Computing (2026) 108:55
https://doi.org/10.1007/s00607-026-01642-z

REGULAR PAPER

GECo: a community-based graph neural network explainer

Domenico Amato[1] · Salvatore Calderaro[1,2] · Giosuè Lo Bosco[1] · Riccardo Rizzo[3] · Filippo Vella[3]

Received: 12 September 2025 / Accepted: 2 March 2026

Abstract
Graph Neural Networks (GNNs) are powerful models that can manage complex data sources and their interconnection links. Still, one of GNNs' main drawbacks is their lack of explainability, which limits their application in sensitive fields. In this paper, we introduce a new approach involving graph communities to address the explainability of graph classification problems, called Graph Neural Network Explainer based on Communities (GECo). Taking into consideration that a community is a subset of graph nodes that are densely connected, GECo exploits the idea that these subgraphs should play a key role in graph classification. This assumption is reasonable, especially if we consider the message-passing mechanism, which is the basic mechanism of GNNs. GECo analyzes the contribution to the classification result of the communities in the graph, building a mask that highlights graph-relevant structures. GECo is tested for Graph Convolutional Networks on six artificial and four real-world graph datasets and is compared to the main explainability methods, such as PGMExplainer, PGExplainer, GNNExplainer, TAGE, and SubgraphX, using six different metrics. The obtained results outperform the other methods for artificial graph datasets and most real-world datasets. Furthermore, GECo is faster than most of the competitors since it does not need to make use of any sampling process.

Keywords Graph neural networks · Explainability · XAI

1 Introduction

Deep Neural Networks (DNNs) have demonstrated the ability to learn from a wide variety of data, including text, images, and temporal series, providing significant contributions in broad application domains. Because of the impressive performance that DNNs demonstrate in general supervised learning tasks, they have also been effec-

Extended author information available on the last page of the article

Published online: 21 March 2026

tively utilized in various biomedical classification tasks [1–4]. However, in some cases, the information is organised in more complex ways, and individual pieces of data share relationships with each other. As a consequence, graphs become the preferred data structure for their capability to represent both information elements and their interconnections. Real examples are social network analysis, bioinformatics, chemistry, and finance, where the relationships among data are just as important as the data itself. The synergy between graph data structure and deep neural networks is expressed in Graph Neural Networks (GNNs). These networks combine the flexibility of neural networks with the ability to manage graph-structured data, making it possible to process graph-like data using machine learning methods. In recent years, GNNs have found widespread applications across various real-world biomedical and bioinformatics domains. In medical imaging, they have been used to improve the accuracy of breast cancer tumor identification by capturing spatial and relational patterns among tissue regions [5]. In neuro-oncology, they have been used for the semantic segmentation of glioblastoma in brain MRI scans, effectively leveraging spatial dependencies between anatomical structures [6]. In biology, they have facilitated the taxonomic classification of bacterial species [7] and the identification of protein transcription factor sequences [8], exploiting the sequential and structural dependencies inherent in biological molecules.

Furthermore, for many machine learning (ML) applications, such as medicine, finance, or support decision systems on sensitive data, it is fundamental to provide a clear and understandable insight into the reasons behind a prediction or a decision. However, explainability is an open challenge for neural networks in general, and the same is true for GNNs.

Many methods of explainability for GNNs have been proposed. Some methods build a local explanation of the model, and others explain the output obtained from a specific input. For example, the same gradient mechanism of the training can be exploited to approximate the importance of some features of the input, or the input perturbation can be used to study the output modifications.

Some explainability algorithms for GNNs extract from the input graph a subgraph containing the most influential features of the output results. Many of these algorithms obtain these subgraphs as the result of an optimization procedure designed to maximize the mutual information between the whole input graph and the explanation subgraph. These methods build the subgraph from scratch, ignoring the structure of the input graph.

This paper proposes an explanation method that exploits the structure of the input graph. One of the main characteristics of graphs is the presence of communities, i.e., parts of the graph densely interconnected that share few edges with the rest of the graph nodes.

The presence of communities reveals parts of the graph that have a strong relationship with each other and sometimes can be considered almost independent from the rest of the graph [9]. These considerations, together with the construction mechanism of the node embedding in the GNNs, constitute the foundation of the proposed method called GECo (**G**raph Neural Network **E**xplainer based on **Co**mmunities).

GECo detects the different communities of the graph, and, for each community, selects the corresponding subgraph. Then, the model is tested on these subgraphs to

see how they alone support the predicted class. Finally, a criterion is established to discriminate the negligible portion from the relevant subset.

The collection of these key communities forms the final explanation, and, from them, the most relevant parts of the graph leading to the classification can be highlighted. We tested the proposed approach's effectiveness on synthetic and real-world datasets, comparing the results with state-of-the-art methodologies and a random baseline.

The remainder of the paper is organized in the following way: in Sect. 2, a review of works in the same field is described; in Sect. 3, the GNNs are described, and the proposed solution is presented in detail, and in the next Sections the used datasets, along with the evaluation criteria, are described. The results are presented in Sect. 4, and the conclusions are drawn in Sect. 5.

2 Related Work

Explaining the prediction made by the GNNs is challenging because graph data are less intuitive compared to images and texts. The traditional explainability methods used for text and images are unsuitable for obtaining convincing explanations for neural graph computation, but some other principles, such as the one based on perturbation, still work.

The explainability methods for GNN can be divided into two groups based on the type of information they provide [10]:

- *Instance-level explanations* that explain deep models by identifying the most important input features for their prediction.
- *Model-level explanations* that study the input graph patterns that lead to a specific prediction.

The *instance-level explanations* can be extracted by these method categories:

- *Perturbation Based Methods*: the output variation corresponding to an input perturbation reflects the input region's importance. These methods identify the nodes/edges/features that must be kept consistent with the original GNN model.
- *Gradient/Features Based Methods:* use the gradient values or the hidden features to approximate the input importance, explaining the predictions, often using backpropagation.
- *Surrogate Methods*: supplementary, more explainable models are trained using the neighboring areas of an input node.
- *Decomposition Methods*: decompose the prediction into several terms, each regarded as the importance score of the corresponding input features.

We describe here the explainability methods for GNNs, which have gained much attention in the recent literature; we will go into deeper detail in instance-level, perturbation-based methods because GECo is part of this category.

The *Perturbation Based Methods* study the output variations concerning the input perturbation. Intuitively, the model's prediction should be the same when vital information is maintained in the perturbed graph. This perturbed graph is a subgraph of the input one, and it is the result of a blind search procedure guided by some function to optimize. GNNExplainer [11] uses mask optimization to learn soft masks for node and edge features to explain the predictions. The process starts with randomly initializing the soft masks, treating them as trainable parameters. Then, the methods combine this mask with the original graph using element-wise multiplication. Next, the masks are optimized, maximizing the mutual information between the original graph and the perturbed graphs' predictions.

PGExplainer [12] learns edge masks to explain the predictions. For this purpose, it trains a mask predictor to predict edge masks. Given the input graph for each edge, the method calculates the edge embedding by concatenating the relative node embeddings. Then, the predictor uses these edge embeddings to predict the probability that each edge is selected and uses these values as an importance score. After this step, the method used the reparametrization trick to sample the masks. Finally, the mask predictor is trained by maximizing the mutual information between the original and new predictions. Xie et al. propose TAGE in [13], a technique very similar to PGExplainer, which can produce explanations for GNNs not tailored to a specific task. It is also independent of the specific models and is trained with self-supervision approaches. TAGE is composed of an embedding explainer and a downstream explainer. The first is trained with conditioned contrastive learning; the latter is based on gradient-based explainers.

Even TAGE, in principle, minimizes the mutual information between the masked embedding of the original graph and the embedding of the explanation subgraph. TAGE is interesting because the authors build the explanation considering connected subgraphs, not simple nodes or links, and they center their attention on the embeddings (and downstream tasks).

GraphMask [14] is a method for explaining the edge importance in each GNN layer. Similar to the previous one, it trains a classifier to predict if an edge can be removed without affecting the original prediction. Learnable connection vectors of the same dimension as the node embeddings replace the dropped edges. The classifier is trained using the whole dataset by minimizing a divergence term, which measures the difference between network predictions. SubgraphX [15] explores subgraph-level explanations using the Monte Carlo Tree Search (MCTS) algorithm to explore different subgraphs via node pruning and select the essential subgraph from the search tree's leaves to explain the prediction. In addition, it uses the Shapley values as a reward of MCTS to measure the importance of subgraphs.

In their paper on SubgraphX, the authors consider the function calculated by the trained GNN useful to estimate an importance score of the subgraph obtained.

The *Gradient/Features-Based Methods* subtype measures the importance of input samples using gradient or hidden feature values. Sensitivity Analysis (SA) [16] produces a local explanation for the prediction using the square norm of the output gradient with respect to the input. The output is a saliency map, which describes the extent to which variations in the input would produce a change in the output.

