GraphSVX: Shapley Value Explanations for Graph Neural Networks

Alexandre Duval, Fragkiskos D. Malliaros

GraphSVX: Shapley Value Explanations for Graph Neural Networks

Alexandre Duval ✉ and Fragkiskos D. Malliaros

Université Paris-Saclay, CentraleSupélec, Inria, France
{alexandre.duval, fragkiskos.malliaros}@centralesupelec.fr

Abstract. Graph Neural Networks (GNNs) achieve significant performance for various learning tasks on geometric data due to the incorporation of graph structure into the learning of node representations, which renders their comprehension challenging. In this paper, we first propose a unified framework satisfied by most existing GNN explainers. Then, we introduce GraphSVX, a post hoc local model-agnostic explanation method specifically designed for GNNs. GraphSVX is a decomposition technique that captures the "fair" contribution of each feature and node towards the explained prediction by constructing a surrogate model on a perturbed dataset. It extends to graphs and ultimately provides as explanation the Shapley Values from game theory. Experiments on real-world and synthetic datasets demonstrate that GraphSVX achieves state-of-the-art performance compared to baseline models while presenting core theoretical and human-centric properties.

1 Introduction

Many aspects of the everyday life involve data without regular spatial structure, known as non-euclidean or geometric data, such as social networks, molecular structures or citation networks [1,10]. These datasets, often represented as graphs, are challenging to work with because they require modelling rich relational information on top of node feature information [37]. Graph Neural Networks (GNNs) are powerful tools for representation learning of such data. They achieve state-of-the-art performance on a wide variety of tasks [8,36] due to their recursive message passing scheme, where they encode information from nodes and pass it along the edges of the graph. Similarly to traditional deep learning frameworks, GNNs showcase a complex functioning that is rather opaque to humans. As the field grows, understanding them becomes essential for well known reasons, such as ensuring privacy, fairness, efficiency, and safety [20].

While there exist a variety of explanation methods [25, 27, 29], they are not well suited for geometric data as they fall short in their ability to incorporate graph topology information. [2, 21] have proposed extensions to GNNs, but in addition to limited performance, they require model internal knowledge and show gradient saturation issues due to the discrete nature of the adjacency matrix.

GNNExplainer [33] is the first explanation method designed specifically for GNNs. It learns a continuous (and a discrete) mask over the edges (and

features) of the graph by formulating an optimisation process that maximizes mutual information between the distribution of possible subgraphs and GNN prediction. More recently, PGExplainer [16] and GraphMask [26] generalize GNNExplainer to an inductive setting; they use re-parametrisation tricks to alleviate the "introduced evidence" problem [5] — i.e. continuous masks deform the adjacency matrix and introduce new semantics to the generated graph. Regarding other approaches; GraphLIME [12] builds on LIME [22] to provide a non-linear explanation model; PGM-Explainer [32] learns a simple Bayesian network handling node dependencies; XGNN [34] produces model-level insights via graph generation trained using reinforcement learning.

Despite recent progress, existing explanation methods do not relate much and show clear limitations. Apart from GNNExplainer, none consider node features together with graph structure in explanations. Besides, they do not present core properties of a "good" explainer [19] (see Sec. 2). Since the field is very recent and largely unexplored, there is little certified knowledge about explainers' characteristics. It is, for instance, unclear whether optimising mutual information is pertinent or not. Overall, this often yields explanations with a poor signification, like a probability score stating how essential a variable is [16, 26, 33]. Existing techniques not only lack strong theoretical grounds, but also do not showcase an evaluation that is sophisticated enough to properly justify their effectiveness or other desirable aspects [24]. Lastly, little importance is granted to their human-centric characteristics [18], limiting the comprehensibility of explanations from a human perspective.

In light of these limitations, first, we propose a unified explanation framework encapsulating recently introduced explainers for GNNs. It not only serves as a connecting force between them but also provides a different and common view of their functioning, which should inspire future work. In this paper, we exploit it ourselves to define and endow our explainer, GraphSVX, with desirable properties. More precisely, GraphSVX carefully constructs and combines the key components of the unified pipeline so as to jointly capture the average marginal contribution of node features and graph nodes towards the explained prediction. We show that GraphSVX ultimately computes, via an efficient algorithm, the *Shapley values* from game theory [28], that we extend to graphs. The resulting unique explanation, thus, satisfy several theoretical properties by definition, while it is made more human-centric through several extensions. In the end, we evaluate GraphSVX on real-world and synthetic datasets for node and graph classification tasks. We show that it outperforms existing baselines in explanation accuracy, and verifies further desirable aspects such as robustness or certainty.

2 Related Work

Explanations methods specific to GNNs are classified into five categories of methods according to [35]: gradient-based, perturbation, decomposition, surrogate, and model-level. We utilise the same taxonomy in this paper to position GraphSVX.

Decomposition methods [2, 21] distribute the prediction score among input features using the weights of the network architecture, through backpropagation. Despite offering a nice interpretation, they are not specific to GNNs and present several major limits such as requiring access to model parameters or being sensitive to small input changes, like **gradient-based methods** discussed in Sec. 1.

Perturbation methods [16, 26, 33] monitor variations in model prediction with respect to different input perturbations. Such methods provide as explanation a continuous mask over edges (features) holding importance probabilities learned via a simple optimisation procedure, affected by the introduced-evidence problem.

Surrogate methods [12, 32] approximate the black box GNN model locally by learning an interpretable model on a dataset built around the instance of interest v (e.g., neighbours). Explanations for the surrogate model are used as explanations for the original model. For now, such approaches are rather intuition-based and consider exclusively node features or graph topology, not both.

Model level methods [34] provide general insights on the model functioning. It supports only graph classification, requires passing a candidate node set as input and is challenged by local methods also providing global explanations [16].

As we will show shortly, GraphSVX bridges the gap between these categories by learning a surrogate explanation model on a perturbed dataset that ultimately decomposes the explained prediction among the nodes and features of the graph, depending on their respective contribution. It also derives model-level insights by explaining subsets of nodes, while avoiding the respective limits of each category.

