# Game Theoretic Explanations for Graph Neural Networks

Ataollah Kamal[1], Céline Robardet[1], and Marc Plantevit[2]

[1] Univ Lyon, INSA Lyon, LIRIS, UMR5205, F-69621 Villeurbanne
[2] EPITA Research Laboratory (LRE), FR-94276, Le Kremlin-Bicêtre, France

**Abstract.** Graph neural networks (GNN) are complex Machine Learning models that solve various graph tasks such as node classification, graph classification or link prediction. Due to their complexity, they are treated as black boxes, and how they perform their prediction is difficult to understand. In recent years, explainers for Machine Learning models have been introduced, among them methods based on game theory. These methods try to explain the decision by computing the importance of the features by considering them as players of a cooperative game who cooperate in order to make the decision. A player's impact on the decision is measured by his marginal contribution to a coalition of players. Different measures built on this principle exist, and they differ in the axioms they satisfy. In this article, we consider two such measures that we adapt to explain GNN.

## 1 Introduction

The advent of deep neural networks has had a significant influence on the fields of machine learning and artificial intelligence, leading to remarkable successes. These networks have demonstrated promising capabilities across diverse research domains such as computer vision and natural language processing, sparking a growing interest in applying deep learning methods to real-world interdisciplinary areas like chemistry, finance, biology, and more. Notably, graph neural networks (GNNs) have emerged as valuable tools for handling graph-related tasks, encompassing node classification, graph classification, and link predictions. GNNs offer the advantage of both bypassing the laborious process of manual feature engineering, which is typically required by pre-neural methods to turn a learning problem on graphs into one based on tabular data, and achieving exceptional performance. Consequently, the success of GNNs has spurred the development of advanced GNN operations, including graph convolution [7, 14], graph attention [18], and graph pooling [27], aimed at further enhancing their performance. However, as for most deep learning models, GNNs lack interpretability, which means that they operate like black boxes without providing an understanding of the underlying mechanisms behind their predictions. This limits their use in critical applications that require transparency, fairness, privacy, and safety. To overcome this limitation and deploy deep models safely and make them trustworthy, it is necessary to provide

both accurate predictions and human-intelligible explanations. Hence, there is a growing need for developing explanation techniques that can explain the workings of deep neural networks.

To cope with this problem, explainers have been introduced for GNNs [6, 26]. In general, these explainers can be categorized into two groups. The instance-level explainers which give explanations for each input, and the model-level ones, which give explanations for a class. Examples of such model-level explainers are XGNN [28] and GNNInterpreter [22], which generate a representative graph for a targeted class, or INSIDE [19], which is a rule-based explainer for a targeted class. The problem with the two former methods is they rely on a strong assumption that a decision can be explained by only a single graph. However, we can observe that this assumption is wrong in many real-world cases, such as molecular toxicity prediction that can be due to the several structures. The limit of INSIDE [19] is that it requires access to the hidden layers of the GNN, which is not possible in many situations (for instance when there are privacy concerns). Instance-level explainers usually provide explanations through a mask [1, 5, 9, 12, 15, 21, 25]. Such mask is obtained by perturbing the input, or studying the gradients. The related methods often optimize metrics on the mask which prevent assessing the individual contribution of entities inside or outside the mask, and leads to potentially misleading interpretations [20].

This drawback can be overcome by addressing the problem as a cooperative game. This type of methods tries to explain the model decision by computing the importance of the characteristics considered as players of a cooperative game who cooperate to make the decision. A player's impact on the decision is measured by his marginal contribution to a coalition of players. Different measures built on this principle exist, and they differ in the axioms they satisfy. The most famous solution is the Shapeley value [16] which fulfils important properties and has been widely used to assess the feature importance in black box decisions [2, 17].

However, Shapeley-based methods have two main limitations. The first limitation is their computational cost, which is, in the general case, NP-Hard due to the combinatorics underlying the coalitions to consider during their computation. To cope with this problem, some efficient alternatives, with a polynomial computation complexity, have been proposed for specific ML models [8]. Another direction to reduce this complexity is to modify the axioms defining the measure, as in [3]. However, such an approach has not yet been applied to GNNs. The second limitation of Shapeley values is their non-consideration of the graph structure within player coalition building, when used to explain GNNs. To overcome this problem, various approaches have been proposed to provide game-theory rooted GNN explanations. GraphSVX [4] computes an approximation of the Shapley values on the node features and nodes themselves by sampling the nodes and features as coalitions. Still, this method does not take into account the graph structure. GraphSHAP [11] uses frequent subgraphs as features. This can be problematic for datasets without node labels like BA2 as the number of such patterns is exponential. GStarX [29] takes ad-

vantage of Hamiache-Navarro value to handle the graph structure in player coalitions. However, this approach suffers from the computational cost of the player's marginal contribution. Even so it considers the structure, only an approximate algorithm can be effective.

