

CRoCExplainer: A Tool to Explain Graph Learning with Improved Connectedness Representation

1st Yanghepu Li

Graduate School of Advanced Integrated Studies
in Human Survivability, Kyoto University
Kyoto, Japan
li.yanghepu.58z@st.kyoto-u.ac.jp

2nd Liang Zhao

Graduate School of Advanced Integrated Studies
in Human Survivability, Kyoto University
Kyoto, Japan
liangzhao@acm.org

Abstract—Graph Neural Network (GNN) is a powerful tool for automatically learning and predicting features of graph data. To study the explainability of a given GNN model, several methods have been proposed, including GNNExplainer, CF², and SubgraphX. Unfortunately, these methods lack sufficient consideration on the connectedness of the output. The challenge is that the connectedness of a graph is difficult to optimize within the framework of Graph Learning (GL), because it is defined as a discrete number of the connected components of the graph, whereas GL uses continuous optimization (e.g., gradient descent).

To address this issue, this article proposes a novel Continuous Representation of Connectedness (CRoC) with time complexity of $O(n^3)$, improving from a previous study Connected-CF². Moreover, an optimizable framework CRoCExplainer is given to integrate both reasoning and connectedness for a given GNN model. Empirical studies show that CRoCExplainer outperforms the state-of-the-art methods in interpretability and efficiency.

Index Terms—Graph Learning; Graph Neural Network; Connectedness of Graph; Explainability of AI;

I. INTRODUCTION

Graph is a general mathematical structure. It uses nodes and edges to denote entities and relations between entities, respectively. Graphs are widely used to model traffic networks, social networks, citation networks, molecules, and more, with numerous algorithms developed so far [1].

Recently, graphs have become objects of Machine Learning (ML) as well. Let $\mathcal{G}$ be a set of graphs for which nodes or edges or both are associated with some features. Let $\mathcal{F}$ be the set of features. This article assumes that $\mathcal{F}$ consists of a finite number of elements, called classes. Let $f : \mathcal{G} \rightarrow \mathcal{F}$ denote the mapping from a graph $G \in \mathcal{G}$ to a feature $f(G) \in \mathcal{F}$. In practice, f is usually unknown, and ML is used to find an approximation f_θ of f from known data with a learnable parameter θ . Linear regression, decision tree, and Support Vector Machine (SVM) are some common ML algorithms for that purpose.

Graph Neural Network (GNN) is a new ML algorithm for studying graphs. It was first introduced by Scarselli et al. [2], who extended recurrent models to handle graphs. In recent years, this field has significant advancements. Gilmer et al. [3] introduced message-passing neural networks (MPNNs) in 2017. It uses Artificial Neural Network (ANN) to learn

the representation of a node by iteratively aggregating the information of its neighbors [4]. More precisely, the feature (i.e., representation) $\mathbf{h}_v^{(k)}$ of a node v at layer k is updated by

$$\mathbf{h}_v^{(k+1)} = \phi(\mathbf{h}_v^{(k)}, \text{AGGREGATE}(\{\mathbf{h}_u^{(k)}, u \in \mathcal{N}(v)\})),$$

where $\mathcal{N}(v)$ denotes the neighbor nodes of v , AGGREGATE and ϕ are learnable functions to aggregate the features from v 's neighbors and to update the feature of v , respectively. Compared to traditional ML methods, GNNs have achieved advanced results in learning and representing graph structures [5], contributing to diverse real-world applications. For instance, Yan et al. [6] used GNNs to learn human action patterns based on skeleton action recognition. Qi et al. [7] identified human-object interaction using GNNs. Zhang et al. [8], Li et al. [9], and Yu et al. [10] applied GNNs to address the traffic prediction problem. For social networks, Bonifazi et al. investigated the COVID-19 vaccine discussions on Twitter [11] and the dynamics of user and community sentiments in a social platform [12].

However, GNNs usually use a large number of variables and implicit functions; thus are difficult for humans to understand [5]. This black-box fashion prevents GNNs from critical applications such as material design and drug discovery where we human need to understand the (simple) reasoning. Therefore, to make the maximum use of advanced GNNs, explainability is considered an important issue.

Given a GNN model f_θ , an explainability study aims to find a simpler “reasoning” on the output $f_\theta(G)$ of f_θ for an input graph G . A common reasoning is to use subgraphs (notice that subsets of features are also possible. In this article, we only consider subgraphs for ease of notation). That is, given a GNN model f_θ , we want to find a (small) subgraph G' for a given graph G such that $f_\theta(G') \approx f_\theta(G)$ (called an instance-level explainer. Note that it is also possible to use f_θ instead of f .) or a (simple) model $f'_{\theta'}$ (e.g., a regression model, a probability distribution, etc.) such that $f'_{\theta'}(G) \approx f_\theta(G)$ for all G (called a model-level explainer).

Early studies attempted to extend feature-based approaches to extract important input features [13], [14]. However, these methods cannot deal with gradient saturation [15] or node classification explanations [16]. Therefore, they are not able to give subgraph explanations of GNNs. To address this issue,

Ying et al. [17] proposed GNNExplainer, a now well-known instance explainer that tries to find a subgraph maximizing the change in the probability of prediction when removed (counterfactual reasoning). On the other hand, CF² proposed by Tan et al. [18] considers both counterfactual and factual reasoning (i.e. to find a small enough subgraph that minimizes the change in the probability of prediction when reserved).