Desirable properties of explanations have received subsequent attention from the social sciences and the machine learning communities, but are often overlooked when designing an explainer. From a theoretical perspective, good explanations are accurate, fidel (*truthful*), and reflect the proportional importance of a feature on prediction (*meaningful*) [4, 35]. They also are stable and consistent (*robust*), meaning with a low variance when changing to a similar model or a similar instance [19]. Besides, they reflect the certainty of the model (*decomposable*) and are as representative as possible of its (*global*) functioning [17]. Finally, since their ultimate goal is to help humans understand the model, explanations should be intuitive to comprehend (*human-centric*). Many sociological and psychological studies emphasise key aspects: only a few motives (*selective*) [31], comparable to other instances (*contrastive*) [14], and interactive with the explainee (*social*).

3 Preliminary Concepts and Background

Notation. We consider a graph $\mathcal{G}$ with N nodes and F features defined by $(\mathbf{X}, \mathbf{A})$ where $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the feature matrix and $\mathbf{A} \in \mathbb{R}^{N \times N}$ the adjacency matrix. $f(\mathbf{X}, \mathbf{A})$ denotes the prediction of the GNN model f, and $f_v(\mathbf{X}, \mathbf{A})$ the

score of the predicted class for node v. Let $\mathbf{X}_{*j} = (X_{1j}, \ldots, X_{Nj})$ with feature values $\mathbf{x}_{*j} = (x_{1j}, \ldots, x_{Nj})$ represent feature j's value vector across all nodes. Similarly, $\mathbf{X}_i = \mathbf{X}_{i*} = (X_{i1}, \ldots, X_{iF})$ stands for node i's feature vector, with $\mathbf{X}_{iS} = \{X_{ik} | k \in S\}$. $\mathbf{1}$ is the all-ones vector.

3.1 Graph Neural Networks

GNNs adopt a message passing mechanism [11] where the update at each GNN layer ℓ involves three key calculations [3]: (i) The propagation step. The model computes a message $m_{ij}^{\ell} = \textsc{Msg}(\mathbf{h}_i^{\ell-1}, \mathbf{h}_j^{\ell-1}, a_{ij})$ between every pair of nodes (v_i, v_j), that is, a function MSG of v_i's and v_j's representations $\mathbf{h}_i^{\ell-1}$ and $\mathbf{h}_j^{\ell-1}$ in the previous layer and of the relation a_{ij} between the nodes. (ii) The aggregation step. For each node v_i, GNN calculates an aggregated message M_i from v_i's neighbourhood $\mathcal{N}_{v_i}$, whose definition vary across methods. $M_i^{\ell} = \textsc{Agg}(m_{ij}^{\ell} | v_j \in \mathcal{N}_{v_i})$. (iii) The update step. GNN non-linearly transforms both the aggregated message M_i^{ℓ} and v_i's representation $\mathbf{h}_i^{\ell-1}$ from the previous layer, to obtain v_i's representation $\mathbf{h}_i^{\ell}$ at layer ℓ: $\mathbf{h}_i^{\ell} = \textsc{Upd}(M_i^{\ell}, \mathbf{h}_i^{\ell-1})$. The representation $\mathbf{z}_i = \mathbf{h}_i^{L}$ of the final GNN layer L serves as final node embedding and is used for downstream machine learning tasks.

3.2 The Shapley value

The Shapley value is a method from Game Theory. It describes how to fairly distribute the total gains of a game to the players depending on their respective contribution, assuming they all collaborate. It is obtained by computing the average marginal contribution of each player when added to any possible coalition of players [28]. This method has been extended to explain machine learning model predictions on tabular data [13, 30], assuming that each feature of the explained instance ($\mathbf{x}$) is a player in a game where the prediction is the payout.

The characteristic function $val : S \to \mathbb{R}$ captures the marginal contribution of the coalition $S \subseteq \{1, \ldots, F\}$ of features towards the prediction $f(\mathbf{x})$ with respect to the average prediction: $val(S) = \mathbb{E}[f(\mathbf{X}) | \mathbf{X}_S = \mathbf{x}_s] - \mathbb{E}[f(\mathbf{X})]$. We isolate the effect of a feature j via $val(S \cup \{j\}) - val(S)$ and average it over all possible ordered coalitions S to obtain its Shapley value as:

$$\phi_j(val) = \sum_{S \subseteq \{1,\ldots,F\} \setminus \{j\}} \frac{|S|! \, (F - |S| - 1)!}{F!} \big(val(S \cup \{j\}) - val(S) \big).$$

The notion of fairness is defined by four axioms (*efficiency*, *dummy*, *symmetry*, *additivity*), and the Shapley value is the unique solution satisfying them. In practice, the sum becomes impossible to compute because the number of possible coalitions (2^{F-1}) increases exponentially by adding more features. We thus approximate Shapley values using sampling [6, 15].

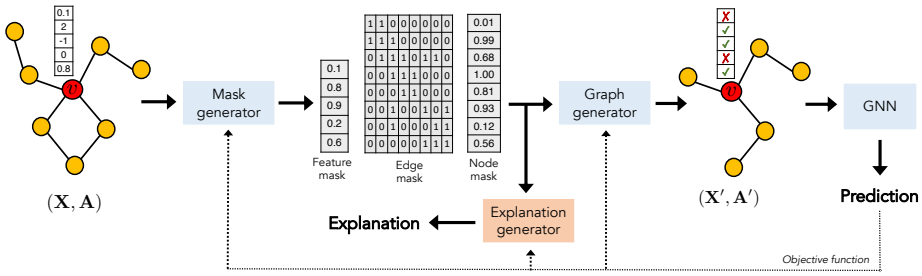


Fig. 1. Overview of unified framework. All methods take as input a given graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, feed it to a mask generator (Mask) to create three masks over nodes, edges, and features. These masks are then passed to a graph generator (Gen) that converts them to the original input space $(\mathbf{X}', \mathbf{A}')$ before feeding them to the original GNN model f. The resulting prediction $f(\mathbf{X}', \mathbf{A}')$ is used to improve the mask generator, the graph generator or the downstream explanation generator (Expl), which ultimately provides the desired explanation using masks and $f(\mathbf{X}', \mathbf{A}')$. This passage through the framework is repeated many times so as to create a proper dataset $\mathcal{D}$ from which each generator block learns. Usually, only one is optimised with a carefully defined optimisation process involving the new and original GNN predictions.