In this paper, we study axiomatic-based methods inspired by game theory approaches for explaining GNNs. We introduce EGO-SHAP which considers ego-networks as features. This graph concept is at the core of GNNs that learn vertex embedding based on what the vertices perceived through message passing. Each node perceives a neighborhood whose size increases with the number of layers considered, and that corresponds to an ego-network of the same size. Furthermore, this allows to partially take into account the graph structure while limiting the number of features, keeping it equal to the number of vertices. The importance of each ego-network in the decision is then computed. To address the issue of Shapeley value computation cost, we propose a method approximating Shapeley values in polynomial time. Then, we introduce the Efficient Symmetric Perturbation Attribution Method (ESPAM) for graphs. Built on FESP [3], it is an exact polynomial-time method that computes the contribution of each node while considering the graph structure. ESPAM_1 is a new method based on FESP. It computes the contribution of each node, by having several cooperative games in which ego-networks are players and GNN is the value function. ESPAM_2 is a direct adaptation of FESP to graphs where the super-pixels are replaced by fixed-size ego-networks. To have more accurate results and respect the structure, we introduce ESPAM_3 which is the generalization of the ESPAM_2 by considering ego-networks with different radius. All of these three methods, not only respect the structure and outperform state-of-the-art explainers, but also produce explanations in polynomial time with an exact algorithm.

The rest of the paper is organized as follows. We introduce EGO-SHAP in Section 2. Then, ESPAM is proposed in Section 3 to deal with the problem of the complexity of exact computation of Shapeley values. We report an extensive experimental study on several datasets against numerous State-of-the-art methods in Section 4. Section 5 concludes this paper and discuss the future directions of research.

## 2 Shapley values to explain decisions of graph classification models

### 2.1 Shapley values

Shapley values, introduced in [16], are defined axiomatically in game theory to measure the impact of each player of a cooperative game on the game outcome. This impact is evaluated by the marginal contribution of the player considering a coalition of players. Let $(N, f)$ be a cooperative game, with $N$ the set of players and $f$ a function $f : 2^N \to \mathbb{R}$ of the set of all possible coalitions of players to the outcome of the game, with $f(\emptyset) = 0$. This function describes the collective

payoff that a set of players can obtain by forming a coalition. Shapley values are real-valued functions $\phi_i(f)$ that measure the contribution of a player $i$ on $f$. Shapley values are a way of distributing total gains to players in a fair way defined by the following axioms:

*Efficiency.* All the gain is distributed among the players: $f(N) = \sum_{i \in N} \phi_i(f)$.

*Null Player.* If the contribution of a player to the game is null, his Shapley value equals zero. A player $i$ does not contribute to the game, if when he participates in a coalition, it does not change the outcome of the game, whatever the considered coalition: $f(S \cup \{i\}) = f(S), \forall S \subseteq N \Rightarrow \phi_i(f) = 0$.

*Symmetry.* If $i$ and $j$ are two equivalent players, that is to say if $f(S \cup \{i\}) = f(S \cup \{j\})$ for every subset $S$ of $N$ which contains neither $i$ nor $j$, then $\phi_i(f) = \phi_j(f)$.

*Additivity.* Considering two cooperative games $(N, f)$ and $(N, g)$, then $\phi_i(f + g) = \phi_i(f) + \phi_i(g), \forall i \in N$.

It has been proved in [16] that Shapley values, defined as

$$\phi_i(f) = \frac{1}{|N|} \sum_{S \subseteq N \setminus \{i\}} \binom{|N|-1)}{|S|} \Big( f(S \cup \{i\}) - f(S) \Big) \tag{1}$$

$$\phi_i(f) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big( f(S \cup \{i\}) - f(S) \Big)$$

are the only payment rule satisfying the four aforementioned axioms.

### 2.2 Shapley values on graphs

Using Shapley values to explain graph classification models requires defining appropriately what is considered a player. The players must, at the same time, represent elementary information of the prediction process, be related to an internal mechanism in the model, and finally be limited in number, due to the complexity of Shapley values computation that increases exponentially with the number of players. In [11], Perotti et al. propose to use frequent connected subgraphs as players, underestimating the complexity generated by the potential large number of players and the difficulty in setting the frequency threshold. Here we limit the number of players by considering ego-networks of fixed diameter $\ell$, with $\ell$ smaller of equal to the number of layers of the GNNs. We do have a reduced set of players, related to the structural information considered by the model.

**Definition 1 ($\ell$-ego-network).** *Given a graph $G$, a $\ell$-ego-network $\epsilon_\ell(v)$, for $v$ a node of $G$, is the subgraph of $G$ that contains $v$ and all the nodes $w$ and edges $e$ at distance at most $\ell$. $\epsilon_\ell(G)$ is the set of all $\ell$-ego-networks of $G$.*