While these subgraph-based approaches [17], [18] are easy to implement, they did not consider the connectedness of the reasoning subgraphs; thus are not useful in the cases where connectedness is important such as finding a community in a social network and a functional group in a chemical compound. In contrast, SubgraphX [19] tries to find small connected subgraphs as the reasoning. Unfortunately, it has exponential time complexity; hence is limited to finding very small connected subgraphs. Moreover, it cannot be used to find explanations with two or more connected components.

Lacking sufficient consideration on the connectedness of the reasoning subgraphs arises because that the connectedness of a graph is difficult to optimize within the framework of Graph Learning (GL), since it is defined as a discrete number of the (connected) components of the graph, whereas GL uses continuous optimization method (e.g., gradient descent). To address this issue, Li proposed Connected-CF² [20] by introducing a real-valued, continuous relaxation of the number of components and including it in the penalty function to be minimized. This relaxation leverages spectral graph theory and linear algebra. Empirical studies show that, comparing to the previous methods, Connected-CF² finds subgraphs with better trade-off between connectedness and explainability. However, it lacks an analysis on the approximation ratio, which complicates the assessment of the discrepancy between the approximated outcomes and the truth. Additionally, Connected-CF² has high computational complexity of $O(n^5)$, where n is the number of nodes of the graph.

Contributions. In this article, we propose a Continuous Representation of Connectedness (CRoC) to address the aforementioned issues. CRoC is in continuous form that can be optimized with a standard continuous optimization method. The time complexity of CRoC is $O(n^3)$, improved from Connected-CF² by two magnitude. We further provide empirical analysis on the approximation accuracy of CRoC. Moreover, we give an optimization framework CRoCExplainer to maximize the reasoning of outputs and ensure corresponding connectedness. Empirical studies show that the proposed CRoCExplainer can give significantly improved explanations with higher efficiency compared with the state-of-the-art methods.

The rest of this article is organized as follows. Section II gives the definition of CRoC and provides the corresponding algorithm, as well as discussions about properties of CRoC and corresponding empirical experiments. Then Section III details the explainer framework. Section IV presents empirical results and discussions, and finally Section V concludes the article.

II. PRELIMINARIES AND THE CROC METRIC

Let $G = (V, E)$ denotes a simple undirected graph with a set $V = \{v_1, v_2, \dots, v_n\}$ of n nodes and a set E of m edges. It is

said connected if there is a path between every pair of nodes in V . A (connected) component of G is a maximal connected subgraph of G . The adjacency matrix $A(G) = (a_{ij}) \in \mathbb{R}^{n \times n}$ of G is the matrix with element $a_{ij} = 1$ if $(v_i, v_j) \in E$ and 0 otherwise, for all $i, j = 1, 2, \dots, n$. Let $d(v)$ denotes the degree of a node v , i.e. the number of edges incident to v . By definition, $d(v_i) = \sum_j a_{ij}$ for all i .

The Laplacian matrix of G is $L(G) = D(G) - A(G) \in \mathbb{R}^{n \times n}$ where $D(G) = (d_{ij}) \in \mathbb{R}^{n \times n}$ is the degree matrix of G , i.e., $d_{ii} = d(v_i)$, and $d_{ij} = 0$ if $i \neq j$. Therefore, denoting $L(G) = (L_{ij})$, we have

$$L_{ij} = \begin{cases} d(v_i) & \text{if } i = j, \\ -1 & \text{if } i \neq j, (v_i, v_j) \in E, \\ 0 & \text{otherwise (i.e., } i \neq j, (v_i, v_j) \notin E). \end{cases} \quad (1)$$

Notice that $L(G)$ is real and symmetric as G is undirected. This implies that it has a complete set of real eigenvalues. In fact, it can be shown that $L(G)$ is positive-semidefinite; hence all eigenvalues of $L(G)$ are nonnegative (see [4]).

Theorem 1 serves as the theoretical foundation of this article. It highlights the intrinsic relationship between the (discrete) connectedness of a graph and its Laplacian matrix.

Theorem 1 (see [21]–[23]). *For any simple undirected graph G , the multiplicity of eigenvalue 0 of $L(G)$ is equal to the number K of the components of G .* $\square$

Now let us introduce the continuous connectedness CRoC for graphs. Given a simple undirected graph G with K components, re-index the nodes so that nodes $v_1, v_2, \dots, v_{k_1}$ induce a component of the largest size k_1 , nodes $v_{k_1+1}, \dots, v_{k_1+k_2}$ induce a component of the second largest size k_2 ($k_2 \leq k_1$), and so on. The last component containing v_n is thus the smallest component. Let $e_1 = (1, \dots, 1, 0, \dots, 0)^T \in \{0, 1\}^n$ where the first k_1 elements are 1s and other elements are 0s. Vector e_1 is called the indicator vector of the first component. Similarly we can define the indicator vector $e_2 = (e_{2i}) \in \{0, 1\}^n$ for the second component so that $e_{2i} = 1$ if and only if $k_1 + 1 \leq i \leq k_1 + k_2$, and so on. The indicator vector $e_K \in \{0, 1\}^n$ is associated with the last component. We have the next theorem.

