ACM Reference format:
Mert Kosan, Zexi Huang, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2025. GCFExplainer: Global Counterfactual Explainer for Graph Neural Networks. *ACM Trans. Intell. Syst. Technol.* 16, 5, Article 99 (August 2025), 23 pages.
https://doi.org/10.1145/3698108

Mert Kosan and Zexi Huang contributed equally to this research.
Mert Kosan did this work prior to joining Visa Inc.
This work is partially funded by NSF via Grants IIS 1817046 and IIS 2229876.
Authors' Contact Information: Mert Kosan (corresponding author), University of California, Santa Barbara, CA, USA; e-mail: mertkosan@gmail.com; Zexi Huang, University of California, Santa Barbara, CA, USA; e-mail: zexihuang.phd@gmail.com; Sourav Medya, University of Illinois, Chicago, IL, USA; e-mail: medya@uic.edu; Sayan Ranu, Indian Institute of Technology, Delhi, India; e-mail: sayanranu@cse.iitd.ac.in; Ambuj Singh, University of California, Santa Barbara, CA, USA; e-mail: ambuj@cs.ucsb.edu.

ACM 2157-6912/2025/8-ART99
https://doi.org/10.1145/3698108

1 Introduction

Graph neural networks (GNNs) [13, 22, 39, 55, 62, 63] are being used in many domains such as drug discovery [17], chip design [37], combinatorial optimization [32], physical simulations [3, 51], event prediction [12, 24, 34], and clustering [4, 53]. Taking the graph(s) as input, GNNs are trained to perform various downstream tasks that form the core of many real-world applications. For example, graph classification has been applied to predict whether a drug would exhibit the desired chemical activity [17]. Similarly, node prediction is used to predict the functionality of proteins in protein–protein interaction networks [7] and categorize users into roles on social networks [67].

Despite the impressive success of GNNs on predictive tasks, GNNs are *black-box* machine learning models. It is non-trivial to explain or reason why a particular prediction is made by a GNN. Explainability of a prediction model is important to understand its shortcomings and identify areas for improvement. In addition, the ability to explain a model is critical toward making it trustworthy. Owing to this limitation of GNNs, there has been significant efforts in recent times toward explanation approaches.

Existing work on explaining GNN predictions can be categorized mainly in two directions [18]: (1) factual reasoning [30, 58, 68, 69], and (2) counterfactual reasoning [1, 2, 29, 50]. Generally speaking, the methods in the first category aim to find an important subgraph that correlates most with the underlying GNN prediction. In contrast, the methods with counterfactual reasoning attempt to identify the smallest amount of perturbation on the input graph that changes the GNN's prediction, for example, removal/addition of edges or nodes.

Compared to factual reasoning, counterfactual explainers have the additional advantage of providing the means for recourse [19, 44, 54, 57, 59]. For example, in the applications of drug discovery [17, 66], mutagenicity is an adverse property of a molecule that hampers its potential to become a marketable drug [20]. In Figure 1, formaldehyde is classified by a GNN to be mutagenic. Factual explainers can attribute the subgraph containing the carbon–hydrogen bond to the cause of mutagenicity, while counterfactual explainers provide an effective way (i.e., a recourse) to turn formaldehyde into formic acid, which is non-mutagenic, by replacing a hydrogen atom with a hydroxyl.

In this work, we focus on counterfactual explanations. Our work is based on the observation that existing counterfactual explainers [30, 58, 68, 69] for graphs take a *local* perspective, generating counterfactual examples for individual input graphs. However, this approach has three key limitations:

- *Lack of global insights*: It is desirable to offer insights that generalize across a multitude of data graphs. For example, instead of providing formic acid as a counterfactual example to formaldehyde, we can summarize global recourse rules such as "*Given any molecule with a carbonyl group (carbon–oxygen double bond), it needs a hydroxy to be non-mutagenic.*" This focus on **global counterfactual explanation (GCE)** promises to provide higher-level insights that are complementary to those obtained from local counterfactual explanations.
- *Information overload*: The primary motivation behind counterfactual analysis is to provide human-intelligible explanations. With this objective, consider real-world graph datasets that routinely contain thousands to millions of graphs. Owing to instance-specific counterfactual explanations, the number of counterfactual graphs grows linearly with the graph dataset size. Consequently, the sheer volume of counterfactual graphs overloads human cognitive ability to process this information. Hence, the initial motivation of providing human-intelligible insights is lost if one does not obtain a holistic view of the counterfactual graphs.
- *Information leakage*: In line with the information overload, local explanations tend to leak information related to the model (i.e., GNN). Therefore, local explanations may need additional

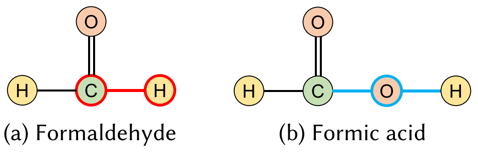

Fig. 1. Formaldehyde (a) is classified by a GNN to be an undesired mutagenic molecule with its important subgraph found by factual reasoning highlighted in red. Formic acid (b) is its non-mutagenic counterfactual example obtained by removing one edge and adding one node and two edges.

procedures to ensure the information remains private. However, global explanations naturally encounter this issue to a lesser extent and leak less information regarding the GNN, making them potentially preferable for model designers.

Contributions. In this article, we study the problem of model-agnostic, *global* counterfactual explanations of GNNs for graph classification. More specifically, given a graph dataset, our goal is to counterfactually explain the largest number of input graphs with a small number of counterfactuals. As we will demonstrate later in our experiments, this formulation naturally forces us to remove redundancy from instance-specific counterfactual explanations and hence has higher information density. Algorithmically, the proposed problem introduces new challenges. We theoretically establish that the proposed problem is NP-hard. Furthermore, the space of all possible counterfactual graphs itself is exponential. Our work overcomes these challenges and makes the following contributions:

— *Novel formulation*: We formulate the novel problem of GCE of GNNs for graph classification. In contrast to existing works on counterfactual reasoning that only generate instance-specific examples, we provide an explanation on the global behavior of the model.
— *Algorithm design*: While the problem is NP-hard, we propose GCFExplainer, which organizes the exponential search space as an *edit map*. We then perform **vertex-reinforced random walks (VRRWs)** on it to generate diverse, representative counterfactual candidates, which are *greedily summarized* as the global explanation. An extension of it based on graph clustering, K-GCFExplainer, is also introduced to better leverage the clustering structure of input datasets.
— *Experiments and case study*: We conduct extensive experiments on real-world datasets to validate the effectiveness of the proposed method. Results show that GCFExplainer not only provides important high-level insights on the model behavior but also outperforms state-of-the-art baselines related to counterfactual reasoning in recourse quality metrics, consistency with input graphs, and robustness against adversarial attacks. We also include a detailed case study demonstrating the usefulness of the generated global counterfactual summary in providing high-level insight for the underlying GNN model in the domain of drug discovery.

While this work is based on our previous conference article [14], we have extended its scope significantly in this version. Specifically, the list of differences between this work and our conference article is as follows:

— We extend GCFExplainer to K-GCFExplainer by incorporating a graph clustering component (Section 4), and provide extensive empirical analysis to demonstrate its superior performance (Section 5.9) and better scalability (Section 5.10) compared to the original GCFExplainer when the datasets have strong clustering structures.

Table 1. Notations

$\phi(.)$	A binary graph classifier
G	An input graph
$\mathbb{G}$	A collection of input graphs
n	The number of input graphs
C	A counterfactual graph
$\mathbb{C}$	A collection of counterfactual graphs
c	The size of the global counterfactual representation
$r(.)$	A recourse function
$d(.,.)$	A distance function (between graphs)
θ	The distance parameter (threshold)
$\mathcal{G}$	A meta-graph (edit map)
h	An average node degree in the meta-graph
$I(.)$	An importance function
M	The number of iterations for the VRRW
k	The number of cluster

— We add new analysis of the consistency between the generated counterfactual summaries and the input graphs for GCFExplainer and baselines, showing our strength in preserving the connectivity of input graphs (Section 5.4).
— We conduct additional experiments measuring the robustness of GCFExplainer against edge flips. Results illustrate that GCFExplainer outperforms the baselines even when the input graphs are noisy or under attacks (Section 5.5).

2 GCEs

This section introduces the GCE problem for graph classification. We start with the background on local counterfactual reasoning. Then, we propose a representation of the global recourse rule that provides a high-level counterfactual understanding of the classifier behavior. Finally, we introduce quality measures for recourse rules and formally define the GCE problem. The notations used in this paper is listed in Table 1.

2.1 Local Counterfactual

Consider a graph $G = (V, E)$, where V and E are the sets of (labelled) nodes and edges, respectively. A (binary) graph classifier (e.g., a GNN) ϕ classifies G into either the undesired class ($\phi(G) = 0$) or the desired one ($\phi(G) = 1$). An explanation of ϕ seeks to answer how these predictions are made. Those based on factual reasoning analyze what properties G possesses to be classified in the current class while those based on counterfactual reasoning find what properties G needs to be assigned to the opposite class.