As an explanation of a graph $G$, we take the most contributing ego-network. We could also consider a set of most contributing ego-networks, either by fixing a number $k$ of such networks or by considering a threshold on the function evaluating the contributions of the ego-networks. For reasons of simplicity, and in order to have a parsimonious explanation, we choose to take a single ego-network. Thus, we are only interested in the order of Shapley values. We propose to approximate these values with Algorithm 1. In this algorithm, $G(S)$ represents the result of masking $G$ by the set $S$ of nodes. It consists in removing all the incident edges of nodes in $S$. Algorithm 1 does not consider all possible coalitions, but those that may impact the Shapley value of a given ego-network. To that end, it uses weighted sampling so that the nodes with lower distances to nodes in the coalition have more chance to be selected. Indeed, two close nodes are more likely to have a mutual effect on each other. In an extreme case, if two nodes are not connected, then their embeddings are not affected by each others in the GNN. To estimate $\phi_v$, the algorithm samples a set of nodes $S$ based on their geodesic distances from $v$ (the minimum number of edges needed to reach a node from $v$), such that the closer the node, the higher chance to be selected. The size of $S$ is uniformly sampled in interval $[\alpha n, \beta n]$, with $\alpha, \beta$ two hyperparameters set to 0.2 and 0.8 respectively. These two values have been set empirically so that the size of $S$ is around a third of the size of the graph. It then uses this set to estimate the difference between the predicted value for $G(S)$ and $G(S \cup \{v\})$. The coefficient applied to this difference is $\frac{1}{\#|S|}$ where $\#|S|$ is the number of times that we sample a set with the same size as $S$. In the end, we divide all the values by the number of nodes to uniformly distribute the contribution between the nodes. Note that if we do not sample any subset twice and $I = 2^{n-1}$ (with $n = |V|$), then the computed value is the exact Shapley value. Finally, the algorithm returns the ego network with the maximum value.

As we will see in Section 4, the computation time of EGO-SHAP is still very long to achieve good quality approximations. Another way to overcome this problem is to modify the axioms defining the attribution measure of the cooperative game.

## 3 Efficient Symmetric Perturbation Attribution Method

We consider the alternative game-theoretic based attribution function proposed in [3]. This class of functions is defined to satisfy the properties of efficiency and symmetry, but the additivity is generalized and changed to linearity, and the null player property is changed to fair treatment. Before explaining the interest of such changes, we recall the definitions of these two new properties.

**Definition 2.** *Considering the cooperative game $(N, f)$ and $\phi$ a contribution function defined on this game, linearity and fair treatment properties are defined by:*

- ***Linearity property.*** *Considering two cooperative games $(N, f)$ and $(N, g)$ and two scalars $\alpha_1, \alpha_2 \in \mathbb{R}$, then $\phi_i(\alpha_1 f + \alpha_2 g) = \alpha_1 \phi_i(f) + \alpha_2 \phi_i(g), \forall i \in N$.*

**Algorithm 1:** EGO-SHAP

**Data:** $G = (V, E)$ a graph with $n = |V|$, $\epsilon_\ell(G)$ the ego-network, $f_c$ the GNN decision for class $c$, $I$ the number of iterations.
**Result:** The most contributing ego network.
**for** $v \in V$ **do**
  $w_v \leftarrow 0$;
  $Sum \leftarrow \sum_w d_G(v, w)$;
  $Size \leftarrow$ Array of zeros of length $n$;
  $D \leftarrow$ Array of zeros of length $n$;
  **for** $u \in V, u \neq v$ **do**
    $w_u \leftarrow \frac{1}{n-2}\left(1 - \frac{d_G(u,v)}{Sum}\right)$
  **end**
  **for** $i = 1$ ***to*** $I$ **do**
    /* Number of elements in the coalition */
    $t \leftarrow \mathcal{U}(\alpha \times n, \beta \times n)$;
    $S \leftarrow \emptyset$;
    **for** $j = 1$ ***to*** $t$ **do**
      Random draw of $s \sim w_s$ ;
      $S \leftarrow S \cup \{s\}$;
    **end**
    $D[t] \leftarrow D[t] + f_c(G(S)) - f_c(G(S \cup \{v\}))$;
    $Size[t] \leftarrow Size[t] + 1$;
  **end**
  $\phi_v \leftarrow \frac{1}{n} \sum_{t=1}^{n} \frac{D[t]}{Size[t]}$;
**end**
**return** $argmax_{v \in V} \phi_v$

– ***Fair treatment property.*** *If the contribution of a player $i$ is lower than the one of another player $j$, for all coalitions, its contribution function value is lower than the one of $j$: For $i, j \in N$, if $f(S \cup \{i\}) \leq f(S \cup \{j\})\ \forall S \subseteq N \setminus \{i, j\}$, then $\phi_i(f) \leq \phi_j(f)$.*

As demonstrated in [3], the only function that meets these four axioms (linearity, symmetry, efficiency, and fair treatment) is defined as:

$$\varphi_i(f) = w \times f(\{i\}) - (1 - w) \times f(N \setminus \{i\}) \quad (2)$$
$$\text{with } w = \frac{f(N) + \sum_{j \in N} f(N \setminus \{j\})}{\sum_{j \in N} f(\{j\}) + \sum_{j \in N} f(N \setminus \{j\})}$$

This function has been used to explain models on texts or images, but to the best of our knowledge, there is no such function to explain models on graphs. Here, we build similar functions for graph models with one difference: they do not meet the fair treatment property. In Subsection 3.3, we argue that it is not a problem, and even more, that it is not reasonable to have this property. We propose three methods which we call ESPAM_1, ESPAM_2 and ESPAM_3. In all

of the following methods, the set of players $N$ is the set of nodes of the graph to be explained.

