Generating Robust Counterfactual Witnesses for Graph Neural Networks

Dazhuo Qiu
Aalborg University
Denmark
dazhuoq@cs.aau.dk

Mengying Wang
Case Western Reserve University
USA
mxw767@case.edu

Arijit Khan
Aalborg University
Denmark
arijitk@cs.aau.dk

Yinghui Wu
Case Western Reserve University
USA
yxw1650@case.edu

***Abstract*—This paper introduces a new class of explanation structures, called *robust counterfactual witnesses* (RCWs), to provide robust, both counterfactual and factual explanations for graph neural networks. Given a graph neural network $\mathcal{M}$, a robust counterfactual witness refers to the fraction of a graph G that are counterfactual and factual explanation of the results of $\mathcal{M}$ over G, but also remains so for any "disturbed" G by flipping up to k of its node pairs. We establish the hardness results, from tractable results to co-NP-hardness, for verifying and generating robust counterfactual witnesses. We study such structures for GNN-based node classification, and present efficient algorithms to verify and generate RCWs. We also provide a parallel algorithm to verify and generate RCWs for large graphs with scalability guarantees. We experimentally verify our explanation generation process for benchmark datasets, and showcase their applications.**

I. Introduction

Graph neural networks (GNNs) have exhibited promising performances in graph analytical tasks such as classification. Given a graph G and a set of test nodes $\mathcal{V}_T$, a GNN-based node classifier aims to assign a correct label to each node $v \in \mathcal{V}_T$. GNN-based classification has been applied for real applications in domain sciences such as social networks, chemistry, and biology [1], [2], [3], [4].

For many GNNs-based tasks such as node classification, data analysts or domain scientists often want to be able to understand the results of GNNs by inspecting intuitive, explanatory structures, in the form where domain knowledge can be directly applied [5]. In particular, such explanation should indicate "invariant" representative structures for similar graphs that fall into the same group, *i.e.,* be "robust" to small changes of the graphs, and be both "factual" (that preserves the result of classification) and "counterfactual" (which flips the result if removed from G) [6]. The need of such robust, and both factual and counterfactual explanation is evident for real-world applications such as drug design or cyber security.

Consider the following real-world examples.

Example 1: [Interpreting "Mutagenics" with Molecular Structures]. In drug discovery, GNNs have been applied to detect *mutagenicity* structures, which has an adverse ability that causes mutations [7], [8]. Consider G_1 depicted in Fig. 1, which is a graph representation of a real-world molecular database [9], [10]. The nodes of G_1 refer to the atoms of the molecules, and the edges of G_1 represent the valence bonds between a pair of atoms. Given a set of "test nodes" V_T in G, a GNN-based classifier M is trained for the task to correctly assign, for each of the nodes in V_T, a label: "mutagenic", if it belongs to a chemical compound that is a mutagenicity structure; and "nonmutagenic", otherwise.

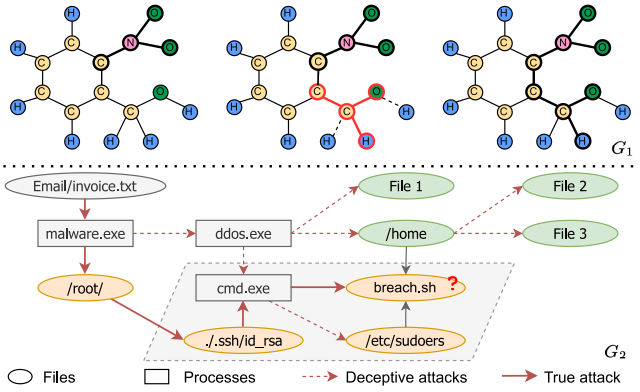

Fig. 1. G_1**: Left-**Bold nodes and edges indicate a counterfactual. **Middle-**Red bold nodes and edges indicate a new counterfactual after deleting dotted edges. **Right-**A counterfactual robust to graph edits. G_2**:** The **shaded area** is a robust counterfactual, also "vulnerable zone" in cyber networks.

A natural interpretation to a chemist would be to identify a subgraph G_1 that contains most of the "mutagenic" nodes as a *toxicophore* structure, that is, a critical fraction of chemical compounds responsible for mutagenicity [11]. Consider the example in Figure 1. On the left is a mutagenic molecule due to the bold nitro group. Meanwhile, the remaining structure after removing the nitro group is considered as non-mutagenic. However, considering a similar molecule that only misses two bonds (dotted edges), which is in the middle, the red bold structure is an aldehyde with a high risk of triggering mutagenicity. To ensure that the discovered mutagenic-related chemical compounds are meaningful to a family of molecules, it is required to find a robust explanation structure, such as the bold indicated structure on the right molecule.

Existing GNN explanation methods may return different structures as the test nodes vary, or carry "noisy" structures such as carbon rings or hydrogen atoms that are not necessarily corresponding toxicophores (as verified in our case study; see Section VII). This may make it difficult for a chemist to discern the potential toxic structure hidden inside normal structures. (e.g., Red bold aldehyde in the middle molecule of Figure 1.) Moreover, instead of analyzing explanations for

each molecule, identifying a common "invariant" explanation that remains the same over a family of similar molecules with few bond differences could improve the efficiency of identifying mutagenic-related chemical compounds. A more effective approach is expected to directly compute explanation structures which are, or contain such toxicophores as robust structures, closer to what a chemist is familiar with. □

Our second example illustrates the need for such structures in cyber security, where new edges are inserted as part of real-world cyber attacks. In such context, the explanation structures themselves may indicate "invariant" structures that needs to be protected *regardless* of attack tactics.

Example 2: [Understanding "Vulnerable Zone" in Cyber Networks]. Graph G_2 in Fig. 1 is a fraction of a provenance graph [12], which shows a multi-stage attack initiated by an email with a malicious attachment disguised as an invoice, targeting a script "breach.sh" to perform a data breach. It consists of nodes that represent files (oval) or processes (rectangle), while the edges depict access actions. Attack tactics are encoded as paths or subgraphs within the graph. A GNN-based node classifier is trained from validated attacks to identify potential attack targets among a set of test nodes, and labels them as "vulnerable" (colored orange in G_2)[13].

An emerging challenge is to effectively detect a two-stage tactic, which first uses, *e.g.,* DDoS as a deceptive attack to various fake targets to exhaust defense resources, to hide the intention of data breaching from an invariant set of high-value files as true target. In such cases, a GNN-classifier can be "fooled" to identify a test node as "vulnerable" simply due to majority of its neighbors wrongly labeled as "vulnerable" under deceptive stage. On the other hand, a robust explanation for those correct labeling of test nodes should stay the same for a set of training attack paths. In such cases, the tactic may switch the first stage targets from time-to-time to maximize the effect of deceptive attacks. As the explanation structures remain invariant despite of various deceptive targets, they may reveal the true intention regardless of how the first stage deceptive targets change. As such, the explanation also suggests a true "vulnerable" files that *should* be protected, thereby help mitigate the impact of deceptive attacks.

Nevertheless, computing such an explanation can be expensive due to the large size of provenance graphs. For example, a single web page loading may already result in 22,000 system calls and yield a provenance graph with thousands of nodes and edges [14], [12]. Especially for sophisticated multi-stage attacks above, it may leave complex and rapidly changing patterns in the provenance graph. Detecting and mitigating such attacks require the ability to process large amounts of data quickly for real-time or near-real-time threat analysis. □

The above examples illustrate the need to generate subgraphs as explanation structures for GNN-based classification, which are (1) *"factual"*, *i.e.,* contributing subgraphs that are consistently responsible for the same assigned class labels to test nodes, (2) *"counterfactual"*, *i.e.,* GNN result changes if such explanation structures are removed, and moreover, (3) *"robust"*, *i.e.,* remain to be a factual and counterfactual explanations upon certain amount of structural disturbance in the graphs. Several methods generate explanation structures (factual or counterfactual) for GNN-based classification [15], [16], [17], [18], [19], yet can not guarantee all the three (factual, counterfactual, and robust) criteria upon edge removal.

Given a pre-trained GNN model M and a graph G, *how do we formally characterize such "invarant" structures?* Moreover, *how can we efficiently compute such "invariant" structures for* GNN*-based node classification over large graphs?*

Contributions. To address the above challenges, we propose a novel class of GNN explanation structures called *robust counterfactual witness*, and develop cost-effective methods to compute them. We summarize our main contributions below.
(1) We formalize the notion of *robust counterfactural witness* (RCW) for GNN-based node classification. An RCW is a subgraph in G that preserves the result of a GNN-based classifier if tested with RCW alone, remains counterfactual (i.e., the GNN gives different output for the remaining fraction of the graph if RCW is excluded), and robust (such that it can preserve the result even if up to k edges in G are changed).
(2) We analyze properties of k-RCWs in regards to the quantification of k*-disturbances*, where k refers to a budget of the total number of node pairs that are allowed to be disturbed. We formulate the problems for verifying (to decide if a subgraph is a k-RCW) and generating RCWs (for a given graph, GNN, and k-disturbance) as explanations for GNN-based node classification. We establish their computational hardness results, from tractable cases to co-NP hardness.
(3) We present feasible algorithms to verify and compute RCWs, matching their hardness results. Our verification algorithm adopts a "*prioritize-expand-verify*" strategy to iteratively expand a k-RCW structure with node pairs that are most likely to change the label of a next test node, such that the explanation remains stable by including such node pairs to prevent the impact of the disturbances.
(4) For large graphs, we introduce a parallel algorithm to generate k-RCWs at scale. The algorithm distributes the expansion and verification process to partitioned graphs, and by synchronizing verified k-disturbances, dynamically refines the search area of the rest of the subgraphs for early termination.
(5) Using real-world datasets, we experimentally verify our algorithms. We show that it is feasible to generate RCWs for large graphs. Our algorithms can produce familiar structures for domain-experts, as well as for large-scale classification tasks. We show that they are robust for noisy graphs. We also demonstrate application scenarios of our explanations.