Another method that produces a saliency map is the Guided Backpropagation (GBP) [16] and CAM method [17]; the first clips negative gradients during the backpropagation phase, concentrating the explanation on the features that have an excitation effect on the output. The method CAM, usable only for graph classification tasks, takes the node embeddings of the last layer and combines different feature maps using weighted summation to obtain an importance score for the input nodes. Grad-CAM [17] extends the CAM method, removing the need for a GAP layer. The idea behind the *Surrogate Methods* is to employ a simple surrogate model to approximate the prediction of a more complex deep model for the neighboring areas of the input example. The explanations obtained from the explainable surrogate model explain the original prediction. GraphLIME [18] is an extension of the LIME algorithm to GNN models that studies the relevance of node features in the context of node classification tasks. This method was developed to explain node classification predictions, but it cannot be directly applied to graph classification models. Another method introduced to study the explainability of node classification models is RelEx [19], which combines the surrogate and perturbation-based methods. PGMExplainer [20] randomly perturbs the node features of several random nodes within the computational graph. This perturbation and its influence on GNN prediction are memorized and processed to explain the same prediction. Conversely, from GraphLime and RelEx, the PGMExplainer method allows an explanation of both node classification and graph classification tasks.

The *Decomposition Methods* study the model parameters to find the relationship between the feature in the input space and the output predictions. It is essential to point out that the conservative property of these methods requires that the sum of the decomposed terms is equal to the original prediction score.

Excitation Back Propagation (BP) [17] extends the method Layer-wise Relevant Propagation (LRP) [16, 21], a method that calculates the contribution of each neuron to the target neuron output. In this case, the probability of a neuron in the current layer is the total probability it outputs to all connected neurons in the next layer. GNN-LRP [22] studies the importance of different graph walks. This method treats the trained GNN as a function and provides a view of high-order Taylor decomposition to develop the score decomposition rule. GNN-LRP follows a backpropagation procedure to approximate the T-order terms. It distributes scores to different graph walks and stores the paths of the distribution processes from layer to layer. Those paths are considered different walks, and the scores are obtained from their corresponding nodes.

The so-called *model-level explanations* are the second main category for explaining a GNN. These methods study the graph patterns that can lead to a particular GNN behavior, e.g., maximizing a target prediction. Among them, one can find the *generation methods* that learn to generate such graph patterns. XGNN [23] belongs to this category. It trains a graph generator for creating graphs that maximize a target graph prediction. Those graphs will represent the explanations for the target prediction.

Curiously, none of the methods described so far takes into account the community structure of a graph, even if it is undeniable that it can be a relevant graph feature and can play a fundamental role in the graph classification task. Community detection algorithms differ in what they treat as a community and in trade-offs between speed,

accuracy, and explainability. Together with the classical approaches for community detection [24, 25], the flourishing of neural approaches has contributed with interesting contributions for the representation, clustering, and detection of communities in graphs [26–28].

3 Methods

In this Section, we begin by introducing the fundamental concepts of GNNs, with a focus on the message-passing framework. Next, we present the proposed GECo algorithm, followed by a description of the synthetic and real-world datasets used to assess its effectiveness. Finally, we outline the evaluation criteria employed in our analysis.

3.1 Graph neural networks

The following introduces the basic notions of GNN to clarify the basic ideas of the proposed explanation method. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Every graph can be associated with a square matrix called an adjacency matrix, defined as $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. For weighted graphs $A_{ij} = w_{ij}$, for unweighted graphs, the ones we consider in this paper, $A_{ij} = 1$ if $(i, j) \in \mathcal{E}$, $A_{ij} = 0$ if $(i, j) \notin \mathcal{E}$

Each graph node is associated with a vector of features $x \in \mathbb{R}^n$ where *n* is the input dimension. The set of all node features can be represented using a matrix $X \in \mathbb{R}^{|\mathcal{V}|\times n}$.

A GNN network, constituted by *K* layers, aims to learn for each layer $k = 1, .., K$ a new set of node features $H^k \in \mathbb{R}^{|\mathcal{V}|\times F^k}$ exploiting the graph topology information and the node attributes, where F^k is the number of features per node predefined for each layer. The idea behind the computation is to update the node representations iteratively, combining them with node representations of their neighbors [29], and this can be done by the "message-passing" paradigm [30].

Under this paradigm, at a generic GNN layer *k*, each node *u* aggregates the messages $m^k_{N(u)}$ received from its neighborhood *N*(*u*). These messages consist of weighted representations of the features of adjacent nodes, which are combined through an aggregation function (AGG):

$$m^k_{N(u)} = AGG^k\{H^k_v, \forall v \in N(u)\} \tag{1}$$

The new values H^{k+1}_u that take into account the "messages" which *u* received are calculated by the update function (UPD) in the following way:

$$H^{k+1}_u = UPD^k\left(H^k_u, m^k_{N(u)}\right) \tag{2}$$

The output of the last layer of the GNN is the matrix $H^K \in \mathbb{R}^{|\mathcal{V}| \times F^K}$, and can be used as the final representation of the nodes for downstream tasks such as classification or regression.

Notice that *AGG* and *UPD* functions have different definitions depending on the kind of GNN, but the reference to $N(u)$ is always present. For example, in the case of Graph Convolutional Neural Networks (GCN), which is the one used to test the GECo algorithm, the *AGG* and *UPD* functions determine the new set of features for the whole graph in the following way:

$$H^{k+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^k W^k\right) \tag{3}$$

where $\tilde{A}$ is an adjacency matrix of an unweighted graph with added self-connections, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, W^k is a trainable weights matrix, and $\sigma(\cdot)$ is a nonlinear activation.

Then, in the graph classification task, an embedding of the whole graph is obtained using summation, average, maximum, or minimum over all nodes or edge features. The average *readout* operation is defined as:

$$h_{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{v \in \mathcal{V}} H_v^K \tag{4}$$

Finally, the classification can be obtained using a multi-layer perceptron (MLP) or a simple linear layer.

Figure 1 shows a representation of a GCN network with the set of convolutional layers, implementing Eq. 3, a readout implementing Eq. 4, and the downstream subsystem indicated as MLP.

3.2 The GECo methodology

Regarding the taxonomy described in Sect. 2, the method presented here belongs to the instance-level methods, particularly the perturbation-based methods, because it

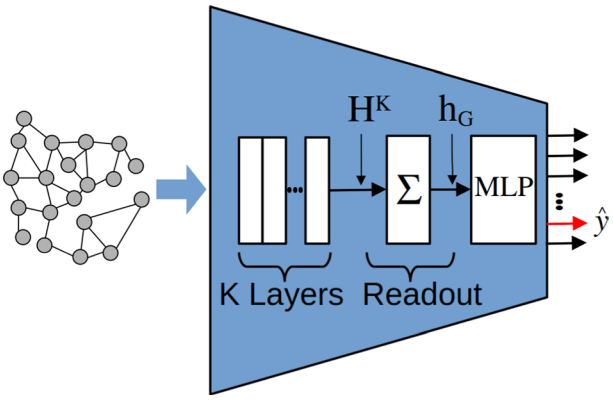


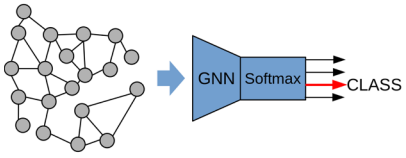
Fig. 1 A representation of a GCN

identifies a perturbation of the input consistent with the prediction of the GNN on the original graph.

GECo aims to find subgraphs responsible for the GNN's output value. The algorithm is based on the hypothesis that a GNN will learn to recognize some structures in the input graph; subgraphs with these structures will produce a high output response. Communities are structures easily recognized in a graph, and considering the AGG step in the GNN algorithm already discussed, relevant communities should generate a very high proportion of the output value in a trained neural network. The community subgraphs are identified and proposed as stand-alone subgraphs to the GNN, and the different output values are memorized. The subgraph corresponding to the highest output values is considered the most important for the classification output.

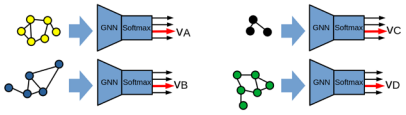
Going into detail, given a trained GNN identified as f, GECo aims to find a mask containing the most relevant nodes for the final classification. The inputs of GECo are f and a graph $\mathcal{G}$. In the first step, $\mathcal{G}$ is given as input to the GNN, obtaining the prediction $\hat{y}$, where $\hat{y}$ is a component of the output vector $MLP(\mathcal{G})$ (see Fig. 2a).

In the second step (Fig. 2b), it is necessary to find the communities of the graph; we decided to use a greedy community detection algorithm based on modularity, a measure that quantifies the density of connections within a community [31]. The greedy solution adopted by GECo is the one proposed by Clauset et al. [32]. For details about this methodology, see Section 1 of the Supplementary Manuscript. Nevertheless, we have performed experiments showing that the community detection algorithm has little influence on GECo, as shown in Figure 1 of the Supplementary Manuscript.

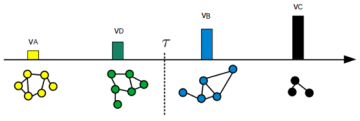


(a) Step 1: Classification of the whole graph $\mathcal{G}$ in the class CLASS.

(b) Step 2: Community detection of the input graph $\mathcal{G}$.



(c) Step 3: Classification of every single community of the input graph $\mathcal{G}$.



(d) Step 4: Decision on τ threshold value.

(e) Step 5: Identification of the most influential community of the input graph $\mathcal{G}$.