Theorem 2. *The set $\{e_1, e_2, \dots, e_K\}$ of indicator vectors as defined above is an orthogonal basis of the eigenspace corresponding to the zero eigenvalue of $L(G)$.*

Proof. It is obvious that $e_i \neq \mathbf{0}$ and $e_i^T e_j = 0$ for any $i \neq j$. Moreover, for any $i = 1, 2, \dots, K$, $L(G)e_i = (D(G) - A(G))e_i = \mathbf{0}$. Since K is the number of components of G , Theorem 1 shows that the multiplicity of zero eigenvalue of $L(G)$ is K . This implies that the eigenspace corresponding to zero eigenvalue of $L(G)$ is of dimension K . Therefore, the set $\{e_1, \dots, e_K\}$ is an orthogonal basis of the eigenspace corresponding to the zero eigenvalue of $L(G)$. $\square$

Let $e_1, \dots, e_K$ be defined as above. We define a vector

$$u(G) = \sum_{i=1}^K i \times e_i. \quad (2)$$

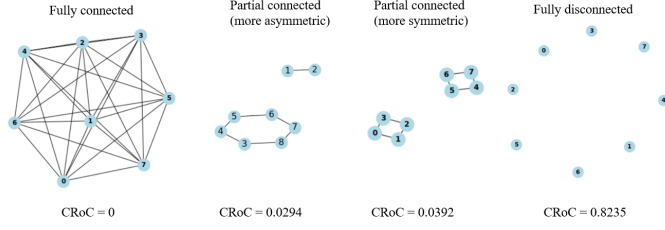


Fig. 1. CRoC values for some graphs with the same number of nodes.

Notice that $u(G) \in \mathbb{R}^n$ is uniquely decided by G . In fact, letting $u(G) = (u_1, \dots, u_n)^T$, we have $u_i = \dots = u_{k_1} = 1, u_{k_1+1} = \dots = u_{k_1+k_2} = 2, \dots, u_n = K$. Let $\bar{u} = \frac{1}{n} \sum_{i=1}^n u_i$ be the average of u_i . We define

$$\text{CRoC}(G) = \frac{4 \sum_{i=1}^n (u_i - \bar{u})^2}{\sum_{i=1}^n i^2} \quad (3)$$

to be the CRoC of G (see Fig. 1 for some examples). Notice that, like the variance, CRoC estimates how far the vector $u(G)$ is from a mean vector. It is easy to see that $\text{CRoC}(G) = 0$ if and only if G is connected (i.e., $K = 1$). Moreover, we can prove the following theorems.

Theorem 3. For two n -nodes graphs G_1 and G_2 with K_1 size- d_1 and K_2 size- d_2 components, respectively, $K_1 > K_2$ implies $\text{CRoC}(G_1) > \text{CRoC}(G_2)$.

Proof. Let $u(G_1) = (u_1, u_2, \dots, u_n)^T$, where $n = d_1 K_1$. By definition, $u_1 = \dots = u_{d_1} = 1, u_{d_1+1} = \dots = u_{2d_1} = 2, \dots, u_{n-d+1} = \dots = u_n = K_1$. Therefore,

$$\bar{u}(G_1) = \frac{d_1(1 + 2 + \dots + K_1)}{n} = \frac{K_1 + 1}{2}.$$

Hence

$$\text{CRoC}(G_1) = \frac{4d_1 \sum_{i=1}^{K_1} (i - \frac{K_1+1}{2})^2}{\sum_{i=1}^n i^2} = \frac{2(K_1^2 - 1)}{2n^2 + 3n + 1}.$$

Analogously, $\text{CRoC}(G_2) = \frac{2(K_2^2 - 1)}{2n^2 + 3n + 1} < \text{CRoC}(G_1)$. $\square$

Remark 1. Notice that $\text{CRoC}(G_i) \rightarrow 1$ when $K_i \approx n$ and $n \rightarrow \infty$ for the graphs considered by Theorem 3.

Remark 2. Theorem 3 shows graphs containing more components with similar symmetry exhibit larger CRoC values. However, the converse may not hold, i.e., a larger CRoC may not imply more components. For instance, consider two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $V_1 = V_2 = \{v_1, v_2, v_3, v_4\}$, $E_1 = \{(v_1, v_2), (v_3, v_4)\}$, and $E_2 = \{(v_2, v_3), (v_3, v_4)\}$. It is easy to verify that $\frac{2}{15} = \text{CRoC}(G_1) > \text{CRoC}(G_2) = \frac{1}{10}$ but $K_1 = K_2$. This example illustrates that, as expected, CRoC can quantify component size asymmetry (variance) in graphs with identical node counts and component counts, in addition to evaluating the connectedness.

The proof for Theorem 3 also shows the next theorem.