Related Work. We summarize the related work as follows.

Graph Neural Networks. Graph neural networks (GNNs) are deep learning models designed to tackle graph-related tasks in an end-to-end manner [20]. Notable variants of GNNs include graph convolution networks (GCNs) [21], attention networks (GATs) [22], graph isomorphism networks (GINs) [23],

APPNP [24], and GraphSAGE [25]). Message-passing based GNNs share a similar feature learning paradigm: for each node, update the features of a node by aggregating the counterparts from its neighbors. The features can then be converted into labels or probabilistic scores for specific tasks. GNNs have demonstrated their efficacy on various tasks, including node and graph classification [21], [23], [26], [27], and link prediction [28]. While this study makes case for GNN-based node classification, our algorithms are model-agnostic – thus, can be extended to generate explanations for other GNN-based tasks.

Explanation for GNNs. Several approaches have been proposed to generate explanations for GNNs [29], [30], [31], [32], [33], [15], [34], [16], [19], [17], [18]. Instance-level methods provide input-dependent explanations for each test graph, whereas model-level methods provide a global understanding of GNNs without considering specific input instances [35]. For example, GNNExplainer [15] learns to optimize soft masks for edges and node features to maximize the mutual information between the original and new predictions and induce important substructures from the learned masks. SubgraphX [16] employs Shapley values to measure a subgraph's importance and considers the interactions among different graph structures. XGNN [19] explores high-level explanations by generating graph patterns to maximize a specific prediction. GCFExplainer [17] studies the global explainability of GNNs through counterfactual reasoning. Specifically, it finds a small set of counterfactual graphs that explain GNNs. CF2 [36] considers a learning based approach to infer substructures that are both factual and counterfactual, yet does not consider robustness guarantee upon graph disturbance. CF-GNNExp [34] generates counterfactual explanations for GNNs via minimal edge deletions. Closer to our work is the generation of robust counterfactual explanations for GNNs [37]. The explanation structure is required to be counterfactual and stable (robust), yet may not be a factual explanation. These methods do not explicitly support all three of our objectives together: robustness, counterfactual, and factual explanations; neither scalable explanation generation for large graphs are discussed.

Robustness of GNNs. Robustness of GNN models have been separately studied, such as certifiable robustness [38]. It verifies if a model remains robust under bounded disturbance of edges. In particular, robust training of GNNs that ensures such robustness has been investigated [39]. A general strategy is to identify adversarial edges that, if removed, changes the results of test nodes. If no such structure can be identified, the nodes are certified to be robust. Compared to those works, the key difference in our problem is that we consider robust substructures of the data that preserves the results of a given fixed model as well as its explanation structure. Therefore, prior robust learning cannot be used to identify our explanation structures due to different objectives.

II. Preliminaries

We start with a review of GNNs. We then introduce our robust subgraph explanation structures. Important notations are summarized in Table I.

TABLE I
Symbols & Notations

Symbol	Meaning
$G = (V, E)$	Graph with nodes V and edges E
(X, A)	X: node feature matrix; A: adjacency matrix
$\mathcal{M}$; M; X^k	A GNN-based classifier; inference function of $\mathcal{M}$;
X^i; $\mathbf{Z}$	node embeddings at layer k of $\mathcal{M}$; logits (output) of inference process
L; F	# layers of $\mathcal{M}$; # features per node
C=(G, G_s, V_T, M, k)	a configuration that specifies graph G; inference function M (of a GNN $\mathcal{M}$); test nodes V_T, and k (in k-disturbance)
CW, RCW, k-RCW	counterfactual witness; robust counterfactual witness; k-robust counterfactual witness
G_w; G_s;	a verified witness; a subgraph to be verified
$m^*_{l,c}(v)$	worst-case margin of test node v

A. Graphs and Graph Neural Networks

Attributed Graphs. We consider a connected graph $G = (V, E, T)$, where V is the set of nodes, and $E \subseteq V \times V$ a set of edges. Each node v carries a tuple $T(v)$ of attributes (or features) and their values.

Graph Neural Networks (GNNs). GNNs [20] are a family of well-established deep learning models that extend traditional neural networks to transform graphs into proper embedding representations for various downstream analysis such as node classification. GNNs employ a multi-layer message-passing scheme, through which the feature representation of a node v in the next layer is aggregated from its counterparts of the neighborhoods of v in G at the current layer. A GNN with L layers iteratively gathers and aggregates information from neighbors of a node v to compute node embedding of v. For example, the Graph Convolution Network (GCN) [21], a representative GNN model, adopts a general form as:

$$X^k = \delta(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}X^{k-1}\Theta^k) \quad (1)$$

Here $\hat{A} = A + I$, where I is the identity matrix and A is the adjacency matrix of G. X^k indicates node feature representation in the k-th layer, (with $X^0 = X$ a matrix of input node features), where each row X_v is a vector encoding of a node tuple $T(v)$.[1]. $\hat{D}$ represents the diagonal node degree matrix of $\hat{A}$, $\delta(.)$ is the non-linear activation function, and Θ^k represents the learnable weight matrix for the k-th layer.

APPNPs, a class of Personalized PageRank based GNNs $\mathcal{M}$ [38], specifies the δ function (Eq. 1 with a Personalized PageRank propagation matrix as $(1-\alpha)(I-\alpha\hat{D}^{-\frac{1}{2}}\hat{A})^{-1}\cdot X\Theta$. Here D is the diagonal matrix of node out-degrees with $D_{ii} = \sum_j A_{ij}$, and α is teleport probability. Other notable GNN variants are GraphSage [25] that sample fixed-size neighbors, and GAT [22] that incorporates self-attention.

GNN-based node classification. Given a graph $G = (V, E)$ and a set of labeled "training" nodes V_L, the node classification task is to learn a GNN-*based classifier* $\mathcal{M}$ to infer the labels of a set of unlabeled test nodes. A GNN-based classifier $\mathcal{M}$ of L layers (1) takes as input G=(X, A) and learns to generate

[1]The encoding can be obtained by, e.g., word embedding or one-hot encoding [40], among other featurization techniques

representations X^L, converts them into labels that best fit the labels of V_L ("training examples") in G, and (2) assigns, for each unlabeled test node $v \in V_T$, a class label $l \in$ Ł.

Inference. The label assignment is determined by an "inference process" M of $\mathcal{M}$. Given a set of test nodes V_T, the process outputs a "logits" matrix $Z \in \mathbb{R}^{|V_T| \times |L|}$ in M's output layer, for each node $v \in V_T$ and each label $l \in L$, to be converted to likelihood scores of l given $v \in V_T$; the higher, the more likely v is assigned with l. A softmax function is then used to convert the logits to corresponding labels. To make a (simplified) difference, (1) we consider the inference process as a polynomial-time computable function $M(v, G) = l$ ($v \in V_T$), with a "result" l for v; and (2) we refer to the output logits Z of M simply as "output" of M.

In particular, we consider three trivial cases: $M(v, v) = l$, and denote $M(v, \emptyset)$ or $M(\emptyset, G)$ as "undefined".

Fixed and deterministic GNN. Given a set of test nodes V_T, we say a GNN $\mathcal{M}$ is *fixed*, if its inference process $M(V_T, G)$ does not change for any fixed input test node in V_T. That is, it has all factors that determines the computation of $M(\cdot)$ such as layers, model parameters, among others, remain fixed. We say $\mathcal{M}$ is *deterministic* if $M(\cdot)$ always generates the same output and results for the same input.

In this work, we consider a fixed, deterministic GNN $\mathcal{M}$. In practice, most of the GNNs are deterministic for the need of consistency and accuracy guarantees.

B. Explanation Structures

To characterize our explanation structures as subgraphs, we introduce a notation of *witnesses*.

Witnesses. Given a graph G, class labels Ł, and a GNN $\mathcal{M}$, a test node $v \in V_T$ for which $M(v, G)$=$l \in$ Ł, we say that a connected subgraph G_w of G is

(1) a *factual witness* of result $M(v, G)$=l, if $M(v, G_w) = l$[2];

(2) a *counterfactual witness* (CW) of $M(v, G)$=l, if (1) it is a factual witness of $M(v, G)$=l, and (2) $M(v, G \setminus G_w) \neq l$; and

(3) a *k-robust counterfactual witness* k-RCW of $M(v, G)$=l, if for *any* graph $\hat{G}$ obtained by a *k-disturbance* on $G \setminus G_w$, G_w remains to be a CW for $M(v, \hat{G})$=l.

Here (1) $G \setminus G_w$ is the graph obtained by removing all the edges of G_w from G, while keeping all the nodes; and (2) a *k-disturbance* to a graph G refers to an operation that "flips" *at most* k node pairs (denoted as E_k) from G, including edge insertions and removals. For a k-disturbance posed on $G \setminus G_w$, it only "flips" a set of node pairs E_k that are not in G_w, *i.e.,* it does not insert nor removes edges of G_w. A k-disturbance may capture a natural structural difference of graphs (*e.g.,* chemical compounds that differ with at most k bonds).

Given a set of test nodes $V_T \subseteq V$, we say that G_w is a k-RCW of V_T *w.r.t.* GNN M, if for every test node $v \in V_T$, G_w is a k-RCW for the result $M(v, G)$=l. Note that a k-RCW G_w of the test nodes V_T naturally contains V_T as its nodes.

[2]Note that this indicates G_w containing v as one of its nodes.