### 3.1 ESPAM_1

We consider the $\ell$-ego-network of node $v$, $\epsilon_\ell(v)$ and $\varphi^\ell$ defined by Equation (2) on the game $(\epsilon_\ell(G), f_c)$, with $f_i(S)$ the predicted probability of the model $f$ for the class $c$ on the subgraph defined as the union of $\epsilon_\ell(v)$, $v \in S$. To define the contribution of each node $v$, we compute the following function:

$$\psi_v(f_c) = \frac{1}{k} \sum_{\ell=1}^{k} \varphi_v^\ell(f_c)$$

$\psi_v(f_c)$ conserves the linearity property, as the function is a linear combination of $\varphi_v^\ell(f_c)$ that satisfies the linearity property. Since $\varphi_v^\ell(f_c)$ is symmetric, the average over $\ell$ of these functions is also. For the efficiency, we have $\sum_{v \in V} \psi_v(f_c) = \sum_{v \in V} \frac{1}{k} \sum_{\ell=1}^{k} \varphi_v^\ell(f_c) = \frac{1}{k} \sum_{\ell=1}^{k} \sum_{v \in V} \varphi_v^\ell(f_c) = \frac{1}{k} \sum_{\ell=1}^{k} f_c(G) = f_c(G)$, as $\varphi_v^\ell(f_c)$ satisfies the efficiency property.

According to [29] a contribution function for structured games (games with a structure on $N$, defined as $(N, E, f)$ where $E \subseteq \{(u, v) : u, v \in N\}$) should meet two axioms to respect the structure. First, if there is no connection between a node $i$ and a coalition $S$, then the effect of $S$ on the contribution of $i$ should be zero. Here, by definition (see equation 2, with $f$ a GNN), all the non-zero contributions are between a node and a coalition connected to the node. Second, the contributions should be impacted by the distance between $i$ and $S$. This is also the case here, as coalitions $S$ at a distance lower than $\ell$ will have more effect than others on $\psi$. Therefore, our formulation respects the graph structure.

Note that in our setting computing the contribution is done in a linear time for each node. Therefore, we have two advantages with respect to the Shapley-based method. First, the time complexity is polynomial, and second, ESPAM_1 respects the structure while, as stated in [29], Shapeley value does not.

### 3.2 ESPAM_2 and ESPAM_3

For the explanation of models on images, [3] propose to use sets of pixels, called superpixels, as features of the model. Then, after computing their contribution, they transfer the contribution of the superpixel to each pixel inside it. We use this idea on graphs where ego-networks act as superpixels. We propagate the contribution of the ego-network on each node of it in a uniform way to preserve efficiency. It should be mentioned that the following two methods are linear since each operator is linear. Similarly, thanks to the invariance of graphs to permutation, the methods preserve symmetry. We propose two approaches: one with $\ell$-ego-networks with fixed $\ell$ (ESPAM_2), and another one based on the average value for a range of radius (ESPAM_3). Algorithm 2 describes ESPAM_3. ESPAM_2 is the special case of ESPAM_3 with $\ell_{min} = \ell_{max}$. In ESPAM_3, for each

$\ell_{min} \leq \ell \leq \ell_{max}$, we compute the contribution of each ego network $s$ in $\epsilon_\ell(G)$ (see equation 2), then we uniformly distribute it to its nodes (the line inside the most inner loop), and finally for each node $v$ we divide this value ($\xi_v$) by the number of radii that we had ($\ell_{max} - \ell_{min} + 1$). More formally:

$$\xi_v(f_c) = \frac{1}{\ell_{max} - \ell_{min} + 1} \sum_{\ell=\ell_{min}}^{\ell_{max}} \sum_{s \in \epsilon_\ell(G) \& v \in V(s)} \frac{\varphi_s(f_c)}{|V(s)|}$$

**Algorithm 2:** ESPAM_3

```
Data: Graph G, ℓ_min , ℓ_max, Decision class c, GNN f
Result: Contribution of each node of G
ξ ← an array of all zeros, the same size as V;
for ℓ = ℓ_min to ℓ_max do
    for s ∈ ε_l(G) do
        t ← φ_s(f_c);
        for v ∈ V(s) do
            ξ_v ← ξ_v + t/|V(s)|;
        end
    end
end
return ξ/(ℓ_max − ℓ_min + 1);
```

### 3.3 Discussion on the Fair Treatment Property

As stated in [13], a measure $\phi$ satisfies the fair treatment property if and only if there exist non-negative constants $\{b_s\}_{s=0}^{n}$ such that

$$\phi_i(f) = \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} \Big( b_{s+1} f(S \cup \{i\}) - b_s f(S) \Big)$$

for $i \in N$, $s = |S|$ and $n = |N|$. By summing up all the contributions, we have:

$$f(N) = \sum_{\ell=1}^{n} c_\ell \sum_{|S|=\ell} f(S) - \sum_{\ell=0}^{n-1} \ell \times c'_\ell \sum_{|S|=\ell} f(S)$$

where $c_\ell = \ell \frac{\ell!(n-\ell-1)!}{n!} b_{\ell-1}$ and $c'_\ell = (n-\ell) \frac{\ell!(n-\ell-1)!}{n!} b_\ell$. By assuming $f(\emptyset) = 0$, we have

$$(1 - c_n) f(N) = \sum_{\ell=1}^{n-1} T_\ell \sum_{|S|=\ell} f(S)$$

with $T_\ell = c_\ell - c'_\ell$. As it can be seen, the probability on the whole graph should be a linear function of the sum of probabilities on the subset of features of the same size. However, regarding how a GNN makes the decision, this does not hold for most of the cases due to the role of the structure in the decision.