Existing counterfactual explanation methods take a local perspective. Specifically, for each input graph G, they find a *counterfactual* (graph) C that is somewhat similar to G but is assigned to a different class. Without loss of generality, let G belong to the undesired class, i.e., $\phi(G) = 0$, then the counterfactual C satisfies $\phi(C) = 1$. The similarity between C and G is quantified by a predefined distance metric d, for example, the number of added/removed edges [2, 29].

In our work, we consider the **graph edit distance (GED)** [47], a more general distance measure, as the distance function to account for other types of changes. Specifically, $\text{GED}(G_1, G_2)$ counts the minimum number of "edits" to convert G_1 to G_2. An "edit" can be adding or removing edges and

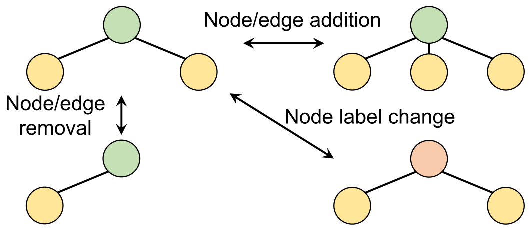

Fig. 2. Edits between graphs.

nodes, or changing node labels (see Figure 2). Moreover, to account for graphs of different sizes, we normalize the GED by the sizes of graphs: $\widehat{\text{GED}}(G_1, G_2) = \text{GED}(G_1, G_2)/(|V_1| + |V_2| + |E_1| + |E_2|)$. Nonetheless, our method can be applied with other graph distance metrics, such as those based on graph kernels (e.g., RW [6], NSPDG [8], WL [49]).

The distance function measures the quality of the counterfactual found by the explanation model. Ideally, the counterfactual C should be very close to the input graph G while belonging to a different class. Formally, we define the counterfactuals that are within a certain distance θ from the input graph as *close counterfactuals*.

Definition 1 (*Close Counterfactual*). Given the GNN classifier ϕ, distance parameter θ, and an input graph G with undesired outcome, i.e., $\phi(G) = 0$; a counterfactual graph, C, is a close counterfactual of G when $\phi(C) = 1$ and $d(G, C) \leq \theta$.

While the (close) counterfactual C found by existing methods explains the classifier behavior for the corresponding input graph G, it is hard to generalize to understand the global pattern. Next, we introduce the global recourse rule that provides a high-level summary of the classifier behavior across different input graphs.

2.2 Global Recourse Representation

The GCE requires a global recourse rule r. Specifically, for any (undesired) input graph G with $\phi(G) = 0$, r provides a (close) counterfactual (i.e., a recourse) for G: $\phi(r(G)) = 1$. While both a recourse rule and a local counterfactual explainer find a counterfactual given an input graph, their goals are different. The goal of the local counterfactual explainer is to find the best (closest) counterfactual possible for each input graph, and therefore, r can be very complicated, e.g., in the form of an optimization algorithm [2, 50]. On the other hand, a recourse rule aims to provide an explanation of the classifier's global behavior, which requires a simpler form that is understandable for domain experts without prior knowledge of deep learning on graphs.

Existing global recourse rules for classifiers with feature vectors as input take the form of short decision trees [44]. However, this is hard to be generalized to graph data with rich structure information. Instead, we propose the representation of a global recourse rule for a graph classifier to be a collection of counterfactual graphs $\mathbb{C}$ in the desired class that are *diverse* and *representative* enough to capture its global behavior. This representation does not require any additional knowledge for domain experts to understand and draw insights from, similar to the local counterfactual examples. It is also easy to find the local counterfactual for a given input graph G based on $\mathbb{C}$ by nominating the closest graph in $\mathbb{C}$: $r(G) = \arg\min_{C \in \mathbb{C}} d(G, C)$.

2.3 Quantifying Recourse Quality

Given a graph classifier ϕ and a set of n input graphs $\mathbb{G}$ in the undesired class, we want to compare the quality of different recourse representations $\mathbb{C}$. Similar to the quality metrics introduced for vector data [44], we aim to account for the following factors:

(1) **Coverage**: Like local counterfactual explainers, we want to ensure that counterfactuals found for individual input graphs are of high quality. Specifically, we introduce recourse coverage—the proportion of input graphs that have close counterfactuals from $\mathbb{C}$ under a given distance threshold θ:
$$\textbf{coverage}(\mathbb{C}) = |\{G \in \mathbb{G} \mid \min_{C \in \mathbb{C}} \{d(G, C)\} \leq \theta\}|/|\mathbb{G}|.$$
(2) **Cost**: Another quality metric based on local counterfactual quality is the recourse cost (i.e., the distance between the input graph and its counterfactual) across the input graphs:
$$\textbf{cost}(\mathbb{C}) = \operatorname*{agg}_{G \in \mathbb{G}} \{\min_{C \in \mathbb{C}} \{d(G, C)\}\},$$
where agg is an aggregation function, e.g., mean or median.
(3) **Interpretability**: Finally, the recourse rule should be easy (small) enough for human cognition. We quantify the interpretability as the size of recourse representation:
$$\textbf{size}(\mathbb{C}) = |\mathbb{C}|.$$

2.4 Problem Formulation and Characterization

An ideal recourse representation should maximize the coverage while minimizing the cost and the size. Formally, we define the GCE problem as follows:

Problem 1 (GCE for Graph Classification). *Given a GNN graph classifier ϕ that classifies n input graphs $\mathbb{G}$ to the undesired class 0 and a budget $c \ll n$, our goal is to find the best recourse representation $\mathbb{C}$ that maximizes the recourse coverage with size c:*
$$\max_{\mathbb{C}} \textbf{\textit{coverage}}(\mathbb{C}) \ \textit{s.t.} \ \textbf{\textit{size}}(\mathbb{C}) = c.$$

We note that in our problem formulation only coverage and size are explicitly accounted for, whereas cost is absent. We make this design choice since cost and coverage are intrinsically opposing forces. Specifically, if we are willing to allow a high cost, coverage increases since we allow for higher individual distances between an input graph and its counterfactual. Therefore, we take the approach of binding the cost to the distance threshold θ in the coverage definition. Nonetheless, an explicit analysis of all these metrics including cost is performed to quantify recourse quality during our empirical evaluation in Section 5. Below we discuss the hardness of GCE.

Theorem 1 (NP-hardness). *The GCE problem is NP-hard.*

Proof. To establish NP-hardness of the proposed problem we reduce it from the classical *Maximum Coverage* problem.

Definition 2 (*Maximum Coverage*). Given a budget c and a collection of subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ from a universe of items $U = \{u_1, \cdots, u_n\}$, find a subset $\mathcal{S}' \subseteq \mathcal{S}$ of sets such that $|\mathcal{S}'| \leq c$ and the number of covered elements $|\bigcup_{\forall S_i \in \mathcal{S}'} S_i|$ is maximized.

We show that given any instance of a Maximum Coverage problem $\langle \mathcal{S}, U \rangle$, it can be mapped to a GCE problem. For u_i, we construct a star graph with a center node with an empty label and n leaf nodes with $n-1$ empty labels and one label u_i. For S_i, we construct a similar star graph with a center node with a special label γ and n leaf nodes with $|S_i|$ labeled with the elements in S_i and $n - |S_i|$ with empty labels. The classifier ϕ classifies a graph as a desired one if and only if it is a star graph with a γ-labeled central node and n leaf nodes with a set of labels among $\mathcal{S} = \{S_1, \cdots, S_m\}$. The allowed edit operations are either adding or deleting a set of labels (as a single edit), but not both together. So, each S_i corresponds to a counterfactual candidate C_i and $d(G_j, C_i) \leq \theta = 1$

if and only if $u_j \in S_i$. With this construction, it is easy to see that an optimal solution for this instance of GCE is the optimal solution for the corresponding instance of the Maximum Coverage problem. □

Owing to NP-hardness, it is not feasible to identify the optimal solution for the GCE problem in polynomial time unless NP = P. In the next section, we will introduce GCFExplainer, an effective and efficient heuristic that solves the GCE problem.

3 GCFExplainer

In this section, we propose GCFExplainer, the first global counterfactual explainer for graph classification. The GCE problem requires us to find a collection of c counterfactual graphs that maximize the coverage of the input graphs. Intuitively, we want each individual counterfactual graph to be a close counterfactual to (i.e., "cover") as many input graphs as possible. Additionally, different counterfactual graphs should cover different sets of input graphs to maximize the overall coverage. These intuitions motivate the design of our algorithm GCFExplainer, which has three major components:

(1) *Structuring the search space*: The search space of counterfactual graphs consists of *all* graphs that are in the same domain as the input graphs and within a distance of θ. In other words, any graph within a distance of θ from an input graph may be a potential counterfactual candidate and therefore needs to be analyzed. The number of potential graphs within θ increases exponentially with θ since the space of graph edits is combinatorial [27, 43]. GCFExplainer uses an *edit map* to organize these graphs as a meta-graph $\mathcal{G}$, where individual nodes are graphs that are created via a different number of edits from the input graphs and each edge represents a single edit.