Theorem 4. For two graphs G_1 and G_2 with K_1 size- d_1 and K_2 size- d_2 components, respectively, $K_1 = K_2$ and $d_1 < d_2$ implies $\text{CRoC}(G_1) > \text{CRoC}(G_2)$. $\square$

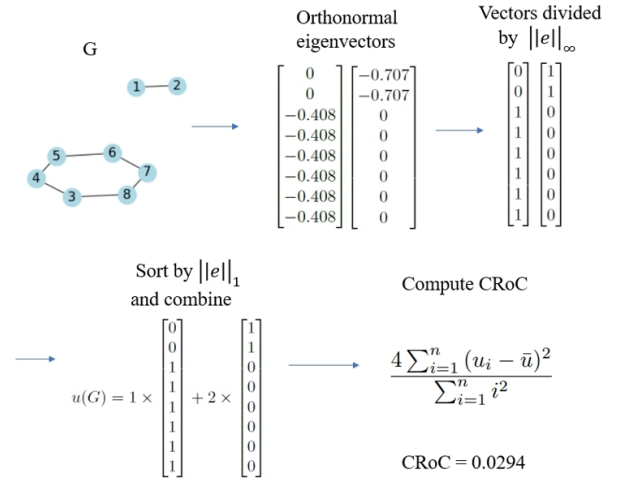


Fig. 2. An illustration on Algorithm 1 working against a given graph (notice that we do not really need to re-index nodes after sorting).

Although the above theorems clarify the definition of CRoC and some intrinsic properties, calculating Eq. 3 requires discretely constructing the basis mentioned in Theorem 2, which is not directly applicable in the context of continuous optimization. Therefore, we propose Algorithm 1 to calculate CRoC from a set of orthogonal eigenvectors corresponding to the eigenvalue 0 for $L(G)$, where the ∞ -norm $\|\cdot\|_\infty$ and the 1-norm $\|\cdot\|_1$ calculate the maximum absolute value among elements and the sum of the absolute values of the elements of a vector, respectively.

Algorithm 1 Finding $\text{CRoC}(G)$

Input: A simple undirected graph G

- 1: Calculate the Laplacian matrix $L(G)$ of G .
- 2: Calculate the set $S = \{e_1, \dots, e_k\}$ of the orthonormal eigenvectors corresponding to eigenvalue 0 for $L(G)$ (see the next remark).
- 3: $e_i = e_i / \|e_i\|_\infty$ for all i .
- 4: Sort S in nondecreasing order of $\|e_i\|_1$.

Output: The value calculated by Eq. 2 and Eq. 3.

Remark. For Step 2, we use `numpy.linalg.eigh()` from the NumPy (<https://numpy.org/>) package, where eigenvalues and eigenvectors are computed using LAPACK [24]. Fig. 2 illustrates Algorithm 1 for a given graph. Notice that the time complexity of Algorithm 1 is $O(n^3)$, dominated by Step 2.

We note that the eigenspace corresponding to eigenvalue 0 for $L(G)$ may have more than one orthonormal basis. Therefore Step 2 in Algorithm 1 may find an orthogonal basis different from the one shown in Theorem 2. Nevertheless, our empirical studies show that Algorithm 1 could find the preferred basis in many cases. Fig. 3 compares the value of Eq. 3 based on the indicator basis and the output by Algorithm 1 for 1,000 random graphs of 20 nodes generated by the Erdős–Rényi model [25], [26]. The probability p for an edge generation follows a normal distribution with mean $\mu_p = 0.03$ and standard deviation $\sigma_p =$

0.01, i.e., $p \sim \mathcal{N}(\mu_p, \sigma_p^2)$. It can be observed that in most cases, the output of Algorithm 1 is within the value of Eq. 3 and three times of that value.

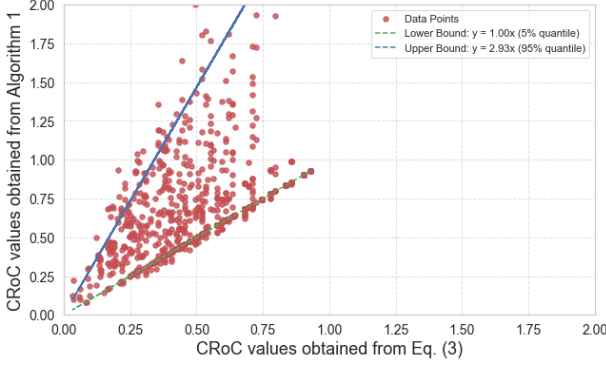


Fig. 3. A comparison between the value of Eq. 3 and the output of Algorithm 1 for 1,000 random graphs. It can be observed that the latter is usually within the former and three times of it.

To further demonstrate that CRoC (here and thereafter, CRoC refers to the results calculated from Algorithm 1) shares similar properties shown by Theorem 3, we conduct several empirical studies.

Firstly, we construct graphs of 100 nodes and increase the number K of the components from 1 to 100 in a symmetric manner such that the sizes of components differ the minimum. For $K = 2$, each component contains 50 nodes. For $K = 3$, the three components contain 33, 33, and 34 nodes, respectively. And so on. For $K = 100$, every component contains a single node. Red circles in Fig. 4 show the results. It be observed that CRoC increases as the number of components increases.