Fig. 2 The steps of the proposed GECo algorithm

For each community, we build a subgraph that contains only the nodes that belong to the considered community. These subgraphs are fed to the GNN, and the probability value corresponding to the predicted class $\hat{y}$ is stored (see Fig. 2c). After this step, we have the associated value of the probability for each community. We use these values to calculate a threshold τ using, for example, the mean or the median of the probability values. Fixed the value of τ, we consider the communities associated with a probability value greater than τ, and we use the nodes that belong to these communities to form the final explanation (see Fig. 2e). For completeness, the pseudo-code describing the GECo algorithm, which summarizes the main steps of the proposed approach and clarifies the overall workflow, is provided in Algorithm 1.

Require: Trained GNN model f, Test graph $\mathcal{G}$
Ensure: Explanation mask E
Step 1: Classify the whole graph $\mathcal{G}$ using f and store the predicted class $\hat{y}$
$\hat{y} \leftarrow argmax(f(\mathcal{G}))$
Step 2: Identify the communities $\{C\}$ of the graph $\mathcal{G}$
$\{C\} \leftarrow$ CommunityDetection($\mathcal{G}$)
for each community C_i in $\{C\}$ **do**
    Step 3: Build the corresponding subgraph $\mathcal{G}_i$ and feed it to f
    $\mathcal{G}_i \leftarrow$ Subgraph($\mathcal{G}, C_i$)
    $p_i \leftarrow f(\mathcal{G}_i)[\hat{y}]$
    Save the probability value p_i
end for
Step 4: Compute the threshold value τ by averaging the probability values $\{p_i\}$
$\tau \leftarrow \frac{1}{|\{C\}|} \sum_i p_i$
Step 5: Select communities with probability values greater than τ and use their nodes to compose the final explanation mask E
$E \leftarrow \bigcup_{i:p_i>\tau} C_i$
return E

Algorithm 1 GECo algorithm

3.3 Synthetic datasets

To evaluate an algorithm that tries to explain the result of GNN, it is necessary to have the ground truth explanation masks that contain the relevant features (node/edges/node features) associated with the graphs. The availability of this mask allows us to compare the result of the explanation algorithm with the ground truth. Furthermore, with the ground truth explanation, we can verify whether the algorithm finds the right features or if the GNN uses different features to make its final decision. For this reason, we created six synthetic datasets containing ground-truth explanations. To generate the synthetic datasets, we considered the following graphs: Erdös–Rényi (ER) and Barabási-Albert (BA).

ER graphs [33, 34] are random graphs introduced by Erdös–Rényi in 1959. There are two variants of the ER random graph model. In the $G(n, M)$ model, the graph is chosen randomly from the graphs with n nodes and M edges. In the $G(n, p)$ model, the graphs are built by randomly connecting labeled nodes; the algorithm to generate a graph using this model is the following:

1. Start with n isolated nodes.

2. Select a pair of nodes and randomly extract a number in [0, 1]. If this number exceeds p, add an edge between the nodes selected; otherwise, leave them disconnected.
3. Repeat step 2) for each of the $\frac{n(n-1)}{2}$ pairs of nodes.

In Fig. 3a, we report an example of an ER graph with 30 nodes built with $p = 0.1$.

BA model [35] is an algorithm to generate a random scale-free network using the technique of "preferential attachment". A random scale-free network is characterized by a degree distribution that follows a power law. In such a network, most nodes have only a few connections (low degree), while a few nodes (called hubs) have many connections (high degree). This contrasts with random networks, where the degree distribution is more uniform, and most nodes have a similar number of connections. The probability p_i that a new node is connected to node i is:

$$p_i = \frac{k_i}{\sum_j k_j} \tag{5}$$

where k_i is the degree of the node i, and the sum is made over all the pre-existing nodes j. Heavily linked nodes tend to accumulate more edges, while nodes with few edges are unlikely to be chosen as the destination for a new link. The new nodes have a "preference" to attach themselves to the already heavily linked nodes. In Fig. 3b, we report an example of a BA graph with 30 nodes

During the creation of the synthetic datasets, we added to the graphs a motif considered as a small recurring pattern that identifies the class of the graph. In particular, we added the following motifs:

- house-motif is a graph with 5 nodes and 6 edges.
- cycle-motif with 5 or 6 nodes.
- wheel-motif is a single hub node connected to each node of the $(n-1)$-node cycle graph. We consider wheel-motifs with 8 nodes.
- grid-motif with a dimension of 3×3.

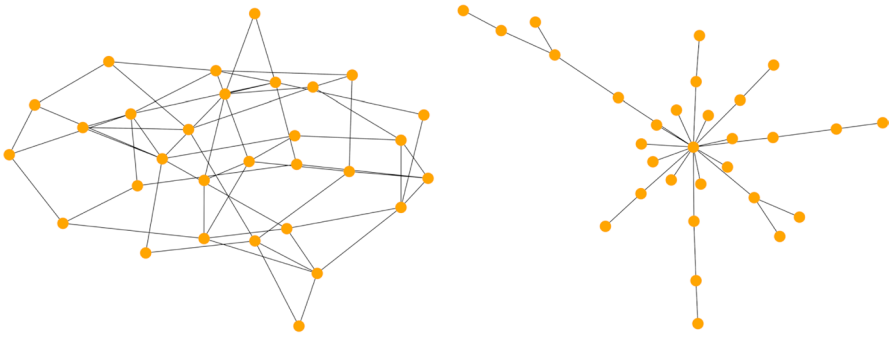


Fig. 3 Two examples of graphs used in synthetic datasets: (a) Erdös–Rényi (ER) graph, (b) Barabasi-Albert (BA)

Fig. 4 reports all the motifs used to build our synthetic datasets.

Using the ER and BA graphs and the mentioned motifs, we built the following synthetic datasets:

- *ba_house_cycle*: containing 1000 BA graphs with 25 nodes; we attach to 500 graphs a house-motif and to the remaining part a cycle-motif with six nodes.
- *ba_cycle_wheel*: containing 1000 BA graphs with 25 nodes; we attach to 500 graphs a cycle-motif with 5 nodes and a wheel-motif with 8 nodes to the remaining part.
- *ba_cycle_wheel_grid*: containing 1500 BA graphs with 25 nodes; We attach to 500 graphs a cycling motif with 5 nodes, a wheel motif with 8 nodes to the other 500 graphs, and a grid motif of dimension 3×3 to the remaining part.
- *er_house_cycle*: containing 1000 ER graphs with 25 nodes; we attach a house cycle to 500 graphs and a cycle motif with 6 nodes to the remaining part.
- *er_cycle_wheel*: containing 1000 ER graphs with 25 nodes; we attach to 500 graphs a cycle-motif with 5 nodes and a wheel-motif with 8 nodes to the remaining part.
- *er_cycle_wheel_grid*: containing 1500 ER graphs with 25 nodes; we attach to 500 graphs a cycle-motif with five nodes, a wheel-motif with eight nodes to the other 500 graphs, and a grid motif of dimension 3×3 to the remaining part.

3.4 Real-world datasets

With the aim of showing the good performance of our method in a generic test case, we performed experiments on real-world datasets. In particular, we choose the datasets described below (molecules) since they are given ground-truth explanations that enable a comparison between the hand-drawn graphs and the ones obtained by our proposal. The chosen datasets present various sub-graphs that act as an explanation for the graph labels assignment, and their characteristics are the following:

- *Benzene* [36]: it consists of 12000 molecular graphs from the ZINC15 database [37]. The molecules belong to two classes, and the goal is to predict whether a given molecule contains a benzene ring. For this dataset, the ground-truth explanations are the atoms (nodes) composing the benzene ring, and if the molecule contains multiple benzene rings, each of these forms the explanation.
- *Fluoride-Carbonyl* [36]: it contains 8761 molecular graphs labeled according to two classes. For this dataset, a positive sample indicates that the molecule comprises both a fluoride (F^-) and carbonyl ($C = O$) functional group, so the ground-true explanations consist of combinations of fluoride atoms and carbonyl

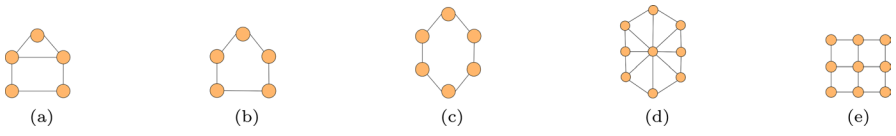


Fig. 4 Motifs used in synthetic datasets: (a) House, (b) Cycle 5, (c) Cycle 6, (d) Wheel, (e) Grid (3×3)

functional groups.

- *Mutagenicity* [38]: the dataset contains 1768 molecular graphs labeled into two classes according to their mutagenicity properties, i.e., the effect on the Gram-negative bacterium S. Typhimurium. The graph labels correspond to the presence or absence of toxicophores: NH_2, NO_2, aliphatic halide, nitroso, and azo-type.
- *Alkane-Carbonyl* [36]: it contains 1125 molecular graphs labeled according to two classes. A positive sample represents a molecule containing both an unbranched alkane and a carbonyl ($C = O$) functional group, so the ground-truth explanations include combinations of alkane and carbonyl functional groups.

To get an overall view, in Table 1, we reported some details about the graphs in the above-described datasets, including the properties used to form the ground-truth explanations. Note that, in the real-world datasets, there are cases in which the ground-truth explanation is composed of unconnected motifs, such as the case of Alkane-Carbonyl (see the left part of Fig. 6).