(2) *VRRW*: To search for good counterfactual candidates, GCFExplainer leverages VRRWs [41] on the edit map $\mathcal{G}$. VRRW has the nice property of converging to a set of nodes that are both important (i.e., cover many input graphs) and diverse (i.e., non-overlapping coverage), which will form a small set of counterfactual candidates for further processing.

(3) *Iterative computation of the summary*: After obtaining good counterfactual candidates from VRRW, GCFExplainer creates the final set of the counterfactual graphs (i.e., the summary) as the recourse representation by iteratively adding the best candidate based on the maximal gain of the coverage given the already added candidates.

3.1 Structuring the Search Space

The search space for counterfactual graphs in GCFExplainer is organized via an edit map $\mathcal{G}$. The edit map is a meta-graph whose nodes are graphs in the same domain as the input graphs and edges connect graphs that differ by a single graph edit. As an example, each graph in Figure 2 represents a node in the edit map, and the arrows denote edges between graphs (nodes) that are one edit away. In the edit map, we only include connected graphs since real graphs of interest are often connected (e.g., molecules, proteins).

While all potential counterfactual candidates are included as its nodes, the edit map has an exponential size and it is computationally prohibitive to fully explore it. However, a key observation is that a counterfactual candidate can only be a few hops away from some input graph. Otherwise, the graph distance between the counterfactual and the input graph would be too large for the counterfactual to cover it. This observation motivates our exploration of the edit map to be focused on the union of close neighborhoods of the input graphs (see Section 3.2.3). Additionally, while we cannot compute the entire edit map, it is easy to chart the close neighborhoods by iteratively performing all possible edits from the input graphs. Next, we introduce the VRRW to efficiently explore the edit map to find counterfactual candidates.

3.2 VRRW

VRRW [41] is a time-variant random walk. Different from other more widely applied random walk processes such as the simple random walk and the PageRank [15, 16, 23, 42], the transition probability $p(u, v)$ of VRRW from node u to node v depends not only on the edge weight $w(u, v)$ but also on the *number of previous visits* in the walk to the target node v, which we denote using $N(v)$. Specifically,

$$p(u, v) \propto w(u, v)N(v). \tag{1}$$

GCFExplainer applies VRRW on the edit map and produces n most frequently visited nodes in the walk as the set of counterfactual candidates $\mathbb{S}$. Next, we formalize VRRW in our setting and explain how it surfaces good counterfactual candidates for GCE.

3.2.1 Vertex-Reinforcement. Our main motivation for using VRRW to explore the edit map instead of other random walk processes is that VRRW converges to a diverse and representative set of nodes [35, 38] in different regions of the edit map. In this way, the frequently visited nodes in instances of VRRW have the potential to be good counterfactual candidates as they would cover a diverse set of input graphs in the edit map. The reason behind the diversity of the highly visited nodes is the previous visit count $N(v)$ in the transition probability. Specifically, nodes with larger visit counts tend to be visited more often later ("richer gets richer"), and thereby dominating all other nodes in their neighborhood. This leads to a bunch of highly visited nodes to "represent" each region of the edit map. We refer the readers to [35] for details on the mathematical basis and the theoretical correctness of this property. Moreover, as our goal is to find counterfactual candidates, we only reinforce (i.e., increase the visit counts of) graphs in the counterfactual class.

3.2.2 Importance Function. While the vertex-reinforcement mechanism ensures diversity of the highly visited nodes, we still need to guide the walker to visit graphs that are good counterfactual candidates. We achieve this by assigning large edge weight $w(u, v)$ to good counterfactual candidates via an importance function $I(v)$:

$$w(u, v) = I(v). \tag{2}$$

The importance function $I(v)$ should capture the quality of a graph v as a counterfactual candidate. It has the following components:

(1) Counterfactual probability $p_\phi(v)$. The graph classifier ϕ predicts a probability for v to be in the counterfactual class ($\phi(v) = 1$). By using it as part of the importance function, the walker is encouraged to visit regions with rich counterfactual graphs.
(2) Individual coverage $\mathbf{coverage}(\{v\})$. The individual coverage of a graph v computes the proportion of input graphs that are close to v. This encourages the walker to visit graphs that cover a large number of input graphs.
(3) Gain of coverage $\mathbf{gain}(v; \mathbb{S})$. Given a graph v and the current set of counterfactual candidates $\mathbb{S}$ (i.e., the n most frequently visited nodes), we can compute the gain between the current coverage and the coverage after adding v to $\mathbb{S}$:

$$\mathbf{gain}(v; \mathbb{S}) = \mathbf{coverage}(\mathbb{S} \cup \{v\}) - \mathbf{coverage}(\mathbb{S})$$

This guides the walker to find graphs that complement the current counterfactual candidates to cover additional input graphs.

The importance function is a combination of these components:

$$I(v) = p_\phi(v)(\alpha\,\mathbf{coverage}(\{v\}) + (1-\alpha)\,\mathbf{gain}(v;\mathbb{S})), \tag{3}$$

where α is a hyperparameter between 0 and 1. With the above importance function, the VRRW in GCFExplainer converges to a set of diverse nodes that have high counterfactual probability and collectively cover a large number of input graphs.

3.2.3 Dynamic Teleportation. The last component of VRRW, teleportation, is to help us manage the exponential search space of the edit map. Since our goal is to find close counterfactuals to the input graphs, the walker only needs to explore the nearby regions of the input graphs. Therefore, we start the walk from the input graphs, and also at each step, let the walker teleport back (i.e., transit) to a random input graph with probability τ.

To decide which input graph to teleport to, we adopt a dynamic probability distribution based on the current counterfactual candidate set $\mathbb{S}$. Specifically, let $g(G) = |\{v \in \mathbb{S} \mid d(v,G) \leq \theta \text{ and } \phi(v) = 1\}|$ be the number of close counterfactuals in $\mathbb{S}$ covering an input graph G. Then the probability to teleport to G is

$$p_\tau(G) = \frac{\exp(-g(G))}{\sum_{G' \in \mathbb{G}} \exp(-g(G'))}. \tag{4}$$

This dynamic teleportation favors input graphs that are not well covered by the current solution set and encourages the walker to explore nearby counterfactuals to cover them after teleportation.

3.3 Iterative Computation of the Summary

We have applied VRRW to generate a good set of n counterfactual candidates $\mathbb{S}$. In the last step of GCFExplainer, we aim to further refine the candidate set and create the final recourse representation (i.e., the summary) with c counterfactual graphs. This summarization problem is also NP-hard and we propose to build $\mathbb{C}$ in an iterative and greedy manner from $\mathbb{S}$.

Specifically, we start with an empty solution set $\mathbb{C}_0$. Then, for each iteration t, we add the graph v to $\mathbb{C}_t$ with the maximal gain of coverage $\mathbf{gain}(v;\mathbb{C}_t)$. This is repeated c times to get the final recourse representation $\mathbb{C}$ with c graphs. It is easy to show that the summarization problem is submodular and therefore, our greedy algorithm provides $(1-1/e)$-approximation.

Notice that the greedy algorithm can also be applied to the local counterfactuals found by existing methods to generate a GCE solution. Here, we highlight three advantages of GCFExplainer:

(1) Existing local counterfactual explainers [1, 2, 29, 50] are only able to generate counterfactuals based on one type of graph edits—edge removal, while GCFExplainer incorporates all types of edits to include a richer set of counterfactual candidates.
(2) The set of counterfactual candidates from GCFExplainer is generated with the GCE objective in mind, while the local counterfactuals from existing methods are optimized for individual input graphs. Therefore, they may not be good candidates to capture the global behavior of the classifier.
(3) It is easy to incorporate domain constraints (e.g., the valence of chemical bonds) into GCF-Explainer by pruning the neighborhood of the edit map, while existing methods based on optimization require non-trivial efforts to customize.

We will empirically demonstrate the superiority of GCFExplainer to this two-stage approach with state-of-the-art local counterfactual explanation methods in our experiments in Section 5.2.