## 4 Experimental Evaluation

To evaluate our work, we used four datasets BA2 [10], Aids [23], Mutagen [30], and BBBP [24], and six baselines GNNExplainer, PGExplainer, Grad, PGMExplainer, GraphSVX and GStarX. The two latter are rooted in cooperative game theory. BA2 is a synthetic dataset in which positive label graphs have a house motif and negative ones have a 5-cycle. The three other datasets are real-world molecule datasets that have information about activity against HIV, mutagenicity, and blood-brain barrier penetration respectively. All the experiments have been done on the machine equipped with 8 Intel(R) Xeon(R) W-2125 CPU @ 4.00GHz cores 126GB main memory, running Debian GNU/Linux.

Each dataset has been divided into three parts (training, validation and test sets). Training and validation sets have been used to learn respectively the parameters and hyperparameters of the GNN models. We evaluate the different explanations on a test set corresponding to ten percent of the data taken at random. Fidelity and Infidelity measure the dependence of the decision on the explanation. Furthermore, Fidelity states how much different the decision of the part of the input not included in the explanation is from the original decision, and Infidelity measures how much the explanation is different from the original graph in terms of the decision. Therefore, high Fidelity and low Infidelity are desirable. Note that if we put the original graph as the explanation, we will have zero Infidelity, even if such an explanation is not a good. To cope with this problem, Sparsity controls the size of the explanation. With higher Sparsity, we have a more human-readable explanation. In the following, we have the formal definitions of these three metrics:

$$\text{Fidelity}(G, g) = f_{c^\star}(G) - f_{c^\star}(G \setminus g)$$

where $c^\star$ is the class with the highest predicted probability by $f$ for $G$.

$$\text{Infidelity(G, g)} = f_{c^\star}(G) - f_{c^\star}(g)$$

$$\text{Sparsity}(G, g) = 1 - \frac{|g|}{|G|}$$

where $|.|$ is the size of the edge set. To capture the effectiveness of explanations on the decision, we have two very similar metrics for Fidelity and Infidelity which we call them Fidelity$^{Acc}$ and Infidelity$^{Acc}$. Their definitions are:

$$\text{Fidelity}^{Acc}(G, g) = 1 - \mathbb{1}_{(C(G)=C(G\setminus g)}$$

**Table 1.** Comparison of methods through four datasets.

| Method | Aids | | | Mutagen | | | BBBP | | | BA2 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Measure | Fidelity | Infidelity | Sparsity | Fidelity | Infidelity | Sparsity | Fidelity | Infidelity | Sparsity | Fidelity | Infidelity | Sparsity |
| EGO-SHAP(R=1) | 0.123 | 0.699 | 0.916 | 0.132 | 0.305 | 0.988 | 0.181 | 0.543 | 0.971 | 0.171 | 0.000 | 0.622 |
| EGO-SHAP(R=2) | 0.118 | 0.707 | 0.912 | 0.279 | 0.210 | 0.969 | 0.166 | 0.440 | 0.932 | 0.189 | 0.219 | 0.622 |
| EGO-SHAP(R=3) | 0.091 | 0.687 | 0.917 | 0.210 | 0.174 | 0.944 | 0.192 | 0.471 | 0.888 | 0.154 | 0.215 | 0.365 |
| ESPAM1 $\gamma = 0.2$ | 0.045 | 0.618 | 0.922 | 0.266 | 0.193 | 0.948 | 0.157 | 0.433 | 0.910 | 0.230 | -0.066 | 0.562 |
| ESPAM1 $\gamma = 0.3$ | 0.056 | 0.660 | 0.886 | 0.263 | 0.191 | 0.929 | 0.187 | 0.461 | 0.873 | 0.225 | -0.057 | 0.473 |
| ESPAM1 $\gamma = 0.4$ | 0.062 | 0.676 | 0.851 | 0.259 | 0.211 | 0.915 | 0.195 | 0.461 | 0.840 | 0.217 | -0.046 | 0.347 |
| ESPAM2 $\gamma = 0.2$ | 0.068 | 0.677 | 0.889 | 0.254 | 0.197 | 0.952 | 0.116 | 0.471 | 0.910 | 0.225 | -0.062 | 0.603 |
| ESPAM2 $\gamma = 0.3$ | 0.077 | 0.679 | 0.846 | 0.255 | 0.235 | 0.936 | 0.143 | 0.482 | 0.876 | 0.181 | -0.031 | 0.414 |
| ESPAM2 $\gamma = 0.4$ | 0.074 | 0.684 | 0.810 | 0.256 | 0.266 | 0.922 | 0.156 | 0.505 | 0.843 | 0.165 | -0.024 | 0.306 |
| ESPAM3 $\gamma = 0.2$ | 0.075 | 0.653 | 0.884 | 0.231 | 0.166 | 0.951 | 0.135 | 0.411 | 0.906 | 0.156 | -0.001 | 0.486 |
| ESPAM3 $\gamma = 0.3$ | 0.077 | 0.664 | 0.843 | 0.243 | 0.193 | 0.934 | 0.165 | 0.451 | 0.873 | 0.147 | -0.010 | 0.385 |
| ESPAM3 $\gamma = 0.4$ | 0.072 | 0.678 | 0.809 | 0.244 | 0.236 | 0.919 | 0.169 | 0.465 | 0.844 | 0.146 | 0 | 0.262 |
| GNNExplainer | 0.036 | 0.494 | 0.501 | 0.177 | 0.237 | 0.505 | 0.100 | 0.099 | 0.501 | 0.093 | 0.223 | 0.619 |
| PGExplainer | 0.032 | 0.038 | 0.547 | 0.157 | 0.157 | 0.515 | 0.098 | 0.098 | 0.534 | 0.004 | 0.353 | 0.955 |
| Grad | 0.078 | 0.766 | 0.910 | 0.223 | 0.357 | 0.978 | 0.171 | 0.447 | 0.938 | 0.195 | 0.334 | 0.804 |
| PGMExplainer | 0.089 | 0.765 | 0.855 | 0.260 | 0.354 | 0.956 | 0.212 | 0.392 | 0.884 | 0.201 | 0.270 | 0.747 |