3.5 Evaluation criteria

Several metrics have been introduced to evaluate the effectiveness of a method in explaining the results from a GNN. In particular, the selected metrics compare the predicted and ground-truth explanations and use user-controlled parameters such as the probability distribution obtained by the GNN or the number of important features that compose the explanation mask.

Fidelity [17] measures how the prediction of the model changes if we remove nodes or edges or node features from the original graph. Let $\mathcal{G}_i$ the i-th graph of the test set, y_i the true label of the graph, $\hat{y_i}^{\mathcal{G}_i}$ the label predicted by the GNN, m_i the mask produced by the explanation algorithm. $\mathcal{G}_i \setminus m_i$ is the graph without the nodes that belong to the mask. $\mathcal{G}_i^{m_i}$ is the graph with only the important features detected by the algorithm. $\hat{y_i}^{\mathcal{G}_i \setminus m_i}$ and $\hat{y_i}^{\mathcal{G}_i^{m_i}}$ are the labels obtained feeding the GNN with the graphs $\mathcal{G}_i \setminus m_i$ and $\mathcal{G}_i^{m_i}$. It is possible to define two measures Fid^+ and Fid^- as:

$$Fid^+ = \frac{1}{N} \sum_{i=1}^{N} \left| g(\hat{y_i}, y_i) - g\left(\hat{y_i}^{\mathcal{G}_i \setminus m_i}, y_i\right) \right| \tag{6}$$

Table 1 Statistics of real-world datasets

Dataset	Benzene	Fluoride-Carbonyl	Mutagenicty	Alkane-Carbonyl
#Graphs	12000	8671	1768	1125
#Nodes	20.58 ± 0.004	21.26 ± 0.004	29.15 ± 0.35	21.13 ± 0.05
#Edges	43.65 ± 0.008	45.37 ± 0.09	60.83 ± 0.75	44.95 ± 0.12
#Features	14	14	14	14
Explanation	Benzene ring	F^- and $C = O$ chemical group	NH_2, NO_2 chemical group	Alkane and $C = O$ chemical group

$$Fid^- = \frac{1}{N}\sum_{i=1}^{N}\left|g(\hat{y}_i, y_i) - g\left(\hat{y}_i^{\mathcal{G}_i^{m_i}}, y_i\right)\right| \tag{7}$$

where:

$$g(y, \hat{y}) = \begin{cases} 1 & y = \hat{y} \\ 0 & y \neq \hat{y} \end{cases} \tag{8}$$

and N is the number of graphs in the test set. Fid^+ studies how the prediction changes if we remove the essential features (edges/features/nodes) identified by the explanation algorithm from the original graph. High values indicate good explanations, so the features detected by the algorithm are the most discriminative. The Fid^- measure studies how the predictions change if we consider only the features detected by the explanation algorithm. Low values indicate that the algorithm identifies the most discriminative features and does not consider the least discriminative ones.

Using the definition of Fid^+ and Fid^- measures, it is possible to define two properties that a reasonable explanation should have: necessity and sufficiency [39]. An explanation is necessary if the model prediction changes when we remove the features belonging to the explanation from the graph. A necessary explanation has a Fid^+ close to 1. Conversely, an explanation is sufficient if it leads independently to the model's original prediction. A sufficient explanation has a Fid^- close to 0.

Recent studies have highlighted some limitations regarding the traditional Fidelity measures due to out-of-distribution (OOD) issues when removing or isolating subgraphs for explainability evaluation [40]. In [40], Zheng et al. address these limitations, introducing a more robust surrogate class of fidelity measures named: $Fid^+_{\alpha_1}$ and $Fid^-_{\alpha_2}$ designed to mitigate distribution shift issues by introducing stochastic edge removal. The method is based on the stochastic sampling function E_α that removes edges with probability α, obtaining the modified fidelity metrics defined as:

$$Fid^+_{\alpha_1} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{M}\sum_{i=1}^{M}\left|g(\hat{y}_i, y_i) - g\left(\hat{y}_i^{G_i \setminus E_{\alpha_1}(m_i)}, y_i\right)\right|\right) \tag{9}$$

$$Fid^-_{\alpha_2} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{M}\sum_{i=1}^{M}\left|g(\hat{y}_i, y_i) - g\left(\hat{y}_i^{E_{\alpha_2}(G_i^{m_i})}, y_i\right)\right|\right) \tag{10}$$

where:

- $\alpha_1 \in [0, 1]$ controls the probability of retaining edges in the explanation subgraph; a higher α_1 ensures that more edges of the explanation are preserved, making $Fid^+_{\alpha_1}$ less sensitive to random perturbations;
- $\alpha_2 \in [0, 1]$ controls the probability of retaining edges outside the explanation subgraph, ensuring a fair evaluation of $Fid^-_{\alpha_2}$ by considering how much the model relies on the non-explanation subgraph;

- $\alpha_1 + \alpha_2 = 1$;
- M represents the number of stochastic perturbation samples applied to each graph, averaging over multiple random realizations to reduce variance in the computed fidelity score.

To summarize, $Fid^+_{\alpha_1}$ evaluates the necessity of the explanation subgraph by measuring how much the model's prediction changes when a stochastically sampled version of the explanation is removed. High values indicate that the explanation is crucial for the model's decision. $Fid^-_{\alpha_2}$ measures the sufficiency of the explanation, checking whether the explanation subgraph alone is enough to retain the original model prediction. Low values indicate that the explanation captures the most discriminative features.

Characterization Score (*charact*) is a global evaluation metric that considers Fid^+ and Fid^- [39]. This measure is helpful because it balances the sufficiency and necessity requirements of explanations. *charact* is the harmonic mean of Fid^+ and $1 - Fid^-$, and it is defined as follows:

$$charact = \frac{\omega_+ + \omega_-}{\frac{\omega_+}{Fid^+} + \frac{\omega_-}{1-Fid^-}} \tag{11}$$

where $\omega_+, \omega_- \in [0, 1]$ are respectively the weights for Fid^+ and $1 - Fid^-$ and satisfy $\omega_+ + \omega_- = 1$.

Graph Explanation Accuracy (*GEA*) [41] is an evaluation strategy that measures the correctness of an explanation using the ground-truth explanation $M^{\mathcal{G}}$. Ground truth and predicted explanations are binary vectors where 0 means an attribute is not essential and 1 means it is important for the model prediction. To measure accuracy, we used the Jaccard index between the ground truth $M^{\mathcal{G}}$ and predicted M^p, and the quantities $TP(M^{\mathcal{G}}, M^p)$ that indicates the number of attributes correctly identified by the mask M^p compared to the ground truth $M^{\mathcal{G}}$ (true positives), $FP(M^{\mathcal{G}}, M^p)$, which indicates the attributes wrongly identified (false positives) and $FN(M^{\mathcal{G}}, M^p)$, which indicates the number of attributes missed by the mask M^p (false negatives):

$$JAC\left(M^{\mathcal{G}}, M^p\right) = \frac{TP}{TP + FP + FN} \tag{12}$$

If we define ζ as the set of all possible ground-truth explanations, where $|\zeta| = 1$ for graphs having a unique explanation. GEA can be calculated as:

$$GEA(\zeta, M^p) = \max\left\{JAC(M^{\mathcal{G}}, M^p)\right\} \; \forall \; M^g \in \zeta. \tag{13}$$

4 Results

The proposed algorithm is tested using a GCN architecture, as described in Sect. 3.1. For the experimental setup, we considered an architecture composed of a GCN that collects the outputs of three layers, an average readout layer, and finally, a linear layer whose number of units is equal to the classes of the considered dataset with softmax activation. We performed a set of preliminary experiments to find a suitable configuration for the best classification results. We train the GNN using the Adam optimization algorithm [42] with a batch size of 64 and a learning rate of 0.05 in both cases. To ensure reliable results, we used 100 different random splits in training and testing with a ratio of 8:2 as a validation protocol. For the GNN computation, each graph node must be associated with a feature vector. For the synthetic datasets, we used the degree of each node as its feature vector. The node degree, which represents the number of edges connected to a node, provides a simple yet informative structural property for distinguishing node roles within the graph. In the real-world datasets, each node has a feature vector with 14 values that represent the chemical properties of the corresponding atom. After the training, we applied the explanation algorithm on the test set and computed the performance indices described above in Subsection 3.5.

For a thorough assessment of our method's performance, we compared the results against a random baseline and five state-of-the-art methodologies: PGMExplainer [20], PGExplainer [12], GNNExplainer [11], SubgraphX [15], and TAGE [13]. The random baseline mask on nodes was obtained using a Bernoulli distribution with a probability value of 0.5.

In the following sections, we report the results on synthetic and real datasets.