Pseudocode and Complexity: The pseudocode of GCFExplainer is presented in Algorithm 1. Lines 1–16 summarizes the VRRW component of GCFExplainer. Specifically, Lines 3–10 determine the

Algorithm 1: GCFExplainer(ϕ, $\mathbb{G}$)

```
1:  $G \leftarrow$ random input graph from $\mathbb{G}$, $N(G) \leftarrow 1$, $\mathbb{S} = \{G\}$
2:  for $i \in 1 : M$ do
3:      Let $\epsilon \sim Bernoulli(\tau)$
4:      if $\epsilon = 0$ then
5:          for $v \in Neighbors(G)$ do
6:              Compute $I(v)$ based on Equation (3)
7:              Compute $p(G, v)$ based on Equation (1)
8:          $v \leftarrow$ random neighbor of $G$ based on $p(G, v)$
9:      else
10:         $v \leftarrow$ random input graph from $\mathbb{G}$ based on Equation (4)
11:     if $\phi(v) = 1$ then
12:         if $v \in \mathbb{S}$ then
13:             $N(v) \leftarrow N(v) + 1$
14:         else
15:             $\mathbb{S} \leftarrow \mathbb{S} + \{v\}$, $N(v) \leftarrow 1$
16:     $G \leftarrow v$
17: $\mathbb{S} \leftarrow$ top $n$ frequently visited counterfactuals in $\mathbb{S}$
18: $\mathbb{C} \leftarrow \emptyset$
19: for $t \in 1 : c$ do
20:     $v \leftarrow \arg\max_{v \in \mathbb{S}}$ gain$(v; \mathbb{C})$
21:     $\mathbb{C} \leftarrow \mathbb{C} + \{v\}$
22: return $\mathbb{C}$, $\mathbb{S}$
```

next graph to visit based on VRRW transition probabilities and dynamic teleportation, and Lines 11–16 update the visit counts and the set of counterfactual candidates. The iterative computation of the counterfactual summary is described in Lines 17–21. The overall complexity of GCFExplainer is $O(Mhn + cn)$, where M is the number of iterations for the VRRW, h is the average node degree in the meta-graph, n is the number of input graphs, and c is the size of the global counterfactual representation. Note that h here depends on the characteristics of input graphs and individual random walk runs, which do not have a close-form bound. In practice, one can set a limit on the number of neighbors to be explored at each VRRW iteration to maintain tractability. We have included an empirical analysis on incorporating this limit (set as 10,000) in our algorithm in Section 5.10. The results show that we can achieve significantly shorter running time (up to 3× times in the Proteins dataset) with only a small effect on the cost (−3.3% on average) in terms of performance. In addition, we also store the computed transition probabilities with a space-saving algorithm [36] to further improve the running time of GCFExplainer.

4 K-GCFExplainer

In this section, we propose K-GCFExplainer as an extension of GCFExplainer. It first puts input graphs into several clusters, and then apply GCFExplainer to each cluster of graphs to generate counterfactuals, which are combined together for a summary. Intuitively, when input graphs exhibit a strong clustering structure (e.g., chemical compounds with different functional groups), incorporating this prior information into our framework can lead to more effective and efficient solutions. More specifically, in K-GCFExplainer:

— VRRW only explores graphs in the same cluster, and the generated counterfactuals are more representative with respect to each cluster.

Algorithm 2: K-GCFExplainer(ϕ, $\mathbb{G}$, k)

```
1:  $\{\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_k\} \leftarrow$ split $\mathbb{G}$ into $k$ clusters
2:  for $i \in 1 : k$ do
3:      $\mathbb{C}_i, \mathbb{S}_i \leftarrow$ GCFExplainer($\phi$, $\mathbb{G}_i$)
4:  $\mathbb{S} \leftarrow \emptyset$
5:  for $i \in 1 : k$ do
6:      $\mathbb{S}_i \leftarrow$ top $|\mathbb{G}_i|$ frequently visited counterfactuals in $\mathbb{S}_i$
7:      $\mathbb{S} \leftarrow \mathbb{S} + \mathbb{S}_i$
8:  $\mathbb{C} \leftarrow \emptyset$
9:  for $t \in 1 : c$ do
10:     $v \leftarrow \arg\max_{v \in \mathbb{S}}$ gain$(v; \mathbb{C})$
11:     $\mathbb{C} \leftarrow \mathbb{C} + \{v\}$
12: return $\mathbb{C}, \mathbb{S}$
```

Table 2. The Statistics of the Datasets

	NCI1	Mutagenicity	AIDS	Proteins
#Graphs	3,978	4,308	1,837	1,113
#Nodes	118,714	130,719	28,905	43,471
#Edges	128,663	132,707	29,985	81,044
#Node Labels	10	10	9	3

—Exploration on each cluster is independent from each other, making our algorithm parallelizable.

We will empirically demonstrate the superiority of K-GCFExplainer to GCFExplainer when the input datasets have a strong clustering structure in Section 5.9.

Pseudocode and Complexity. The pseudocode for K-GCFExplainer is presented in Algorithm 2. Line 1 applies a clustering algorithm to the input graphs. For each cluster, Lines 2–3 execute GCFExplainer, as described in Algorithm 1. Lines 4–7 combine frequently visited counterfactuals from each cluster. Lines 8–11 generate the counterfactual summary, following a process similar to that in Algorithm 1. The overall complexity of K-GCFExplainer is $O(\sum_{i=1}^{k} M_i hn + cn)$, where k is the number of clusters, M_i is the number of VRRW iterations for the ith cluster, h is the average node degree in the meta-graph, n is the number of input graphs, and c is the size of the global counterfactual representation. Since we set $\sum_{i=1}^{k} M_i = M$, the complexity of GCFExplainer and K-GCFExplainer remains the same. Importantly, K-GCFExplainer can be parallelized for each cluster, enhancing its overall efficiency.

5 Experiments

We provide empirical results for the proposed GCFExplanier along with baselines on commonly used graph classification datasets. Our code is available at https://github.com/mertkosan/GCFExplainer.

5.1 Experimental Settings

5.1.1 Datasets. We use four different real-world datasets for graph classification benchmark with their statistics in Table 2. Specifically, NCI1 [60], Mutagenicity [20, 45], and AIDS [45] are collections of molecules with nodes representing different atoms and edges representing chemical bonds between them. The molecules are classified by whether they are anticancer, mutagenic,

Table 3. Accuracy of the GNN Graph Classifier

	NCI1	**Mutagenicity**	**AIDS**	**Proteins**
Training	0.8439	0.8825	0.9980	0.7800
Validation	0.8161	0.8302	0.9727	0.8198
Testing	0.7809	0.8000	0.9781	0.7297

and active against HIV, respectively. Proteins [7, 10] is a collection of proteins classified into enzymes and non-enzymes, with nodes representing secondary structure elements and edges representing structural proximity. For all datasets, we filter out graphs containing rare nodes with label frequencies smaller than 50.

5.1.2 Graph Classifier. We follow [58] and train a GNN with three convolution layers [22] of embedding dimension 20, a max pooling layer, and a fully connected layer for classification. Note that our method only requires the predictions from the graph classifier and can be applied to any black-box graph classifiers beyond GNNs. The model is trained with the Adam optimizer [21] and a learning rate of 0.001 for 1,000 epochs. The datasets are split into 80%/10%/10% for training/validation/testing with the model accuracy shown in Table 3.

5.1.3 Baselines. To the best of our knowledge, GCFExplainer is the first global counterfactual explainer. To validate its effectiveness, we compare it against state-of-the-art local counterfactual explainers RCExplainer [2] and CFF [50] combined with the greedy summarization algorithm described in Section 3.3. Additionally, we design another baseline called Ground-Truth that uses graphs belonging to the desired class from the original dataset as local counterfactuals.

Note that one can interpret the factual explanation from a factual explainer in a counterfactual way to allow a comparison between factual and counterfactual approaches. That is, if a factual explainer identifies an important subgraph that leads to its predicted class, the residual graph after removing this explanatory subgraph could be treated as a counterfactual explanation. However, this usually leads to unfavorable comparison for the factual explainers as they are not designed with counterfactual reasoning in mind.

5.1.4 Explainer Settings. We use a distance threshold θ of 0.05 for training all explainers. Since computing the exact GED is NP-hard, we apply a state-of-the-art neural approximation algorithm [43]. For GCFExplainer, we set the teleportation probability $\tau = 0.1$ and tune α, the weight between individual coverage and gain of coverage, from $\{0, 0.5, 1\}$. A sensitivity analysis is presented in Section 5.8. The number of VRRW iterations M is set to 50,000, which is enough for convergence as shown in Section 5.7. For K-GCFExplainer, we employ K-medoids [40] for the initial clustering of input graphs based on their GED. Each cluster i utilizes the same hyperparameters as GCFExplainer, with the number of VRRW iterations M_i proportional to the cluster size and $\sum_{i=1}^{k} M_i = 50{,}000$, where the k—the number of clusters—is chosen from $\{2, 3, 4, 5\}$. For baselines, we tune their hyperparameters to achieve the best local counterfactual rates while maintaining an average distance to input graphs that is smaller than the distance threshold θ.

5.1.5 Experiment Environment. All our experiments are run on a machine with 2 NVIDIA GeForce RTX 2080 GPU (8 GB of RAM) and 32 Intel Xeon CPUs (2.10 GHz and 128 GB of RAM).

5.2 Recourse Quality

We start by comparing the recourse quality between GCFExplainer and baselines. Table 4 shows the recourse coverage with $\theta = 0.1$ and median recourse cost of the top 10 counterfactual graphs

Table 4. Recourse Coverage ($\theta = 0.1$) and Median Recourse Cost Comparison between GCFExplainer and Baselines for a 10-Graph Global Explanation

	NCI1		**Mutagenicity**		**AIDS**		**Proteins**	
	Coverage	Cost	Coverage	Cost	Coverage	Cost	Coverage	Cost
Ground-Truth	16.54%	<u>0.1326</u>	28.96%	<u>0.1275</u>	0.41%	0.2012	8.47%	<u>0.2155</u>
RCExplainer	15.22%	0.1370	<u>31.99%</u>	0.1290	<u>8.96%</u>	<u>0.1531</u>	<u>8.74%</u>	0.2283
CFF	<u>17.61%</u>	0.1331	30.43%	0.1327	3.39%	0.1669	3.83%	0.2557
GCFExplainer	**27.85%**	**0.1281**	**37.08%**	**0.1135**	**14.66%**	**0.1516**	**10.93%**	**0.1856**

GCFExplainer consistently and significantly outperforms all baselines across different datasets. We highlight the best-performing method in bold and the second best-performing method with underlines.

