Explaining Graph Neural Networks with Structure-Aware Cooperative Games

Anonymous Author(s)
Affiliation
Address
email

Abstract

Explaining predictions made by machine learning models is important and have attracted an increased interest. The Shapley value from cooperative game theory has been proposed as a prime approach to compute feature importance towards predictions, especially for images, text, tabular data, and recently graph neural networks (GNNs) on graphs. In this work, we revisit the appropriateness of the Shapley value for GNN explanation, where the task is to identify the most important subgraph and constituent nodes for GNN predictions. We purport that the Shapley value is a non-ideal choice for graph data because it is by definition not structure-aware. We propose a Graph Structure-aware eXplanation (GStarX) method to leverage the critical graph structure information to improve the explanation. Specifically, we define a scoring function based on a new structure-aware value from the cooperative game theory called the HN value. When used to score node importance, the HN value utilizes graph structures to attribute cooperation surplus between neighbor nodes, resembling message passing in GNNs, so that node importance scores reflect not only the node feature importance, but also the structural roles. We demonstrate that GStarX produces qualitatively more intuitive explanations, and quantitatively improves explanation fidelity over strong baselines on chemical graph property prediction and text graph sentiment classification.

1 Introduction

Explainability is crucial for complex machine learning (ML) models in sensitive applications, helping establish user trust and providing insights for potential improvements. Many efforts focus on explaining models on images, text, and tabular data. In contrast, the explainability of models on graph data is yet underexplored. Since explainability can be especially critical for many graph tasks like drug discovery, and interest in deep graph models is growing rapidly, further investigation of graph explainability is warranted. In this work, we study graph ML explanation with graph neural networks (GNNs) as the target models, given their popularity and widespread use.

In ML explainability, the Shapley value [1] has been deemed as a "fair" scoring function for computing feature importance. Originally from the cooperative game theory, many values, including the Shapley value, have been proposed for allocating a total payoff to players in a game. When used for scoring the feature importance of a data instance, the model prediction is treated as the total payoff and the features are considered as players. In particular, for an instance with n features $\{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\}$, the Shapley value of its ith feature $\boldsymbol{x}_i$ is computed via aggregating $m(i, S)$, which are the marginal contributions of $\boldsymbol{x}_i$ to sets of other features $\boldsymbol{x}_S \subseteq \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \setminus \{\boldsymbol{x}_i\}$. Each $\boldsymbol{x}_S$ is called a *coalition*. $m(i, S)$ is computed as the difference between model outputs for $\boldsymbol{x}_S \cup \{\boldsymbol{x}_i\}$ and $\boldsymbol{x}_S$, which is meant to capture the interaction between $\boldsymbol{x}_i$ and $\boldsymbol{x}_S$. The Shapley value is widely used for explaining ML models on images, text, and tabular data, when the features are pixels, words, and attributes [2, 3].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

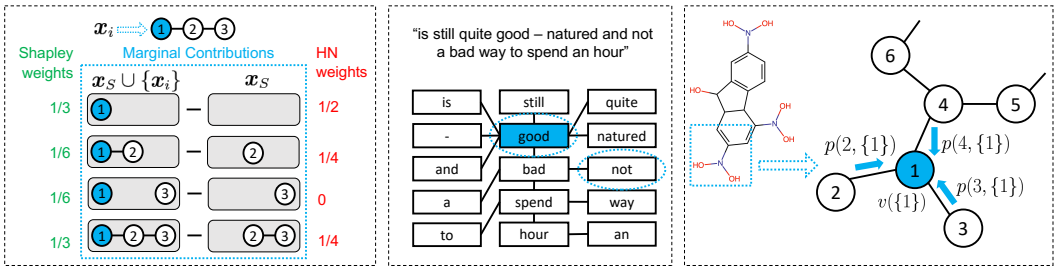

Figure 1: Explanations on graphs with structure-aware values (like HN) offers advantages over non-structure-aware values (like Shapley). **(a) Synthetic graph (left):** the Shapley value assigns weights to $m(i, S)$ only based on $|\boldsymbol{x}_S|$, while the HN value assigns weights considering structures and in particular gives 0 weight to the disconnected $\boldsymbol{x}_S$. **(b) Text graph (middle):** for a sentence classified as positive, the {"not", "good"} coalition shouldn't be considered when they are not connected by "bad". **(c) Chemical graph (right):** for a chemical graph with mutagenic functional group -NO2, the importance of the atom N (node 1) is better recognized if decided locally within the functional group.

The Shapley value has recently been extended to explain GNNs on graphs by feature importance scoring as above, where features are nodes [4] or supernodes [5]. We argue that the Shapley value is a non-ideal choice for (super)node importance scoring because its contribution aggregation is **non-structure-aware**. The Shapley value aggregation assumes no structural relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_S$ even though they are both parts of the input graph (a review of the Shapley value in Section 2.2). Since the graph structure generally contains critical information and is crucial to the success of GNNs, we consider properly leveraging the structure with a better **structure-aware** scoring function.

We propose Graph Structure-aware eXplanation (GStarX), where we construct a structure-aware node importance scoring function based on the HN value [6] from the cooperative game theory. Recall that GNNs make predictions via message passing (MP), during which node representations are updated by aggregating messages from their neighbors. MP aggregates both node feature information and structure information, resulting in powerful structure-aware models [7]. The HN value shares a similar idea to MP by allocating the payoff surplus generated from the cooperation between neighboring players (nodes). When used as a scoring function to explain node importance, the HN value captures both features and structural interactions between nodes (details in Section 4). An example comparing the Shapley value and the HN value is shown in Figure 1**(a)**. In this example, their difference boils down to different aggregation weights of marginal contributions, where the former is uniform and the latter is structure-aware (details in Section 3.2). In summary, we make the following contributions:

- Identify the non-structure-aware limitation of the Shapley value for GNN explanation.
- Introduce the structure-aware HN value from the cooperative game theory to the graph machine learning community and connect it to the GNN message passing and GNN Explanation.
- Propose a new HN-value-based GNN explanation method GStarX, and demonstrate the superiority of GStarX over strong baselines for explaining GNNs on chemical and text graphs.

2 Preliminaries

2.1 Graph neural networks

Consider a graph $\mathcal{G}$ with (feature-enriched) nodes $\mathcal{V}$ and edges $\mathcal{E}$. We denote $\mathcal{G}$ as $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$, where $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ denotes d-dimensional features of n nodes in $\mathcal{V}$, and $\boldsymbol{A} \in \{0,1\}^{n \times n}$ denotes the adjacency matrix specifying edges in $\mathcal{E}$. GNNs make predictions on $\mathcal{G}$ by learning representations via the *message-passing* mechanism. During message passing, the representation of each node $u \in \mathcal{V}$ is updated by aggregating its only representation and representations (messages) from its neighbors, denoted as $\mathcal{N}(u)$. This aggregation is recursively applied, so u can collect messages from its multi-hop neighbors and produce structure-aware representations [7]. With $\boldsymbol{h}_i^{(l)}$ denotes the representation of node i at iteration l, and $\text{AGGR}(\cdot,\cdot)$ denotes the aggregation operation, e.g. summation, the representation update is shown in Equation 1.

$$\boldsymbol{h}_u^{(l)} = \text{AGGR}(\boldsymbol{h}_u^{(l-1)}, \{\boldsymbol{h}_i^{(l-1)} | i \in \mathcal{N}(u)\}) \tag{1}$$

2.2 Cooperative games

A cooperative game denoted by (N, v), is defined by a set of players $N = \{1, \ldots, n\}$, and a *characteristic function* $v : 2^N \to \mathbb{R}$. v takes a subset of players $S \subseteq N$, called a *coalition*, and maps it to a payoff $v(S)$, where $v(\emptyset) := 0$. A *solution function* ϕ is a function maps each given game (N, v) to $\phi(N, v) \in \mathbb{R}^n$. The vector $\phi(N, v)$, called a *solution*, represents a certain allocation of the total payoff $v(N)$ generated by all players to each individual, with the ith coordinate $\phi_i(N, v)$ being the payoff attributed to player i. $\phi(N, v)$ is also called the "value" of the game when it satisfies certain properties, and different values were proposed to name solutions with different properties [1, 8].