4 A Unified Framework for GNN Explainers

As detailed in the previous section, existing interpretation methods for GNNs are categorised and often treated separately. In this paper, we approach the explanation problem from a new angle, proposing a unified view that regroups existing explainers under a single framework: GNNExplainer, PGExplainer, GraphLIME, PGM-Explainer, XGNN, and the proposed GraphSVX. The key differences across models lie in the definition and optimisation of the three main blocks of the pipeline, as shown in Fig. 1:

- Mask generates discrete or continuous masks over features $\mathbf{M}_F \in \mathbb{R}^F$, nodes $\mathbf{M}_N \in \mathbb{R}^N$ and edges $\mathbf{M}_E \in \mathbb{R}^{N \times N}$, according to a specific strategy.
- Gen outputs a new graph $\mathcal{G}' = (\mathbf{X}', \mathbf{A}')$ from the masks $(\mathbf{M}_E, \mathbf{M}_N, \mathbf{M}_F)$ and the original graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$.
- Expl generates explanations, often offered as a vector or a graph, using a function g whose definition vary across baselines.

In the following, we show how each baseline fits the pipeline. $\odot$ stands for the element wise multiplication operation, σ the softmax function, $||$ the concatenation operation, and $\mathbf{M}^{\text{ext}}$ describes the extended vector $\mathbf{M}$ with repeated entries, whose size makes the operation feasible. All three masks are not considered for a single method; some are ignored as they have no effect on final explanations; one often studies node feature $\mathbf{M}_F$ or graph structure ($\mathbf{M}_E$ or $\mathbf{M}_N$).

GNNExplainer's key component is the mask generator. It generates both $\mathbf{M}_F$ and $\mathbf{M}_E$, where $\mathbf{M}_E$ has continuous values and $\mathbf{M}_F$ discrete ones. They are both

randomly initialised and jointly optimised via a mutual information loss function $MI(Y, (\mathbf{M}_E, \mathbf{M}_F)) = H(Y) - H(Y|\mathbf{A}', \mathbf{X}')$, where Gen gives $\mathbf{A}' = \mathbf{A} \odot \sigma(\mathbf{M}_E)$ and $\mathbf{X}' = \mathbf{X} \odot \mathbf{M}_F^{\text{ext}}$. Y represents the class label and $H(\cdot)$ the entropy term. Expl simply returns the learned masks as explanations, via the identity function $g(\mathbf{M}_E, \mathbf{M}_F) = (\mathbf{M}_E, \mathbf{M}_F)$.

PGExplainer is very similar to GNNExplainer. Mask generates only an edge mask $\mathbf{M}_E$ using a multi-layer neural network MLP_ψ and the learned matrix $\mathbf{Z}$ of node representations: $\mathbf{M}_E = \text{MLP}_\psi(\mathcal{G}, \mathbf{Z})$. The new graph is constructed with $\text{Gen}(\mathbf{X}, \mathbf{A}, \mathbf{M}_E) = (\mathbf{X}, \mathbf{A} \odot \rho(\mathbf{M}_E))$, where ρ denotes a reparametrisation trick. The obtained prediction $f_v(\mathbf{X}, \mathbf{A}')$ is also used to maximise mutual information with $f_v(\mathbf{X}, \mathbf{A})$ and backpropagates the result to optimise Mask. As for GNNExplainer, Expl provides $\mathbf{M}_E$ as explanations.

GraphLIME is a surrogate method with a simple and not optimised mask generator. Although it measures feature importance, it creates a node mask $\mathbf{M}_N$ using the neighbourhood of v (i.e., $\mathcal{N}_v$). The kth mask (or sample) is defined as $\mathbf{M}_{N,i}^k = 1 \quad \text{if } v_i = \mathcal{N}_v[k]$ and 0 otherwise. $\text{Gen}(\mathbf{X}, \mathbf{A}, \mathbf{M}_N) = (\mathbf{X}, \mathbf{A})$, so in fact, it computes and stores the original model prediction. $\mathbf{X}$ and $f(\mathbf{X}, \mathbf{A})$ are then combined with the mask $\mathbf{M}_N$ via simple dot products $\mathbf{M}_N^\top \cdot \mathbf{X}$ and $\mathbf{M}_N^\top \cdot f(\mathbf{X}, \mathbf{A})$ respectively, to isolate the original feature vector and prediction of the kth neighbour of v. These two elements are treated as input and target of an HSIC Lasso model g, trained with an adapted loss function. The learned coefficients constitute importance measures that are given as explanations by Expl.

PGM-Explainer builds a probabilistic graphical model on a local dataset that consists of random node masks $\mathbf{M}_N \in \{0, 1\}^N$. The associated prediction $f_v(\mathbf{X}', \mathbf{A}')$ is obtained by posing $\mathbf{A}' = \mathbf{A}$ and $\mathbf{X}' = \mathbf{M}_N^{\text{ext}} \odot \mathbf{X} + (\mathbf{1} - \mathbf{M}_N^{\text{ext}} \odot \boldsymbol{\mu}^{\text{ext}})$, with $\boldsymbol{\mu} = (E[\mathbf{X}_{*1}], \ldots, E[\mathbf{X}_{*F}])^\top$. This means that each excluded node feature ($\mathbf{M}_{N,j} = 0$) is set to its mean value across all nodes. This dataset is fed sequentially to the main component Expl, which learns and outputs a Bayesian Network g with input $\mathbf{M}_N$ (made sparser by looking at the Markov-blanket of v), *BIC score* loss function, and target $I(f_v(\mathbf{X}', \mathbf{A}'))$, where $I(\cdot)$ is a specific function that quantifies the difference in prediction between original and new prediction.