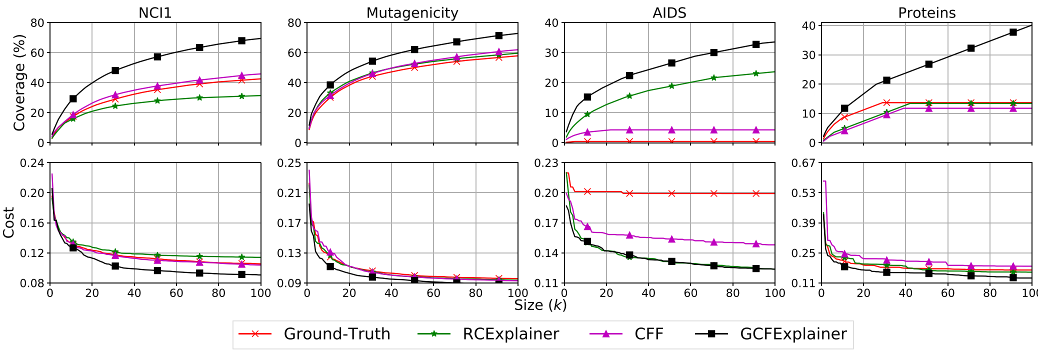

Fig. 3. Coverage and cost performance comparison between GCFExplainer and baselines based on different counterfactual summary sizes. GCFExplainer consistently outperforms the baselines across different sizes.

(i.e., $k = 10$). We first notice that the two state-of-the-art local counterfactual explainers have similar performance as Ground-Truth, consistent with our claim that local counterfactual examples from existing methods are not good candidates for a global explanation. The proposed GCFExplainer, on the other hand, achieves significantly better performance for global recourse quality. Compared to the best baseline, RCExplainer, GCFExplainer realizes a *46.9%* gain in recourse coverage and a *9.5%* reduction in recourse cost.

Next, we show the recourse coverage and cost for different sizes of counterfactual summary in Figure 3. As expected, adding more graphs to the recourse representation increases recourse coverage while decreasing recourse cost, at the cost of interpretability. And GCFExplainer maintains a constant edge over the baselines.

We also compare the recourse coverage based on different distance thresholds θ, with results shown in Figure 4. While coverage increases for all methods as the threshold increases, GCFExplainer consistently outperforms the baselines across different thresholds.

5.3 Global Counterfactual Insight

We have demonstrated the superiority of GCFExplainer based on various quality metrics for global recourse. Here, we show how GCFExplainer provides global insights compared to local counterfactual examples. Figure 5 illustrates (a) four input undesired graphs with a similar structure from the AIDS dataset, (b) corresponding local counterfactual examples (based on RCExplainer and CFF), and (c) the representative global counterfactual graph from GCFExplainer covering the

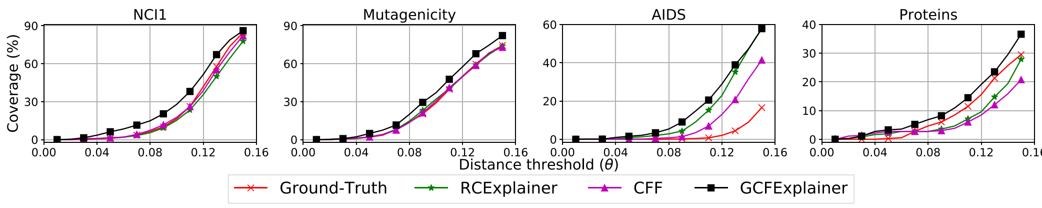

Fig. 4. Recourse coverage comparison between GCFExplainer and baselines based on different distance threshold values (θ). GCFExplainer consistently outperforms the baselines across different θ.

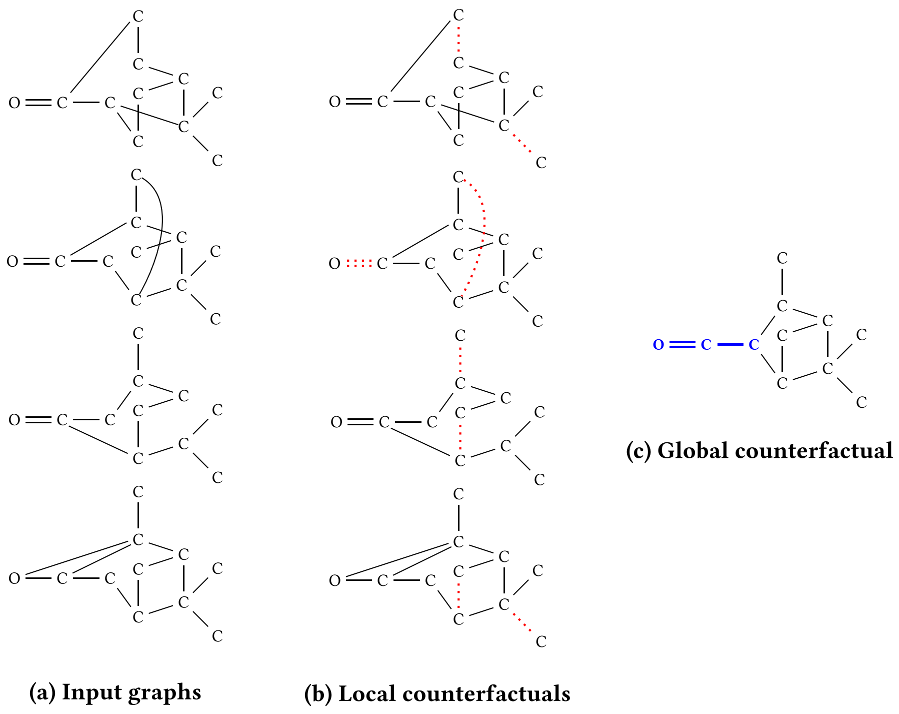

Fig. 5. Illustration of global and local counterfactual explanations for the AIDS dataset. The global counterfactual graph (c) presents a high-level recourse rule—changing ketones and ethers into aldehydes (shown in blue)—to combat HIV, while the edge removals (shown in red) recommended by local counterfactual examples (b) are hard to generalize.

input graphs. Our goal is to understand why the input graphs are inactive against AIDS (undesired) and how to obtain the desired property with minimal changes.

The local counterfactuals in Figure 5(b) attribute the classification results to different edges in individual graphs (shown as red dotted lines) and recommend their removal to make input graphs active against HIV. Note that while only two edits are proposed for each individual graph, they appear at different locations, which are hard to generalize for a global view of the model behavior. In contrast, the global counterfactual graph from GCFExplainer presents a high-level recourse rule. Specifically, the carbon atom with the carbon–oxygen bond is connected to *two* other carbon atoms in the input graphs, making them ketones (with a C=O bond) or ethers (with a C–O bond). On the other hand, the global counterfactual graph highlights a different functional group, aldehyde (shown in blue), to be the key for combating AIDS. In aldehydes, the carbon atom with a

Table 5. The Ratio of Connected Graphs for Both Input Graphs and Counterfactual Explanations across All Methods and Datasets

	NCI1		**Mutagenicity**		**AIDS**		**Proteins**	
	Ratio	Difference	Ratio	Difference	Ratio	Difference	Ratio	Difference
Input Graphs	0.9188		0.9651		0.9973		0.9617	
RCExplainer	0.9681	<u>0.0493</u>	0.8621	<u>−0.1030</u>	0.9493	<u>−0.0480</u>	0.9671	**0.0054**
CFF	0.5263	−0.3925	0.6848	−0.2803	0.5192	−0.4781	0.7487	−0.2130
GCFExplainer	0.9229	**0.0041**	0.9578	**−0.0073**	0.9847	**−0.0126**	0.9754	<u>0.0137</u>

GCFExplainer produces counterfactuals with connectivity closer (i.e., low difference) to input graphs. We highlight the best-performing method in bold and the second best-performing method with underlines.

carbon–oxygen bond is only connected to *one* other carbon atom, leading to different chemical properties compared to ketones and ethers. Indeed, aldehydes have been shown to be effective HIV protease inhibitors [48].

Finally, this case study also demonstrates that counterfactual candidates found by GCFExplainer are better for global explanation than local counterfactuals. We note that while the GED between the local counterfactuals and their corresponding input graphs is only 2, they do not cover other similarly structured input graphs (with distance > 5). Meanwhile, our global counterfactual graph covers all input graphs (with distance ≤ 4).