We define several *trivial cases*. (1) A single node $v \in V_T$ is a *trivial* factual witness of $M(v, G)$=l, as $M(v, v)$=l trivially; it is a *trivial* CW because $M(\emptyset, G)$ is undefined. (2) The graph G is a trivial factual witness as $M(v, G)$ trivially equals to itself; a trivial CW for $M(v, G)$=l, as it is a trivial factual witness, and $M(v, \emptyset)$ is undefined; and also a trivial k-RCW, since no k-disturbance can be applied to $G \setminus G = \emptyset$. In practice, we aim at finding "non-trivial" k-RCWs as connected subgraphs with at least an edge, and are not G themselves.

Example 3: Consider graph G_2 in Fig. 1, where the target file 'breach.sh' is a test node v. A data breach gains admin access from critical system files such as '/.ssh/id_rsa' or '/etc/sudoers', to execute 'breach.sh' via 'cmd.exe'.

(1) There are two primary *witnesses* to explain its label: (cmd.exe, /.ssh/id_rsa, breach.sh) and (cmd.exe, /etc/sudoers, breach.sh), each includes a factual path that leads to the breach and serves as evidence to explain v's label 'vulnerable'.

(2) Furthermore, a CW is shown as the combined subgraph of these two witnesses, denoted as G_{w_2} (shaded area of G_2 in Fig. 1). Removing edges in G_{w_2} from G will reverse the classification of v to 'not vulnerable', as the remaining part is insufficient for a breach path.

(3) G_{w_2} is also a k-RCW for v's label, where k=3, which is the maximum length of deceptive attacks' path. It indicates that for any attack paths involving up to k deceptive steps, G_{w_2} remains unchanged. It hence suggests the real important files to be protected to mitigate the impact of deceptive attacks. □

We further justify that RCWs serve as a desirable explanation structure by showing an invariant property.

Lemma 1: *Given graph G, a set of test nodes V_T and a fixed deterministic* GNN M, *a k-*RCW *for V_T is a k'-*RCW *for V'_T, for any $k' \in [0, k]$ and any subset $V'_T \subseteq V_T$.* □

Proof: We first prove that for any node $v \in V_T$, a k-RCW G_w for $M(v, G)$ remains to be a k'-RCW for $M(v, G)$, for any $k' \in [0, k]$. One can verify the above result by contradiction. Assume that there is a $k' \in [0, k]$, such that a k-RCW G_s is not a k'-RCW, then (a) G_w is not a factual witness, or it is not counterfactual, both contradicts to that G_w is an RCW; or (b) G_w is a CW but is not robust. Then there exists a k'-disturbance with k' edges E'_k in $G \setminus G_w$, which "disproves" that G_w is a k-RCW. As $k' \leq k$, the same k'-disturbance prevents G_w to be a k-RCW by definition. This contradicts to that G_w is a k-RCW. Hence, G_w remains to be a k'-RCW for $M(v, G)$. As the test node v ranges over V_T, the above analysis holds to verify that G_w remains to be a k'-RCW for any subset $V'_T \subseteq V_T$. Lemma 1 thus follows.

We next study two classes of fundamental problems for RCWs, notably, verification and generation.

III. Verification of Witnesses

For the ease of presentation, we introduce a input configuration for verification problem.

Configuration. A configuration $C = (G, G_s, V_T, M, k)$ specifies a graph G with a set of test nodes V_T, a fixed, deterministic GNN model M, an integer k to specify k-RCW, and a subgraph G_s of G. Specifically, a counterfactual witness CW is a 0-RCW (k=0), *i.e.,* no robustness requirement is posed.

A. Verification Problem

Given a configuration $C = (G, G_s, V_T, M, k)$, the verification problem is to decide if G_s is a k-RCW for V_T *w.r.t.* M. We first study the complexity of two special cases.

Verification of factual witnesses. The witness verification problem takes as input a configuration $C = (G, G_s, V_T, M, 0)$ and decides if G_s is a factual witness for V_T, *i.e.,* for every test node $v \in V_T$, G_s is a factual witness.

We have the following result.

Lemma 2: *The witness verification problem is in* PTIME. □

Similarly, the verification problem for CW is to decide if a given subgraph G_s of G is a counterfactual witness for V_T. We show that this does not make the verification harder.

Lemma 3: *The verification problem for* CW *is in* PTIME. □

We prove the above two results with two verification algorithms, verifyW, and verifyCW, which invokes the inference process M of GNN $\mathcal{M}$ to check if the label $M(v, G_s)$ and $M(v, G \setminus G_s)$ remains to be l, respectively. One may justify that verifyW and verifyCW are in PTIME by observing that the inference function M is PTIME computable, with a cost typically determined by L, $|V|$, $|E|$, d, and F, where L is the number of layers of the GNN $\mathcal{M}$, $d = \frac{|E|}{|V|}$ is the average degree of G, and F is the number of features per node. For example, the inference cost of representative message-passing based GNNs (*e.g.,*GCNs, GATs, GraphSage), or PageRank based APPNPs are in $O(L|E|F + L|V|F^2)$ [41], [42].

We leave the detailed proofs in [43].

Despite the verification of witnesses and CW is tractable, its RCW counterpart is nontrivial.

Theorem 1: *The k-*RCW *verification problem is* NP*-hard.* □

Proof sketch: We make case for APPNPs. Given G_s, we first invoke verifyW and verifyCW to check if it remains to be a CW. If G_s is not a factual witness or a CW, we assert that G_s is not a k-RCW. Otherwise, the problem equals to verify if a CW G_s is a k-RCW. We show that the problem is equivalent to finding a set of k edges from $G \setminus G_s$, s.t. PageRank scores of V_T are maximized. This is achieved via the verification of certifiable robustness of nodes [38], which shows that the output of M that assigns the label l is determined by the personalized PageRank scores of V_T.

We then show the hardness of verification of k-RCW, by performing a PTIME reduction from the *Max-PageRank* problem [44]. Given a graph G_p and a set of edges E, it is to find from E a subset E_k of k edges such that the PageRank scores of a set of target nodes are maximized. By setting G_p as G'=$G \setminus G_s$, and the target nodes as V_T, we construct a PTIME reduction from Max-PageRank to an instance of k-RCW verification problem. As Max-PageRank is NP-complete, the verification of k-RCW is NP-hard. □

Algorithm. We outline a general verification algorithm, denoted as verifyRCW. Given a configuration C, the algorithm first invokes verifyW and verifyCW to check if G_s is a factual witness and a CW, both in PTIME (Lemma 2 and Lemma 3). If G_s remains to be a CW, it next performs a k-round of verification, to verify if G_s is an j-RCW in the j-th round. It early terminates whenever a j-th disturbance is identified, which already disproves that G_s is a k-RCW, given Lemma 1. We present the detailed algorithm and analysis in [43].

B. A Tractable Case

While it is in general intractable to verify k-RCW, not all is lost: We show that there exists a tractable case for the verification problem when the k-disturbance is further constrained by a "local budget", for the class of APPNPs. We introduce two specifications below.

(k, b)-disturbance. Given a graph G, a k-disturbance is a (k, b)-disturbance to G if it disturbs at most k edges, and meanwhile at most b edges for each involved nodes in G. Here b is a pre-defined constant as a local "budget" for a permissible disturbance. In practice, a local budget may indicate the accumulated total cost of "local attack" of a server, or variants of bonds in-degree as seen in chemical compound graphs [10], which are typically small constants (*e.g.,* ≤ 5).

Node robustness. We start with a characterization of node robustness that extends certifiable robustness [38], which verifies if a predicted label can be changed by a k-disturbance. This work makes a case for APPNPs, a class of Personalized PageRank based GNNs $\mathcal{M}$ as aforementioned. Given a configuration $C = (G, G_s, V_T, M, k)$, where M is the inference process for a fixed deterministic APPNP $\mathcal{M}$, a node $v \in V_T$ is *robust w.r.t.* configuration C, if a "worst-case margin" $m^*_{l,*}(v) = \min_{c \neq l} m^*_{l,c}(v) > 0$, where $M(v, G) = l$ is the (true) [3] label of v, and c is any other label ($c \neq l$). We define $m^*_{l,c}(v)$ as:

$$\begin{aligned} m^*_{l,c}(v) &= \min_{E_k \subseteq G \setminus G_s} m_{l,c}(v) \\ &= \min_{E_k \subseteq G \setminus G_s} \pi_{E_k}(v)^T \left(Z_{\{:,l\}} - Z_{\{:,c\}}\right) \end{aligned} \tag{2}$$

where E_k ranges over all the possible (k, b)-disturbances that can be applied to $G' = G \setminus G_s$, and $\pi_{E_k}(v) = \mathbf{\Pi}_{v,:}$ is the PageRank vector of node v in the PageRank matrix $\mathbf{\Pi} = (1 - \alpha)(I_N - \alpha D^{-1} A')^{-1}$, with A' obtained by disturbing the adjacency matrix of G with E_k, *e.g.,* $A'[i, j] = 0$ (resp. 1) if $(v_i, v_j) \in E_k$ (resp. $(v_i, v_j) \notin E_k$). Note that we do not explicitly remove the edges and change G or G', but reflect the tentative disturbing by computing A'.

By verifying $m^*_{l,*}(v) > 0$ (*i.e.,* $m^*_{l,c}(v) > 0$ for any $c \in$ Ł) other than the correct label l of v), it indicates that under any

[3] Notice that here the "robustness" is not a direct guarantee for M to assign "correct" label. Rather to simplify our discussion, we assume a high quality GNN $\mathcal{M}$ with inference process M that assigns the true label for node v.

disturbance of size k in $G \setminus G_s$, M always predicts the label of node v as l *w.r.t.* the output Z.