XGNN is a model-level approach that trains an iterative graph generator (add one edge at a time) via reinforcement learning. This causes two key differences with previous approaches: (1) the input graph at iteration t ($\mathcal{G}_t$) is obtained from the previous iteration and is initialised as the empty graph; (2) we also pass a candidate node set $\mathcal{C}$, such that $\mathbf{X}_\mathcal{C}$ contains the feature vector of all distinct nodes across all graphs in dataset. Mask generates an edge mask $\mathbf{M}_E = \mathbf{A}_t$ and a node mask $\mathbf{M}_{N_t} \in \{0, 1\}^{|\mathcal{C}|}$ specifying the latest node added to $\mathcal{G}_t$, if any. Gen produces a new graph $\mathcal{G}_{t+1}$ from $\mathcal{G}_t$ by predicting a new edge, possibly creating a new node from $\mathcal{C}$. This is achieved by applying a GCN and two MLP networks. Then, $\mathcal{G}_{t+1}$ is fed to the explained GNN. The resulting prediction is used to

update model parameters via a policy gradient loss function. EXPL stores nonzero $\mathbf{M}_{N_t}$ at each time step and provides $g(\{\mathbf{M}_{N_t}\}_t, \mathbf{X}_C, \mathbf{M}_E)) = (||_t \mathbf{M}_{N_t} \cdot \mathbf{X}_C, \mathbf{M}_E)$ as explanation — i.e. the graph generated at the final iteration, written $\mathcal{G}_T$.

GraphSVX. As we will see in the next section, the proposed GraphSVX model carefully exploits the potential of this framework through a better design and combination of complex mask, graph and explanation generators–in the perspective of improving performance and embedding desirable properties in explanations.

5 Proposed Method

GraphSVX is a post hoc model-agnostic explanation method specifically designed for GNNs, that jointly computes graph structure and node feature explanations for a single instance. More precisely, GraphSVX constructs a perturbed dataset made of binary masks for nodes and features $(\mathbf{M}_N, \mathbf{M}_F)$, and computes their marginal contribution $f(\mathbf{X}', \mathbf{A}')$ towards the prediction using a graph generator $\text{GEN}(\mathbf{X}, \mathbf{A}, \mathbf{M}_F, \mathbf{M}_N) = (\mathbf{X}', \mathbf{A}')$. It then learns a carefully defined explanation model on the dataset $(\mathbf{M}_N || \mathbf{M}_F, f(\mathbf{X}', \mathbf{A}'))$ and provides it as explanation. Ultimately, it produces a unique deterministic explanation that decomposes the original prediction and has a real signification (Shapley values) as well as other desirable properties evoked in Sec. 2. Without loss of generality, we consider a node classification task for the presentation of the method.

5.1 Mask and graph generators

First of all, we create an efficient mask generator algorithm that constructs discrete feature and node masks, respectively denoted by $\mathbf{M}_F \in \{0,1\}^F$ and $\mathbf{M}_N \in \{0,1\}^N$. Intuitively, for the explained instance v, we aim at studying the joint influence of a subset of features and neighbours of v towards the associated prediction $f_v(\mathbf{X}, \mathbf{A})$. The mask generator helps us determine the subset being studied. Associating 1 with a variable (node or feature) means that it is considered, 0 that it is discarded. For now, we let MASK randomly sample from all possible (2^{F+N-1}) pairs of masks $\mathbf{M}_F$ and $\mathbf{M}_N$, meaning all possible coalitions S of features and nodes (v is not considered in explanations). Let $\mathbf{z}$ be the random variable accounting for selected variables, $\mathbf{z} = (\mathbf{M}_F \| \mathbf{M}_N)$. This is a simplified version of the true mask generator, which we will come back to later, in Sec. 5.4.

We now would like to estimate the joint effect of this group of variables towards the original prediction. We thus isolate the effect of selected variables marginalised over excluded ones, and observe the change in prediction. We define $\text{GEN} : (\mathbf{X}, \mathbf{A}, \mathbf{M}_F, \mathbf{M}_N) \rightarrow (\mathbf{X}', \mathbf{A}')$, which converts the obtained masks to the original input space, in this perspective. Due to the message passing scheme of GNNs, studying jointly node and features' influence is tricky. Unlike GNNExplainer, we avoid any overlapping effect by considering feature values of v (instead of the whole subgraph around v) and all nodes except v. Several options are possible to cancel out a node's influence on the prediction, such as

replacing its feature vector by random or expected values. Here, we decide to isolate the node in the graph, which totally removes its effect on the prediction. Similarly, to neutralise the effect of a feature, as GNNs do not handle missing values, we set it to the dataset expected value. Formally, it translates into:

$$\mathbf{X}' = \mathbf{X} \quad \text{with} \quad \mathbf{X}'_v = \mathbf{M}_F \odot \mathbf{X}_v + (\mathbf{1} - \mathbf{M}_F) \odot \boldsymbol{\mu} \tag{1}$$

$$\mathbf{A}' = (\mathbf{M}_N^{\text{ext}\top} \cdot \mathbf{A} \cdot \mathbf{M}_N^{\text{ext}}) \odot I(\mathbf{A}), \tag{2}$$

where $\boldsymbol{\mu} = (\mathbb{E}[\mathbf{X}_{*1}], \ldots, \mathbb{E}[\mathbf{X}_{*F}])^\top$ and $I(\cdot)$ captures the indirect effect of k-hop neighbours of v ($k > 1$), which is often underestimated. Indeed, if a 3-hop neighbour w is considered alone in a coalition, it becomes disconnected from v in the new graph $\mathcal{G}'$. This prevents us from capturing its indirect impact on the prediction since it does not pass information to v anymore. To remedy this problem, we select one shortest path $\mathcal{P}$ connecting w to v via *Dijkstra's* algorithm, and include $\mathcal{P}$ back in the new graph. To keep the influence of the new nodes (in $\mathcal{P} \setminus \{w, v\}$) switched off, we set their feature vector to mean values obtained by Monte Carlo sampling.

To finalize the perturbation dataset, we pass $\mathbf{z}' = (\mathbf{X}', \mathbf{A}')$ to the GNN model f and store each sample $(\mathbf{z}, f(\mathbf{z}'))$ in a dataset $\mathcal{D}$. $\mathcal{D}$ associates with a subset of nodes and features of v their estimated influence on the original prediction.