The Shapley value is one popular solution of cooperative games. The main idea is to assign each player a "fair" share of the total payoff by considering all possible player interactions. For example, when player i cooperates with a coalition S, the total payoff $v(S \cup \{i\})$ may be very different from $v(S) + v(\{i\})$ because of i's interaction with S. Thus the marginal contribution of i to S is defined as by $m(i, S) = v(S \cup \{i\}) - v(S)$. Then the formula of the Shapley value for i is shown in Equation 2, where marginal contributions to all possible coalitions $S \subseteq N \backslash \{i\}$ are aggregated. The first identify in Equation 2 shows that the aggregation weights are first uniformly distributed among coalition sizes k (outer average), then uniformly distributed among all coalitions with the same size (inner average).

$$\phi_i(N, v) = \overbrace{\frac{1}{n} \sum_{k=0}^{n-1}}^{\text{Average over } k} \overbrace{\frac{1}{\binom{n-1}{k}} \sum_{\substack{S \subseteq N \backslash \{i\} \\ |S| = k}}}^{\text{Average over } S \text{ s.t. } |S| = k} m(i, S) = \sum_{S \subseteq N \backslash \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} m(i, S) \quad (2)$$

Games with communication structures. Although the Shapley value is widely used for cooperative games, its assumption of fully flexible cooperation among all players may not be achievable. Some coalitions may be preferred over others and some may even be impossible due to limited communication among players. Thus, [9] uses a graph $\mathcal{G}$ as the *communication structure* of players to represent cooperation preference. A game with a communication structure is defined by a triple $(N, v, \mathcal{G})$, with N being node set of $\mathcal{G}$. It is more practical when cooperation preference is available. Several such values with different properties have been proposed [9, 10, 11, 12] including the HN value [6].

3 GNN explanation via feature importance scoring

3.1 Problem formalization

A general approach to formalize a ML explanation problem is through feature importance scoring [3, 13], where features may refer to pixels of images, word of text, or nodes/edges/subgraphs of graphs. Let $f(\cdot)$ denote a to-be-explained GNN, $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$ denote an input graph, and $0 \leq \gamma \leq 1$ denote a sparsity constraint to enforce concise explanation. GNN explanation via subgraph scoring is aimed to find a subgraph g that maximizes a given evaluation metric $\textsc{Eval}(\cdot, \cdot, \cdot)$, which measures the faithfulness of g to $\mathcal{G}$ regarding making predictions with $f(\cdot)$, i.e.

$$g^* = \underset{g \subseteq \mathcal{G}, |g| \leq \gamma |\mathcal{G}|}{\arg\max} \ \textsc{Eval}(f(\cdot), \mathcal{G}, g) \quad (3)$$

When the task is graph classification and $f(\cdot)$ outputs a one-sum vector $f(\mathcal{G}) \in [0, 1]^C$ containing probabilities for $\mathcal{G}$ belongs to C classes, an example $\textsc{Eval}$ can be the prediction probability drop for removing g from $\mathcal{G}$, i.e. $\textsc{Eval}(f(\cdot), \mathcal{G}, g) = [f(\mathcal{G})]_{c^*} - [f(\mathcal{G} \backslash g)]_{c^*}$ with $c^* = \arg\max_c [f(\mathcal{G})]_c$.

In practice, since the number of subgraphs is combinatorial in the number of nodes, the objective is often relaxed to finding a set of important nodes or edges first, and then inducing the subgraph [14, 15, 4]. Then a more tractable objective of finding the optimal set of nodes $S^* \subseteq \mathcal{V}$ [1] is given by

$$S^* = \underset{S \subseteq \mathcal{V}, |S| \leq \gamma |\mathcal{V}|}{\arg\max} \sum_{i \in S} \textsc{Score}(f(\cdot), \mathcal{G}, i) \quad (4)$$

Existing methods often boil down to Equation 4 with different scoring functions ($\textsc{Score}$), and defining a proper $\textsc{Score}$ is non-trivial. For example, a simple example of $\textsc{Score}$ is to evaluate

[1] A similar objective can be defined as S over edges $\mathcal{E}$. We define it over nodes as nodes often contain richer features than edges and are more flexible. One advantage of this choice will be made clear in Section 5.3

each node i directly as $\text{SCORE}(f(\cdot), \mathcal{G}, i) = [f(\{i\})]_{c^*}$. However, this choice misses interactions between nodes and corresponds to a trivial case of no message-passing but only a feed-forward network applied on $\{i\}$. Another possible choice is to use EVAL as SCORE, like the example above: $\text{SCORE}(f(\cdot), \mathcal{G}, i) = [f(\mathcal{G})]_{c^*} - [f(\mathcal{G} \backslash \{i\})]_{c^*}$. However, this again fails to capture interactions between nodes; for example, two nodes i and j may be both important but also complimentary, so their contribution to $\mathcal{G}$ can only be observed when they are missing simultaneously.

3.2 Scoring functions from cooperative games

Given the challenges for defining a proper SCORE, solutions to cooperative games, like the Shapley value, have been proposed with $f(\cdot)$ as the characteristic function, i.e. $\text{SCORE}(f(\cdot), \mathcal{G}, i) = \phi_i(|\mathcal{G}|, f(\cdot))$ [5, 4]. However, existing works only use the non-structure-aware Shapley value. In contrast, values defined on games $(N, v, \mathcal{G})$ with communication structures $\mathcal{G}$ are naturally structure-aware but were never considered. Below we discuss the non-structure-aware limitation of the Shapley value in detail and motivating structure-aware values with practical examples in GNN explanation.

The Shapley value is defined on games (N, v), which by definition take no graph structures. It assumes flexible cooperation between players and uniform distribution of coalition importance that only depends on $|S|$ (see Equation 2). Even if a $\mathcal{G}$ is given and the game is defined as $(N, v, \mathcal{G})$, the Shapley value will overlook $\mathcal{G}$ when aggregating $m(i, S)$. In contrast, structure-aware values on $(N, v, \mathcal{G})$ can be interpreted as a weighted aggregation of coalitions with more reasonable weights. Although different solutions $\phi(N, v, \mathcal{G})$ have their nuances in weight adjustments [11, 6, 9, 12], they share two key properties: **(1)** the weight is 0 if i and S are disconnected because they are interpreted as players without communication channels [9], and **(2)** the weight is impacted by the nature of connections between i and S because it is easier for better-connected nodes to communicate.

A synthetic example. We take the HN value (definition in Section 4.1) as an example structure-aware value and compare it to the Shapley value in a simple graph in Figure 1**(a)**. To compute $\phi_1(N, v, \mathcal{G})$, both values aggregates $m(1, S)$ for $S \in \{\emptyset, \{2\}, \{3\}, \{2, 3\}\}$. The Shapley value first assigns a uniform weight $\frac{1}{3}$ to three different $|S|$, and then splits weights uniformly for the $|S| = 1$ case to be $\frac{1}{6}$. However, the HN value assigns weight 0 for $S = \{3\}$ because 1 and 3 are disconnected in coalition $\{1, 3\}$ and are assumed to be two independent graphs that shouldn't interact (property **(1)**). Their interaction is rather captured in the $S = \{2, 3\}$ case, when 1 and 3 are connected by the bridging node 2. This case is also downweighted from $\frac{1}{3}$ to $\frac{1}{4}$, as 3 is relatively far from 1 (property **(2)**).

A practical example. The good properties of structure-aware values can help explain graph tasks. The example in Figure 1**(b)** is from `GraphSST2` (dataset description in Section 5.1), where the graph for sentiment classification is constructed from the sentence *"is still quite good-natured and not a bad way to spend an hour"* with edges generated by the Biaffine parser [16]. Assuming a model can correctly classify it as positive. Intuitively, "good" and "not a bad" are central to the human explanation. To compute the Shapley value of the word "good", the coalition "not good" will contribute negatively and diminish the positive importance of "good", despite the two words lacking any direct connection. A structure-aware value can instead eliminate the {"not", "good"} coalition, and only consider interactions between "not" and "good" (in fact, "not" and any other word) when the bridging "bad" appears, hence better binding "not" with "bad" and improving the salience of "good".

4 GStarX: Graph Structure-aware eXplanation

We propose GStarX, which uses a structure-aware HN-value-based SCORE to explain GNNs. We first state the definition of the HN value in cooperative game theory (4.1), and then connect it to the GNN message passing (4.2), and finally give the GStarX algorithm for GNN explanation (4.3).

4.1 The HN value

For any game $(N, v, \mathcal{G})$ with a communication structure $\mathcal{G}$ and $S \subseteq N$ being a coalition, we use $S^* = \cup_{i \in S}\{\mathcal{N}(i)\} \cup S$ to denote the union of S and its neighbors in $\mathcal{G}$. We use $S/\mathcal{G}$ to denote the node partition of S, which is determined by the connected components of the induced subgraph of S in $\mathcal{G}$ (denoted as $\mathcal{G}[S]$). For example, in Figure 1**(b)**, when $S =$ {"is", "an", "hour"}, S^* will be {"is", "good", "an", "hour", "spend"}, and $S/\mathcal{G}$ will be {{"is"}, {"an", "hour"}}.

Definition 4.1 (Surplus). $p(j, S)$ representing the surplus generated by a coalition S cooperating with its neighbor j is defined as

$$p(j, S) = v(S \cup \{j\}) - v(S) - v(\{j\}) \tag{5}$$

Intuitively, $p(j, S)$ is generated because S is actively cooperating. Thus, when evaluating a fair payoff to S, a portion of $p(j, S)$ should be added to its own payoff $v(S)$. This idea leads to the definition of associated games regarding the original games, where surplus allocation is performed.