5.4 Consistency

The consistency of graph properties between counterfactual explanations and the input graphs is an important factor for their practical application in real-world scenarios. For example, in drug discovery, we want the drug candidates (i.e., counterfactuals) to be valid molecules that can also be manufactured in an easy manner, like those provided as input graphs. While evaluating the consistency of such complicated and domain-specific graph characteristics is beyond the scope of this work, here, we look at a simpler property, the connectivity of our generated counterfactuals, and compare it against the original input graphs. Table 5 presents the percentage of connected graphs for input graphs, counterfactual explanations by all methods, and their differences across the datasets. GCFExplainer generally outperforms the baselines in terms of maintaining the original connectivity. The primary reason for this advantage is that, during our VRRW, we refrain from removing any bridges from the graph. This approach ensures that the number of connected components remains unchanged in the graph.

5.5 Robustness

The input graphs to the GNN explainers can be noisy or incomplete [33, 64], and sometimes even under adversarial attacks [5, 9]. Here, we evaluate the robustness of GCFExplainer against random edge flips. Specifically, we randomly choose m node pairs from each input graph and perform an edge flip—removing the edge if it already exists or adding it otherwise. We evaluate the robustness of our model across different values of m, ranging from 1 to 5.

Table 6 compares the recourse coverage of GCFExplainer, its attacked variants (denoted as GCF$_{attack=m}$), and the baselines. Note that the reported performance of baselines is without any noise or attack, yet their explanations are actually vulnerable to them [25]. While the coverage performance of GCFExplainer decreases when the noise level increases, the loss is not substantial enough to negate its superior performance compared to the baselines without noise. The performance loss is most significant in AIDS, which is expected as graphs in this dataset are generally

Table 6. Comparison of Coverage Performance for GCFExplainer, Its Attacked Variants, and Baselines across Different Summary Sizes for All Datasets

	NCI1						**Mutagenicity**					
Summary Size	1	5	10	25	50	100	1	5	10	25	50	100
GCFExplainer	0.057	0.179	0.279	0.441	0.568	0.694	0.117	0.282	0.371	0.508	0.617	0.728
GCF$_{attack=1}$	0.051	0.179	0.264	0.411	0.536	0.660	0.097	0.262	0.346	0.485	0.585	0.690
GCF$_{attack=2}$	0.050	0.160	0.235	0.382	0.488	0.599	0.097	0.240	0.329	0.468	0.570	0.673
GCF$_{attack=3}$	0.044	0.145	0.221	**0.348**	**0.440**	**0.529**	0.101	**0.249**	**0.336**	**0.468**	**0.562**	**0.650**
GCF$_{attack=4}$	**0.045**	**0.138**	**0.198**	0.292	0.367	0.437	**0.110**	0.230	0.293	0.398	0.494	0.588
GCF$_{attack=5}$	0.038	0.110	0.155	0.235	0.295	0.342	0.085	0.206	0.272	0.365	0.452	0.537
RCExplainer	0.031	0.101	0.152	0.227	0.278	0.314	0.098	0.242	0.320	0.435	0.521	0.596
CFF	0.043	0.116	0.176	0.293	0.376	0.458	0.095	0.235	0.304	0.429	0.525	0.619
	AIDS						**Proteins**					
Summary Size	1	5	10	25	50	100	1	5	10	25	50	100
GCFExplainer	0.037	0.107	0.147	0.206	**0.263**	**0.335**	0.022	0.068	0.109	0.194	0.265	0.402
GCF$_{attack=1}$	0.029	**0.065**	**0.090**	**0.141**	0.184	0.234	0.019	0.060	0.093	0.150	0.219	0.339
GCF$_{attack=2}$	**0.022**	0.054	0.079	0.119	0.154	0.191	0.014	0.046	0.077	0.134	0.202	0.303
GCF$_{attack=3}$	0.016	0.046	0.063	0.091	0.109	0.118	0.014	0.049	0.071	0.112	0.180	0.186
GCF$_{attack=4}$	0.014	0.046	0.065	0.098	0.118	0.122	0.014	0.041	0.055	0.096	**0.156**	**0.156**
GCF$_{attack=5}$	0.009	0.028	0.041	0.060	0.071	0.071	**0.011**	**0.038**	**0.052**	**0.093**	0.117	0.117
RCExplainer	0.019	0.060	0.090	0.140	0.187	0.236	0.008	0.030	0.046	0.087	0.134	0.134
CFF	0.011	0.025	0.034	0.043	0.043	0.043	0.006	0.025	0.038	0.079	0.118	0.118

Despite GCFExplainer experiencing performance losses with increasing attack strength, GCFExplainer still outperforms the baselines in most cases. We highlight such cases with the strongest attack strength in bold. For example, the coverage of GCF$_{attack=4}$ for NCI1 with summary size 1, 0.045, is highlighted in bold, demonstrating GCFExplainer still outperforms all baselines while its input graphs are attacked by 4 edge flips. We highlight the best-performing variant in bold.

smaller. Intuitively, our method can counter the attacks by negating them first during the random walk exploration and then proceeding with the original input graph, as long as such a direction leads to higher importance values. Formally evaluating this effect as well as further enhancing the robustness of our current method can be an interesting future direction.

5.6 Ablation Study

We then conduct an ablation study to investigate the effectiveness of GCFExplainer components. We consider three alternatives:

— GCFExplainer-NVR: no vertex-reinforcement ($N(v) = 1$)
— GCFExplainer-NIF: no importance function ($I(v) = 1$)
— GCFExplainer-NDT: no dynamic teleportation ($p_\tau(G) = 1/|\mathbb{G}|$)

The coverage results are shown in Table 7. We observe decreased performance when any of GCFExplainer components is absent.

5.7 Convergence Analysis

In this subsection, we show the empirical convergence of VRRW based on the Mutagenicity dataset in Figure 6. We observe that the coverage performance for different summary sizes starts to converge after 15,000 iterations and fully converges after 50,000 iterations, which is the number we applied in our experiments.

Table 7. Ablation Study Results Based on Recourse Coverage

	NCI1	Mutagenicity	AIDS	Proteins
GCFExplainer-NVR	24.56%	35.44%	11.33%	8.56%
GCFExplainer-NIF	13.29%	29.16%	4.54%	6.83%
GCFExplainer-NDT	27.34%	36.35%	14.05%	9.28%
GCFExplainer	**27.85%**	**37.08%**	**14.66%**	**10.93%**

We highlight the best-performing variant in bold.

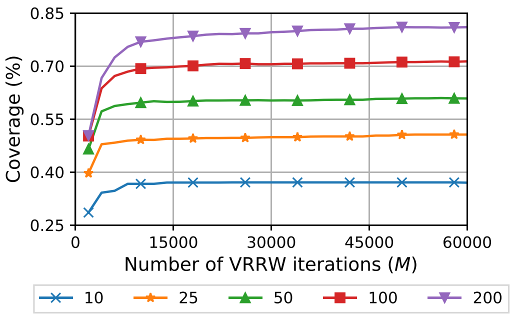

Fig. 6. Convergence of VRRW for the Mutagenicity dataset based on recourse coverage with different summary sizes. VRRW fully converges after M = 50,000 iterations.

Table 8. Sensitivity Analysis on α, the Weight between Individual Coverage and Gain of Coverage in the Importance Function

	NCI1	Mutagenicity	AIDS	Proteins
$\alpha = 0.0$	**27.85%**	36.87%	12.83%	10.11%
$\alpha = 0.5$	27.50%	36.59%	**14.66%**	10.38%
$\alpha = 1.0$	22.27%	**37.08%**	13.99%	**10.93%**

We highlight the best-performing variant in bold.

5.8 Sensitivity Analysis

The only hyperparameter of GCFExplainer we tune is α in Equation (3) that weights the individual coverage and gain of coverage for the importance function. Table 8 shows the results based on different α. While GCFExplainer outperforms baselines with all different α, we observe that individual coverage works better for NCI1 and gain of cumulative coverage works better for other datasets.

5.9 K-GCFExplainer Performance

In this subsection, we present the performance of K-GCFExplainer with different values of k (number of clusters) compared to the original model GCFExplainer. Table 9 illustrates the coverage performance of GCFExplainer and K-GCFExplainer with k set to 2, 3, 4, and 5. The results suggest that while the best k for K-GCFExplainer differs across datasets, it still demonstrates performance improvement across different summary sizes. This is especially evident for the NCI1 and Mutagenicity datasets. On the other hand, GCFExplainer achieves better performance in AIDS.