4.1 Results on synthetic datasets

We reported the results and the relative comparison in Table 2, where up-arrows indicate measures that ideally should be one, and down-arrows indicate measures that ideally should be zero. From the table, we observe that the GECo algorithm consistently excels in explainability across different datasets, as indicated by its high Fid^+ values and near-zero Fid^- values. For instance, in the ba_house_cycle dataset, GECo achieves $Fid^+ = 0.929$ and $Fid^- = 0$, effectively identifying key features while excluding irrelevant ones. This suggests that GECo focuses on the most discriminative features used in decision-making. In contrast, methods like GNNExplainer, despite detecting important features, tend to include irrelevant ones. For example, GNNExplainer reports $Fid^+ = 0.478$ and $Fid^- = 0.257$, indicating a less precise explanation compared to GECo. The *charact* metric further demonstrates GECo's ability to balance sufficiency and necessity, outperforming methods like PGExplainer, which struggles with lower scores due to weaker performance in either aspect. Regarding explanation correctness, as measured by the *GEA* metric, GECo consistently provides reliable and accurate explanations. For instance, it aligns well with ground-truth explanations in synthetic datasets like ba_cycle_wheel, where it correctly detects wheel motifs, as illustrated in Fig. 5. Note that since *GEA* is a Jaccard-based measure, it allows us to measure the redundancy in the retrieved explanation part. In particular, the larger and more redundant the explanation with respect

to the ground truth, the lower is the corresponding GEA. This behaviour has never been observed in our experiment; our GEA is always higher than that of our counterparts, also with respect to SubgraphX, whose criteria involve the search for the most informative non-redundant motif (see the comparison of GEA values between GECo and SubgraphX in the 4-th column of Table 3). Furthermore, to test the effectiveness of GECo also in an OOD scenario, we used the robust Fidelity measures previously described: $Fid_{\alpha_1^+}$ and $Fid_{\alpha_2^-}$. The obtained results are reported in the last two columns of Table 2. We can notice that GECo consistently outperforms all the other methods across all datasets, both in terms of $Fid^+_{\alpha_1}$ and $Fid^-_{\alpha_2}$. The bold results in the table show GECo's superiority, as it has the highest $Fid^+_{\alpha_1}$ values (ranging from 0.422 to 0.641) and the lowest $Fid^-_{\alpha_2}$ values (ranging from 0.004 to 0.066). These results suggest that GECo is highly effective in generating high-fidelity explanations with minimal false positives, critical for explainability in real-world applications. TAGE, while not consistently the best, shows competitive results, especially in some datasets like er_cycle_wheel and ba_cycle_wheel_grid. The method excels in certain metrics, achieving some of the lowest $Fid^-_{\alpha_2}$ values (such as 0.035 and 0.066), albeit with slightly lower $Fid^+_{\alpha_1}$ values compared to GECo. This suggests that TAGE might be more conservative in its explanations, potentially at the cost of lower overall fidelity. SubgraphX, while often in the lower range for $Fid^+_{\alpha_1}$, particularly struggles with $Fid^-_{\alpha_2}$ on many datasets. This is highlighted by its high values in several datasets (e.g., 0.395 in ba_house_cycle and 0.391 in er_cycle_wheel). Although it occasionally has promising results (like 0.222 in ba_cycle_wheel_grid), its variability suggests that SubgraphX may not provide stable, reliable explanations across all datasets. Finally, PGMExplainer and PGExplainer, while offering decent results, generally perform worse than GECo, with consistently lower $Fid^+_{\alpha_1}$ scores and higher $Fid^-_{\alpha_2}$ scores. The performance difference between GECo and these methods further emphasizes GECo's superior ability to provide high-quality explanations with fewer false positives. In summary, GECo emerges as the clear leader in terms of both the robustness and accuracy of its explanations, as indicated by its high $Fid^+_{\alpha_1}$ and low $Fid^-_{\alpha_2}$ across all datasets. While other methods like TAGE show competitive performance in specific cases, none match GECo's consistency. SubgraphX and PGExplainer, while offering some interesting insights, appear to struggle with stability and reliability, especially in the critical $Fid^-_{\alpha_2}$ metric. Furthermore, we measured explanation time (in seconds) for each test set graph, averaging over 100 splits. GECo consistently outperforms others, with under 3 s on average, while SubgraphX takes the longest, over 100 s. GNNExplainer and PGExplainer are slower than PGMExplainer, each taking around 100 s across datasets. TAGE results in the second fastest with about 7 s. These times refer to the ba_cycle_wheel dataset, but the same trend is observable for the other datasets. Overall, GECo stands out as the most consistent and robust explainability method across both binary and multiclass datasets. It maintains an excellent balance between sufficiency and necessity requirements (high Fid^+, low Fid^-, and high *charact* scores) while closely aligning with ground truth (high *GEA*). This makes GECo the most reliable option for generating clear, concise, and accurate explanations in GNN-based models.

Table 2 Results synthetic datasets and comparison of GECo with state-of-the-art methods, using the chosen 6 evaluation metrics. The best results are in bold and report mean±standard deviation over 100 runs; up-arrows indicate measures that ideally should be one, and down-arrows indicate measures that ideally should be zero

Dataset	Method	$Fid^+\uparrow$	$Fid^-\downarrow$	*charact*↑	*GEA*↑	$Fid^+_{\alpha_1}\uparrow$	$Fid^-_{\alpha_2}\downarrow$
ba_house_cycle	Random	0.411 ± 0.067	0.410 ± 0.053	0.477 ± 0.055	0.145 ± 0.004	0.256 ± 0.058	0.413 ± 0.038
	PGMExplainer	0.270 ± 0.056	0.436 ± 0.048	0.360 ± 0.048	0.089 ± 0.006	0.105 ± 0.038	0.409 ± 0.031
	PGExplainer	0.217 ± 0.137	0.459 ± 0.101	0.292 ± 0.152	0.213 ± 0.152	0.147 ± 0.080	0.446 ± 0.089
	GNNExplainer	0.478 ± 0.044	0.257 ± 0.066	0.579 ± 0.037	0.190 ± 0.001	0.309 ± 0.065	0.449 ± 0.040
	SubgraphX	0.191 ± 0.181	0.356 ± 0.252	0.270 ± 0.233	0.269 ± 0.147	0.129 ± 0.094	0.395 ± 0.229
	TAGE	0.426 ± 0.079	0.269 ± 0.072	0.533 ± 0.071	0.132 ± 0.021	0.281 ± 0.084	0.406 ± 0.068
	GECo	0.929 ± 0.043	0.000 ± 0.002	0.952 ± 0.024	0.305 ± 0.029	0.443 ± 0.073	0.042 ± 0.017
ba_cycle_wheel	Random	0.297 ± 0.033	0.294 ± 0.034	0.417 ± 0.035	0.172 ± 0.005	0.064 ± 0.018	0.309 ± 0.033
	PGMExplainer	0.133 ± 0.024	0.360 ± 0.040	0.219 ± 0.034	0.113 ± 0.007	0.006 ± 0.005	0.343 ± 0.030
	PGExplainer	0.111 ± 0.193	0.424 ± 0.133	0.139 ± 0.237	0.155 ± 0.149	0.014 ± 0.019	0.412 ± 0.124
	GNNExplainer	0.491 ± 0.035	0.026 ± 0.018	0.652 ± 0.032	0.234 ± 0.003	0.007 ± 0.005	0.343 ± 0031
	SubgraphX	0.329 ± 0.231	0.218 ± 0.300	0.439 ± 0.306	0.380 ± 0.181	0.032 ± 0.066	0.326 ± 0.089
	TAGE	0.499 ± 0.065	0.005 ± 0.005	0.662 ± 0.059	0.221 ± 0.025	0.337 ± 0.087	0.346 ± 0.125
	GECo	0.607 ± 0.100	0.000 ± 0.000	0.750 ± 0.076	0.553 ± 0.032	0.501 ± 0.054	0.020 ± 0.012
er_house_cycle	Random	0.368 ± 0.041	0.372 ± 0.040	0.461 ± 0.030	0.154 ± 0.005	0.227 ± 0.048	0.381 ± 0.039
	PGMExplainer	0.203 ± 0.053	0.430 ± 0.040	0.295 ± 0.058	0.097 ± 0.006	0.097 ± 0.031	0.423 ± 0.035
	PGExplainer	0.286 ± 0.154	0.426 ± 0.141	0.367 ± 0.169	0.265 ± 0.174	0.166 ± 0.090	0.406 ± 0.116
	GNNExplainer	0.471 ± 0.039	0.084 ± 0.030	0.621 ± 0.035	0.197 ± 0.021	0.197 ± 0.039	0.387 ± 0.026
	SubgraphX	0.260 ± 0.111	0.349 ± 0.110	0.363 ± 0.132	0.262 ± 0.130	0.032 ± 0.066	0.326 ± 0.089
	TAGE	0.435 ± 0.087	0.317 ± 0.148	0.526 ± 0.093	0.122 ± 0.036	0.337 ± 0.087	0.346 ± 0.125
	GECo	0.791 ± 0.090	0.000 ± 0.001	0.880 ± 0.057	0.391 ± 0.069	0.481 ± 0.062	0.051 ± 0.038

Table 2 (continued)