Theorem 2: *Given the configuration C that specifies* APPNP *$\mathcal{M}$, and when only (k, b)-disturbance is allowed, the verification problem for k-*RCW *is in* PTIME. □

As a constructive proof, we next outline a PTIME algorithm, denoted as verifyRCW. The algorithm verifies if a given subgraph G_s is a k-RCW for a single node $v \in V_T$ *w.r.t.* $\mathcal{M}$ under (k, b)-disturbance. For a configuration C with test set V_T, it suffices to invoke verifyRCW for each $v \in V_T$. We show that such PTIME solution exists with the following condition.

Lemma 4: *Given the configuration C that specifies* APPNP *$\mathcal{M}$, and when only (k, b)-disturbance is allowed, let G_s be a verified* CW *of $M(v, G) = l$ for a node $v \in V_T$, then G_s is a k-*RCW *of $M(v, G)$=l, if and only if $M(v, G \setminus E_k^*) = l$, where $E_k^* = \arg\max_{E_k \subseteq G \setminus G_s, c \neq l} (\mathbf{Z}_{\{:,c\}} - \mathbf{Z}_{\{:,l\}})^T \pi_{E_k}(v)$.* □

Proof sketch: Let $\hat{G}$ be the graph obtained by disturbing E_k^* in $G \setminus G_s$. For the **If** condition, let E_k^* be the node pairs to be disturbed in the corresponding (k, b)-disturbance in $G \setminus G_s$. As E_k^* maximizes $Z_{\{:,c\}} - Z_{\{:,l\}})^T \pi_{E_k}(v)$, which indicates that it minimizes the gap between the likelihoods that *some* label $c \neq l$ being assigned to v, quantified by the worst-case margin $m_{l,c}^*$, for *every* label c that is not l. That is, disturbing E_k^* is "most likely" to change the current result $M(v, G)$=l, *i.e.,* $M(v, \hat{G})$=$c \neq l$, for any label $c \in L$. Then, if $m_{l,c}^* > 0$ for every $c \neq l$ under disturbance E_k^*, $M(v, \hat{G})$ will remain to be l. This means that no (k, b)-disturbance in $G \setminus G_s$ will change v's label, hence G_s is a k-RCW of $M(v, G)$=l by definition.

The **Only If** condition can be shown by contradiction. Assuming there exists a set E_k^* that makes $M(v, \hat{G}) \neq l$, and G_s is still a k-RCW of $M(v, G) = l$; since $\mathcal{M}$ is fixed and deterministic, $M(v, G_s) = l$ is not affected by disturbing E_k "outside" of G_s; yet $M(v, \hat{G}) \neq l$, hence G_s is not a factual witness for $M(v, \hat{G}) = l$, and hence not a CW for $M(v, \hat{G}) = l$, violating the third condition in the definition of k-RCW for G. Hence G_s is not a k-RCW of $M(v, G) = l$. This contradicts to that G_s is a k-RCW of $M(v, G)$=l. □

Based on the above result, it suffices to show that there is a PTIME solution to the optimization problem that computes E_k^*, which maximizes $Z_{\{:,c\}} - Z_{\{:,l\}})^T \pi_{E_k}(v)$. Following the analysis in [38], we present a greedy edge selection strategy to construct E_k^* in PTIME. This gives us the following verification algorithm, as a proof by construction for Theorem 2.

Algorithm. Algorithm 1 first invokes verifyW and verifyCW to check if G_s is a factual witness and a CW (lines 1-5). If G_s remains to be a CW, it then follows an iterative "optimize-and-verify" process, which first computes a set of edges E_k that can optimize $Z_{\{:,c\}} - Z_{\{:,l\}})^T \pi_{E_k}(v)$, and then verifies if the (k, b)-disturbance changes the label.

(1) verifyRCW nontrivially optimizes the policy iteration procedure [38], which goes through multiple rounds of top b edges selection process (lines 6-13). In each round i, it augments an optimal edge set E_k^i by selecting at most b edges E_b^i, explored in the neighboring area of the node v. The edges are selected as the top b edges that can most likely update the label l of v with c. This is achieved by greedily selecting top b adjacent edges (v, v') with the highest "margin" scores $s(v, v')$ as computed in line 12. Let $M(v, G) = l$. A *margin score* for a node $v \in V_T$, a label $l' \in$ Ł $(l \neq l')$ and an edge (v, v') quantifies the gap between the likelihood of v being assigned to l (which should be the largest) and l', if edge (v, v') is removed.

(2) Each time an (i, b)-disturbance is constructed, it invokes verifyCW to verify if the label of v changes by applying (i, b)-disturbances (line 15). If so, G_s is already not a k-RCW as "disapproved" by such an (i, b)-disturbance (Lemma 1) and returns "no". Otherwise, the robustness of G_s is asserted under the tests of all k-disturbances with most influence to model output as assured by the greedy selection (line 17).

Algorithm 1 Algorithm verifyRCW-PPNP (single node)

Input: A configuration $C = \{G, G_s, v, \mathcal{M}, k\}$, integer b;
Output: true if G_s is a k-RCW, false otherwise.

1: **if** verifyW(C) = false **then**
2: **return** false;
3: construct graph $G' := G \setminus G_s$;
4: **if** verifyCW(G', C) = false **then**
5: **return** false;
6: initializes bitmaps A and $\hat{A}$ as adjacency matrix of G'; initializes integer i:=0;
7: initialize edge set E_k^0 with k random edges in G'; a queue Q:=$\{V_T\}$;
8: **while** $E_k^i \neq E_k^{i-1}$ **and** $|E_k^i| < k$ **do**
9: update $\hat{A}$ by "disturbing" E^i in G; /*flip entries*/
10: update X^L s.t $I_N - \alpha \hat{D}^{-\frac{1}{2}} \hat{A})^{-1} \cdot X^L$=$\mathbf{Z}_{\{:,c\}} - \mathbf{Z}_{\{:,l\}}$;
11: **for each** $v \in Q$ **do**
12: choose top-b adjacent edges E_b^i of v in G' with highest scores $s(v, v')$=1-2 $\cdot A_{(v,v')}$ $(X_{v'} - \frac{X_v - X'_v}{\alpha})$
13: $E_k^i := E_k^i \cup E_b^i$; i:=$i + 1$;
14: Q:=$Q \setminus \{v\} \cup \{v'\}$ $(v' \in E_b^i)$
15: **if** $M(v, G \setminus E_k^i) \neq l$ **then**
16: **return** false;
17: **if** $M(v, G \setminus E_k^i) = l$ **then**
18: **return true**;
19: **return false**

Analysis. The correctness is ensured by showing that the greedy selection process correctly solves the Personalized PageRank maximization problem, which meanwhile minimizes the worst-case margin (Lemma 4) following the analysis in [38], where a PTIME solution is provided. For time cost, verifyRCW verifies at most $k \times b$ disturbances, and both verifyW and verifyCW remains to be in PTIME under (k, b)-disturbance (Lemma 2 and Lemma 3). The total cost is thus in $O(kbLF(|E| + |V|F))$ for representative GNNs. We leave the detailed analysis in [43].

IV. RCW Generation Problem

The generation problem, unlike verification, is to compute a nontrivial k-RCW if exists, for a given configuration $C = (C, \emptyset, V_T, M, k)$, (or simply (C, V_T, M, k) as the k-RCW is to be computed). We can verify the following result.

Theorem 3: *Given a configuration $C = (C, V_T, M, k)$, the k-RCW generation problem is in general co-NP-hard.* □

Proof sketch: It suffices to consider a "single node". A problem is co-NP-hard if its complementary problem is NP-complete. The decision problem of k-RCW generation for a configuration $C = (C, v, M, k)$ is to decide if there exists a subgraph G_s such that for any k-disturbance, G_s remains to be a CW for $M(v, \hat{G})$=l, where $\hat{G}$ is obtained by disturbing G with the k-disturbance (see Section II). Its complementary problem is to verify if no subgraph G_s in G can be a k-RCW, *i.e.,* for any subgraph G_s, there always exists some k-disturbance that "disaproves" G_s to be a k-RCW. We make a case for APPNPs for the complementary problem. (1) There is an NP algorithm that non-deterministically guesses a subgraph G_s and a k-disturbance, and solves a verification problem given G_s and k-disturbance in PTIME with inference test. (2) The NP-hardness follows from Theorem 1, which shows that the verification problem of k-RCW for APPNPs is already NP-hard. Putting these together, Theorem 3 follows. □

One may consider a more feasible case when the verification problem is tractable. Following Theorem 2, we identify a tractable case for APPNPs under (k, b)-disturbances.

Lemma 5: *Given a configuration $C = (C, V_T, M, k, b)$ and only (k, b)-disturbances are allowed, the k-RCW generation problem for APPNPs is in PTIME.* □

We next present a feasible algorithm to generate k-RCW for GNN $\mathcal{M}$. The algorithm ensures to output a k-RCW in general cases; and in particular, ensures a PTIME process for APPNPs under (k, b)-disturbance.

V. Generating Robust Witnesses

We present our main result below.

Theorem 4: *Given a configuration C that specifics G, $\mathcal{M}$, V_T and k, there is an algorithm that generates a k-RCW in $O((N + |G|)(L|E|F + L|V|F^2)$ time, where N is the total number of verified k-disturbances. Specifically for APPNPs under (k, b)-disturbances, it computes a k-RCW in $O(kbLF(|E| + |V|F) + k|V_T|)$ time.* □

Our idea, inspired by Lemma 4, processes a set of test nodes V_T "one node at a time", and iteratively grows G_s with a "prioritize-expand-verify" strategy for each node: (1) reorder unvisited test nodes in V_T to find the node that G_s is likely to continue to be a k-RCW, *i.e.,* its label is *not* likely to be changed given current G_s and k-disturbances in $G \setminus G_s$ (see Procedure PrioritizeQ); (2) *approximate* the optimal set of node pairs E_k^* that can minimize the worst-case margin $m_{l,c}^*$ of v (Eq. 2), *i.e.,* most likely to change its label if "flipped" [4] (see Procedure Expand); and (3) augment G_s to contain E_k^*, such that it "secures" G by preventing disturbances from E_k^*. As such, $M(v, G \setminus G_s)$ is most likely[5] to be l, and G_s will remain to be a k-RCW for $M(v, G \setminus G_s)$ determined by the next round of verification.