Next, we construct graphs containing 100 nodes and increase the number K of components from 1 to 100 in the “most asymmetric” manner. For $K = 2$, one component contains 99 nodes, and the other is an isolated node. For $K = 3$, one component contains 98 nodes, and the other two are isolated nodes. And so on. For $K = 100$, each component contains a single node. Blue triangles in Fig. 4 show that CRoC increases as the number K of components increases as well.

Then, we construct a graph with 100 nodes and 2 connected components. We systematically vary the ratio r of the sizes of the two components, starting from the most asymmetric case $r = 1/99$ and gradually increasing the symmetry until it reaches the most symmetric case $r = 1$. This allows us to explore how CRoC changes with respect to the symmetry between components. Fig. 5 highlights that, for this fixed number of components, the CRoC increases as the symmetry between the components increases. **Notably, a comparison between Fig. 4 and Fig. 5 shows that although both the component count and symmetry of components influence the CRoC, the effect of the component count to CRoC is far more pronounced.**

Finally, to empirically study CRoC in real-world scenarios, we use one hundred random networks, all with 100 nodes and generated by the Erdős–Rényi model [25], [26]. The probability p for an edge generation follows a normal distribution with mean $\mu_p = 0.03$ and standard deviation $\sigma_p = 0.01$, i.e.,

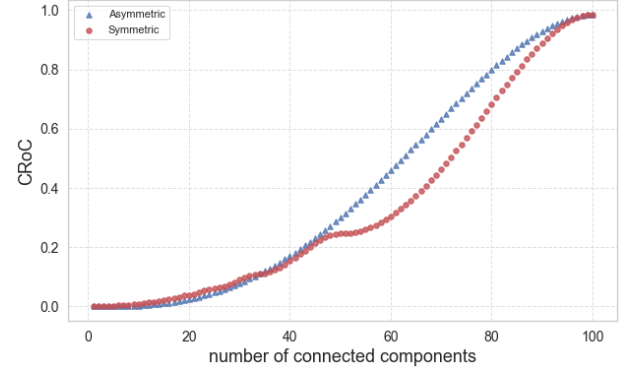


Fig. 4. Correlations between CRoC (calculated from Algorithm 1) and the number of connected components. Red circles represent the symmetric graph, whereas blue triangles denote the asymmetric graph. The total number of nodes of the graph is fixed to 100.

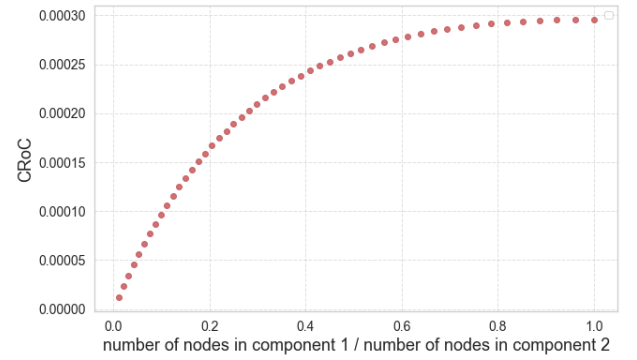


Fig. 5. Correlations between CRoC (calculated from Algorithm 1) and the ratio of the sizes of the two components. The total number of nodes of the graph is fixed to 100.

$p \sim \mathcal{N}(\mu_p, \sigma_p^2)$. For each network, we calculate its CRoC value by Algorithm 1, the number K of components, and the variance in its component sizes defined by $\text{Var} = \frac{\sum_{i=1}^K (n_i - \bar{n})^2}{K}$ where n_i is the size of component i and $\bar{n}$ is their mean value. Fig. 6 and Fig. 7 show the results of this study, including the relationship between CRoC and the number of components, as well as between CRoC and the variance of component sizes, respectively. The results suggest that graphs with a greater number of components generally exhibit higher CRoC values. Additionally, increased level of symmetry among components tends to result in larger CRoC values. This suggests we may use it in the explainer (see the next section).

To end this section, we compare the CRoC metric obtained from Algorithm 1 with several existing approximations for graph connectedness. Connected-CF² [20] introduced an approximation for graph connectedness as

$$\det(L(G)) + \exp(-\theta \sum_{i,j=1}^n C_{ij}^2), \quad (4)$$

where $\det(\cdot)$ denotes the determinant of a matrix and C_{ij} denotes the **first minor** (i.e., the determinant of the submatrix by subtracting the i -th row and j -th column from the original matrix) of $L(G)$. When this approximation converges to zero,

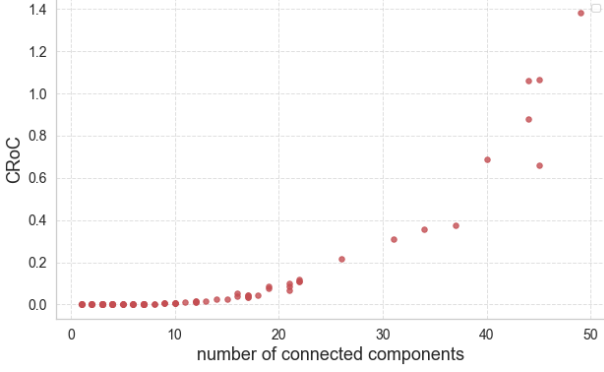


Fig. 6. Correlations between CRoC (calculated by Algorithm 1) and the number of components for random Erdős-Rényi networks.