5.2 Explanation generator

In this section, we build a surrogate model g on the dataset $\mathcal{D} = \{(\mathbf{z}, f(\mathbf{z}'))\}$ and provide it as explanation. More rigorously, an explanation ϕ of f is normally drawn from a set of possible explanations, called interpretable domain Ω. It is the solution of the following optimisation process: $\boldsymbol{\phi} = \arg\min_{g \in \Omega} \mathcal{L}_f(g)$, where the loss function attributes a score to each explanation. The choice of Ω has a large impact on the type and quality of the obtained explanation. In this paper, we choose broadly Ω to be the set of interpretable models, and more precisely the set of Weighted Linear Regression (WLR).

In short, we intend our model to learn to calculate the individual effect of each variable towards the original prediction from the joint effect $f(\mathbf{z}')$, using many different coalitions S of nodes and features. This is made possible by the definition of the input dataset $\mathcal{D}$ and is enforced by a cross entropy loss function, as follows:

$$\begin{aligned} \mathcal{L}_{f,\boldsymbol{\pi}}(g) &= \sum_{\mathbf{z}} \left[g(\mathbf{z}) - f(\mathbf{z}')\right]^2 \boldsymbol{\pi}_{\mathbf{z}}, \\ \text{where} \qquad \boldsymbol{\pi}_{\mathbf{z}} &= \frac{F + N - 1}{(F + N) \cdot |\mathbf{z}|} \cdot \binom{F + N - 1}{|\mathbf{z}|}^{-1}. \end{aligned} \tag{3}$$

$\boldsymbol{\pi}$ is a kernel weight that attributes a high weight to samples $\mathbf{z}$ with small or large dimension, or in different terms, groups of features and nodes with few or many elements—since it is easier to capture individual effects from the combined effect in these cases.

In the end, we provide the learned parameters of g as explanation. Each coefficient corresponds to a node of the graph or a feature of v and represents its estimated influence on the prediction $f_v(\mathbf{X}, \mathbf{A})$. In fact, it approximates the extension of the Shapley value to graphs, as shown in next paragraph.

5.3 Decomposition model

We first justify why it is relevant to extend the Shapley value to graphs. Looking back at the original theory, each player contributing to the total gain is allocated a proportion of that gain depending on its fair contribution. Since a GNN model prediction is fully determined by node feature information ($\mathbf{X}$) and graph structural information ($\mathbf{A}$), both edges/nodes and node features are players that should be considered in explanations. In practice, we extend to graphs the four Axioms defining fairness (please see the extended version [9]), and redefine how is captured the influence of players (features and nodes) towards the prediction as $val(S) = \mathbb{E}_{\mathbf{X}_v}[f_v(\mathbf{X}, \mathbf{A}_S)|\mathbf{X}_{vS} = \mathbf{x}_{vS}] - \mathbb{E}[f_v(\mathbf{X}, \mathbf{A})]$. $\mathbf{A}_S$ is the adjacency matrix where all nodes in $\overline{S}$ (not in S) have been isolated.

Assuming model linearity and feature independence, we show that GraphSVX, in fact, captures via $f(\mathbf{z}')$ the marginal contribution of each coalition S towards the prediction:

$$\begin{aligned}
\mathbb{E}_{\mathbf{X}_v}[f_v(\mathbf{X}, \mathbf{A}_S)|\mathbf{X}_{vS}] &= \mathbb{E}_{\mathbf{X}_{v\overline{S}}|\mathbf{X}_{vS}}[f_v(\mathbf{X}, \mathbf{A}_S)] \\
&\approx \mathbb{E}_{\mathbf{X}_{v\overline{S}}}[f_v(\mathbf{X}, \mathbf{A}_S)] && \text{by independence} \\
&\approx f_v(\mathbb{E}_{\mathbf{X}_{v\overline{S}}}[\mathbf{X}], \mathbf{A}_S) && \text{by linearity} \\
&= f_v(\mathbf{X}', \mathbf{A}'),
\end{aligned}$$

where $\mathbf{A}' = \mathbf{A}_S$ and $\mathbf{X}'_{ij} = \begin{cases} \mathbb{E}[\mathbf{X}_{*j}] \text{ if } i = v \text{ and } j \in \overline{S} \\ \mathbf{X}_{ij} \text{ otherwise.} \end{cases}$
Using the above, we prove that GraphSVX calculates the Shapley values on graph data. This builds on the fact that Shapley values can be expressed as an additive feature attribution model, as shown by [15] in the case of tabular data.

In this perspective, we set $\boldsymbol{\pi}_v$ such that $\boldsymbol{\pi}_v(\mathbf{z}) \to \infty$ when $|\mathbf{z}| \in \{0, F+N\}$ to enforce the *efficiency* axiom: $g(\mathbf{1}) = f_v(\mathbf{X}, \mathbf{A}) = \mathbb{E}[f_v(\mathbf{X}, \mathbf{A})] + \sum_{i=1}^{F+N} \phi_i$. This holds due to the specific definition of Gen and g (i.e., Expl), where $g(\mathbf{1}) = f_v(\mathbf{X}, \mathbf{A})$ and the constant ϕ_0, also called base value, equals $\mathbb{E}_{\mathbf{X}_v}[f_v(\mathbf{X}, \mathbf{A}_v)] \approx \mathbb{E}[f_v(\mathbf{X}, \mathbf{A})]$, so the mean model prediction. $\mathbf{A}_v$ refers to $\mathbf{A}_\emptyset$, where v is isolated.

Theorem 1. *With the above specifications and assumptions, the solution to* $\min_{g \in \Omega} \mathcal{L}_{f,\boldsymbol{\pi}}(g)$ *under Eq.* (3) *is a unique explanation model* g *whose parameters compute the extension of the Shapley values to graphs.*

Proof. Please see the extended version of this paper [9].

5.4 Efficient approximation specific to GNNs

Similarly to the euclidean case, the exact computation of the Shapley values becomes intractable due to the number of possible coalitions required. Especially that we consider jointly features and nodes, which augments exponentially the complexity of the problem. To remedy this, we derive an efficient approximation via a smart mask generator.

Firstly, we reduce the number of nodes and features initially considered to $D \leq N$ and $B \leq F$ respectively, without impacting performance. Indeed, for a GNN model with k layers, only k-hop neighbours of v can influence the prediction for v, and thus receive a non-zero Shapley value. All others are allocated a null importance according to the *dummy* axiom[1] and can therefore be discarded. Similarly, each feature j of v whose value is comprised in the confidence interval $I_j = [\mu_j - \lambda \cdot \sigma_j, \mu_j + \lambda \cdot \sigma_j]$ around the mean value μ_j can be discarded, where σ_j is the corresponding standard deviation and λ a constant.