Definition 4.2 (HN Associated Game). Given $0 \leq \tau \leq 1$ representing the portion of surplus that will be allocated to a coalition S. The HN associated game $(N, v_\tau^*, \mathcal{G})$ of $(N, v, \mathcal{G})$ is defined as

$$v_\tau^*(S) = \begin{cases} v(S) + \tau \sum_{j \in S^* \setminus S} p(j, S) & \text{if } |S/\mathcal{G}| = 1 \quad (6) \\ \sum_{T \in S/\mathcal{G}} v_\tau^*(T) & \text{otherwise} \quad (7) \end{cases}$$

The HN value is a solution on $(N, v, \mathcal{G})$. It is computed by iteratively constructing a series of HN associated games until it converges to a *limit game* $(N, \tilde{v}, \mathcal{G})$. In other words, we first construct v_τ^* from v by surplus allocation. Then we construct v_τ^{**} from v_τ^* by allocating the surplus generated from the v_τ^* and so on. The convergence of the limit game is guaranteed and the result $\tilde{v}$ is independent of τ under mild conditions as shown in [6]. The HN value of each player is uniquely determined by applying $\tilde{v}$ to that player, i.e. $\phi_i(N, v, \mathcal{G}) = \tilde{v}(\{i\})$. We state the formal definitions of the limit game and the uniqueness theorem of the HN value in Appendix E.2.

4.2 Connecting GNNs and the HN surplus allocation through the message passing lens

Both the GNN message passing (Equation 1) and the associated game surplus allocation (Equation 6) are iterative aggregation algorithms, with considerable alignment. In fact, the surplus allocation on each singular node set $S = \{i\}$ is exactly a message passing operation: Equation 6 becomes an instantiation of Equation 1 with $\text{AGGR}(a, \boldsymbol{b}) = a + \tau \sum_j \boldsymbol{b}_j$ on a scalar node value a and a neighbor set $\boldsymbol{b}$. The algorithms differ in that HN surplus allocation applies more broadly to cases where $|S| \geq 1$; it treats S as a supernode when nodes in S form a connected component in $\mathcal{G}$, and handles disconnected S component-wise via Equation 7.

We illustrate the surplus allocation using a real chemical graph example. The molecule shown in Figure 1**(c)** is taken from `MUTAG` (dataset description in Section 5.1). It is known to be classified as *mutagenic* because of the -NO2 group (nodes 1, 2, and 3) [17]. When we compute $v_\tau^*(\{1\})$, the surplus $p(2, \{1\})$, $p(3, \{1\})$, and $p(4, \{1\})$ are allocated to node 1 (like messages passed to a central node in GNN). Then surplus are aggregated together with $v(\{1\})$ following Equation 6 to form $v_\tau^*(\{1\})$. For graphs, the surplus allocation approach has two advantages over the uniform aggregation used in the Shapley value: **(1)**: the aggregated payoff in each v_τ^* is structure-aware like representations learned by GNNs [7], and **(2)** the iterative computation preserves locality, which is what GNNs preserve [18]. In other words, close neighbors heavily influence each other due to cooperation in many iterations, while far away nodes have a lesser influence on each other due to little cooperation. In this example, since the local -NO2 piece generates a high payoff for the mutagenicity classification, locally allocating the payoff helps us better understand the importance of the nitrogen atom and the oxygen atoms. Whereas aggregating over many unnecessary coalitions with far-away carbon atoms can obscure the true contribution of -NO2. We will revisit this example in Section 5.3.

4.3 The GStarX algorithm

We now state our main algorithm for explaining GNNs with GStarX, which essentially implements and solves the objective in Equation 4. We first introduce our HN-value-based SCORE for the objective. To use the HN value as the SCORE, we need to define the players and the characteristic function of the game, and then apply the HN value formula in Equation 6 and 7. Suppose the inputs include a graph $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$ and label $y \in \{1, \ldots, C\}$, a GNN $f(\cdot)$ outputs a probability vector $f(\mathcal{G}) \in [0, 1]^C$, and the predicted class $c^* = \arg\max_c [f(\mathcal{G})]_c$. We let $\mathcal{V}$ be players, and let the normalized probability of the predicted class be the characteristic function v:

$$v(S) = [f(\mathcal{G}[S])]_{c^*} - f_{c^*}^0 \quad \forall S \subseteq \mathcal{V} \tag{8}$$

Algorithm 1 GStarX: Graph Structure-Aware Explanation

Input: Graph instance $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$, trained model $f(\cdot)$, empirical expectation f^0, hyperparameter τ, maximum sample size m, number of samples J, sparsity γ
Get the predicted class $c^* = \arg\max_c [f(\mathcal{G})]_c$
Define the characteristic function $v(S) = [f(g_S)]_{c^*} - f^0_{c^*}$
if $n \leq m$ **then**
  $\phi = \text{Compute-HN-Values}(\mathcal{G}, \mathcal{V}, v(\cdot), \tau)$
else
  $\phi = \text{Compute-HN-Values-MC}(\mathcal{G}, \mathcal{V}, v(\cdot), \tau, m, J)$
end if
Sort ϕ in descending order with indices $\{\pi_1, \ldots, \pi_n\}$
$k = \lfloor \gamma |\mathcal{V}| \rfloor$
Return: $S^* = \{u_{\pi_1}, \ldots, u_{\pi_k}\}$

Here $\mathcal{G}[S]$ denotes the subgraph induced by nodes in $S \subseteq \mathcal{V}$. The normalization term $f^0_{c^*} = \mathbb{E}\left[[f(G)]_{c^*}\right]$ is the expectation over a random variable G representing the general graph. In practice, we approximate it using the empirical expectation over all $\mathcal{G}$ in the dataset. The function SCORE will thus be the HN value of the game, i.e. $\text{SCORE}(f(\cdot), \mathcal{G}, i) = \phi_i(\mathcal{V}, v, \mathcal{G}) = \tilde{v}(\{i\})$.

Given the SCORE function, we solve the objective by first computing the scores $\phi \in \mathbb{R}^n$ then selecting the top $\lfloor \gamma |\mathcal{V}| \rfloor$ scores greedily as in Algorithm 1. Practically, like other game-theoretic methods, the exact computation of the HN value is infeasible when the number of players n is large (see Appendix E.3). We thus do an exact computation for small graphs (the if-branch) and Monte-Carlo sampling for large graphs (the else-branch). See Appendix E.3 for detailed algorithms for Compute-HN-Values and Compute-HN-Values-MC. Also, even though the algorithm is stated for graph classification, GStarX works for node classification as well. This can be easily seen since GNNs classify nodes u_i by processing an ego-graph centered at u_i, so the task can be converted to graph classification, where the label of u_i is used as the label of the ego-graph. We focus on graph classification in the main text for simpler illustration and discuss more about node classification in Appendix B.

5 Experiments

5.1 Experiment settings

Datasets. We conduct experiments on datasets including synthetic data, chemical data, and text data. A brief description of the datasets is shown below with more detailed statistics in Appendix A.1

- **Chemical graph property prediction.** `MUTAG` [17], `BACE` and `BBBP` [19] contain molecule graphs for graph classification, with atoms as nodes, bonds as edges, and chemical properties as labels.
- **Text graph sentiment classification.** `GraphSST2` and `Twitter` [20] contain graphs constructed from text. Nodes are words with pre-trained BERT embeddings as features. Edges are generated by the Biaffine parser [16]. Graphs are labeled as positive or negative sentiment.
- **Synthetic graph motif detection.** `BA2Motifs` [15] contains graphs with a Barabasi-Albert (BA) graph of size 20 and a 5-node motifs in each graph. Node features are 10-dimensional all-one vectors. The motif can be a house-like structure or a cycle, which separates graphs into two classes.

GNNs and explanation baselines. We evaluate GStarX by explaining GCNs [21] on all datasets in our major experiment. In later analysis, we also evaluate on GIN [22] and GAT [23] on certain datasets following [5]. All models are trained to convergence with hyperparameters and performance shown in Appendix A.2. We compare with 5 strong baselines representing the SOTA methods for GNN explanation: GNNExplainer [14], PGExplainer [15], SubgraphX [5], GraphSVX [4], and OrphicX [24]. In particular, SubgraphX and GraphSVX use Shapley-value-based scoring functions.

Evaluation metrics. Evaluating explanations is non-trivial due to the lack of ground truth. We follow [5, 20] to employ Fidelity, Inverse Fidelity (Inv-Fidelity), and Sparsity as our evaluation metrics. Fidelity and Inv-Fidelity measure whether the prediction is faithfully important to the model prediction by removing the selected nodes or only keeping the selected nodes respectively. Sparsity promotes fair comparison, since including more nodes generally improves Fidelity, and explanations with different sizes are not directly comparable. Ideal explanations are high Fidelity, low Inv-Fidelity, and high Sparsity, indicating relevance and conciseness. Equations 9-11 show their formulas.