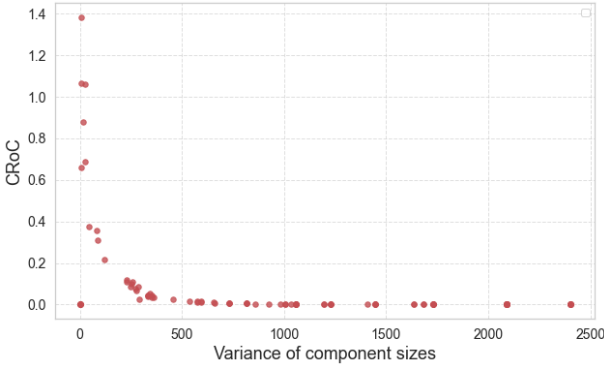


Fig. 7. Correlations between CRoC (calculated by Algorithm 1) and the variance of component sizes for random Erdős-Rényi networks.

meaning that $\det(L(G)) \rightarrow 0$ and $\exp(-\theta \sum_{i,j=1}^n C_{ij}^2) \rightarrow 0$ (note that both terms are non-negative), it follows that $L(G)$ has determinant of 0 and at least one non-zero first minor. This implies $\text{rank}(L(G)) = n - 1$. From Theorem 1, we can conclude that G is connected. To compute (4), we need to calculate one determinant of dimension n and n^2 determinants of dimension $n - 1$. Thus, the computational complexity is $O(n^3 + n^2 \times (n - 1)^3) = O(n^5)$. With such high computational cost, (4) can only evaluate whether a graph is connected, which can be achieved by CRoC at zero value with a significantly lower complexity of $O(n^3)$.

Another approximation is a variant from heat kernel [27] to approximate the number of components in graphs by

$$\sum_{i=1}^n \exp\left(-\frac{\lambda_i^2}{\sigma^2}\right), \quad (5)$$

where λ_i are the eigenvalues of the Laplacian matrix $L(G)$, σ is a small positive parameter controlling the sensitivity to eigenvalues near zero. When σ is sufficiently small, (5) is only sensitive to near-zero eigenvalues. In this case, when $\lambda_i \rightarrow 0$, $\exp(-\frac{\lambda_i^2}{\sigma^2}) \rightarrow 1$; otherwise, $\exp(-\frac{\lambda_i^2}{\sigma^2}) \rightarrow 0$. Therefore, if σ is carefully selected,

$$\text{Number of Components} \approx \sum_{i=1}^n \exp\left(-\frac{\lambda_i^2}{\sigma^2}\right).$$

Note that (5) requires computing eigenvalues, which has the same computational complexity as CRoC, namely $O(n^3)$. However, CRoC offers additional capabilities, such as evaluating the asymmetry between different components.

In summary, compared to existing approximations, CRoC not only surpasses their computational efficiency but also provides enhanced capabilities. Therefore, we consider CRoC is a more versatile and powerful tool for analyzing graph structures, and will use it in the proposed explainer framework.

III. EXPLAINER FRAMEWORK CROCEXPLAINER

Given a GNN model $f_\theta : \mathcal{G} \rightarrow \mathcal{F}$ and a graph $G \in \mathcal{G}$, we consider an explanation for them as a subgraph G' of G such that $f_\theta(G') \approx f(G)$. Following Tan et al. [18], we also handle an explanation by an edge mask matrix $M \in \{0, 1\}^{n \times n}$, i.e., the spanning subgraph G_M of G with adjacency matrix $A(G_M) = M \odot A(G)$ is an explanation for f_θ and G , where $\odot$ denotes element-wise multiplication.

An exact definition of $f_\theta(G') \approx f(G)$ is called a reasoning. In the literature, factual reasoning and counterfactual reasoning are widely used. The former tries to find sufficient explanations, whereas the latter tries to find necessary explanations. More precisely, factual reasoning finds an edge mask matrix M such that $f_\theta(G_M)$ is as close as possible to $f(G)$, whereas counterfactual reasoning finds M such that $f_\theta(G_{\bar{M}})$ is as far as possible from $f(G)$, where $\bar{M} = \mathbf{1}^{n \times n} - M$ denotes the edge mask matrix inverse to M . Both have their pros and cons. While most explainer models use one of them, Tan et al. [18] reported that combining both excels others that use only one. Following their framework, we also use both reasonings.