Algorithm 2 Algorithm RoboGExp

Input: A configuration $C = \{G, V_T, M, k\}$, integer b (local budget);
Output: A k-RCW G_s of V_T *w.r.t.* $\mathcal{M}$.

1: Initializes G_s:=$\mathcal{V}_T$; set E_k:=$\emptyset$; queue Q := V_T;
2: **while** $G \setminus G_s \neq \emptyset$ **do**
3: update C:=(G, Q, M, k);
4: **if** verifyRCW(C, G_s)=false **then**
5: **return** G;
6: **while** $Q \neq \emptyset$ **do**
7: Q:={$v | v$ is not in $G_s, v \in V_T$};
8: Q:=PrioritizeQ(Q, C, G_s);
9: node v:=Q.dequeue();
10: G_s:=Expand (v, G_s, C);
11: **if** verifyRCW(C, G_s)=false **then**
12: **return** G;
13: **return** G_s;
14: **return** G_s;

Procedure: Expand(v, G_s, C)

1: set N_k:= {$u | u$ is in k-hop neighbor of v};
2: queue $v.Q_k$:=$\emptyset$; set E_k:=$\emptyset$;
3: **while** there is an unvisited pair (u, u') in $N_k \cap G \setminus G_s$ **do**
4: $v.Q_k$:=(u, u');
5: update worst-case margin $m_{l,c}^*(v)$;
6: update E_k as all pairs (u, u') that maximizes the worst-case margin;
7: augment G_w with E_k;
8: **return** G_s;

Algorithm. The algorithm, denoted as RoboGExp and illustrated in Algorithm 2, outputs a non-trivial k-RCW G_s if exists; and terminates with trivial k-RCW G or V_T by default. It uses (i) a queue Q to prioritize the processing of the test nodes V_T, (ii) for each test node $v \in V_T$, a queue of node pairs $v.Q_k$, to track the k-disturbance that most likely to change its own label. (1) It initializes G_s as a trivial CW set V_T (line 1), and grows G_s up to the trivial k-RCW G to guarantee the termination (lines 2-12). (2) The expansion of G_s is controlled by the prioritized node-by-node process (lines 7-12). Given a verified k-RCW G_s for a fraction of V_T (lines 3-5); it invokes procedure PrioritizeQ to prepare the next test node v to be processed (line 8), and invokes procedure Expand to process v and grow G_s. (3) The process continues until either a nontrivial k-RCW G_s for V_T is identified and returned (line

[4]For APPNPs, maximizing Personalized PageRank score as in Lemma 4.
[5]Here the optimality of E_k^* is not guaranteed by the procedure, and needs to be verified, for general cases

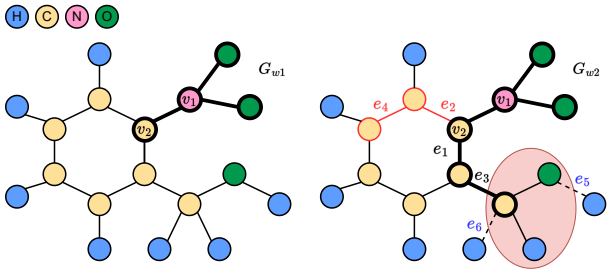

Fig. 2. Generating robust witness for G_1 with expansion and verification.

13), or no such structure can be found, and trivial explanations are returned (lines 5, 12 and 14).

Procedure Expand. The procedure Expand iteratively explores an unvisited node pair (u, u') in $G \setminus G_s$ and iteratively estimates if the worst-case margin is affected by including (u, u') in a k-disturbance. If $m^*_{l,c}$ is less than 0, a k-disturbance is enlisted to $v.Q_k$ to augment G_s for further verification.

In particular for APPNPs under (k, b)-disturbance, Expand implements the greedy policy iteration process in the procedure verifyRCW-PPNP (which replaces lines 5-6 with an implementation that simulates lines 7-16 of verifyRCW-PPNP), to expand G_s with the optimal edge set E^*_k.

Example 4: Figure 2 illustrates the expansion and verification process. Consider the mutagenic molecule graph G_1 with two test nodes $V_T = \{v_1, v_2\}$, and set k=2, b=1. Both nodes v_1 and v_2 are labeled 'mutagenic' by a APPNPs $\mathcal{M}$. A currently verified 2-RCW G_{w1} is identified for node v_1 (with bolded edge), which contains v_1 and its three direct neighbors.

(1) Given G_{w1}, RoboGExp prioritized the test nodes and starts with v_2. On the right side, RoboGExp invokes Expand to estimate, from the 1-hop neighboring area of both test nodes v_1 and v_2, those node pairs that are most likely to maximize the worst-case margin of v_2 from 'mutagenic' to 'nonmutagenic' if disturbed. This promotes options to expand G_{w1} with either e_1 or e_2, and removal of e_5.

(2) In the next round of expansion (line 3 of Expand), Expand continues to explore unvisited node pairs from the 2-hop of the test nodes, and promotes to add either e_3 or e_4, respectively, and remove e_6. In the verification phase, G_{w1} with additional e_2 and e_4 is verified to be not a CW. On the other hand, G_{w1} with additional e_1 and e_3, with e_5 removed (3 node pairs) is verified to be a CW, and is updated as a new 2-RCW G_{w2}. Indeed, we found that the expanded explanation with enforced factual and counterfactual robustness correctly incorporated the aldehyde structure (red area) a main toxic fraction that explains the label of v_2. □

Analysis. Algorithm RoboGExp always terminates with either a non-trivial k-RCW, or trivial cases. For the time cost, (1) it takes a total verification cost in $O(N \times I)$ (line 4), where N is the total number of k-disturbances being verified, and I is the GNN-specific single inference cost, which is typically bounded in $O(L|E|F + L|V|F^2)$ for representative GNNs (*e.g.,*GCNs, GATs, GraphSage, APPNPs) [41], [42]. Procedure Expand takes $O(|G|I)$ time, as G_s grows at most $|V| + |E|$ times[6]. The total cost is thus in $O((N + |G|)L|E|F + L|V|F^2)$, for representative GNNs.

Specifically for APPNPs, the algorithm RoboGExp invokes procedure verifyRCW-PPNP to verify at most $k \times b$ (k, b)-disturbances for each node $v \in V_T$, and for each verification, takes time $O(L|E|F + L|V|F^2)$ for L-layered APPNPs with F features per node. Thus the total verification cost is in $O(kbLF(|E| + |V|F))$. Observe that the optimal node pairs E^*_k of size k for a node v only needs to be computed once in verifyRCW-PPNP (line 11) or by Expand (line 6) via a deterministic process, hence the procedureExpand *does not* incur additional verification, and incurs a total cost in $O(k|V_T|)$. Thus the total cost is in $O(kbLF(|E| + |V|F) + k|V_T|)$.

Given the above analysis, Lemma 5 and Theorem 4 follow.

Minimality of k-RCWs. It is often desirable to compute small explanations in terms of the number of edges to be inspected. We capture this with a minimality measure. A subgraph G_s of G is a *minimal* k-RCW, if for any subgraph G'_s obtained by disturbing an edge from G_s, G'_s is not a k-RCW of V_T *w.r.t.* $\mathcal{M}$. We show that with minor revisions, algorithm RoboGExp can readily generates minimal RCWs without incurring additional overhead. Specifically, RoboGExp uses a revised Expand and verifyRCW as follows. (1) Instead of a batch estimation and expansion of multiple node pairs, Expand is enforced to perform a "best-at-a-time" growth in the procedure Expand, such that the explanation is verified each time an edge is inserted to G_s. (2) Upon the expanded G_s, verifyRCW performs at most $|G_s|$ round of additional inference tests (by invoking M), and in each round, removes an edge e in G_s to test if the rest remains a k-RCW. Note that it suffices for verifyRCW to *incrementally* check if $G_s \setminus \{e\}$ remains to be a k-RCW, as G_s is a verified k-RCW for $M(v, G)$=l, and verifyRCW only considers additional k-disturbances that must include e, which "hot starts" from $k-1$-RCW verification in verifyRCW.

Due to limited space, we present the details in [43].

VI. Parallel Witness Generation

When the graph is large, the verification of k-RCW can be expensive and is a major bottleneck. We next present a parallel algorithm, denoted as paraRoboGExp and illustrated as Algorithm VI. Our idea is to parallelize the verification into a smaller manageable tasks $<G_i, G_s, j>$, which checks "if there is a disturbance from G_i alone that disprove G_s to be a j-RCW of $M(v, G)$=l", and perform parallelized verification for each fragment G_i of G.

Given Lemma 1, we observe the following result.

Lemma 6: *Given a subgraph G_i of $G \setminus G_s$, If there is a j-disturbance E^i_j in $G_i \setminus G_s$, and applying E^i_j to G changes the result $M(v, G) = l$, then G_s is not a k-RCW for $M(v, G)$=l, for any $k \geq j$.* □

[6]Note that although the growth considers node pairs in $|V| \times |V|$, G_s does not re-add node pairs and grows up to size $|G|$.

Algorithm 3 Algorithm paraRoboGExp

Input: Graph $G = \{G_1, G_2, \ldots, G_n\}$, test nodes V_T, and k.
Output: A k-RCW G_s of V_T *w.r.t.* $\mathcal{M}$.