The complexity is now $\mathcal{O}(2^{B+D})$ and we further drive it down to $\mathcal{O}(2^B + 2^D)$ by sampling separately masks of nodes and features, while still considering them jointly in g. In other words, instead of studying the influence of possible combinations of nodes and features, we consider all combinations of features with no nodes selected, and all combinations of nodes with all features included: $(2^B + 2^D)$. We observe empirically that it achieves identical explanations with fewer samples, while it seems to be more intuitive to capture the effect of nodes and features on prediction (expressed by Axiom 1).

Axiom 1 (Relative efficiency) *Node contribution to predictions can be separated from feature contribution, and their sum decomposes the prediction with respect to the average one, as* $\begin{cases} \sum_{j=1}^{B} \phi_j = f_v(\mathbf{X}, \mathbf{A}_v) - \mathbb{E}[f_v(\mathbf{X}, \mathbf{A})] \\ \sum_{i=1}^{D} \phi_{B+i} = f_v(\mathbf{X}, \mathbf{A}) - f_v(\mathbf{X}, \mathbf{A}_v). \end{cases}$

Lastly, we approximate explanations using $P \ll 2^B + 2^D$ samples, where P is sufficient to obtain a good approximation. We reduce P by greatly improving Mask, as evoked in Sec. 5.1. Assuming we have a budget of P samples, we develop a smart space allocation algorithm to draw in priority coalitions of order k, where k starts at 0 and is incremented when all coalitions of the order are sampled. This means that we sample in priority coalitions with high weight, so with nearly all or very few players. If they cannot all be chosen (for current k) due to space constraints, we proceed to a smart sampling that favours unseen players. The pseudocode and an efficiency evaluation lie in the extended version [9].

5.5 Desirable properties of explanations

In the end, GraphSVX generates fairly distributed explanations $\sum_j \phi_j = f_v(\mathbf{X}, \mathbf{A})$, where each ϕ_j approximates the average marginal contribution of a node or feature j towards the explained GNN prediction (with respect to the average prediction

[1] Axiom: If $\forall S \in \mathcal{P}(\{1, \ldots, p\})$ and $j \notin S$, $val(S \cup \{j\}) = val(S)$, then $\phi_j(val) = 0$.

ϕ_0). By definition, the resulting explanation is unique, consistent, and stable. It is also truthful and robust to noise, as shown in Sec. 6.2. The last focus of this paper is to make them more selective, global, contrastive and social; as we aim to design an explainer with desirable properties. A few aspects are detailed here.

Contrastive. Explanations are contrastive already as they yield the contribution of a variable with respect to the average prediction $\phi_0 = \mathbb{E}[f(\mathbf{X}, \mathbf{A})]$. To go futher and explain an instance with respect to another one, we could substitute $\mathbf{X}_v$ in Eq. (1) by $\mathbf{X}'_v = \mathbf{M}_F \odot \mathbf{X}_v + (\mathbf{1} - \mathbf{M}_F) \odot \boldsymbol{\xi}$, with $\boldsymbol{\xi}$ being the feature vector of a specific node w, or of a fictive representative instance from class C.

Global. We derive explanations for a subset U of nodes instead of a single node v, following the same pipeline. The neighbourhood changes to $\bigcup_i^U \mathcal{N}_i$, Eq. (1) now updates $\mathbf{X}_U$ instead of $\mathbf{X}_v$ and $f(\mathbf{z}')$ is calculated as the average prediction score for nodes in U. Also, towards a more global understanding, we can output the global importance of each feature j on v's prediction by enforcing in Eq. (1) $\mathbf{X}_{\mathcal{N}_v \cup \{v\},j}$ to a mean value obtained by Monte Carlo sampling on the dataset, when $\mathbf{z}_j = 0$. This holds when we discard node importance, otherwise the overlapping effects between nodes and features render the process obsolete.

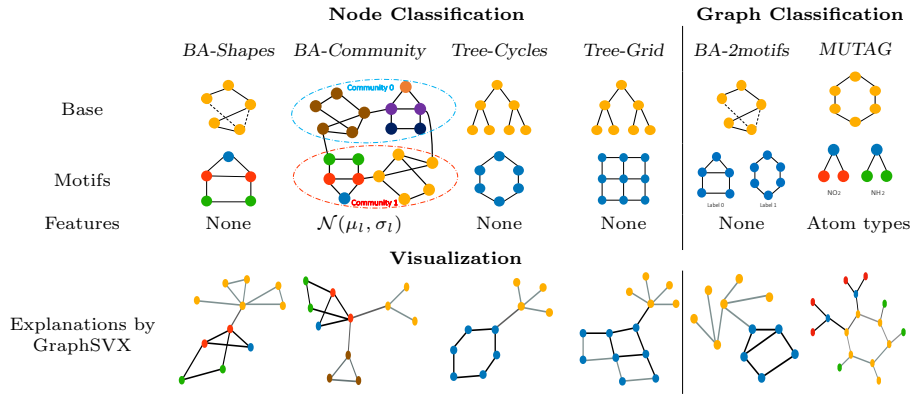
Graph classification. Until now, we had focused on node classification but the exact same principle applies for graph classification. We simply look at $f(\mathbf{X}, \mathbf{A}) \in \mathbb{R}$ instead of $f_v(\mathbf{X}, \mathbf{A})$, derive explanations for all nodes or all features (not both) by considering features across the whole dataset instead of features of v, like our global extension.

6 Experimental Evaluation

In this section, we conduct several experiments designed to determine the quality of our explanation method, using synthetic and real world datasets, on both node and graph classification tasks. We first study the effectiveness of GraphSVX in presence of ground truth explanations. We then show how our explainer generalises to more complex real world datasets with no ground truth, by testing GraphSVX's ability to filter noisy features and noisy nodes from explanations. Detailed dataset statistics, hyper-parameter tuning, properties' check and further experimental results including ablation study, are given in the extended version [9]. The source code is available at `https://github.com/AlexDuvalinho/GraphSVX`.