We want to find compact, effective and connected explanations. Following [18], we define the compactness of an explanation M by $\|M\|_0$, i.e., the number of non-zero values in M . To define the effectiveness, we need more details on GNN. In general, a GNN model outputs a feature $\hat{y} \in \mathcal{F}$ that maximizes the probability P_θ for a given graph G , i.e.,

$$\hat{y} = \arg \max_{y \in \mathcal{F}} \{P_\theta(y|A(G))\},$$

where P_θ can be obtained by a softmax layer before the output. Tan et al. [18] calls M an effective factual reasoning if

$$P_\theta(\hat{y}|A(G_M)) > \max_{y \in \mathcal{F}, y \neq \hat{y}} P_\theta(y|A(G_M)),$$

and an effective counterfactual reasoning if

$$P_\theta(\hat{y}|A(G_{\bar{M}})) < \min_{y \in \mathcal{F}, y \neq \hat{y}} P_\theta(y|A(G_{\bar{M}})).$$

For estimating the connectedness, we use the output of Algorithm 1 applied to G_M , denoted by $\text{CRoC}(M)$. Ideally we want to find an M^* that solves the following problem.

$$\begin{aligned} (P) \quad & \text{minimize} \quad \|M\|_0 \\ & \text{s.t., } P_\theta(\hat{y}|A(G_M)) > \max_{y \in \mathcal{F}, y \neq \hat{y}} P_\theta(y|A(G_M)), \\ & P_\theta(\hat{y}|A(G_{\bar{M}})) < \min_{y \in \mathcal{F}, y \neq \hat{y}} P_\theta(y|A(G_{\bar{M}})), \\ & \text{CRoC}(M) = 0. \end{aligned}$$

This optimization problem is difficult to solve due to its combinatorial nature [18]. We relax it to a continuous optimization problem instead. For the objective function, we relax the domain of the elements of M from $\{0, 1\}$ to $[0, 1]$. Accordingly, we substitute the L0-norm $\|M\|_0$ of M with the differential $\|M\|_2$. Compared with the L1-norm applied by Tan et al. [18], which is non-differentiable at zero, L2-norm is differentiable everywhere. For the reasoning constraints, we follow [18] to relax them with

$$L_f = \text{ReLU}(\zeta - (P_\theta(\hat{y}|A(G_M)) - \max_{y \in \mathcal{F}, y \neq \hat{y}} P_\theta(y|A(G_M))))$$

and

$$L_c = \text{ReLU}(\zeta - (\min_{y \in \mathcal{F}, y \neq \hat{y}} P_\theta(y|A(G_{\bar{M}})) - P_\theta(\hat{y}|A(G_{\bar{M}}))))$$

respectively, where ζ is a pre-defined value (we used 0.5 in the empirical study). For the connectedness, CRoC is differential when M is relaxed to real-valued. Finally, we consider the Lagrangian Relaxation problem (P) as the follows:

$$(P') \text{ minimize } \|M\|_2 + \lambda(\alpha L_f + (1 - \alpha)L_c) + \beta \text{CRoC}(M)$$

Parameters λ and β adjust the trade-off between effectiveness and connectedness. Parameter α adjusts the trade-off between factual and counterfactual reasoning. To get an integer solution M^* to (P) from a relaxed solution M to (P'), we simply round the elements to 1 if it is greater or equal to 1/2, otherwise 0.

We remark that the proposed framework can adapt to node classification in a straightforward manner.

IV. EMPIRICAL STUDIES

In this section, we first introduce the datasets, followed by an overview of the baseline methods. Then we describe the experimental setup, and finally we present the experimental results along with discussion.

A. Datasets and Baselines

We use two synthetic datasets and one real-world dataset, all of which contain ground-truth labels. The synthetic datasets Tree-Cycles and BA-shapes were provided by [17]. The real-world dataset Mutag₀ is a subset of the Mutag dataset [28] extracted by Tan et al. [18]. It contains only mutagenic compounds with benzene-NO₂ and non-mutagenic compounds without benzene-NO₂. See Table I for the detail.

TABLE I
DATASETS USED IN THE EMPIRICAL STUDY.

Datasets	#graph	#avg node	#avg edge	#class	#feature	task
BA-shapes	1	700	4100	4	-	node
Tree-Cycles	1	871	1950	2	-	node
Mutag ₀	2301	31.74	32.54	2	14	graph

For the baselines, we want them to find explanations of subgraphs and do not have ‘‘critical’’ conditions on datasets, e.g., PGExplainer [29] requires extra motif information to generate reasoning. In addition, SubgraphX [19] is not included in the baselines due to its prohibitive computational cost. As the result, the following models are chosen.

- **GNNExplainer** [17]: It finds explanations by maximizing the mutual information.
- **CF-GNNExplainer** [30]: It is an extended version of GNNExplainer with counterfactual reasoning.
- **CF²** [18]: It considers both factual reasoning and counterfactual reasoning.
- **Connected-CF²** [20]: It adds connectedness consideration to CF².