/*At Coordinator s_0*/
1: Initializes G_s:=$\mathcal{V}_T$; set E_k:=$\emptyset$; queue Q := V_T;
2: initialize a bitmap array B;
3: **while** $Q \neq \emptyset$ **do**
4: Q:={$v | v$ is not in $G_s, v \in V_T$};
5: Q:=PrioritizeQ(Q, C, G_s);
6: node v:=Q.dequeue();
7: **for** $j = 1$ to k **do**
8: $G^i_{s_j}$:= paraExpand(C, G_i, G_s, j, B_j); /*parallel expansion*/
9: paraverifyRCW$(C, G^i_{s_j}, G_s, j, B_j)$;/*Parallel verification*/
10: synchronize B;
11: G_{s_j}= $\bigcup_{i=1}^{n} G^i_{s_j}$;
12: **if** verifyRCW(C, G_{s_j}, B_j)=false **then**
13: update B_j; **continue**;
14: **return** G_s;

Partition. The algorithmparaRoboGExp works with a coordinator site S_0 and n workers $\{S_1, \ldots S_n\}$. Given a configuration $C = (G, M, V_T, k)$, it ensures a "inference preserving partition" such that local verification and inference can be conducted. (1) The GNN model $\mathcal{M}$ is duplicated at each site. (2) The graph G is fragmented into n partitions $\{G_1, \ldots G_n\}$ where each worker S_i processes one fragment G_i. For each "border node" v that resides in two fragments G_i and G_k, its k-hop neighbors are duplicated in both G_i and G_k, to ensure no data exchange is needed for parallel verification. (3) All local fragments share a compressed bitmap encoding of the adjacency matrix B of G, the node features X^0, and the common V_T, such that $M(v, G)$ can be inferred locally without expensive data communication.

Algorithm. AlgorithmparaRoboGExp is illustrated as Algorithm VI. It starts with the initialization similar to its sequential counterpart RoboGExp. In addition, it uses a global set of bitmaps B to synchronize the k-disturbances that have been verified, to avoid redundant verification. It mainly parallelizes the computation of two procedures: Expand, with procedure paraExpand (line 8), and verifyRCW, with procedure paraverifyRCW (line 9) (not shown). It assembles an expanded global subgraph G_{s_j} as the union of all verified local counterparts (line 11). As this does not necessarily ensure all disturbances are explored, it continues to complete a coordinator side verification, yet does not repeat the verified local ones (lines 12-13) using the assembled bitmap array B.

The procedure paraExpand grows the current verified $j-1$-RCW $G_{s_{j-1}}$ at each site in parallel, and sends the locally expanded $G^i_{s_j}$ to the coordinator with locally expanded node pairs that can optimize the worst case margin $m^*_{l,c}(v)$. Similarly, the procedure paraverifyRCW examines and only sends $G^i_{s_j}$ that remains to be a CW, and updates the bitmap array B with the verified local k-disturbance.

Analysis. The algorithm paraRoboGExp correctly parallelizes the computation of its sequential counterpart RoboGExp, as guarded by the coordinator side verification, and reduces unnecessary data shipment and verification, given Lemma 6. For the parallel cost, there are at most $k|V_T|$ rounds of parallel computation. The total time cost (including communication cost) is in $O(\frac{|G|+|B|)(L|E|F+L|V|F^2)}{n} + I_0 \cdot L|E|F + L|V|F^2)$. Here I_0 refers to the number of k-disturbances verified at the coordinator. paraRoboGExp *avoids* unnecessary redundant verification. This analysis verifies that paraRoboGExp scales well as more processors are used. We leave the details in [43].

VII. Experimental Study

We experimentally evaluate our algorithms with three real-world and one synthetic datasets. We evaluate the following: (**RQ1**): (a) the quality of the k-RCWs generated by RoboGExp in terms of robustness and fidelity measures; (b) the impact of critical factors, including test node size $|V_T|$ and degree of disturbances (k), to the quality; (**RQ2**): (a) the efficiency of RoboGExp and paraRoboGExp for large graphs, (b) the impact of $|V_T|$ and k to efficiency, and (**RQ3**): The scalability of paraRoboGExp. We also perform two case study analyses to showcase real applications of RoboGExp. The codes and datasets are made available at [45], with the full version [43].

A. Experiment settings

Datasets. Our datasets are summarized below (Table II):
(1) BAHouse [15], a synthetic dataset, uses a Barabási-Albert graph as the base with a house motif. Each node has 5 neighbors on average, with motifs labeled 1, 2, and 3 as 'roof', 'middle', and 'ground'. The remaining nodes are labeled 0.
(2) PPI [46], a protein-protein interaction dataset, contains human proteins (nodes) and their interactions (edges). Node features include motif gene sets and immunological signatures, labeled by gene ontology sets reflecting functional properties.
(3) CiteSeer [47] is a citation dataset for computer science. Each node is a publication with edges as citation relations. There are 6 classes: Agents, AI, DB, IR, ML, and HCI. "Agents" specifically pertains to papers related to the field of autonomous agents and multi-agent systems. Each node features a binary vector for keyword occurrences.
(4) Reddit [48] is a large-scale social network dataset with hundreds of millions of edges and a set of nodes (posts) with features and node classes. Node features are word vectors, and node labels are communities in which the post belongs.

For injecting k-disturbances, we adopt a strategy that mainly removes existing edges to capture cases that establishing new links in real networks may be expensive [7].

[7]Other disturbance strategy exists such as insertion-only or random attacks; the best choice is application-specific.

TABLE II
STATISTICS OF DATASETS

Dataset	# nodes	# edges	# node features	# class labels
BAHouse	300	1500	-	4
PPI	2,245	61,318	50	121
CiteSeer	3,327	9,104	3,703	6
Reddit	232,965	114,615,892	602	41

GNN Classifiers. Following the evaluation of GNN explanation [16], [15], [18], [17], we adopt standard message-passing graph convolutional network (GCNs) configured with 3 convolution layers, each featuring an embedding dimension of 128. We remark that our solutions are model-agnostic and generalize to GNN specifications. We leave the details of training and inference settings in [43].

GNN Explainers. We compare RoboGExp against two recent GNN explainers. We used the original codes and authors' configurations for a fair comparison. (1) CF-GNNExp [34] generates counterfactual explanations for GNNs via minimal edge deletions, but overlooks factual explanations. (2) CF^2 [49] integrates factual and counterfactual reasoning for a specified test node into a single optimization problem, allowing the generation of GNN explanations that are both necessary and sufficient, but without robustness guarantees.

We are aware of two more recent methods [17] and [37]. Both are optimized for counterfactual explanations. The former has source codes relying on a pre-trained model; and the latter provides no available codes for node classification based explanation generation. Neither discusses efficiency and scalability of explanation generation. We thus cannot provide a fair experimental comparison with them.

Evaluation metrics. We evaluate RoboGExp and other explainers based on the following set of metrics.
(1) Normalized GED. Graph edit distance (GED) measures the number of edits required to transform a graph to another. We use a normalized GED (defined in Eq. 3; where the graph size refers to the total number of nodes and edges) to quantify the structural similarity of generated RCWs, over their counterparts from disturbed graphs. The GED quantifies the ability that a k-RCW G'_w from a disturbed graph by a k-disturbance remains to be an "invariant" compared to its original counterparts G_w before k-disturbances, which indicates their robustness: the smaller, the more "robust".

$$\text{normalized GED}(G_w, G'_w) = \frac{\text{GED}(G_w, G'_w)}{\max(|G_w|, |G'_w|)} \quad (3)$$

(2) Fidelity+ [35]. quantifies the deviations caused by removing the explanation subgraph from the input graph. Fidelity+ evaluates the counterfactual effectiveness. A higher Fidelity+ score indicates better distinction, hence the better. Here $\mathbb{1}$ is 1 if $M(v, G) = l$, and 0 otherwise.

$$Fidelity+ = \frac{1}{|V_T|} \sum_{v \in V_T} (\mathbb{1}(M(v, G) = l) - \mathbb{1}(M(v, G \setminus G_s) = l))$$

(3) Fidelity− [35]. It quantifies the difference between the results over G and the generated RCWs. indicating a factual accuracy. Ideal Fidelity- scores are low, even negative, indicating perfectly matched or even stronger predictions.

TABLE III
QUALITY OF EXPLANATIONS (CiteSeer)

	NormGED	Fidelity+	Fidelity-	Size
RoboGExp	0.32	0.79	0.05	66
CF^2	0.68	0.47	0.06	132
CF-GNNExp	0.72	0.65	0.13	78

$$Fidelity- = \frac{1}{|V_T|} \sum_{v \in V_T} (\mathbb{1}(M(v, G) = l) - \mathbb{1}(M(v, G_s) = l))$$

(4) Generation Time. The total response time on generating explanations for all the test nodes $|V_T|$. To evaluate the impact of the factor k on k-disturbances to generation time, we report the total response time it takes to "re-generate" the explanations for each method. The learning-based methods CF^2 and CF-GNNExp require retraining upon the change of graphs. We include the training and generation costs for both.

Environment. Our algorithms are implemented in Python 3.8.1 by PyTorch-Geometric framework [50]. All experiments are conducted on a Linux system equipped with 2 CPUs, each possessing 16 cores and 256 GB RAM.

B. Experiment results

Exp-1: Effectiveness: quality of explanations (RQ1(a)). Using CiteSeer, we first evaluate the effectiveness of RoboGExp, CF^2, and CF-GNNExp in terms of the quality of the explanations they generated. The results over other datasets are consistent, hence omitted.