Dataset	Method	$Fid^{+}\uparrow$	$Fid^{-}\downarrow$	*charact*↑	*GEA*↑	$Fid^{+}_{\alpha_1}\uparrow$	$Fid^{-}_{\alpha_2}\downarrow$
er_cycle_wheel	Random	0.348 ± 0.055	0.352 ± 0.058	0.448 ± 0.038	0.170 ± 0.005	0.245 ± 0.086	0.396 ± 0.054
	PGMExplainer	0.213 ± 0.057	0.442 ± 0.046	0.303 ± 0.056	0.113 ± 0.005	0.115 ± 0.055	0.434 ± 0.052
	PGExplainer	0.222 ± 0.164	0.414 ± 0.134	0.295 ± 0.190	0.227 ± 0.167	0.121 ± 0.131	0.432 ± 0.094
	GNNExplainer	0.454 ± 0.044	$\underline{0.048 \pm 0.036}$	$\underline{0.613 \pm 0.040}$	0.227 ± 0.003	0.106 ± 0.050	0.427 ± 0.046
	SubgraphX	0.167 ± 0.124	0.392 ± 0.133	0.243 ± 0.160	$\underline{0.263 \pm 0.135}$	0.073 ± 0.129	0.391 ± 0.096
	TAGE	$\underline{0.466 \pm 0.097}$	0.164 ± 0.151	0.594 ± 0.109	0.208 ± 0.045	$\underline{0.413 \pm 0.140}$	$\underline{0.265 \pm 0.112}$
	GECo	0.866 ± 0.108	0.002 ± 0.018	0.923 ± 0.070	0.407 ± 0.050	0.487 ± 0.093	0.038 ± 0.036
ba_cycle_wheel_grid	Random	0.467 ± 0.034	0.468 ± 0.036	0.495 ± 0.023	0.185 ± 0.004	0.155 ± 0.040	0.487 ± 0.032
	PGMExplainer	0.229 ± 0.035	0.127 ± 0.005	0.299 ± 0.030	0.127 ± 0.005	0.032 ± 0.016	0.538 ± 0.049
	PGExplainer	0.214 ± 0.172	0.568 ± 0.099	0.260 ± 0.168	0.159 ± 0.114	0.080 ± 0.050	0.575 ± 0.097
	GNNExplainer	$\underline{0.664 \pm 0.032}$	0.147 ± 0.038	$\underline{0.746 \pm 0.026}$	0.256 ± 0.003	0.033 ± 0.017	0.538 ± 0.052
	SubgraphX	0.562 ± 0.142	0.217 ± 0.197	0.650 ± 0.161	$\underline{0.527 \pm 0.115}$	0.201 ± 0.096	$\underline{0.222 \pm 0.025}$
	TAGE	0.644 ± 0.074	$\underline{0.111 \pm 0.058}$	0.744 ± 0.057	0.236 ± 0.024	$\underline{0.479 \pm 0.079}$	0.239 ± 0.067
	GECo	0.887 ± 0.052	0.000 ± 0.002	0.939 ± 0.029	0.561 ± 0.036	0.501 ± 0.054	0.020 ± 0.012
er_cycle_wheel_grid	Random	0.520 ± 0.039	0.521 ± 0.034	0.496 ± 0.016	0.185 ± 0.004	0.331 ± 0.065	0.551 ± 0.042
	PGMExplainer	0.331 ± 0.051	0.623 ± 0.036	0.349 ± 0.032	0.128 ± 0.005	0.148 ± 0.037	0.605 ± 0.029
	PGExplainer	0.285 ± 0.153	0.557 ± 0.135	0.333 ± 0.151	0.205 ± 0.143	0.180 ± 0.103	0.561 ± 0.102
	GNNExplainer	0.628 ± 0.041	$\underline{0.074 \pm 0.024}$	0.748 ± 0.033	0.246 ± 0.002	0.147 ± 0.036	0.561 ± 0.102
	SubgraphX	0.325 ± 0.087	0.410 ± 0.095	0.415 ± 0.089	$\underline{0.378 \pm 0.098}$	0.229 ± 0.090	0.066 ± 0.088
	TAGE	$\underline{0.648 \pm 0.073}$	0.091 ± 0.051	$\underline{0.754 \pm 0.061}$	0.256 ± 0.014	$\underline{0.604 \pm 0.072}$	0.291 ± 0.072
	GECo	0.915 ± 0.028	0.001 ± 0.000	0.954 ± 0.017	0.510 ± 0.038	0.641 ± 0.063	0.066 ± 0.029

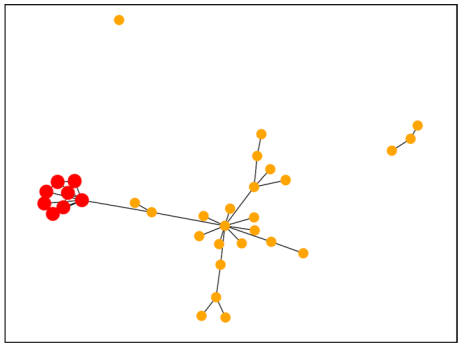


Fig. 5 An example of an explanation for a graph belonging to the wheel-motif class. The red nodes are the ones composing the explanation mask

4.2 Results on real datasets

For the real-world dataset, each layer's dimensionality is 64. In our experimental activity, we considered datasets containing molecules since we could obtain ground-truth explanations for these graphs and compare them with those obtained by our proposal. It is also essential to point out that the ground-truth explanation is only available for the positive class in the molecular datasets. For instance, in the Benzene dataset, a molecule belonging to the positive class contains a benzene ring, so the ground-truth explanation comprises the nodes corresponding to the atoms composing the benzene ring. Conversely, molecules of the negative class do not include the benzene ring, so the ground-truth explanation does not contain any node. However, all the explanation algorithms always respond and supporting the net response, providing an explanation mask where the most relevant features of the decision-making process were found. As a result, when calculating the *GEA* metric (see Eq. 13), a graph belonging to the negative class always yields a *GEA* of 0 since $TP = 0$, leading to overall low *GEA* values. Furthermore, to avoid division by zero, when $TP + FP + FN = 0$, we added a small constant $\epsilon = 1 \times 10^{-9}$ to the denominator. Table 3 reports the results of the GECo algorithm and the relative comparison with state-of-the-art approaches, where up-arrows indicate measures that ideally should be one, and down-arrows indicate measures that ideally should be zero.

GECo demonstrates superior performance in identifying necessary features, as indicated by its Fid^+ values, where it outperforms other approaches in all datasets except for the Mutagenicity dataset. This indicates that the features selected by GECo can nearly perfectly predict the model's original output.

This means that the subgraph parts that GECo selects are all essential for the output result. It is not perfect, but it is better than the other methods at selecting all the important parts. In terms of Fid^- values, GECo also excels, suggesting it can effectively identify sufficient features for accurate predictions independently. With all the datasets, GECo produces the best Fid^- values, finding the subgraph that triggers a given label. This trend is reinforced by the *charact* metric, where GECo achieves the highest scores, reflecting its excellent balance of necessary and sufficient requirements across most datasets.

In real-world applications, it is essential to ensure that the features identified by an explanation algorithm correspond to those recognized by domain experts, and

Table 3 Results real-world datasets and comparison of GECo with state-of-the-art methods, using the chosen 4 evaluation metrics

Dataset	Method	$Fid^{+}\uparrow$	$Fid^{-}\downarrow$	*charact*↑	*GEA*↑	$Fid^{+}_{\alpha_1}\uparrow$	$Fid^{-}_{\alpha_2}\downarrow$
Benzene	Random	0.437 ± 0.023	0.436 ± 0.023	0.491 ± 0.009	0.111 ± 0.002	0.290 ± 0.040	0.449 ± 0.022
	PGMExplainer	0.219 ± 0.045	0.243 ± 0.048	0.336 ± 0.053	0.085 ± 0.002	0.329 ± 0.043	0.421 ± 0.024
	PGExplainer	0.211 ± 0.053	0.504 ± 0.046	0.292 ± 0.052	0.048 ± 0.041	0.290 ± 0.066	0.488 ± 0.058
	GNNExplainer	0.499 ± 0.016	0.106 ± 0.009	0.640 ± 0.013	0.143 ± 0.002	0.427 ± 0.034	0.357 ± 0.021
	SubgraphX	0.290 ± 0.071	0.274 ± 0.085	0.407 ± 0.074	0.133 ± 0.081	0.196 ± 0.088	0.186 ± 0.067
	TAGE	0.507 ± 0.037	0.132 ± 0.053	0.639 ± 0.041	0.184 ± 0.009	0.343 ± 0.036	0.321 ± 0.032
	GECo	0.710 ± 0.053	0.015 ± 0.010	0.824 ± 0.039	0.236 ± 0.018	0.366 ± 0.085	0.100 ± 0.033
Fluor.-Carb.	Random	0.173 ± 0.070	0.172 ± 0.069	0.276 ± 0.088	0.034 ± 0.001	0.109 ± 0.042	0.148 ± 0.062
	PGMExplainer	0.066 ± 0.011	0.099 ± 0.027	0.123 ± 0.069	0.023 ± 0.001	0.126 ± 0.052	0.142 ± 0.053
	PGExplainer	0.129 ± 0.042	0.288 ± 0.092	0.215 ± 0.060	0.058 ± 0.025	0.081 ± 0.040	0.237 ± 0.082
	GNNExplainer	0.193 ± 0.096	0.084 ± 0.020	0.309 ± 0.121	0.039 ± 0.002	0.130 ± 0.066	0.134 ± 0.045
	SubgraphX	0.133 ± 0.040	0.162 ± 0.081	0.227 ± 0.060	0.020 ± 0.009	0.076 ± 0.035	0.149 ± 0.065
	TAGE	0.588 ± 0.094	0.332 ± 0.150	0.606 ± 0.077	0.046 ± 0.003	0.179 ± 0.107	0.122 ± 0.077
	GECo	0.615 ± 0.083	0.021 ± 0.008	0.751 ± 0.065	0.038 ± 0.005	$\mathbf{0.182 \pm 0.115}$	$\mathbf{0.065 \pm 0.026}$
Mutagenicity	Random	0.561 ± 0.112	0.563 ± 0.111	0.468 ± 0.030	0.033 ± 0.003	0.480 ± 0.148	0.591 ± 0.102
	PGMExplainer	0.464 ± 0.128	0.600 ± 0.076	0.408 ± 0.051	0.031 ± 0.004	0.326 ± 0.160	0.569 ± 0.115
	PGExplainer	0.244 ± 0.137	0.311 ± 0.106	0.344 ± 0.160	0.038 ± 0.049	0.354 ± 0.171	0.429 ± 0.043
	GNNExplainer	0.640 ± 0.066	0.086 ± 0.016	0.750 ± 0.054	0.033 ± 0.003	0.405 ± 0.111	0.317 ± 0.035
	SubgraphX	0.216 ± 0.119	0.275 ± 0.076	0.319 ± 0.140	0.025 ± 0.034	0.291 ± 0.151	0.355 ± 0.083
	TAGE	0.356 ± 0.078	0.357 ± 0.078	0.451 ± 0.063	0.054 ± 0.006	0.537 ± 0.115	0.363 ± 0.107
	GECo	0.481 ± 0.058	0.004 ± 0.003	0.647 ± 0.051	0.111 ± 0.011	$\mathbf{0.666 \pm 0.119}$	$\mathbf{0.288 \pm 0.104}$