where $C(.)$ is the predicted class by the model.

$$\text{Infidelity}^{Acc}(G, g) = 1 - \mathbb{1}_{C(G)=C(g)}$$

**Table 2.** $Fidelity^{Acc}$ and $Infidelity^{Acc}$ comparison.

| Method | Aids | | Mutagen | | BBBP | | BA2 | |
|---|---|---|---|---|---|---|---|---|
| Measure | $Fidelity^{Acc}$ | $Infidelity^{Acc}$ | $Fidelity^{Acc}$ | $Infidelity^{Acc}$ | $Fidelity^{Acc}$ | $Infidelity^{Acc}$ | $Fidelity^{Acc}$ | $Infidelity^{Acc}$ |
| EGO-SHAP(R=1) | 0.030 | 0.683 | 0.219 | 0.514 | 0.137 | 0.699 | 0.43 | 0.15 |
| EGO-SHAP(R=2) | 0.066 | 0.769 | 0.433 | 0.323 | 0.144 | 0.644 | 0.43 | 0.57 |
| EGO-SHAP(R=3) | 0.045 | 0.755 | 0.352 | 0.306 | 0.156 | 0.692 | 0.43 | 0.57 |
| ESPAM1 $\gamma = 0.2$ | 0.035 | 0.713 | 0.396 | 0.290 | 0.139 | 0.629 | 0.43 | 0.05 |
| ESPAM1 $\gamma = 0.3$ | 0.055 | 0.723 | 0.424 | 0.346 | 0.172 | 0.675 | 0.43 | 0.02 |
| ESPAM1 $\gamma = 0.4$ | 0.050 | 0.738 | 0.403 | 0.405 | 0.178 | 0.688 | 0.43 | 0.04 |
| ESPAM2 $\gamma = 0.2$ | 0.035 | 0.713 | 0.396 | 0.290 | 0.139 | 0.629 | 0.43 | 0.05 |
| ESPAM2 $\gamma = 0.3$ | 0.055 | 0.723 | 0.424 | 0.346 | 0.172 | 0.675 | 0.43 | 0.02 |
| ESPAM2 $\gamma = 0.4$ | 0.050 | 0.738 | 0.403 | 0.405 | 0.178 | 0.688 | 0.43 | 0.04 |
| ESPAM3 $\gamma = 0.2$ | 0.035 | 0.713 | 0.396 | 0.290 | 0.139 | 0.629 | 0.43 | 0.05 |
| ESPAM3 $\gamma = 0.3$ | 0.055 | 0.723 | 0.424 | 0.346 | 0.172 | 0.675 | 0.43 | 0.02 |
| ESPAM3 $\gamma = 0.4$ | 0.050 | 0.738 | 0.403 | 0.405 | 0.178 | 0.688 | 0.43 | 0.04 |
| GNNExplainer | 0.010 | 0.010 | 0.165 | 0.193 | 0.072 | 0.092 | 0.18 | 0.43 |
| PGExplainer | 0.0 | 0.025 | 0.007 | 0.349 | 0.0 | 0.132 | 0.03 | 0.43 |
| Grad | 0.020 | 0.768 | 0.268 | 0.485 | 0.086 | 0.662 | 0.18 | 0.43 |
| PGMExplainer | 0.015 | 0.768 | 0.235 | 0.782 | 0.066 | 0.668 | 0.43 | 0.43 |

For each dataset, the average values of Fidelity, Infidelity and Sparsity are reported in Table 1. Complementary results with Fidelity$^{Acc}$ and Infidelity$^{Acc}$ are provided in Table 2. Note that for the Shapley approximation, the explanation is already defined and for the ESPAMs, we need a human-interpretable explanation. A contribution vector itself cannot demonstrate the significant part of the graph responsible for the prediction. To this end, an explanation is the induced subgraph on the 1-hop of the top $\gamma$ percent nodes with respect to their contribution score. $\gamma$ is a hyperparameter which, in practice, we set to 0.2, 0.3, and 0.4 values. Regarding Tables 1 and 2, ESPAMs and EGO-SHAP outperform SOTA methods on Mutagen and BA2 datasets. On Aids, they had superior Fidelity and Fidelity$^{acc}$ than SOTA methods. However, in terms of Infidelity and Infidelity$^{acc}$, their values are less than most of the SOTA methods but it should be mentioned that for those SOTAs with the lower Infidelity and Infidelity$^{acc}$, the Sparsity is low. We mentioned before why such an explanation is not good. Finally, on the BBBP, ESPAMs have Fidelity, Fidelity$^{acc}$, and Sparsity advantage over the SOTA methods.