Quality of Explanations. Setting k = 20, and $|V_T|$ = 20, we report the quality of the explaination structures of RoboGExp, CF^2, and CF-GNNExp in Table III. The table shows that RoboGExp outperforms both CF^2 and CF-GNNExp for all the three metrics. (1) RoboGExp outperforms CF^2 and CF-GNNExp by twice and 2.3 times in terms of GED. This demonstrates that it can generate explanations that is much more stable in structural similarity over the disturbance of underlying graphs. (2) RoboGExp achieves best scores in Fidelity+ and Fidelity−, outperforming CF^2 and CF-GNNExp. These indicate that RoboGExp can generate explanations as RCWs that stays consistent with the results of GNN classification simultaneously as factual, and counterfactual explanations.

We remark that the RCWs, by definition, should be theoretically achieving Fidelity+ as 1.0 and Fidelity− 0.0, as counterfactual and factual witnesses, respectively. The reason that it does not achieve the theoretical best scores is due to that not all the test nodes retain the same labels, leading to non-existence of non-trivial k-RCWs in some cases.

We also analyzed the size of the explanations. RoboGExp generated smallest size, which is on average half of their CF^2 counterparts. Due to space limit, we report it in [43].

Exp-2: Effectiveness: impact of factors (RQ1(b)). We next investigate the impact of the size of disturbance k and test size $|\mathcal{V}_T|$ on the effectiveness of RoboGExp, CF^2, and CF-GNNExp. We report their performances in Fig 3.

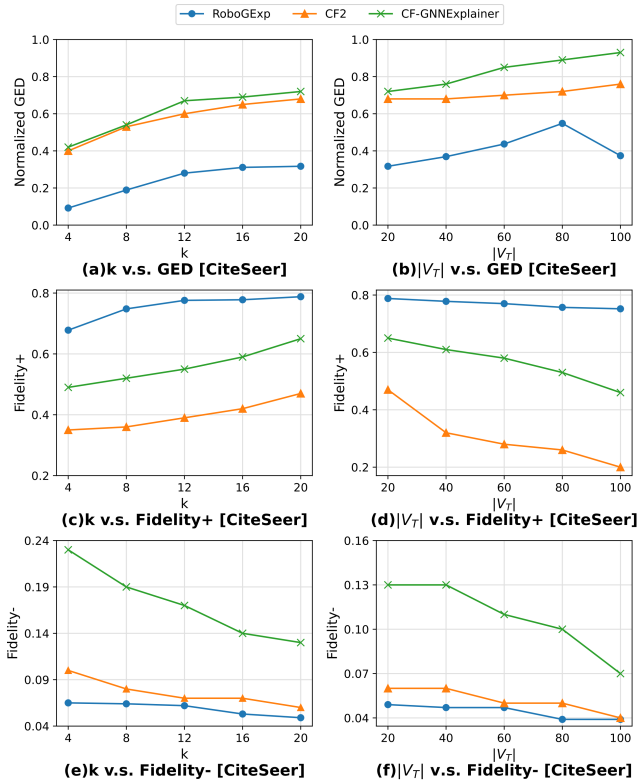

Fig. 3. Effectiveness with Impact Factors

Varying k. Fixing $|\mathcal{V}_T| = 20$, we vary k from 4 to 20, and report the results in Fig 3(a), (c), and (e).

(1) Fig 3(a) shows us that for all approaches, GED increases for larger k. Increasing k implies a more substantial disturbance, and accordingly larger variance to the underlying graph, hence larger differences in RCWs to retain the label of V_T. Due to the similar reason, CF^2 and CF-GNNExp exhibit larger GED. On the other hand, RoboGExp consistently outperforms CF^2 and CF-GNNExp. For example, it outperforms both when $k = 20$, with a GED that remains better than the best scores CF^2 and CF-GNNExp can reach.

(2) Fig 3(c) verifies that in general, all three methods achieve higher Fidelity+ as k becomes larger. Meanwhile, RoboGExp maintains higher and more stable Fidelity+ than CF^2 and CF-GNNExp, for all cases. We found that with $|V_T|$ unchanged, larger k allows RoboGExp to explore more k-disturbances "outside" of RCWs, and identify edges that are more likely to change the node labels. By incorporating such edges into RCWs, the generated explanations are also closer to the decision boundary. This verifies that RoboGExp is able to response better to larger structural variance of graphs and find counterfactual structures that better approximate the decision boundary of GNNs. RoboGExp remains least sensitive due to its ability to find robust explanations under disturbances. CF-GNNExp outperforms CF^2 due to its specialized learning goal towards counterfactual property.

(3) In Fig 3(e), all methods have improved results with smaller Fidelity− as k becomes larger. The erratic trend in CF^2's Fidelity− score is indicative of its less consistent factual explanation generation process, particularly when the graph is subjected to disturbances. Comparing CF^2 and CF-GNNExp, CF^2 is able to achieve Fidelity− scores that are closer to RoboGExp, due to that it considers both factual and counterfactual explanations, while CF-GNNExp is optimized for generating counterfactual explanations. Similar to Fidelity+, larger k allows RoboGExp to explore more disturbances and increases its chance to identify factual and counterfactual structures, resulting in a lower Fidelity−.

Varying $|\mathcal{V}_T|$. Fixing $k = 20$, we vary $|\mathcal{V}_T|$ from 20 to 100, and present the results in Fig 3(b), (d), and (f).

(1) Fig 3(b) depicts the following. RoboGExp remains to outperform CF^2 and CF-GNNExp constantly in all cases. Both RoboGExp and CF^2 exhibit a relatively stable performance in GED as $|\mathcal{V}_T|$ varies from 20 to 100. CF^2 is quite stable due to its goal of seeking factual and counterfactual explanations, which may have helped it identifying similar structures. CF-GNNExp is quite sensitive to larger $|V_T|$, due to larger test nodes introducing more structurally different explanations.

(2) In Fig 3(d), all methods have lower Fidelity+ scores as $|V_T|$ becomes larger. As more test nodes are introduced, it becomes difficult to maintain a counterfactual explanation due to more diverse structures. CF^2 optimizes on both factual and counterfactual explanations, and demonstrates more sensitive behavior. CF-GNNExp outperforms CF^2 in this case due to its optimization for only counterfactual explanations. On the other hand, RoboGExp remains to outperform CF^2 and CF-GNNExp, and is least sensitive to $|V_T|$. It is likely that the goal to find stable explanation for all test nodes greatly mitigates the impact of the additional diversity introduced by larger V_T.

(3) In Fig 3(f), all three methods find it difficult to maintain Fidelity− as $|V_T|$ is larger. RoboGExp continues to achieve the best Fidelity− scores. The difference is that CF^2 in turn outperforms CF-GNNExp and is able to achieve Fidelity− scores closer to RoboGExp, due to its optimization on both factual and counterfactual explanations. In general, RoboGExp remains the least sensitive to V_T, due to its prioritization strategy and enforcement of robustness verification.

Exp-3: Efficiency (RQ2). We report the efficiency of RoboGExp, CF^2, and CF-GNNExp in Fig 4. By default, we set $k = 20$, $|\mathcal{V}_T| = 20$, and test with CiteSeer.

Overall efficiency. Fig 4(a) shows the response time of RoboGExp, CF^2 and CF-GNNExp for three real-world datasets: BAHouse, CiteSeer, and PPI. RoboGExp constantly outperforms CF^2 and CF-GNNExp for all datasets. We found that the latter two incurs major overhead in the learning process to infer the explanations, which remains a main bottleneck and sensitive to large graphs with enriched features. Our methods explore bounded numbers of k-disturbances with efficient expansion and verification algorithms, and avoid unnecessary GNN inferences. For example, RoboGExp takes only 58.6% of the time of CF-GNNExp and 12.03% of the time of CF^2 to find explanations that are factual, counterfactual and robust. Moreoever, upon the disturbance of graph G, CF^2 and CF-GNNExp require retraining and "re-generate" explanations,

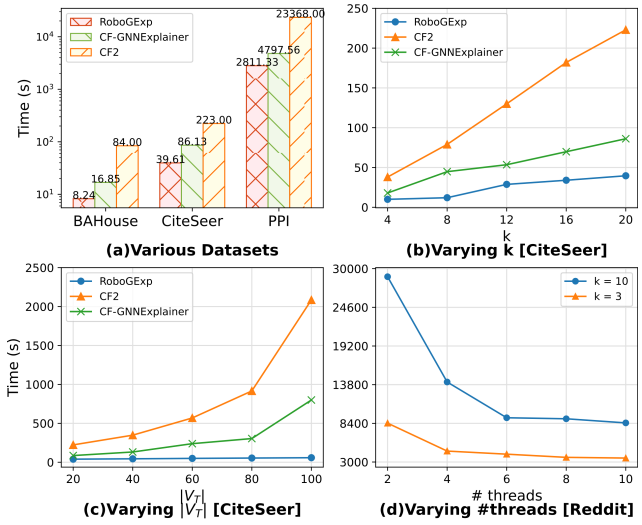

Fig. 4. Scalability and Efficiency

while RoboGExp can be applied to find robust explanations "once-for-all" for any variants of G under k-disturbance.

Impact of k. Using the same setting as in its effectiveness counterparts in Figs. 3 (a), (c), and (e), we report the response time of RoboGExp, CF^2, and CF-GNNExp as k varies from 4 to 20. All methods take more time as k becomes larger. As expected, RoboGExp takes more time to verify a larger number of k-disturbances, yet still benefits from its localizes search in "nearby" area of the explanations. Both CF^2 and CF-GNNExp incurs increased overhead in learning; the difference is that CF^2 directly re-learns the matrix representation, and CF-GNNExp re-learns to remove edges to sparsify the matrix, hence benefits better from removal-heavy disturbances.