Table 3 (continued)

Dataset	Method	$Fid^+\uparrow$	$Fid^-\downarrow$	*charact*↑	*GEA*↑	$Fid^+_{\alpha_1}\uparrow$	$Fid^-_{\alpha_2}\downarrow$
Alk.-Carb.	Random	0.253 ± 0.039	0.255 ± 0.037	0.375 ± 0.041	0.041 ± 0.005	0.121 ± 0.029	0.249 ± 0.043
	PGMExplainer	0.094 ± 0.024	0.299 ± 0.051	0.165 ± 0.037	0.027 ± 0.005	0.146 ± 0.032	0.231 ± 0.031
	PGExplainer	0.090 ± 0.080	0.382 ± 0.062	0.146 ± 0.106	$\underline{0.045 \pm 0.021}$	0.086 ± 0.077	0.386 ± 0.056
	GNNExplainer	0.304 ± 0.051	$\underline{0.092 \pm 0.022}$	$\underline{0.454 \pm 0.057}$	0.041 ± 0.003	0.210 ± 0.045	0.219 ± 0.027
	SubgraphX	0.188 ± 0.081	0.149 ± 0.094	0.302 ± 0.114	0.041 ± 0.019	0.052 ± 0.060	$0, 230 \pm 0.090$
	TAGE	$\underline{0.325 \pm 0.089}$	0.279 ± 0.081	0.435 ± 0.071	0.044 ± 0.013	$\underline{0.246 \pm 0.076}$	$\underline{0.126 \pm 0.0058}$
	GECo	0.575 ± 0.046	0.001 ± 0.003	0.728 ± 0.038	0.066 ± 0.009	$\mathbf{0.334 \pm 0.057}$	$\mathbf{0.102 \pm 0.005}$

The best results are in bold and report mean±standard deviation over 100 runs; up-arrows indicate measures that ideally should be one, and down-arrows indicate measures that ideally should be zero

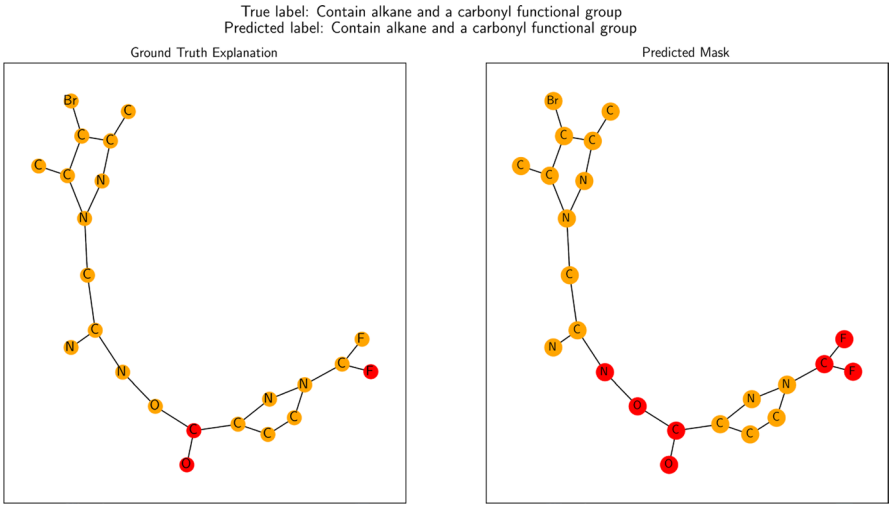


Fig. 6 Explanation example for a graph belonging to the Alkane-Carbonyl dataset. On the left, we have the ground-truth explanation; on the right, we have the predicted mask. In either case, the red nodes are the ones composing the mask

this is evaluated using the *GEA* metric. GECo outperforms other methods in this regard, aligning its predicted explanations closely with ground-truth explanations. For instance, in the Alkane-Carbonyl dataset, illustrated in Fig. 6, GECo correctly identifies the unconnected motifs constituted by *CO* and *F*. Considering the two other performances measure $Fid^{+}_{\alpha_1}$ and $Fid^{-}_{\alpha_2}$, the GECo algorithm still presents better performances than the other algorithms. Across all datasets, GECo consistently achieves the highest $Fid^{+}_{\alpha_1}$ and the lowest $Fid^{-}_{\alpha_2}$, indicating superior capability in identifying faithful and discriminative substructures. Notably, on the Mutagenicity dataset, GECo outperforms the closest competitor (TAGE) by a significant margin in both metrics, showcasing its robustness and precision. While GNNExplainer achieves the highest $Fid^{+}_{\alpha_1}$ on the Benzene dataset, GECo attains the lowest $Fid^{-}_{\alpha_2}$, suggesting a more conservative yet precise explanation. For more challenging datasets like Fluoride-Carbonyl and Alkane-Carbonyl, GECo maintains a clear lead, especially in reducing $Fid^{-}_{\alpha_2}$, which is crucial for avoiding misleading explanations. These results confirm that GECo is not only competitive but often superior to existing methods in generating reliable and explainable graph-level explanations, also tested in an OOD scenario.

GECo proves to be the fastest algorithm, taking about 18 s to explain the Benzene dataset, while PGMExplainer and TAGE take around 130 s. PGExplainer, GNNExplainer, and SubgraphX exceed 700 s. We observed the same trend for the other datasets. Furthermore, an investigation about trade-off computation between explanation time and fidelity shows that GECo achieves the best balance among the considered methods (see Section 3 of the Supplementary manuscript for details). GECo excels in identifying sufficient features for model predictions while filtering out irrelevant ones, maintaining a balance between necessity and sufficiency. It aligns well with ground truth, with minor discrepancies in localization. A good explanation should

be sparse, meaning that it should focus only on the most relevant parts of the input graph, discarding the irrelevant ones. To numerically quantify this property, it is possible to use the Sparsity metric, which measures the proportion of the original graph that is retained in an explanation. Lower sparsity values correspond to more compact explanations. Additional results included in Section 2 of the Supplementary manuscipt show that GECo achieves a good balance between sparsity and fidelity. Its explanations remain both concise and highly faithful to the model's predictions across varying sparsity levels.

5 Conclusions

In this paper, we propose GECo, a novel method for explaining GNNs, aimed at providing explainable and coherent explanations for graph classification tasks. Differently from previously proposed approaches that focus their attention on single nodes or small subgraphs extracted through a masking procedure or perturbation process, it leverages communities as standalone components, evaluating their collective contribution to the model's prediction. The experiments were conducted on a wide range of datasets, including both synthetic graphs (with known ground truth on the relevant structures) and real graphs. The obtained results confirm the ability of GECo to identify relevant subgraphs, both in terms of necessary and sufficient substructures for the model's decision. Furthermore, a significant advantage of GECo is the cumpotational efficency, obtaining competitive results in significantly shorter times than other state-of-the-art methods. The communities selected by GECo often exhibit strong coherence with the regions labeled as relevant in the ground truth, empirically validating the hypothesis that the communities represent relevant structures for explainability of the model. An additional advantage of our approach lies in the modularity of the pipeline. GECo can be easily incorporate other GNN models, regardless of the specific architecture, and can benefit from incremental improvements in the community detection modules. In summary, GECo represents a significant contribution to the field of GNN explainability, offering a good compromise between explanation quality, model fidelity, structural consistency with the original graph, and computational efficiency. The method is suitable for application in many real-world contexts where explainability is crucial, such as bioinformatics, medicine, and complex network analysis. Future work will explore several promising directions. In particular, we aim to integrate more advanced or adaptive community detection algorithms to improve the resolution and granularity of the extracted explanations.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00607-026-01642-z.