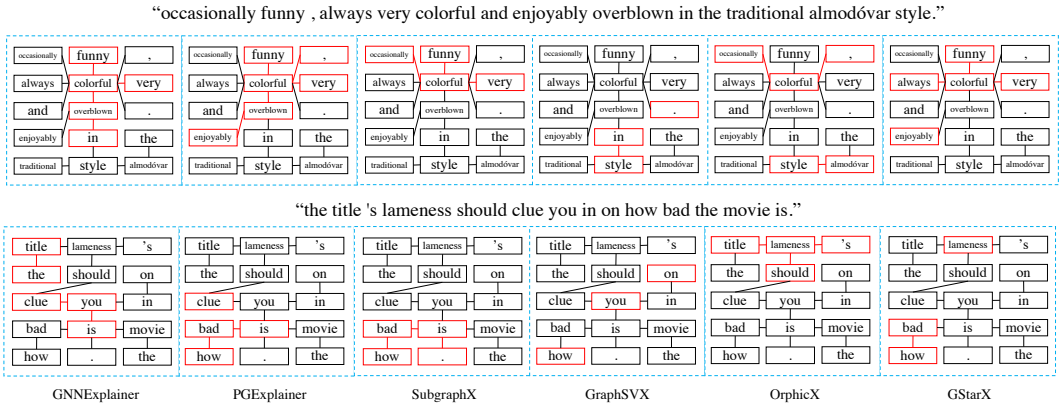

Figure 2: Explanations on sentences from `GraphSST2`. We show the explanation of one positive sentence (upper) and one negative sentence (lower). Red outlines indicate the selected nodes/edges as the explanation. GStarX identifies the sentiment words more accurately compared to baselines.

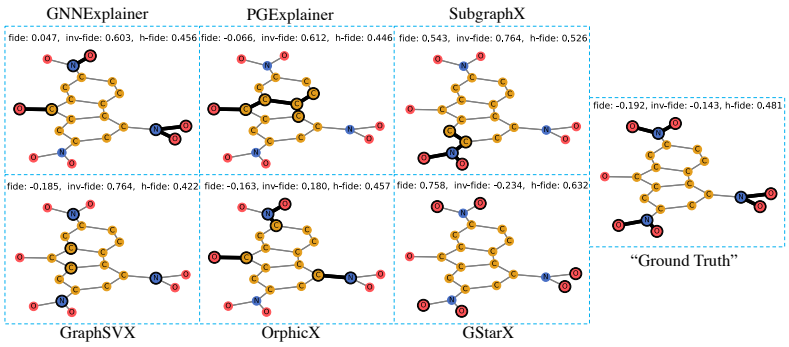

Figure 3: Explanations on a mutagenic molecule in `MUTAG`. Carbon atoms (C) are in yellow, nitrogen atoms (N) are in blue, and oxygen atoms are in red (O). Dark outlines indicate the selected nodes/edges as the explanation. We report the explanation Fidelity (fide), Inv-Fidelity (inv-fide), and H-Fidelity (h-fide). GStarX gives a significantly better explanation than other methods in terms of these metrics.

$$\text{Fidelity}(\mathcal{G}, g) = [f(\mathcal{G})]_{c^*} - [f(\mathcal{G} \backslash g)]_{c^*} \tag{9}$$

$$\text{Inv-Fidelity}(\mathcal{G}, g) = [f(\mathcal{G})]_{c^*} - [f(g)]_{c^*} \tag{10}$$

$$\text{Sparsity}(\mathcal{G}, g) = 1 - |g|/|\mathcal{G}| \tag{11}$$

Fidelity and Inv-Fidelity are complementary and are both important for a good explanation g. Fidelity justifies the necessity for g to be included to predict correctly. Inv-Fidelity justifies the sufficiency of a standalone g to predict correctly. As they are analogous to precision and recall, we draw an analogy to the F1 score to propose a single-scalar-metric "harmonic fidelity" (H-Fidelity), where we normalize them by Sparsity and take their harmonic mean; see Appendix A.3 for the formula.

Hyperparameters. GStarX includes three hyperparameters: τ for the associated game, maximum sample size m, and number of samples J. In our experiments we choose $\tau = 0.01$ since we need $\tau < \frac{2}{n}$ for convergence (Appendix E.2) and all graphs have less than 200 nodes. Bigger m and J should be better for the MC approximation, and we found $m = 10$ and $J = n$ work well empirically.

5.2 Quantitative studies

We report averaged test set H-Fidelity in Table 1. We conduct 8 different runs to get results with Sparsity ranging from 0.5-0.85 in 0.05 increments (Sparsity cannot be precisely guaranteed, hence it has minor variations across methods) and report the best H-Fidelity for each method. GStarX outperforms others on 4/6 datasets and has the highest average. We also follow [5] to show the Fidelity vs. Sparsity plots for all 8 sparsity in Appendix A.4.

Table 1: The best H-Fidelity (higher is better) of 8 different Sparsity for each dataset. GStarX shows higher H-Fidelity on average and on 4/6 datasets.

Dataset	GNNExplainer	PGExplainer	SubgraphX	GraphSVX	OrphicX	GStarX
BA2Motifs	0.4841	0.4879	**0.6050**	0.5017	0.5087	0.5824
BACE	0.5016	0.5127	0.5519	0.5067	0.4960	**0.5934**
BBBP	0.4735	0.4750	**0.5610**	0.5345	0.4893	0.5227
GraphSST2	0.4845	0.5196	0.5487	0.5053	0.4924	**0.5519**
MUTAG	0.4745	0.4714	0.5253	0.5211	0.4925	**0.6171**
Twitter	0.4838	0.4938	0.5494	0.4989	0.4944	**0.5716**
Average	0.4837	0.4934	0.5569	0.5114	0.4952	**0.5732**

5.3 Qualitative studies

We visualize the explanations of graphs in GraphSST2 in Figure 2 and compare them qualitatively. We show explanations selected with high and comparable Sparsity on a positive (upper) graph and a negative (lower) graph. GStarX concisely captures the important words for sentiment classification without including extraneous ones for both sentences. Baseline methods generally select some-but-not-all important sentiment words, with extra neutral words as well. Among baselines, SubgraphX gives more reasonable results. However, it cannot cover two groups of important nodes with a limited budget because it can only select a connected subgraph as the explanation; e.g. to cover the negative word "lameness" in the second sentence, SubgraphX needs at least 3 more nodes along the way, which will significantly decrease Sparsity while including undesirable, neutral words.

We visualize explanations selected with high and comparable Sparsity of a mutagenic molecule from MUTAG in Figure 3. Explanations on chemical graphs are harder to evaluate than text graphs as they require domain knowledge. MUTAG has been widely used as a benchmark for evaluating GNN explanations because human experts recognize -NO2 as mutagenic [17], which makes MUTAG a dataset with "ground truth"[2]. Surprisingly, we found that GStarX generates much better H-Fidelity/Fidelity/Inv-Fidelity than other methods and even the "ground truth" by only selecting the -O in -NO2 as explanations. In particular, the -0.234 Inv-Fidelity of GStarX means the selected subgraph has an even better prediction result than the original whole graph (0 Inv-Fidelity) and the ground truth (-0.143 Inv-Fidelity) because nodes not significant to the GNN prediction are removed. Fidelity metrics of baselines are inferior to GStarX because they include other non-discriminative carbon atoms despite they capture -NO2 to some extent. This suggests that even though **human experts identify -NO2** as the "ground truth" of mutagenicity, the **GNN only needs -O** to classify mutagenic molecules. With the goal being understand model behavior, GStarX explanation is better. Moreover, SubgraphX is the only baseline that has better H-Fidelity than the "ground truth", but it can only capture one -NO2 because its search algorithm requires the explanation to be connected, so its Inv-Fidelity is not optimal. In fact, GNNExplainer, PGExplainer, and SubgraphX can never generate explanations including only disconnected -O without -N like GStarX, because the former two solve the explanation problem by optimizing edges (as opposed to Equation 4), and the latter requires connectedness. More MUTAG explanation visualizations are in Appendix H.

5.4 Model-agnostic explanation

GStarX makes no assumptions about the model architecture and can be applied to various GNN backbones. We use GCN for all datasets in our major experiment for consistency, and now further investigate performance on two more popular GNNs: GIN and GAT. We follow [5] to train GIN on MUTAG and GAT on GraphSST2[3], and show results in Table 2. For both settings, GStarX outperforms the baselines, which is consistent with results on GCN.

5.5 Efficiency study

The GStarX algorithm scales in $O(J)$ with practical $J \propto |\mathcal{V}|$. Following [5], we study the empirical efficiency of GStarX by explaining 50 randomly selected graphs from BBBP. We report the average run

[2]Carbon rings were also claimed as mutagenic by human experts, but we found it is not discriminative as they exist in both mutagenic and non-mutagenic molecules in MUTAG.