Table 9. Comparison of Coverage Performance for GCFExplainer and K-GCFExplainer with Different Values of k across Various Summary Sizes for All Datasets

	NCI1						**Mutagenicity**					
Summary Size	1	5	10	25	50	100	1	5	10	25	50	100
GCFExplainer	0.057	0.179	0.279	0.441	0.568	0.694	0.117	0.282	0.371	0.508	0.617	0.728
K-GCFExplainer$_{k=2}$	0.055	0.176	0.273	0.434	0.569	0.698	0.109	0.281	0.375	**0.518**	**0.624**	**0.728**
K-GCFExplainer$_{k=3}$	**0.059**	0.180	0.279	0.443	0.563	0.693	**0.118**	**0.288**	**0.376**	0.516	0.619	0.723
K-GCFExplainer$_{k=4}$	0.049	**0.183**	**0.281**	0.444	0.569	0.699	0.115	0.280	0.372	0.512	0.614	0.724
K-GCFExplainer$_{k=5}$	0.050	0.179	0.275	**0.446**	**0.573**	**0.702**	0.109	0.260	0.368	0.511	0.620	0.727
	AIDS						**Proteins**					
Summary Size	1	5	10	25	50	100	1	5	10	25	50	100
GCFExplainer	0.037	0.107	0.147	0.206	0.263	0.335	0.022	0.068	0.109	0.194	0.265	0.402
K-GCFExplainer$_{k=2}$	0.035	0.096	0.130	0.185	0.239	0.298	0.022	**0.077**	0.109	0.191	0.262	0.399
K-GCFExplainer$_{k=3}$	0.035	0.097	0.132	0.189	0.244	0.308	0.022	0.071	0.101	0.178	0.246	0.383
K-GCFExplainer$_{k=4}$	0.028	0.092	0.127	0.187	0.240	0.301	0.022	**0.077**	0.109	0.178	0.246	0.369
K-GCFExplainer$_{k=5}$	0.028	0.087	0.125	0.183	0.238	0.298	0.019	0.066	0.098	0.180	0.251	0.380

We highlight the best-performing method from K-GCFExplainer in bold and the performance of K-GCFExplainer when it either matches or outperforms GCFExplainer underlined. Notably, K-GCFExplainer exhibits consistently superior performance compared to GCFExplainer for NCI1 and Mutagenicity. The AIDS dataset is the only dataset where we have not seen performance gains, which has the lowest silhouette scores (or weakest clustering structure), as shown in Figure 7.

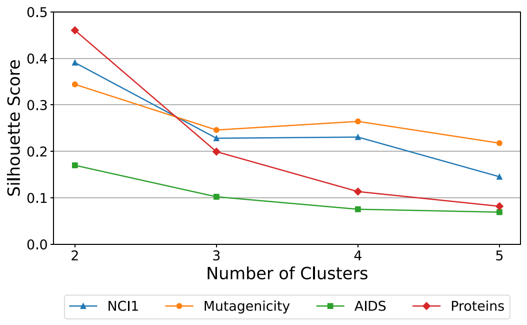

Fig. 7. Silhoutte scores based on the number of clusters for input graphs of each dataset. Higher silhouette scores are associated with stronger clustering structures. We see that the clustering structure of AIDS is the weakest among all datasets, explaining the absence of performance gains for clustering-based K-GCFExplainer.

We hypothesize that the input graphs of AIDS datasets cannot be easily clustered compared to other datasets. To validate this hypothesis, we compute the silhouette scores [46] for the clustering results in Figure 7. Higher sihouette scores are associated with stronger clustering structures. In our case, AIDS indeed has a much lower silhouette score, indicating its lack of a strong clustering structure.

The incorporation of a graph clustering component into GCFExplainer has the potential to enhance the effectiveness of the generated counterfactual summary when carefully designed. Additionally, note that K-GCFExplainer is a two-step algorithm that initially calculates clusters, and then runs GCFExplainer. One could argue that training these two steps end-to-end may further improve performance. We believe that this can be a promising future direction in the problem of global counterfactual explainers.

Table 10. Counterfactual Candidates Generation Time Comparison in Terms of Seconds

	NCI1	**Mutagenicity**	**AIDS**	**Proteins**
RCExplainer	30,454	52,549	29,047	8,444
CFF	22,794	31,749	21,296	**6,412**
GCFExplainer	<u>19,817</u>	<u>24,006</u>	<u>2,615</u>	19,246
GCFExplainer-S	**19,365**	**18,798**	**2,539**	<u>7,429</u>
K-GCFExplainer$_{k=3}$	*13,209*	*18,035*	*1,078*	*10,360*
K-GCFExplainer-S$_{k=3}$	*12,437*	*11,987*	*968*	*6,258*

GCFExplainer (-S) has a competitive running time albeit exploring more counterfactual graphs compared to baselines. Additionally, K-GCFExplainer scales better compared to GCFExplainer for both the original and sampled versions. We highlight the best-performing and second best-performing method among GCFExplainer and the baselines in bold and with underlines. Results for K-GCFExplainer are italicized to demonstrate their improved running time compared to GCFExplainer.

5.10 Running Time

Table 10 summarizes the running times of generating counterfactual candidates based on different methods. GCFExplainer has a competitive running time albeit exploring more counterfactual graphs in the process. We also include results for GCFExplainer-S which samples a maximum of 10,000 neighbors for computing the importance at each step. It achieves better running time at a negligible cost of 3.3% performance loss on average. Additionally, K-GCFExplainer(-S) exhibits better scalability compared to GCFExplainer, as each cluster run is independent and can be parallelized. We set $k = 3$ and report the maximum running time of three cluster runs. Finally, summarizing the counterfactual candidates takes less than a second for all methods.

6 Related Work

Explanations for GNN. There is much research on explaining GNNs. The first proposed method, GNNExplainer [68], finds the explanatory subgraph and subfeatures by maximizing the mutual information between the original prediction and the prediction based on the subgraph and subfeatures. Later, PGExplainer [30] provides an inductive framework extracting GNN node embeddings and learns to map embedding pairs to the probability of edge existence in the explanatory weighted subgraph. PGMExplainer [58] builds a probabilistic explanation model that learns new predictions from perturbed node features, performs variable selection using Markov blanket of variables, and then produces a Bayesian network via structure learning. In XGNN [69], the authors find model-level explanations by a graph generation module that outputs a sequence of edges using reinforcement learning. TAGExplainer [65] has a two-step approach where the first step involves task-agnostic self-supervised training and the second step is for downstream tasks. GEM [28] extracts subgraphs with a Granger causality and an autoencoder while SubgraphX [70] finds subgraphs using a monte carlo tree search. These explanation methods focus on *factual* reasoning while the goal of our work is to provide a global *counterfactual* explanation for GNNs.

Counterfactual Explanations. There have been attempts to explain GNNs via counterfactual reasoning [1, 2, 29, 50, 56, 71]. CF-GNNExplainer [29], one of the earlier methods, provides counterfactual explanations by parameterizing and applying closed formula of GCN on a learnable perturbed adjacency matrix, which leads to a prediction flip for for node classification. On the other hand, RCExplainer [2] aims to find a robust subset of edges whose removal changes the prediction of the remaining graph by modeling the implicit decision regions based on GNN embeddings. In [1], the authors investigate counterfactual explanations for a more specific class of graphs—the brain networks—that share the same set of nodes by greedily adding or removing edges using a heuristic. More recently, the authors of CFF [50] argue that a good explanation for GNNs should consider both factual and counterfactual reasoning and they explicitly incorporate those objective functions when searching for the best explanatory subgraphs and subfeatures. Another recent method CLEAR [31] provides a generative approach that uses graph variational autoencoder to generate counterfactual graphs. InduCE [56] proposes a new approach to inductive counterfactual reasoning, expanding the perturbation space to include edge additions. All the above methods produce *local* counterfactual examples while our work aims to provide a *global* explanation in terms of a summary of representative counterfactual graphs. Also, recently, [61] applied reinforcement learning to explore chemically valid global counterfactuals. Besides the graph-domain, global counterfactual reasoning is also applied in vector data [11, 26, 52].

7 Conclusion and Future Work

We have proposed GCFExplainer, the first global counterfactual explainer for graph classification. Compared to local explainers, GCFExplainer (and its extension K-GCFExplainer) provide a high-level picture of the model behavior and effective global recourse rules. We hope that our work will not only deepen our understanding of GNNs but also build a bridge for experts from other domains to leverage deep learning models for high-stakes decision-making.

Promising future directions of this research include:

—Extending our work to graph tasks other than graph classification, such as node classification and link prediction.
—Incorporating domain-specific constraints (e.g., the valence of chemical bonds in drug discovery setting) into our explainer framework to generate more useful counterfactuals.
—Collaborating with domain experts to evaluate the usefulness/actionability of the generated counterfactuals.
—Developing a human-in-the-loop system to allow interactions between the algorithm and experts for a more effective drug discovery process.
—Exploring the application scenarios of graph counterfactual explanation in settings beyond drug discovery and validating/adapting the proposed algorithm.

Acknowledgement

The authors thank Sevgi Kosan and Ilhan Kosan for their helpful comments on the chemical properties in the global counterfactual analysis.

References

[1] Carlo Abrate and Francesco Bonchi. 2021. Counterfactual graphs for explainable classification of brain networks. In *SIGKDD*.
[2] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. 2021. Robust counterfactual explanations on graph neural networks. In *NeurIPS*.
[3] Ravinder Bhattoo, Sayan Ranu, and N. M. Krishnan. 2022. Learning articulated rigid body dynamics with lagrangian graph neural network. In *NeurIPS*.
[4] Aritra Bhowmick, Mert Kosan, Zexi Huang, Ambuj Singh, and Sourav Medya. 2024. DGCLUSTER: A neural framework for attributed graph clustering via modularity maximization. In *AAAI*, Vol. 38, 11069–11077.
[5] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial attacks on node embeddings via graph poisoning. In *ICML*.
[6] Karsten Borgwardt, Nicol Schraudolph, and S. V. N. Vishwanathan. 2006. Fast computation of graph kernels. In *NeurIPS*.