Author contributions Conceptualization: Domenico Amato, Salvatore Calderaro, Giosuè Lo Bosco, Riccardo Rizzo, and Filippo Vella. Methodology: Domenico Amato, Salvatore Calderaro, Giosuè Lo Bosco, Riccardo Rizzo, and Filippo Vella. Software: Domenico Amato and Salvatore Calderaro. Validation: Domenico Amato, Salvatore Calderaro, Giosuè Lo Bosco, Riccardo Rizzo, and Filippo Vella. Writing—original draft preparation: Domenico Amato, Salvatore Calderaro, Giosuè Lo Bosco, Riccardo Rizzo, and Filippo Vella. Writing—review and editing: Domenico Amato, Salvatore Calderaro, Giosuè

Lo Bosco, Riccardo Rizzo, and Filippo Vella. All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by Università degli Studi di Palermo within the CRUI-CARE Agreement.

Data and code availability The official GECo implementation and the synthetic datasets produced for this study can be found at the following GitHub Repository [43]. An official Python package is also published on PyPI with the MIT License [44].

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. Amato D, Di Gangi MA, Lo Bosco G, Rizzo R (2020) Recurrent deep neural networks for nucleosome classification. In: Raposo M, Ribeiro P, Sério S, Staiano A, Ciaramella A (eds) Computational intelligence methods for bioinformatics and biostatistics. Springer, Cham, pp 118–127
2. Amato D, Di Gangi MA, Fiannaca A, La Paglia L, La Rosa M, Lo Bosco G, Rizzo R, Urso A (2021) In: Elloumi, M. (ed.) Classification of sequences with deep artificial neural networks: representation and architectural issues, pp 27–59. Springer, Cham. https://doi.org/10.1007/978-3-030-71676-9_2
3. Amato D, Calderaro S, Lo Bosco G, Rizzo R, Vella F (2024) Explainable histopathology image classification with self-organizing maps: a granular computing perspective. Cognitive Computation, 1–21
4. Amato D, Calderaro S, Lo Bosco G, Rizzo R, Vella F (2023) Metric learning in histopathological image classification: opening the black box. Sensors 23(13):6003
5. Calderaro S, Lo Bosco G, Vella F, Rizzo R (2023) Breast cancer histologic grade identification by graph neural network embeddings. In: International work-conference on bioinformatics and biomedical engineering, pp 283–296. Springer
6. Amato D, Calderaro S, Lo Bosco G, Rizzo R, Vella F (2024) Semantic segmentation of gliomas on brain mris by graph convolutional neural networks. In: 2024 international conference on AI X data and knowledge engineering (AIxDKE), pp 143–149. IEEE
7. Amato D, Calderaro S, Lo Bosco G, Rizzo R, Vella F (2024) Bacteria taxonomic classification using graph neural networks. In: 2024 IEEE international conference on evolving and adaptive intelligent systems (EAIS), pp 1–6. IEEE
8. Amato D, Calderaro S, Lo Bosco G, Vella F, Rizzo R (2025) Proteins transcription factor prediction using graph neural networks. In: International meeting on computational intelligence methods for bioinformatics and biostatistics, pp 15–27. Springer
9. Fortunato S, Castellano C (2007) Community structure in graphs. arXiv preprint arXiv:0712.2716
10. Yuan H, Yu H, Gui S, Ji S (2022) Explainability in graph neural networks: a taxonomic survey. IEEE Trans Pattern Anal Mach Intell 45(5):5782–5799

11. Ying Z, Bourgeois D, You J, Zitnik M, Leskovec J (2019) Gnnexplainer: Generating explanations for graph neural networks. Adv Neural Inform Processing Syst **32**
12. Luo D, Cheng W, Xu D, Yu W, Zong B, Chen H, Zhang X (2020) Parameterized explainer for graph neural network. Adv Neural Inf Process Syst 33:19620–19631
13. Xie Y, Katariya S, Tang X, Huang E, Rao N, Subbian K, Ji S (2022) Task-agnostic graph explanations. In: Proceedings of the 36th international conference on neural information processing systems. NIPS '22. Curran Associates Inc., Red Hook, NY, USA
14. Schlichtkrull MS, Cao ND, Titov I (2021) Interpreting graph neural networks for NLP with differentiable edge masking. In: International conference on learning representations
15. Yuan H, Yu H, Wang J, Li K, Ji S (2021) On explainability of graph neural networks via subgraph explorations. In: International Conference on Machine Learning, pp 12241–12252. PMLR
16. Baldassarre F, Azizpour H (2019) Explainability techniques for graph convolutional networks. In: International conference on machine learning (ICML) workshops, 2019 workshop on learning and reasoning with graph-structured representations
17. Pope PE, Kolouri S, Rostami M, Martin CE, Hoffmann H (2019) Explainability methods for graph convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
18. Huang Q, Yamada M, Tian Y, Singh D, Chang Y (2022) Graphlime: Local interpretable model explanations for graph neural networks. IEEE Transactions on Knowledge and Data Engineering
19. Zhang Y, Defazio D, Ramesh A (2021) Relex: A model-agnostic relational model explainer. In: Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society, pp 1042–1049
20. Vu M, Thai MT (2020) Pgm-explainer: probabilistic graphical model explanations for graph neural networks. Adv Neural Inf Process Syst 33:12225–12235
21. Schwarzenberg R, Hübner M, Harbecke D, Alt C, Hennig L (2019) Layerwise relevance visualization in convolutional text graph classifiers. In: Proceedings of the thirteenth workshop on graph-based methods for natural language processing (TextGraphs-13), pp. 58–62. Association for Computational Linguistics, Hong Kong
22. Schnake T, Eberle O, Lederer J, Nakajima S, Schütt KT, Müller K-R, Montavon G (2022) Higher-order explanations of graph neural networks via relevant walks. IEEE Trans Pattern Anal Mach Intell 44(11):7581–7596
23. Yuan H, Tang J, Hu X, Ji S (2020) 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, pp. 430–438. Association for Computing Machinery, New York, NY, USA
24. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech: Theory Exp 2008(10):10008
25. Traag VA, Waltman L, Van Eck NJ (2019) From louvain to leiden: guaranteeing well-connected communities. Sci Rep 9(1):1–12
26. Wu X, Hu J, Quan Y, Miao Q, Sun PG (2025) Motif-based contrastive graph clustering with clustering-oriented prompt. Inform Processing Manag 62(5):104208. https://doi.org/10.1016/j.ipm.2025.104208
27. Wu X, Lu W, Quan Y, Miao Q, Sun PG (2024) Deep dual graph attention auto-encoder for community detection. Expert Syst Appl 238:122182. https://doi.org/10.1016/j.eswa.2023.122182
28. Wu X, Hu J, Zhang A, Quan Y, Miao Q, Sun PG (2025) Structure-enhanced contrastive learning for graph clustering. IEEE Transactions on Network Science and Engineering
29. Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: International Conference on Learning Representations
30. Hamilton WL (2020) Graph representation learning. Morgan & Claypool Publishers, San Rafael, CA(USA)
31. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69:026113
32. Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70:066111
33. Erdös P, Rényi A (1959) On random graphs i. Publ math debrecen 6(290–297):18
34. Gilbert EN (1959) Random graphs. Ann Math Stat 30(4):1141–1144
35. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
36. Sanchez-Lengeling B, Wei J, Lee B, Reif E, Wang P, Qian W, McCloskey K, Colwell L, Wiltschko A (2020) Evaluating attribution for graph neural networks. Adv Neural Inf Process Syst 33:5898–5910
37. Sterling T, Irwin JJ (2015) Zinc 15-ligand discovery for everyone. J Chem Inf Model 55(11):2324–2337

38. Kazius J, McGuire R, Bursi R (2005) Derivation and validation of toxicophores for mutagenicity prediction. J Med Chem 48(1):312–320
39. Amara K, Ying Z, Zhang Z, Han Z, Zhao Y, Shan Y, Brandes U, Schemm S, Zhang C (2022) Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. In: The First Learning on Graphs Conference
40. Zheng X, Shirani F, Wang T, Cheng W, Chen Z, Chen H, Wei H, Luo D (2024) Towards robust fidelity for evaluating explainability of graph neural networks. In: The twelfth international conference on learning representations
41. Agarwal C, Queen O, Lakkaraju H, Zitnik M (2023) Evaluating explainability for graph neural networks. Sci Data 10(1):144
42. Kingma D, Ba J (2015) Adam: A method for stochastic optimization. In: International conference on learning representations (ICLR), San Diego, CA, USA
43. Calderaro S (2026) GECo explainer: official github repository. https://github.com/salvatorecalderaro/geco_explainer Accessed 2026-01-24
44. Python Package for GECo Explainer. https://pypi.org/project/geco-explainer/ Accessed 2026-01-24

Authors and Affiliations

Domenico Amato[1] · Salvatore Calderaro[1,2] · Giosuè Lo Bosco[1] · Riccardo Rizzo[3] · Filippo Vella[3]

✉ Salvatore Calderaro
salvatore.calderaro01@unipa.it

Domenico Amato
domenico.amato01@unipa.it

Giosuè Lo Bosco
giosue.lobosco@unipa.it

Riccardo Rizzo
riccardo.rizzo@icar.cnr.it

Filippo Vella
filippo.vella@icar.cnr.it

[1] Department of Mathematics and Computer Science, University of Palermo, Via Archirafi 34, Palermo 90123, Italy

[2] Department of Physics and Chemistry - Emilio Segrè, University of Palermo, Viale Delle Scienze Ed. 18, Palermo 90128, Italy

[3] Institute for high performance computing and networking, National Research Council of Italy, Via Ugo La Malfa 153, Palermo 90146, Italy