6.1 Synthetic and real datasets with ground truth

Synthetic node classification task. We follow the same setting as [16] and [33], where four kinds of datasets are constructed. Each input graph is a combination of a base graph together with a set of motifs, which both differ across datasets. The label of each node is determined based on its belonging and role in the motif.



Explanation Accuracy

GNNExplainer	0.83	0.75	0.86	0.84	0.68	0.65
PGM-Explainer	0.96	0.92	0.95	0.87	0.91	0.72
PGExplainer	0.92	0.81	0.96	0.88	0.85	**0.79**
GraphSVX	**0.99**	**0.93**	**0.97**	**0.93**	**0.99**	0.77

Table 1. Evaluation of GraphSVX and baseline GNN explainers on various datasets. The top part describes the construction of each dataset, with its base graph, the motif added, and the node features generated. Node labels are represented by colors. Then, we provide a visualisation of GraphSVX's explanations, where an important substructure is drawn in bold, as well as a quantitative evaluation based on the accuracy metric.

As a consequence, the explanation for a node in a motif should be the nodes in the same motif, which creates ground truth explanation. This ground truth can be used to measure the performance of an explainer via an accuracy metric.

Synthetic and real-world graph classification task. With a similar evaluation perspective, we measure the effectiveness of our explainer on graph classification, also using ground truth. We use a synthetic dataset *BA-2motifs* that resembles the previous ones, and a real life dataset called *MUTAG*. It consists of $4,337$ molecule graphs, each assigned to one of 2 classes based on its mutagenic effect [23]. As discussed in [7], carbon rings with groups NH_2 or NO_2 are known to be mutagenic, and could therefore be used as ground truth.

Baselines. We compare the performance of GraphSVX to the main explanation baselines that incorporate graph structure in explanations, namely GNNExplainer, PGExplainer and PGM-Explainer. GraphLIME and XGNN are not applicable here, since they do not provide graph structure explanations for such tasks.

Experimental setup and metrics. We train the same GNN model – 3 graph convolution blocks with 20 hidden units, (maxpooling) and a fully connected classification layer – on every dataset during $1,000$ epochs, with relu activation, Adam optimizer and initial learning rate 0.001. The performance is measured

with an accuracy metric (node or edge accuracy depending on the nature of explanations) on top-k explanations, where k is equal to the ground truth dimension. More precisely, we formalise the evaluation as a binary classification of nodes (or edges) where nodes (or edges) inside motifs are positive, and the rest negative.

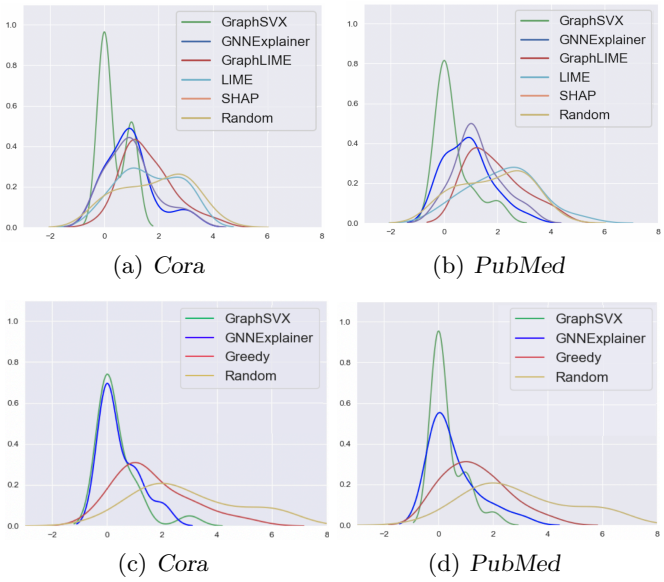
Results. The results on both synthetic and real-life datasets are summarized in Table 1. As shown both visually and quantitatively, GraphSVX correctly identifies essential graph structure, outperforming the leading baselines on all but one task, in addition to offering higher theoretical guarantees and human-friendly explanations. On *MUTAG*, the special nature of the dataset and ground truth favours edge explanation methods, which capture slightly more information than node explainers. Hence, we expect PGExplainer to perform better. For *BA-Community*, GraphSVX demonstrates its ability to identify relevant features and nodes together, as it also identifies important node features with 100% accuracy. In terms of efficiency, our explainer is slower than the scalable PGExplainer despite our efficient approximation, but is often comparable to GNNExplainer.

6.2 Real-world datasets without ground truth

Previous experiments involve mostly synthetic datasets, which are not totally representative of real-life scenarios. Hence, in this section, we evaluate GraphSVX on two real-world datasets without ground truth explanations: *Cora* and *PubMed*. Instead of looking if the explainer provides the correct explanation, we check that it does not provide a bad one. In particular, we introduce noisy features and nodes to the dataset, train a new GNN on the latter (which we verify do not leverage these noisy variables) and observe if our explainer includes them in explanations. In different terms, we investigate if the explainer filters useless features/nodes in complex datasets, selecting only relevant information in explanations.

Datasets. *Cora* is a citation graph where nodes represent articles and edges represent citations between pairs of papers. The task involved is document classification where the goal is to categorise each paper into one out of seven categories. Each feature indicates the absence/presence of the corresponding term in its abstract. *PubMed* is also a publication dataset with three classes and 500 features, each indicating the TF-IDF value of the corresponding word.

Noisy features. Concretely, we artificially add 20% of new "noisy" features to the dataset. We define these new features using existing ones' distribution. We re-train a 2-layer GCN and a 2-layer GAT model on this noisy data, whose test accuracy is above 75%. We then produce explanations for 50 test samples using different explainer baselines, on *Cora* and *PubMed*, and we compare their performance by assessing how many noisy features are included in explanations among top-k features. Ultimately, we compare the resulting frequency distributions using a kernel density estimator (KDE). Intuitively, since features are noisy, they are not used by the GNN model, and thus are unimportant. Therefore, the less noisy features are included in the explanation, the better the explainer.



(a) *Cora* (b) *PubMed*

(c) *Cora* (d) *PubMed*

Fig. 2. Frequency distributions of noisy features (a), (b) and nodes (c), (d) using a GAT model on *Cora* and *PubMed*.