[3]As some baselines take over 24 hours on full GraphSST2, we randomly select 30 graphs for this analysis.

Table 2: GStarX shows higher H-Fidelity for both GAT on GraphSST2 and GIN on MUTAG.

Dataset	GNNExplainer	PGExplainer	SubgraphX	GraphSVX	OrphicX	GStarX
GraphSST2	0.4951	0.4918	0.5484	0.5132	0.4997	**0.5542**
MUTAG	0.5042	0.4993	0.5264	0.5592	0.5152	**0.6064**

Table 3: Average running time on 50 graphs in BBBP

Method	GNNExplainer	PGExplainer	SubgraphX	GraphSVX	OrphicX	GStarX
Time(s)	11.92	0.03 (train 720)	75.96	3.06	0.15 (train 915)	31.24

time in Table 3. Our result on the baselines is similar to [5]. GStarX is not the fastest method, but it is more than two times faster than SubgraphX. Since explanation usually doesn't have strict efficiency requirements in real applications, considering GStarX generates higher-quality explanations than the baselines, we believe the time complexity of GStarX is acceptable.

6 Related work

GNN explanation aims to produce an explanation for GNN predictions on a given graph, usually as a subgraph induced by important nodes or edges. Many existing methods work by scoring nodes or edges and are thus similar to this work. For example, the scoring function of GNNExplainer [14] is the mutual information between a masked graph and the prediction on the original graph, where a soft mask on edges and node features is generated by direct parameter learning. PGExplainer [15] uses the same scoring function but generates a discrete mask on edges by training an edge mask predictor. SubgraphX [5] uses the Shapley value as its scoring function on subgraphs selected by Monte Carlo Tree Search (MCTS), and GraphSVX [4] uses a least-square approximation to the Shapley value to score nodes and their features. While SubgraphX and GraphSVX were shown to perform better than prior alternatives, as we show in Section 3, the Shapley value they try to approximate is non-ideal as it is non-structure-aware. Although SubgraphX and GraphSVX use L-hop subgraphs and thus technically they use the graph structure, we discuss why this approach has limitations in achieving structure-awareness in Appendix G. While there are many other GNN explanation methods from very different perspectives, i.e. gradient analysis [25], model decomposition [26], surrogate models [27], and causality [28, 24], we defer their details to Appendix C given their lesser relevance.

Cooperative game theory originally studies how to allocate payoffs among a set of players in a cooperative game. Recently, certain ideas from this domain have been successfully used in feature importance scoring for ML model explanation [2, 29, 3]. When used for explanation, the players of the game are the features, e.g. pixels for images. The value of the game is allocated to the features as importance scores in the model explanation. The vast majority of works in this line, like the ones cited above, deem the Shapley value [1] to be the only choice. In fact, there are many other values with different properties and used in different situations in cooperative game theory. However, to the best of our knowledge, only [30] mentions the Myerson value [9] in the context of proposing a connected Shapley (C-Shapley) value for explaining sequence data, so it is not directly comparable to ours. A detailed discussion of the Myerson value and the C-Shapley value can be found in Appendix F. In this work, we follow the cooperative game theory approach to explain models on graph data using a new HN value [6]. We show it is a better choice than the Shapley value as it is structure-aware.

7 Conclusion and future work

In summary, we study GNN explanation on graph data via feature importance scoring. In particular, we identify the non-structure-aware challenge of existing Shapley-value-based approaches and propose GStarX to assign importance scores to each node via a structure-aware HN value. We build connections between the surplus allocation in HN value and message passing in GNNs and demonstrate the superior performance of GStarX over strong baselines on chemical graph property prediction and text graph sentiment classification. As there is a rich literature of cooperative game theory beyond the Shapley value, more values are possible for explaining ML models. For graph data, edge based values can potentially be applied to an alternative edge-based objective like Equation 4. Other values may be appropriate to more data types beyond graphs. We leave these as future work.

References

[1] Lloyd Shapley. A value fo n-person games. Ann. Math. Study28, Contributions to the Theory of Games, ed. by HW Kuhn, and AW Tucker, pages 307–317, 1953.

[2] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. Applied Stochastic Models in Business and Industry, 17(4):319–330, 2001.

[3] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 4765–4774. Curran Associates, Inc., 2017.

[4] Alexandre Duval and Fragkiskos D Malliaros. Graphsvx: Shapley value explanations for graph neural networks. arXiv preprint arXiv:2104.10482, 2021.

[5] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 12241–12252. PMLR, 18–24 Jul 2021.

[6] Gérard Hamiache and Florian Navarro. Associated consistency, value and graphs. International Journal of Game Theory, 49(1):227–249, 2020.

[7] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? arXiv preprint arXiv:2002.04025, 2020.

[8] Lester G Telser. The usefulness of core theory in economics. Journal of Economic Perspectives, 8(2):151–164, 1994.

[9] Roger B Myerson. Graphs and cooperation in games. Mathematics of operations research, 2(3):225–229, 1977.

[10] Peter Borm, Guillerom Owen, and Stif Tijs. On the position value for communication situations. SIAM Journal on Discrete Mathematics, 5(3):305–320, 1992.

[11] Gérard Hamiache. A value with incomplete communication. Games and Economic Behavior, 26(1):59–78, 1999.

[12] Atsushi Kajii, Hiroyuki Kojima, and Takashi Ui. A refinement of the myerson value. IMS Preprint Series, 25, 2006.

[13] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. arXiv preprint arXiv:1705.07857, 2017.

[14] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. Advances in neural information processing systems, 32:9240, 2019.

[15] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 19620–19631. Curran Associates, Inc., 2020.

[16] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. arXiv preprint arXiv:1803.07640, 2018.

[17] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. Journal of medicinal chemistry, 34(2):786–797, 1991.

[18] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013.

[19] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. Chemical science, 9(2):513–530, 2018.

[20] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. arXiv preprint arXiv:2012.15445, 2020.

[21] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[22] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018.

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

[24] Wanyu Lin, Hao Lan, Hao Wang, and Baochun Li. Orphicx: A causality-inspired latent variable model for interpreting graph neural networks. arXiv preprint arXiv:2203.15209, 2022.

[25] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10772–10781, 2019.

[26] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks, 2019.

[27] Minh Vu and My T. Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 12225–12235. Curran Associates, Inc., 2020.

[28] Wanyu Lin, Hao Lan, and Baochun Li. Generative causal explanations for graph neural networks. In International Conference on Machine Learning, pages 6666–6679. PMLR, 2021.

[29] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. Knowledge and information systems, 41(3):647–665, 2014.

[30] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. In International Conference on Learning Representations, 2019.

[31] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[32] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, Keqiang Yan, Haoran Liu, Cong Fu, Bora M Oztekin, Xuan Zhang, and Shuiwang Ji. DIG: A turnkey library for diving into graph deep learning research. Journal of Machine Learning Research, 22(240):1–9, 2021.

[33] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks, 2020.

[34] Gérard Hamiache. Associated consistency and shapley value. International Journal of Game Theory, 30(2):279–289, 2001.

[35] S Hart and A Mas-Colell. Potential, value, and consistency. Econometrica, 57(3):589–614, 1989.

[36] Morton Davis and Michael Maschler. The kernel of a cooperative game. Naval Research Logistics Quarterly, 12(3):223–259, 1965.

[37] AI Sobolev. Characterization of the principle of optimality for cooperative games through functional equations. Mathematical Methods in the Social Sciences, Vipusk, 6:92–151, 1975.

[38] Bezalel Peleg. On the reduced game property and its converse. International Journal of Game Theory, 15(3):187–200, 1986.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experiment details

A.1 Datasets

In Table 4, we provided the basic statistics of datasets used in our experiments.

Table 4: Dataset Statistics.

Dataset	# Graphs	# Test Graphs	# Nodes (avg)	# Edges (avg)	# Features	# Classes
MUTAG	188	20	17.93	19.79	7	2
BACE	1,513	152	34.01	73.72	9	2
BBBP	2,039	200	24.06	25.95	9	2
GraphSST2	70,042	1821	9.20	10.19	768	2
Twitter	6,940	692	21.10	40.20	768	3
BA2Motifs	1,000	100	25	25.48	10	2

A.2 Model architectures and implementation

In Table 5, we provided the hyperparameters and test accuracy for the GCN model used in our major experiments. In Table 2, we provided the hyperparameters and test accuracy for the GIN and GAT model used in our analysis experiment. Most parameters are following [5], with small changes to further boost the test accuracy.

We run all experiments on a machine with 80 Intel(R) Xeon(R) E5-2698 v4 @ 2.20GHz CPUs, and a single NVIDIA V100 GPU with 16GB RAM. Our implementations are based on Python 3.8.10, PyTorch 1.10.0, PyTorch-Geometric 1.7.1 [31], and DIG [32]. We adapt the GNN implementation and most baseline explainer implementation from the DIG library, except GraphSVX where we use the official implementation. For the baseline hyperparameters, we closely follow the setting in [5] and [4] for a fair comparison. Please refer to [5] Section 4.1 and [4] Appendix E for details.