[7] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, suppl 1 (2005), i47–i56.

[8] Fabrizio Costa and Kurt De Grave. 2010. Fast neighborhood subgraph pairwise distance kernel. In *ICML*.

[9] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *ICML*.

[10] Paul D. Dobson and Andrew J. Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology* 330, 4 (2003), 771–783.

[11] Riccardo Guidotti. 2022. Counterfactual explanations and how to find them: Literature review and benchmarking. *Data Mining and Knowledge Discovery* (2022), 1–55.

[12] Shubham Gupta, Sahil Manchanda, Srikanta Bedathur, and Sayan Ranu. 2022. TIGGER: Scalable generative modelling for temporal interaction graphs. In *AAAI*.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[14] Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2023. Global counterfactual explainer for graph neural networks. In *WSDM*.

[15] Zexi Huang, Arlei Silva, and Ambuj Singh. 2021. A broader picture of random-walk based graph embedding. In *SIGKDD*.

[16] Zexi Huang, Arlei Silva, and Ambuj Singh. 2022. POLE: Polarized embedding for signed networks. In *WSDM*.

[17] Mingjian Jiang, Zhen Li, Shugang Zhang, Shuang Wang, Xiaofeng Wang, Qing Yuan, and Zhiqiang Wei. 2020. Drug–target affinity prediction using graph neural network and contact maps. *RSC Advances* 10, 35 (2020), 20701–20712.

[18] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. 2023. A survey on explainability of graph neural networks. arXiv:2306.01958. Retrieved from https://arxiv.org/pdf/2306.01958

[19] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-agnostic counterfactual explanations for consequential decisions. In *AISTATS*.

[20] Jeroen Kazius, Ross McGuire, and Roberta Bursi. 2005. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry* 48, 1 (2005), 312–320.

[21] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/pdf/1412.6980

[22] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[23] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.

[24] Mert Kosan, Arlei Silva, Sourav Medya, Brian Uzzi, and Ambuj Singh. 2021. Event detection on dynamic graphs. arXiv:2110.12148. Retrieved from https://arxiv.org/pdf/2110.12148

[25] Mert Kosan, Samidha Verma, Burouj Armgaan, Khushbu Pahwa, Ambuj Singh, Sourav Medya, and Sayan Ranu. 2024. GNNX-BENCH: Unravelling the utility of perturbation-based GNN explainers through in-depth benchmarking. In *ICLR*.

[26] Dan Ley, Saumitra Mishra, and Daniele Magazzeni. 2023. GLOBE-CE: A translation based approach for global counterfactual explanations. In *ICML*. PMLR, 19315–19342.

[27] Yongjiang Liang and Peixiang Zhao. 2017. Similarity search in graph databases: A multi-layered indexing approach. In *ICDE*, 783–794.

[28] Wanyu Lin, Hao Lan, and Baochun Li. 2021. Generative causal explanations for graph neural networks. In ICML. PMLR, 6666–6679.

[29] Ana Lucic, Maartje A. Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. 2022. CF-GNNExplainer: Counterfactual explanations for graph neural networks. In *AISTATS*.

[30] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized explainer for graph neural network. In *NeurIPS*.

[31] Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. 2022. CLEAR: Generative counterfactual explanations on graphs. arXiv:2210.08443. Retrieved from https://arxiv.org/pdf/2210.08443

[32] Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. 2020. GCOMB: Learning budget-constrained combinatorial algorithms over billion-sized graphs. In *NeurIPS*.

[33] Travis Martin, Brian Ball, and Mark E. J. Newman. 2016. Structural inference for uncertain networks. *Physical Review E* 93, 1 (2016), 012306.

[34] Sourav Medya, Mohammad Rasoolinejad, Yang Yang, and Brian Uzzi. 2022. An exploratory study of stock price movements from earnings calls. In *WebConf*.

[35] Qiaozhu Mei, Jian Guo, and Dragomir Radev. 2010. Divrank: The interplay of prestige and diversity in information networks. In *SIGKDD*.

[36] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. 2005. Efficient computation of frequent and top-k elements in data streams. In *ICDT*.

[37] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe W. J. Jiang, Ebrahim M. Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Anand Babu, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. 2020. Chip placement with deep reinforcement learning. arXiv:2004.10746.

[38] Dheepikaa Natarajan and Sayan Ranu. 2016. A scalable and generic framework to mine top-k representative subgraph patterns. In *ICDM*.

[39] Sunil Nishad, Shubhangi Agarwal, Arnab Bhattacharya, and Sayan Ranu. 2021. GraphReach: Position-aware graph neural network using reachability estimations. In *IJCAI*.

[40] Hae-Sang Park and Chi-Hyuck Jun. 2009. A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications* 36, 2 (2009), 3336–3341.

[41] Robin Pemantle. 1992. Vertex-reinforced random walk. *Probability Theory and Related Fields* 92, 1 (1992), 117–136.

[42] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.

[43] Rishab Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakravarthy, Yogish Sabharwal, and Sayan Ranu. 2022. GREED: A neural framework for learning graph distance functions. In *NeurIPS*.

[44] Kaivalya Rawal and Himabindu Lakkaraju. 2020. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. In *NeurIPS*.

[45] Kaspar Riesen and Horst Bunke. 2008. IAM graph database repository for graph based pattern recognition and machine learning. In *SSPR & SPR*. Springer, 287–297.

[46] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20 (1987), 53–65.

[47] Alberto Sanfeliu and King-Sun Fu. 1983. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics* 3 (1983), 353–362.

[48] Edoardo Sarubbi, Pier Fausto Seneci, Michael R. Angelastro, Norton P. Peet, Maurizio Denaro, and Khalid Islam. 1993. Peptide aldehydes as inhibitors of HIV protease. *FEBS Letters* 319, 3 (1993), 253–256.

[49] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-lehman graph kernels. *JMLR* 12, 9 (2011), 2539–2561.

[50] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *WebConf.*

[51] Abishek Thangamuthu, Gunjan Kumar, Suresh Bishnoi, Ravinder Bhattoo, N. M. Anoop Krishnan, and Sayan Ranu. 2022. Unravelling the performance of physics-informed graph neural networks for dynamical systems. In *NeurIPS*.

[52] Stratis Tsirtsis and Manuel Gomez Rodriguez. 2020. Decisions, counterfactual explanations and strategic behavior. In *NeurIPS,* Vol. 33, 16749–16760.

[53] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2023. Graph clustering with graph neural networks. *Journal of Machine Learning Research* 24, 127 (2023), 1–21.

[54] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *FAT*.

[55] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[56] Samidha Verma, Burouj Armgaan, Sourav Medya, and Sayan Ranu. 2024. InduCE: Inductive counterfactual explanations for graph neural networks. *Transactions on Machine Learning Research* (2024). Retrieved from https://openreview.net/forum?id=RZPN8cgqST

[57] Paul Voigt and Axel Von dem Bussche. 2017. *The EU General Data Protection Regulation (GDPR). A Practical Guide* (1st. ed.) Vol. 10, Springer International Publishing, 383 pages.

[58] Minh Vu and My T. Thai. 2020. PGM-Explainer: Probabilistic graphical model explanations for graph neural networks. In *NeurIPS*.

[59] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology* 31 (2017), 841.

[60] Nikil Wale and George Karypis. 2006. Comparison of descriptor spaces for chemical compound retrieval and classification. In *ICDM*.

[61] Danqing Wang, Antonis Antoniades, Kha-Dinh Luong, Edwin Zhang, Mert Kosan, Jiachen Li, Ambuj Singh, William Yang Wang, and Lei Li. 2024. Global human-guided counterfactual explanations for molecular properties via reinforcement learning. In *KDD,* 2991–3000.

[62] Yuke Wang, Boyuan Feng, and Yufei Ding. 2022. QGTC: Accelerating quantized graph neural networks via GPU tensor core. In *PPoPP*.

[63] Yuke Wang, Boyuan Feng, Gushu Li, Shuangchen Li, Lei Deng, Yuan Xie, and Yufei Ding. 2021. GNNAdvisor: An efficient runtime system for GNN acceleration on GPUs. In *OSDI*.

[64] Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. 2019. End to end learning and optimization on graphs. In *NeurIPS*.

[65] Yaochen Xie, Sumeet Katariya, Xianfeng Tang, Edward Huang, Nikhil Rao, Karthik Subbian, and Shuiwang Ji. 2022. Task-agnostic graph explanations. In *NeurIPS*.

[66] Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. 2021. Graph neural networks for automated de novo drug design. *Drug Discovery Today* 26, 6 (2021), 1382–1393.

[67] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011. Like like alike: Joint friendship and interest propagation in social networks. In *WebConf*.

[68] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating explanations for graph neural networks. In *NeurIPS*.

[69] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. XGNN: Towards model-level explanations of graph neural networks. In *SIGKDD*.

[70] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. In *ICML*. PMLR, 12241–12252.

[71] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. 2022. Learning from counterfactual links for link prediction. In *ICML*.

Received 28 December 2023; revised 30 June 2024; accepted 2 September 2024