Baselines include GNNExplainer, GraphLIME (described previously) as well as the well-known SHAP [15] and LIME [22] models. We also compare GraphSVX to a method based on a Greedy procedure, which greedily removes the most contributory features/nodes of the prediction until the prediction changes, and to the Random procedure, which randomly selects k features/nodes as the explanations for the prediction being explained.

The results are depicted in Fig. 2 (a)-(b). For all GNNs and on all datasets, the number of noisy features selected by GraphSVX is close to zero, and in general lower than existing baselines—demonstrating its robustness to noise.

Noisy nodes. We follow a similar idea for noisy neighbours instead of noisy features. Each new node's connectivity and feature vector are determined using the dataset's distribution. Only a few baselines (GNNExplainer, Greedy, Random) among the ones selected previously can be included for this task since GraphLIME, SHAP, and LIME do not provide explanations for nodes.

As before, this evaluation builds on the assumption that a well-performing model will not consider as essential these noisy variables. We check the validity of this assumption for the GAT model by looking at its attention weights. We retrieve the average attention weight of each node across the different GAT layers and compare the one attributed to noisy nodes versus normal nodes. We expect it to be lower for noisy nodes, which proves to be true: 0.11 vs. 0.15.

As shown in Fig. 2 (c)-(d), GraphSVX also outperforms all baselines, showing nearly no noisy nodes in explanations. Nevertheless, GNNExplainer achieves

almost as good performance on both datasets (and in several evaluation settings).

7 Conclusion

In this paper, we have first introduced a unified framework for explaining GNNs, showing how various explainers could be expressed as instances of it. We then use this complete view to define GraphSVX, which conscientiously exploits the above pipeline to output explanations for graph topology and node features endowed with desirable theoretical and human-centric properties, eligible of a good explainer. We achieve this by defining a decomposition method that builds an explanation model on a perturbed dataset, ultimately computing the Shapley values from game theory, that we extended to graphs. After extensive evaluation, we not only achieve state-of-the-art performance on various graph and node classification tasks but also demonstrate the desirable properties of GraphSVX.

Acknowledgements. Supported in part by ANR (French National Research Agency) under the JCJC project GraphIA (ANR-20-CE23-0009-01).

References

1. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM (2011)
2. Baldassarre, F., Azizpour, H.: Explainability techniques for graph convolutional networks. arXiv (2019)
3. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., et al.: Relational inductive biases, deep learning, and graph networks. arXiv (2018)
4. Burkart, N., Huber, M.F.: A survey on the explainability of supervised machine learning. JAIR **70**, 245–317 (2021)
5. Dabkowski, P., Gal, Y.: Real time image saliency for black box classifiers. In: NeurIPS (2017)
6. Datta, A., Sen, S., Zick, Y.: Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In: IEEE symposium on security and privacy (SP) (2016)
7. Debnath, K., et al.: 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)
8. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NeurIPS (2016)
9. Duval, A., Malliaros, F.D.: GraphSVX: Shapley Value Explanations for Graph Neural Networks. arXiv preprint arXiv:2104.10482 (2021)
10. Duvenaud, D., et al.: Convolutional networks on graphs for learning molecular fingerprints. In: NeurIPS (2015)
11. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS (2017)
12. Huang, Q., Yamada, M., Tian, Y., Singh, D., Yin, D., Chang, Y.: GraphLIME: Local interpretable model explanations for graph neural networks. arXiv (2020)

13. Lipovetsky, S., Conklin, M.: Analysis of regression in game theory approach. ASMBI **17**(4), 319–330 (2001)
14. Lipton, P.: Contrastive explanation. Royal Institute of Philosophy Supplement **27**, 247–266 (1990)
15. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: NeurIPS (2017)
16. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network. In: NeurIPS (2020)
17. Miller, T.: Explanation in Artificial Intelligence: Insights from the social sciences. Artificial intelligence **267**, 1–38 (2019)
18. Miller, T., Howe, P., Sonenberg, L.: Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences. arXiv (2017)
19. Molnar, C.: Interpretable Machine Learning. Lulu. com (2020)
20. O'neil, C.: Weapons of math destruction: How big data increases inequality and threatens democracy. Broadway Books (2016)
21. Pope, P.E., Kolouri, S., Rostami, M., Martin, C.E., Hoffmann, H.: Explainability methods for graph convolutional neural networks. In: CVPR (2019)
22. Ribeiro, M.T., Singh, S., Guestrin, C.: 'Why should I trust you?' Explaining the predictions of any classifier. In: KDD (2016)
23. Riesen, K., Bunke, H.: Iam graph database repository for graph based pattern recognition and machine learning. In: Joint IAPR - SPR & SSPR. Springer (2008)
24. Robnik-Šikonja, M., Bohanec, M.: Perturbation-based explanations of prediction models. In: Human and machine learning, pp. 159–175. Springer (2018)
25. Saltelli, A.: Sensitivity analysis for importance assessment. Risk analysis **22**(3), 579–590 (2002)
26. Schlichtkrull, M.S., Cao, N.D., Titov, I.: Interpreting graph neural networks for NLP with differentiable edge masking. In: ICLR (2021)
27. Selvaraju, R., et al.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: ICCV (2017)
28. Shapley, L.S.: A value for n-person games. Contributions to the Theory of Games **2**(28), 307–317 (1953)
29. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: ICML (2017)
30. Strumbelj, E., Kononenko, I.: An efficient explanation of individual classifications using game theory. JMLR **11**, 1–18 (2010)
31. Ustun, B., Rudin, C.: Methods and models for interpretable linear classification. arXiv (2014)
32. Vu, M.N., Thai, M.T.: PGM-Explainer: Probabilistic graphical model explanations for graph neural networks. In: NeurIPS (2020)
33. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNNExplainer: Generating explanations for graph neural networks. In: NeurIPS (2019)
34. Yuan, H., Tang, J., Hu, X., Ji, S.: XGNN: Towards model-level explanations of graph neural networks. In: KDD (2020)
35. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: A taxonomic survey. arXiv preprint arXiv:2012.15445 (2020)
36. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. (2018)
37. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. arXiv (2018)