Table 5: GCN architecture hyperparameters according to results in Table 5

Dataset	#Layers	#Hidden	Pool	Test Acc
BA2Motifs	3	20	mean	0.9800
BACE	3	128	max	0.8026
BBBP	3	128	max	0.8634
MUTAG	3	128	mean	0.8500
GraphSST2	3	128	max	0.8808
Twitter	3	128	max	0.6908

Table 6: GIN and GAT architecture hyperparameters according to results in Table 2. For GAT, we use 10 attention heads with 10 dimension each, and thus 100 hidden dimensions.

Dataset	#Layers	#Hidden	Pool	Test Acc
GraphSST2(GAT)	3	10 ×10	max	0.8814
MUTAG(GIN)	3	128	max	1.0

A.3 Exact formula for evaluation metrics

We showed the formula of Fidelity, Inv-Fidelity, and Sparsity in Equation 9, 10, and 11. In Equation 12, 13, and 14, we show the formula for normalized fidelity (N-Fidelity), normalized inverse fidelity (N-Inv-Fidelity), and harmonic fidelity (H-Fidelity). Both the N-Fidelity and N-Inv-Fidelity are in $[-1, 1]$. The H-Fidelity flips N-Inv-Fidelity, rescales both values to be in $[0, 1]$, and takes their harmonic mean.

$$\text{N-Fidelity}(\mathcal{G}, g) = \text{Fidelity}(\mathcal{G}, g) \cdot (1 - \frac{|g|}{|\mathcal{G}|}) \tag{12}$$

$$\text{N-Inv-Fidelity}(\mathcal{G}, g) = \text{Inv-Fidelity}(\mathcal{G}, g) \cdot (\frac{|g|}{|\mathcal{G}|}) \tag{13}$$

Let $m1 = \text{N-Fidelity}(\mathcal{G}, g)$, $m2 = \text{N-Inv-Fidelity}(\mathcal{G}, g)$

$$\begin{aligned}\text{H-Fidelity}(\mathcal{G}, g) &= \frac{2}{(\frac{1+m1}{2})^{-1} + (\frac{1-m2}{2})^{-1}} \\ &= \frac{(1+m1)\cdot(1-m2)}{(2+m1-m2)}\end{aligned} \tag{14}$$

A.4 Fidelity vs. sparsity plots

In Table 1, we reported the best H-Fidelity among 8 different sparsities for each method. We also follow [5] to show the Fidelity vs. Sparsity plots in Figure 4. Note that GraphSVX tends to give sparse explanations on some datasets, we still pick 8 different sparsities for it but mostly on the higher end.

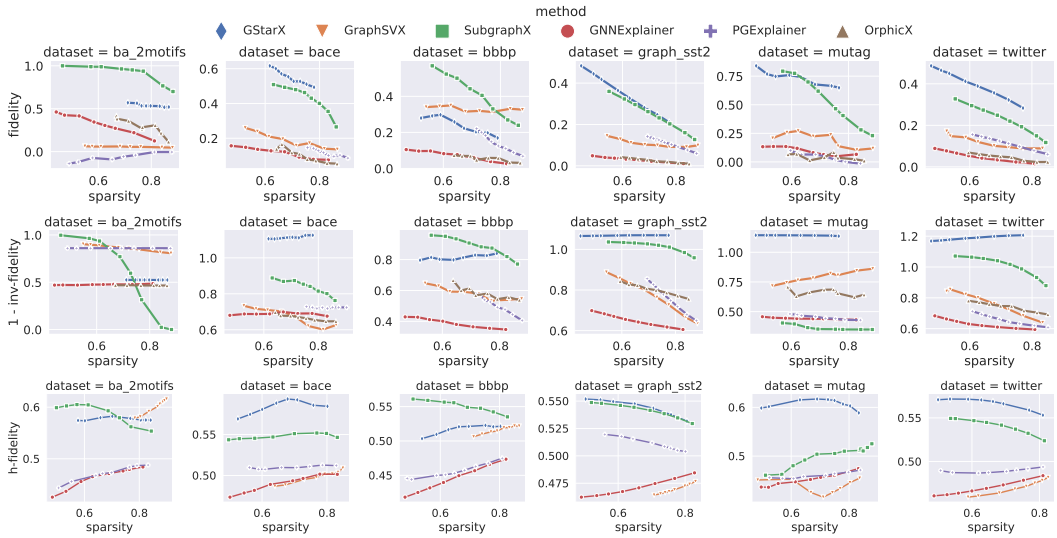

Figure 4: Fidelity (row1), 1 - Inv-Fidelity (row2), and H-Fidelity (row3) vs. Sparsity on all datasets corresponding to the results shown in Table 1. All three metrics are the higher the better. We see that GStarX outperforms the other methods

B GStarX for node classification

Even the GStarX algorithm is stated for graph classification, it works for node classification as well. This can be easily seen as the GNN node classification can be covert to classify an ego-graph. Given a graph $\mathcal{G}$ with $\mathcal{V} = \{u_1, \ldots, u_n\}$. Node classification on u_i with an L-layer GNN can be converted to a graph classification. The target graph to classify will be the L-hop ego-graph centered at u_i, because this is the receptive field of the GNN for classifying u_i and nodes further away won't have influence on the result. The label of the graph will be the label of u_i. In this case, the final readout layer of the GNN will be indexing u_i instead of pooling. Given this kind of conversion, everything we showed in Section 4 follows.

We conduct experiments on a standard node classification classification benchmark. With dataset description shown below and results shown in Table 7.

`BAShape` [14] is a synthetic dataset similar to `BA2Motifs`but for node classification. It has one graph containing a 300-node BA graph and 80 5-node motifs randomly attached to the BA graph. Node features are 10-dimensional all-1 vectors. Each node in `BAShape` is labeled with its the membership of a motif and the location within the motif.

Table 7: The best H-Fidelity (higher is better) of 8 different Sparsity. GStarX shows higher H-Fidelity over baselines.

Dataset	GNNExplainer	PGExplainer	SubgraphX	GraphSVX	OrphicX	GStarX
BAShape	0.4772	0.5042	0.6050	0.4916	0.5081	**0.5321**

C More related work

GNN explanation continued Besides the perturbation-based method we mentioned in Section 6, there are several other type of approaches for GNN explanation. Gradient-based methods are widely used for explaining ML models on images and text. The key idea is to use the gradients as the approximations of input importance. Such methods like contrastive gradient-based (CG) saliency maps, Class Activation Mapping (CAM), and gradient-weighted CAM (Grad-CAM) have been generalized to graph data in [25]. Decomposition based methods are a popular way to explain deep NNs for images. They measure the importance of input features by decomposing the model predictions and regard the decomposed terms as importance scores. Decomposition methods including Layer-wise Relevance Propagation (LRP) and Excitation Backpropagation (EB) have also been extended to graphs [25, 26]. Surrogate-based methods work by approximating a complex model using an explainable model locally. Possible options to approximate GNNs include linear model as in GraphLIME [33], additive feature attribution model with the Shapley value as in GraphSVX [4], and Bayesian networks as in [27]. GNN explainability has also been studied from the causal perspective. In [28, 24], generative models were constructed to learn causal factors, and explanations were produced by analyzing the cause-effect relationship in the causal graph.

D Properties of the Shapley value

The Shapley value was proposed as the unique solution of a game (N, v) that satisfies three properties shown below, i.e. *efficiency*, *symmetry*, and *additivity* [1]. These three properties together are referred as an axiomatic characterization of the Shapley value. The *associated consistency* properties introduced in Section 4.1 provides a different axiomatic characterization.

Property D.1 (Efficiency).

$$\sum_{i \in N} \phi_i(N, v) = v(N)$$

Property D.2 (Symmetry). If $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \in N \backslash \{i, j\}$, then

$$\phi_i(N, v) = \phi_j(N, v)$$

Property D.3 (Additivity). Given two games (N, v) and (N, w),

$$\phi(N, v + w) = \phi(N, v) + \phi(N, w)$$

The efficiency property states that the value should fully distribute the payoff the game. The symmetry property states that if two players make equal contributions to all possible coalitions formed by other players (including the empty coalition), then they should have the same value. The additivity property states that the value of two independent games should be added player by player. It is the most useful for a system of independent games.

E More about the HN value

E.1 Consistency and associated games

One reason for the Shapley value's popularity is its *axiomatic characterization*, indicating that it is the unique solution which satisfies a set of desirable properties (see Appendix D). Then [34] proposed a new axiomatic characterization of the Shapley value based on a different *associated consistency* property. The *consistency* property is a common analysis tool used in game theory [35, 36, 37, 38]. The idea is to analyze a game (N, v) by defining other reduced games (S, v_S) for $S \subseteq N$, and a solution function ϕ is called *consistent* when $\phi(N, v)$ yields the same payoff as $\phi(S, v_S)$ on each S. When (S, v_S) is defined with desired properties, these good properties can be enforced for a solution

by requiring consistency. The associated consistency in [34] is a special case of consistency between (N, v) and only one other game (N, v^*), which is called the *associated game*. [34] shows that a carefully designed associated game uniquely characterizes the Shapley value. Associated consistency is also the key idea of the HN value.