We empirically study in Fig. 1 the number of iterations needed by EGO-SHAP to converge. This requires approximately 1000 iterations for an instance of BBBP. This implies too long running times when explaining several instances (e.g., about 11 hours on BBBP test instances as demonstrated in Table 4). This makes EGO-SHAP non competitive according to other explainers except GstarX which took 2 days.

Note that Fidelity, Infidelity and Sparsity must be considered together. In case we focus on one of them, we probably get satisfying metric values but not a good explanation. E.g., the whole graph would have the best Fidelity but also the worst Sparsity. As so, we have another metric H-Fidelity which is a combination of Fidelity and Infidelity normalized by Sparsity [29]. To define

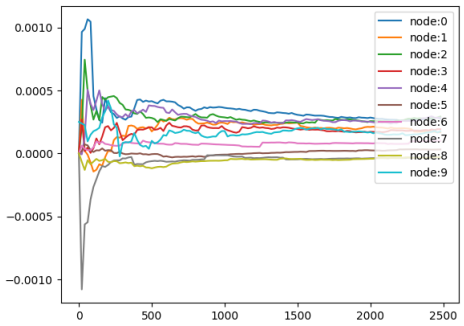


**Fig. 1.** Number of iterations needed to converge for BBBP for EGO-SHAP.

**Table 3.** Comparison of ESPAMs and EGO-SHAP with game-theoretic baselines using H-Fidelity.

| Method | BA2 | Aids | Mutagen | BBBP |
|---|---|---|---|---|
| EGO-SHAP(R=1) | 0.525 | 0.507 | 0.525 | 0.501 |
| EGO-SHAP(R=2) | 0.503 | 0.505 | 0.550 | 0.490 |
| EGO-SHAP(R=3) | 0.471 | 0.488 | 0.532 | 0.484 |
| ESPAM1 | **0. 532** | **0.510** | 0.573 | **0.542** |
| ESPAM2 | 0.531 | **0.510** | **0.574** | 0.534 |
| ESPAM3 | 0.530 | **0.510** | 0.565 | 0.538 |
| GstarX$\gamma = 0.2$ | 0.507 | 0.496 | 0.489 | 0.481 |
| GstarX$\gamma = 0.3$ | 0.500 | 0.498 | 0.494 | 0.483 |
| GstarX$\gamma = 0.4$ | 0.495 | 0.499 | 0.498 | 0.485 |
| GraphSVX$\gamma = 0.2$ | 0.503 | 0.505 | 0.523 | 0.508 |
| GraphSVX$\gamma = 0.3$ | 0.491 | 0.499 | 0.524 | 0.499 |
| GraphSVX$\gamma = 0.4$ | 0.481 | 0.493 | 0.527 | 0.489 |

the equations, we first define the normalized Fidelity:

$$\text{N-Fidelity} = \text{Fidelity}(G, g).(1 - \frac{|g|}{|G|})$$

Similarly, the normalized Infidelity is defined as:

$$\text{N-Infidelity} = \text{Infidelity}(G, g).\frac{|g|}{|G|}$$

Now assume that $m_1$ = N-Fidelity and $m_2$ = N-Infidelity. Then, harmonic Fidelity (H-Fidelity) is:

$$\text{H-Fidelity}(G, g) = \frac{(1 + m_1)(1 - m_2)}{(2 + m_1 - m_2)}$$

To compare our methods by the mean of this metric, we have used the same dataset and baselines of the same family of our methods (i.e. game theoretic

methods). As the baselines, we use GraphSVX [4] and GstarX [29]. Now that we have only one metric, in the ESPAMs method, instead of setting $\gamma$ manually, we use dual annealing to optimize H-Fidelity by $\gamma$. Dual Annealing is a stochastic global optimization algorithm based on simulated annealing and local search algorithms. Results are shown in Table 3. ESPAMs outperform their competitors in terms of H-Fidelity. The execution times needed to provide explanations on the test instances are reported in Table 4. The only structure-preserver among the cooperative game competitors is GstarX which had less H-Fidelity and took two days for each dataset to calculate the explanations while ESPAMs had a maximum execution time of less than five minutes. EGO-SHAP provides effective explanations since it outperforms GstarX and GraphSVX on H-Fidelity. However, it is computationally expensive. Considering this, ESPAM methods are an efficient alternative.

**Table 4.** Time Comparison (in Seconds).

| Method | Aids | Mutagen | BBBP | BA2 |
|---|---|---|---|---|
| EGO-SHAP(R=1) | 23411.46 | 202171.66 | 42409.054 | 22158.78 |
| ESPAM1 | 36.19 | 217.85 | 217.85 | 26.08 |
| ESPAM2 | 13.28 | 78.26 | 16.52 | 9.28 |
| ESPAM3 | 36.46 | 225.45 | 45.33 | 28.84 |
| GNNExplainer | 215.82 | 77.89 | 221.82 | 217.98 |
| PGExplainer | 23.02 | 16.46 | 11.44 | 2.75 |
| Grad | 17.25 | 77.89 | 55.85 | 4.237 |
| PGMExplainer | 6114.68 | 4104.46 | 7755.01 | 10055.79 |

In Figure 2, we report some explanations made by three cooperative game explainers: ESPAM1, GstarX, and GraphSVX for an instance of BA2 dataset with a negative label (i.e. it has a $C_5$ as an induced subgraph). As it can be seen, the only explainer that captured $C_5$ is ESPAM1.