Varying $|\mathcal{V}_T|$. Using the same setting as in its effectiveness counterparts in Figs. 3 (b),(d) and (f), we evaluate the impact of $|V_T|$ by varying it from 20 to 100. As shown in Figure 4(c), RoboGExp scales well and is insensitive in response time, due to its prioritization strategy that favors the processing of test nodes which are unlikely to have labels changed given current explanations. In contrast, both CF^2 and CF-GNNExp are sensitive, due to that both re-generate explanations each time V_T changes, and incur larger cost with more test nodes.

Exp-4: Scalability (RQ3). We next evaluate the scalability of paraRoboGExp, in terms of the number of threads and k. Fig 4(d) verifies the result over a large real-world dataset Reddit. paraRoboGExp scales well as the number of threads increases, with consistent impact due to larger k. The generation time is improved by 70.7% as the number of threads varies from 2 to 10 when k=10. The result verifies that it is practical to generate explanations over large graphs, and the response time can be effectively better improved for larger-scale disturbances (i.e., when k is larger).

Exp-5: Case Analysis. We next present two case studies, to showcase the application scenarios of RoboGExp.

Deciphering invariant in drug structures Fig. 5 depicts a chemical compound graph G_3 with a test node v_3 classified as

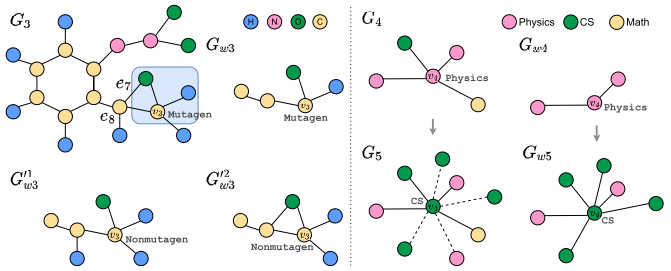

Fig. 5. Case Study: Left: a small RCW that capture an invariant structure for three variants of a drug structure; Right: an RCW that explains topic change with matching new citations.

"Mutagenic". On the top right is the RCW G_{w3} generated by RoboGExp, which preserves the aldehyde structure in the blue area – crucial for recognizing v as a component from a toxic fraction. CF^2 generated a larger explanation G'_{w3}. For two molecule variants G_3^1 and G_3^2 of G_3 obtained by removing the edge e_7 and edge e_8, respectively, RoboGExp verifies that G'_{w3} remains to be a 1-RCW for the family of all three compounds, while CF^2 generated two different explanations G'^1_{w3} and G'^2_{w3}. Observe that in both G'^1_{w3} and G'^2_{w3}, their aldehyde structure contains one redundant connection to a hydrogen atom, which will change the label of v_3 to "nonmutagen". Indeed, while the structure O-C-H is easy to be recognized as a toxic fragment responsible for mutagenic property, and is preserved in G_{w3}, additional "noisy" substructure that contains O, C, and two H is included to CF^2 generated structures, making it hard to distinguish the node to be in a mutagenic fraction or not.

Explaining topic change with new references. Our second case shows that RoboGExp is able to response quickly to the change of true labels with new explanation. Consider G_4 from CiteSeer with a node (a paper) about Quantum Computing, which has a label 'Physics', and has an explanation G_{w4} found by RoboGExp that identifies its relevant papers, all in physics area. Recent citations "disturbed" G_4 to G_5 with new neighbors that cited the paper from computer sciences, via which the GCN $\mathcal{M}$ updated its label to be "Computer Science". RoboGExp responded by discovering a new explanation G_{w5}, with small changes that include a majority of major computer sciences citations and less from physics. This verifies RoboGExp is able to strike a balance between factual and robustness as the true label changes.

VIII. Conclusion

We proposed k-robust counterfactual witnesses (k-RCW), an explanation structure for GNN-based classification, that are factual, counterfactual and robust to k-disturbances. We established the hardness and feasibility results, from tractable cases to co-NP-hardness, for both verification and generation problems. We introduced feasible algorithms to tackle the verification and generation problems, and parallel algorithms to generate RCWs for large graphs. Our experimental study verified the efficiency of explanation generation, and the quality of the explanations, and their applications. A future topic is to enhance our solution to generate minimum explanations, and evaluate their application for otherGNNs-based tasks.

REFERENCES

[1] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[2] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[3] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2011, pp. 1082–1090.

[4] L. Wei, H. Zhao, Z. He, and Q. Yao, "Neural architecture search for gnn-based graph classification," *ACM Transactions on Information Systems*, 2023.

[5] T. Chen, D. Qiu, Y. Wu, A. Khan, X. Ke, and Y. Gao, "View-based explanations for graph neural networks," in *ACM International Conference on Management of Data (SIGMOD)*, 2024.

[6] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, "Evaluating the quality of machine learning explanations: A survey on methods and metrics," *Electronics*, vol. 10, no. 5, p. 593, 2021.

[7] J. Xiong, Z. Xiong, K. Chen, H. Jiang, and M. Zheng, "Graph neural networks for automated de novo drug design," *Drug Discovery Today*, vol. 26, no. 6, pp. 1382–1393, 2021.

[8] M. Jiang, Z. Li, S. Zhang, S. Wang, X. Wang, Q. Yuan, and Z. Wei, "Drug–target affinity prediction using graph neural network and contact maps," *RSC advances*, vol. 10, no. 35, pp. 20 701–20 712, 2020.

[9] L. David, A. Thakkar, R. Mercado, and O. Engkvist, "Molecular representations in ai-driven drug discovery: a review and practical guide," *Journal of Cheminformatics*, vol. 12, no. 1, pp. 1–22, 2020.

[10] A. M. Albalahi, A. Ali, Z. Du, A. A. Bhatti, T. Alraqad, N. Iqbal, and A. E. Hamza, "On bond incident degree indices of chemical graphs," *Mathematics*, vol. 11, no. 1, p. 27, 2022.

[11] J. Kazius, R. McGuire, and R. Bursi, "Derivation and validation of toxicophores for mutagenicity prediction," *Journal of Medicinal Chemistry*, vol. 48, no. 1, pp. 312–320, 2005.

[12] H. Ding, J. Zhai, D. Deng, and S. Ma, "The case for learned provenance graph storage systems," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3277–3294.

[13] T. Bilot, N. El Madhoun, K. Al Agha, and A. Zouaoui, "Graph neural networks for intrusion detection: A survey," *IEEE Access*, 2023.

[14] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1172–1189.

[15] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[16] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *International Conference on Machine Learning (ICML)*, 2021, pp. 12 241–12 252.

[17] Z. Huang, M. Kosan, S. Medya, S. Ranu, and A. Singh, "Global counterfactual explainer for graph neural networks," in *WSDM*, 2023.

[18] S. Zhang, Y. Liu, N. Shah, and Y. Sun, "Gstarx: Explaining graph neural networks with structure-aware cooperative games," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[19] H. Yuan, J. Tang, X. Hu, and S. Ji, "Xgnn: Towards model-level explanations of graph neural networks," in *ACM international conference on knowledge discovery and data mining (KDD)*, 2020, pp. 430–438.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[23] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019.

[24] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *ICLR*, 2019.

[25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1024–1034.

[26] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[27] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[28] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[29] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision (ECCV)*, 2014, pp. 818–833.

[30] R. Schwarzenberg, M. Hübner, D. Harbecke, C. Alt, and L. Hennig, "Layerwise relevance visualization in convolutional text graph classifiers," in *Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs@EMNLP)*, 2019, pp. 58–62.

[31] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[32] M. S. Schlichtkrull, N. De Cao, and I. Titov, "Interpreting graph neural networks for nlp with differentiable edge masking," in *ICLR*, 2021.

[33] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 19 620–19 631, 2020.

[34] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, "Cf-gnnexplainer: Counterfactual explanations for graph neural networks," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, vol. 151, 2022, pp. 4499–4511.

[35] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5782–5799, 2023.

[36] J. Tan, S. Geng, Z. Fu, Y. Ge, S. Xu, Y. Li, and Y. Zhang, "Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1018–1027.

[37] M. Bajaj, L. Chu, Z. Y. Xue, J. Pei, L. Wang, P. C.-H. Lam, and Y. Zhang, "Robust counterfactual explanations on graph neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 5644–5655, 2021.

[38] A. Bojchevski and S. Günnemann, "Certifiable robustness to graph perturbations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[39] S. Guan, H. Ma, and Y. Wu, "Robognn: Robustifying node classification under link perturbation," in *IJCAI*, 2022.

[40] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform," *arXiv preprint arXiv:1803.07640*, 2018.

[41] M. Chen, Z. Wei, B. Ding, Y. Li, Y. Yuan, X. Du, and J.-R. Wen, "Scalable graph neural networks via bidirectional propagation," *Advances in neural information processing systems*, vol. 33, pp. 14 556–14 566, 2020.

[42] H. Zhou, A. Srivastava, H. Zeng, R. Kannan, and V. Prasanna, "Accelerating large scale real-time gnn inference using channel pruning," *Proc. VLDB Endow.*, vol. 14, no. 9, 2021.

[43] "Full version," 2023, *https://github.com/DazhuoQ/RoboGExp/blob/main/RoboGExp_ful*

[44] B. C. Csáji, R. M. Jungers, and V. D. Blondel, "Pagerank optimization by edge selection," *Discrete Applied Mathematics*, vol. 169, pp. 73–87, 2014.

[45] "Code and datasets," 2023, *https://github.com/DazhuoQ/RoboGExp*.

[46] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[47] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the Third ACM Conference on Digital Libraries*, 1998, p. 89–98.

[48] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, p. 1025–1035, 2017.

[49] J. Tan, S. Geng, Z. Fu, Y. Ge, S. Xu, Y. Li, and Y. Zhang, "Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning," in *Proceedings of the ACM Web Conference 2022*, 2022, p. 1018–1027.

[50] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.