E.2 Limit game and the axiomatic characterization

The HN value is established on an special associated game as we discussed in Section 4.1. We can actually write this associated game in a more compact matrix form, where we slightly abuse notation and use v and v^*_τ to represent vectors of payoffs for all $S \subseteq N$ under the original and associated game respectively. In other words, $v(S)$, which used to represent evaluating the coalition S using the characteristic function v, now can also be interpreted as indexing the vector v with index S.

Lemma E.1. *A matrix form of the associated game $(N, v^*_\tau, \mathcal{G})$ is given by*

$$v^*_\tau = \boldsymbol{H}_{\{\tau,n,\mathcal{G}\}} v \tag{15}$$

The matrix $\boldsymbol{H}_{\{\tau,n,\mathcal{G}\}}$ depends on the hyperparameter τ, number of players n, and the graph $\mathcal{G}$. When these variables are clear from the context, we drop them and write $v^*_\tau = \boldsymbol{H} v$. Please refer to [6] for the proof of Lemma E.1.

With the matrix form, we can define the limit game.

Definition E.2. Given a game $(N, v, \mathcal{G})$, its limit game $(N, \tilde{v}, \mathcal{G})$ is defined by

$$\tilde{v} = \lim_{p \to \infty} \boldsymbol{H}^p v \tag{16}$$

Notice that although the matrix $\boldsymbol{H}$ is constructed from the associated game and depends on τ, the powers of $\boldsymbol{H}$ actually converges to a limit independent from τ, when τ is sufficiently small. The general condition depends on the actual graph, but $0 < \tau < \frac{2}{n}$ is proven to be sufficient for the complete graph case [34]. As we discussed in Section 4.1, the limit game can be seen as constructing associated games repeatedly until the characteristic function converges.

An axiomatic characterization of the HN value regarding its uniqueness is given by the following theorem based on the limit game. The associated consistency is the core property related to this work. We encourage the readers to check [6] for the other two properties.

Theorem E.3. *There exists a unique solution ϕ that verifies the associated consistency, i.e. $\phi_i(N, v, \mathcal{G}) = \phi_i(N, v^*_\tau, \mathcal{G})$, inessential game, and continuity. ϕ is given by*

$$\phi_i(N, v, \mathcal{G}) = \tilde{v}(\{i\}) \tag{17}$$

E.3 The algorithm for computing the HN value

We show the algorithm for Compute-HN-Values (Algorithm 2) and Compute-HN-Values-MC (Algorithm 3) mentioned in Section 4.3. The algorithm is a combination of Equation 15, 16, and 8.

Algorithm 2 The Compute-HN-Values-MC Function

Input: Graph instance $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$, characteristic function v, hyperparameter τ, maximum sample size m, number of samples J
Let $\psi_1, \ldots, \psi_n$ be n empty lists
for $j = 1$ **to** J **do**
  Sample g_{S^j} from $\mathcal{G}$ s.t. $S^j = \{u_{j_1}, \ldots, u_{j_l}\}$ and $l < m$
  $\phi^j =$ Compute-HN-Values$(g_{S^j}, S^j, v(\cdot), \tau)$
  for $k = 1$ **to** l **do**
    Append ϕ^j_k to ψ_{j_k}
  end for
end for
Set ϕ_i to be the mean of ψ_i
Return: ϕ

Algorithm 3 The Compute-HN-Values Function

Input: Graph instance $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$, characteristic function v, hyperparameter τ
for S in 2^N **do**
  Compute payoff $v(S)$ {Equation 8}
end for
Construct matrix $\boldsymbol{H}$ from τ, n, and $\mathcal{G}$ {Equation 15}
repeat
  $\boldsymbol{H} = \boldsymbol{H}\boldsymbol{H}$
until $\boldsymbol{H}$ converges
Get the limit game $\tilde{v} = \boldsymbol{H}v^e$ {Equation 16}
Assign the first n entries of $\tilde{v}$ to ϕ
Return: ϕ

F The myerson value and the C-Shapley value

F.1 The Myerson value

In the study of cooperative games, [9] proposed to characterize the cooperation possibilities between players using a graph structure $\mathcal{G}$, which leads to the communication structure introduced in Section 2.2 and the Myerson value as a solution for this special type of games $(N, v, \mathcal{G})$. The Myerson value is closely related to the Shapley value. In fact, it is the Shapley value on a transformed game where players are partitioned by the graph. We now formally introduce the partition and the transformed game.

Definition F.1 (Partition). Given a set of players N and a graph $\mathcal{G}$. For any coalition $S \subseteq N$, the partition of S is denoted by $S/\mathcal{G}$ and defined by

$$S/\mathcal{G} = \{\{i | i \text{ and } j \text{ are connected in S by } \mathcal{G}\} | j \in S\}$$

and a member of the set $S/\mathcal{G}$ is called a component of S.

Definition F.2 (Transformed Game). Given a game $(N, v, \mathcal{G})$, we can transform it to a new game $v/\mathcal{G}$ such that for all $S \subseteq N$

$$(v/\mathcal{G})(S) = \sum_{T \in S/\mathcal{G}} v(T)$$

Intuitively, given a coalition S, the transformed game treats each connected component of S as independent, evaluates them separately, and sums their payoff as the payoff of S.

The Shapley value has a axiomatic characterization that uniquely determines it as we introduced in Appendix D. Likewise, the Myerson value was proposed to be a unique solution that satisfies the *component efficiency* and the *fairness* property defined below.

Property F.3 (Component Efficiency). For a game $(N, v, \mathcal{G})$ and any connected component $S \in N/\mathcal{G}$, a solution is component efficient if

$$\sum_{i \in S} \phi_i(N, v, \mathcal{G}) = v(S)$$

Property F.4 (Fairness). For a game $(N, v, \mathcal{G})$ and any edge (i, j) in $\mathcal{G}$, let $\tilde{\mathcal{G}}$ be $\mathcal{G}$ with the edge (i, j) removed, a solution is fair if

$$\phi_i(N, v, \mathcal{G}) - \phi_i(N, v, \tilde{\mathcal{G}}) = \phi_j(N, v, \mathcal{G}) - \phi_j(N, v, \tilde{\mathcal{G}})$$

The component efficiency property is an extension of the regular efficiency property to games with a communication structure. It requires efficiency to hold for each disconnected piece because these pieces are assumed as independent from each other. The fairness property states that if breaking an edge (i, j) changes the value of player i, then the value of player j should be changed by the same amount.

Theorem F.5 (Myerson Value). *There exists a unique solution ϕ of game $(N, v, \mathcal{G})$ satisfying component efficiency and fairness. With $\tilde{\phi}$ represents the Shapley value, the solution is given by the formula*

$$\phi(N, v, \mathcal{G}) = \tilde{\phi}(N, (v/\mathcal{G}))$$

For games with a communication structure, the Myerson value is a better choice than the Shapley value as it uses the graph structure. However, it also suffer from some criticisms. For example, the fairness assumption may not be realistic. When an existing edge is broken, the value changes for players on the two edge ends can be asymmetric. Intuitively, if the edge connects a popular hub player i to a leaf player j, then the change of i can be less significant than j since j becomes isolated when (i, j) is removed. This is also the case when the game value is used for model explanation. For example in Figure 1 (b), when the edge ("good", "quite") is broken, the value of "quite" should change a lot. It used to contribute positively together with "good", and thus gets some payoff allocation, but it now becomes an isolated node, which is neutral by itself. On the other hand, the word "good" can still contribute positively by itself and interact with other nodes through its other edges, and thus its value shouldn't change too much. Because of such criticisms, we choose to use the HN value as our scoring function, which characterize the value by associated consistency rather than fairness.

F.2 The C_Shapley value

The Myerson value was also mentioned in [30] for model explanation on text, where the C_Shapley value was proposed an approximation of the Shapley value, and it was claimed to be equal to the Myerson value. We have discussed why Shapley value and Myerson are not ideal choice for explaining graph data in Section 3 and Appendix F.1. These are partially the reason why our HN value based method is better than the C_Shapley value. However, the major reason why we don't do direct comparison to the C_Shapley value as a baseline is because its formula only works for line graphs like sequence data, and not even all nodes in line graphs. In contrast, our target task is general graph prediction for graph with possibly complicated topological structures.

We now clarify a mistake of the C_Shapley value formula and explain why it won't work for general graphs. The notations are following the [30], where d is the number of players corresponding to n in our notation, and $[d]$ corresponding to N.