## 5 Conclusion

In this paper, we addressed the problem of explaining GNN decisions. To this end, we devised EGO-SHAP and ESPAM methods rooted in cooperative game theory. EGO-SHAP computes the Shapeley value of each ego-network of a given radius. This allows to take into account the graph structure while keeping a linear number of players. However, the Shapeley value is costly to approximate. We then introduced ESPAM methods to compute the contribution of each ego-network in polynomial time. We reported an empirical study on four datasets against several state-of-the-art methods. Both EGO-SHAP and ESPAM outperform state-of-the-art methods. While EGO-SHAP needs time to compute the explanations, ESPAMs obtain explanations in a very short amount of time. In addition, ESPAMs respect the graph structure in the player coalitions while the

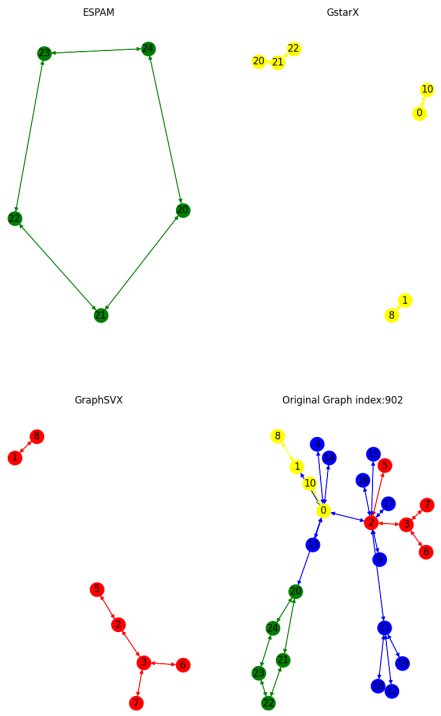


**Fig. 2.** Comparison of the explanations made by ESPAM1, GraphSVX and GstarX on BA2 for an instance with the negative label.

only game-theoretic GNN explainer with such a property – GStarX – takes days to compute explanations. Our findings pave the way for further research and development in this area, including exploring other ways to better take into account the graph structure, and handling multiple features on vertices as well as measures to assess the importance of the structure against the content itself.

## References

1. Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(1):47–56, jan 2005.
2. Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.
3. Charles Condevaux, Sébastien Harispe, and Stéphane Mussard. Fair and efficient alternatives to shapley-baszhanged attribution methods. *Machine Learning and Knowledge Discovery in Databases*, page 309–324, 2023.
4. Alexandre Duval and Fragkiskos D. Malliaros. Graphsvx: Shapley value explanations for graph neural networks, 2021.

5. Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. Graphlime: Local interpretable model explanations for GNNs. *arXiv:2001.06216*, 2020.
6. Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. A survey on explainability of graph neural networks. *arXiv preprint arXiv:2306.01958*, 2023.
7. Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
8. Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):56–67, Jan 2020.
9. 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.
10. Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network, 2020.
11. Alan Perotti, Paolo Bajardi, Francesco Bonchi, and André Panisson. Graphshap: Motif-based explanations for black-box graph classifiers, 2022.
12. Phillip E. Pope, Soheil Kolouri, Mohammad Rostami, Charles E. Martin, and Heiko Hoffmann. Explainability methods for GCN. In *IEEE CVPR 2019*, pages 10772–10781, 2019.
13. Tadeusz Radzik and Theo Driessen. On a family of values for tu-games generalizing the shapley value. *Mathematical Social Sciences*, 65(2):105–111, 2013.
14. Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
15. Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T. Schütt, Klaus-Robert Müller, and Grégoire Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7581–7596, 2022.
16. Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.
17. Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR, 2020.
18. Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
19. Luca Veyrin-Forrer, Ataollah Kamal, Stefan Duffner, Marc Plantevit, and Céline Robardet. What does my GNN really capture? on exploring internal GNN representations. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 747–752. ijcai.org, 2022.

20. Luca Veyrin-Forrer, Ataollah Kamal, Stefan Duffner, Marc Plantevit, and Céline Robardet. On gnn explainability with activation rules. *Data Mining and Knowledge Discovery*, 2022.
21. Minh N. Vu and My T. Thai. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. In *NeurIPS 2020*, 2020.
22. Xiaoqi Wang and Han-Wei Shen. Gnninterpreter: A probabilistic generative model-level explanation for graph neural networks. *arXiv preprint arXiv:2209.07924*, 2022.
23. Bo Wu, Yang Liu, Bo Lang, and Lei Huang. Dgcnn: Disordered graph convolutional neural network based on the gaussian mixture model, 2017.
24. 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, 2017.
25. Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for GNNs. In *NeurIPS 2019*, pages 9240–9251, 2019.
26. 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*, 45(05):5782–5799, may 2023.
27. Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.
28. Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 430–438, New York, NY, USA, 2020. Association for Computing Machinery.
29. Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. In *Advances in Neural Information Processing Systems*, 2022.
30. Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical graph pooling with structure learning, 2019.