The formula for the C_Shapley value is given in Equation 6 in Definition 2 in the paper, and it is stated for "a graph G" without mentioning any assumptions of the graph. However, from the proof of this formula in Appendix B.2 in the paper, the line graph assumption can be seen in two places. The first place is Equation 20, where the set $\mathcal{C}$ is explicitly defined only for subsequences. The second place is Equation 22, the first line converts $\sum_{A:U_S(A)=U}$ to $\sum_{i=0}^{d-|U|-2}$, which is implicitly saying $V_S(A)$ can be picked from all d but $|U|-2$ nodes. However, this conversion is only possible when there are exactly 2 edges between U and $[d]\backslash U$, i.e. the middle part of a line graph. If there are l edges between U and $[d]\backslash U$, then the summation should go up to $d-|U|-l$. When $l=0$, i.e. U equals $[d]$ or a connected component of $[d]$, no partition is needed and the coefficient simply evaluates to 1. By correcting all these cases, the final formula for the C_Shapley value coefficients of marginal contributions thus becomes

$$\sum_{i=0}^{d-|U|-l} \frac{1}{\binom{d-1}{i+|U|-1}} \binom{d-|U|-l}{i} \tag{18}$$

$$= \frac{d}{(|U|+l)\binom{|U|+l-1}{|U|-1}} \tag{19}$$

$$= \frac{dl}{(|U|+l)(|U|+l-1)\cdots|U|} \tag{20}$$

for $l>0$, and 1 for $l=0$.

The correct formula for the C_Shapley value of general graphs will be

$$\phi_X(i) = \begin{cases} \sum_{U\in\mathcal{C}} \frac{l}{(|U|+l)\cdots|U|} m_X(U,i) & \text{if } l>0 \\ \frac{1}{d} & \text{if } l=0 \end{cases} \tag{21}$$

with l represents the edges between U and $[d]\backslash U$ and $\mathcal{C}$ represents all connected subgraphs in $[d]$ containing i.

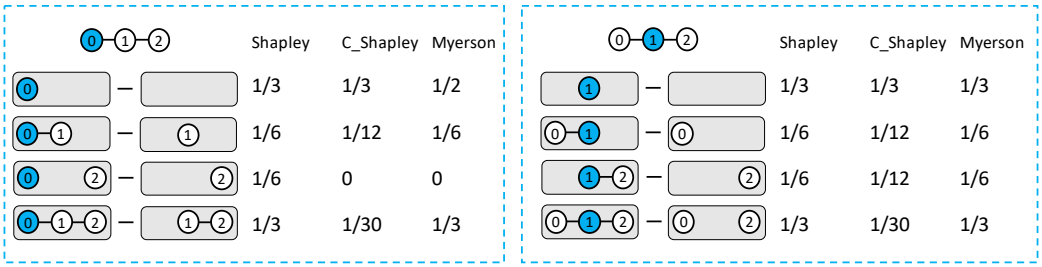

Figure 5: A toy 3-node graph example for comparing the mariginal contribution coefficients between the Shapley, the C_Shapley, and the Myerson value. **(a)** Value computation for node 0 (left). **(b)** Value computation for node 1 (right).

To verify this formula with the 3-node toy graph in Figure 5. When computing the value of node 0 (left), the three connected components containing 0 are $\mathcal{C} = \{\{0\}, \{0, 1\}, \{0, 1, 2\}\}$. Since 0 is an end node and has no leaf nodes to its left, l for these three components will be 1, 1, and 0 respectively. According to our new formula in Equation 21, the coefficients will be $\frac{1}{2}$, $\frac{1}{6}$, and $\frac{1}{3}$ respectively, with the disconnected $\{0, 2\}$ case removed. This matches the original idea of Myerson value, where the $\{0, 2\} - \{2\}$ case is reduced to the $\{0\} - \emptyset$ case, which turns the Shapley coefficients from $[\frac{1}{3}, \frac{1}{6}, \frac{1}{6}, \frac{1}{3}]$ to $[\frac{1}{3} + \frac{1}{6}, \frac{1}{6}, \frac{1}{6} - \frac{1}{6}, \frac{1}{3}]$, which is $[\frac{1}{2}, \frac{1}{6}, 0, \frac{1}{3}]$. However, the original CShapley formula from Equation 6 in the [30] evaluates to $[\frac{1}{3}, \frac{1}{12}, 0, \frac{1}{30}]$, which doesn't match the Myerson value and not even sum up to 1. Another example of computing the value of node 1 is shown in Figure 5 right.

The C_Shapley, even with the correct formula, eventually boils down to an approximation of the Shapley value or the Myerson value, which as we discussed are less idea than the HN value. Also, the correct formula in Equation 21 requires generating all possible subgraphs U containing the node i and specify the edges between U and $[d] \backslash U$. This makes the computation very complicated, we thus skip the comparison to the C_Shapley value.

G Use the graph structure via an L-hop cutoff

Although the Shapley value itself is not structure-aware, we do note the existing Shapley-value-based GNN explanation methods uses a L-hop cutoff to help approximate the Shapley value [5, 4]. Technically, this operation uses the graph structure, so we can't strictly refer these explanation methods as not structure aware. However, we argue that the L-hop cutoff is a naive way of utilizing the graph structure. It has several concerns, and it is not the same structure-aware as the HN value.

The L-hop cutoff approximates the Shapley value of node i by considering only the L-hop neighbors of i when explaining a L-layer GNN. The rationale of this operation is that an L-layer GNNs only propagate messages within L-hops so a node more than L-hop away from i has never passed any messages to i which means no interactions are possible. In existing Shapley-value-based GNN explanation methods, this L-hop cutoff operation was meant for reducing the exponentially growing computations of the Shapley value, and the ultimate goal is still compute the Shapley value. The L-hop cutoff operation has several issues making it a less desirable choice. **1)** Even meant to save computation, there are still many nodes involved in the computation after applying the L-hop cutoff, since the number of nodes grow exponentially as L grow. For advanced GNNs, the L can be large. When L is larger than the diameter of the graph, which is actually the case for many recent deep GNNs, the L-hop cutoff is not effective anymore. **2)** When constructing coalitions of nodes within the local graph of L-hops, the computation still follows the Shapley value formula. This means the useful graph structure information among these nodes is forfeited which causes the structure-awareness concern of Shapley value as we discussed in Section 3,

H More explanation visualizations

Under the same setting as Figure 3, we visualize more explanations in 6.

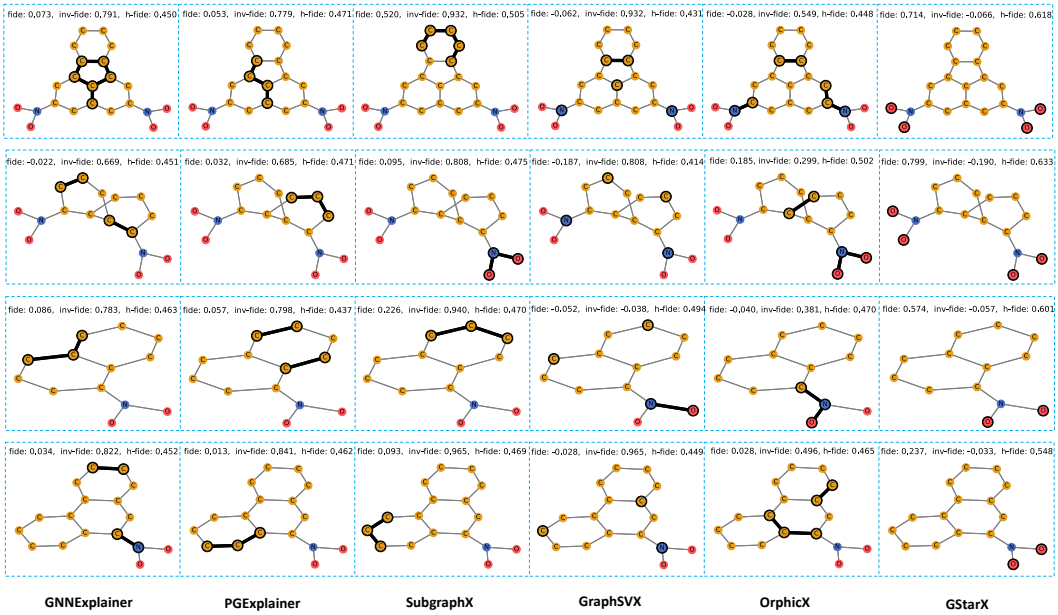

Figure 6: Explanations on a mutagenic molecule from the MUTAG dataset. Carbon atoms (C) are in yellow, nitrogen atoms (N) are in blue, and oxygen atoms (O) are in red. We use dark outlines to indicate the selected subgraph explanation and report the Fidelity (fide), Inv-Fidelity (inv-fide), and H-Fidelity (h-fide) of each explanation. GStarX gives a significant better explanation than other methods in terms of these metrics.