B. Setup

The experiments consist of two phases: (1) training a fixed baseline GNN model for classification, and (2) solving the optimization problem to find corresponding explanations. For the baseline model, we use Graph Convolutional Network (GCN) [31]. GCN is a popular GNN with the following update rule:

$$\mathbf{h}_v^{(k+1)} = \sigma \left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\tilde{\mathbf{D}}_{vv} \tilde{\mathbf{D}}_{uu}}} \mathbf{h}_u^{(k)} \mathbf{W}^{(k)} \right),$$

where $\mathbf{h}_v^{(k)}$ is the feature of node v at layer k , $\sigma(\cdot)$ is the activation function, $\mathcal{N}(v)$ denotes the neighbors of node v , $\tilde{\mathbf{D}}$ is the degree matrix of the original graph adding a self-loop, i.e., $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$, and $\mathbf{W}^{(k)}$ is a learnable weight matrix at layer k . We use a fixed GCN as the base GNN model, which consists of three convolutional layers of hidden dimension 16. After each layer, except for the final one, we apply the ReLU activation function, and a softmax layer is used before the final classification. The learning rate is set to 0.001, with the training set comprising 80% of the total dataset. Table II provides detailed information on the base GCN model used in the experiments.

TABLE II
DETAIL ON THE TRAINED BASIC GCN.

Datasets	BA-Shapes	Tree-Cycles	Mutag ₀
Epochs	3000	3000	1000
Accuracy (in %)	97.86	98.29	98.05

In the explanation phase, we keep all hyperparameters in CRoCExplainer consistent across datasets. The value of λ is set to 500 for BA-Shapes and Tree-Cycles, and 1000 for the Mutag₀ dataset. We set α to 0.6 to prioritize factual reasoning over counterfactual reasoning, and β is set to 100.

C. Results and Discussion

Since datasets Tree-Cycles, BA-Shapes, and Mutag₀ all contain ground-truth labels, we use common classification metrics Accuracy, Precision, Recall, and F_1 scores to evaluate the explanations on the test datasets for different methods. **Accuracy, Precision, Recall and F_1 are defined as follows:**

$$\text{Accuracy} = \frac{TP + TF}{TP + FP + TN + FN},$$

$$\text{Precision} = \frac{TP}{TP + FP},$$

$$\text{Recall} = \frac{TP}{TP + FN},$$

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

here, TP , TN , FP and FN correspond to an edge both in the given explanation and the ground-truth, an edge neither in the explanation nor in the ground-truth, an edge in the explanation while not in the ground-truth, and an edge not in the explanation while in the ground-truth, respectively. Note that Accuracy, Precision, Recall, and F_1 are computed separately for each individual explanation, we extend these metrics in **bold** form to evaluate model performance across entire datasets. Specifically, we calculate dataset-level aggregates by taking the mean values across all instances:

$$\text{Accuracy} = \frac{1}{|E|} \sum_{e \in E} \text{Accuracy}(e),$$

$$\text{Precision} = \frac{1}{|E|} \sum_{e \in E} \text{Precision}(e),$$

$$\text{Recall} = \frac{1}{|E|} \sum_{e \in E} \text{Recall}(e),$$

$$F_1 = \frac{1}{|E|} \sum_{e \in E} F_1(e),$$

where $e \in E$ denotes an explanation generated by the Explainer for a graph G in the target dataset D , and $|E|$ represents the total number of explanations. This aggregation strategy provides a unified performance measure across all input instances.

Table III presents the results of the explanations for each dataset in comparison to the ground truth. Across all datasets, CRoCExplainer demonstrates high **Accuracy** and **F_1** scores, highlighting its strong and robust explanatory capability. Notably, in Tree-Cycles dataset, CRoCExplainer outperforms other methods. Moreover, CRoCExplainer often achieves a higher **Recall** value, indicating its tendency to provide explanations that align closely with ground-truth data (see Fig. 8). This performance results from CRoCExplainer’s consideration of connectedness in its explanations.

Finally, we provide a running time comparison between CRoCExplainer and Connected-CF². The experiments were conducted on the same machine with an AMD Ryzen 7 5800H processor, 32 GB DDR4 RAM and an NVIDIA GeForce RTX 3060 GPU. As shown in Table IV, CRoCExplainer significantly reduces computational complexity compared to Connected-CF².

V. CONCLUSION AND FUTURE WORK

In this article, we introduce a novel metric CRoC to measure the connectedness in a graph, which can be optimized using continuous optimization methods. Based on this metric, we propose CRoCExplainer to generate explanations for GNN predictions. Empirical results demonstrate that CRoCExplainer provides highly effective explanations with greater efficiency, highlighting the effectiveness of the CRoC metric.

There are also some limitations, including lacking a theoretical study on the trade-off between connectedness and asymmetry in CRoC. Additionally, CRoCExplainer focuses on

interpreting the output of GNNs rather than directly analyzing and explaining the internal mechanisms of GNNs. Furthermore, the current implementation of CRoCExplainer does not fully leverage the potential of CRoC. In the future, we plan to investigate the trade-off properties of CRoC more deeply and explore additional variants.

REFERENCES

- [1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, fourth edition edition, 2022.
- [2] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.
- [3] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.
- [4] William L Hamilton. *Graph representation learning*. Springer, 2020.
- [5] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021.
- [6] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [7] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 401–417, 2018.
- [8] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 2018.
- [9] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.
- [10] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI’18*, page 3634–3640. AAAI Press, 2018.
- [11] Gianluca Bonifazi, Bernardo Breve, Stefano Cirillo, Enrico Corradini, and Luca Virgili. Investigating the covid-19 vaccine discussions on twitter through a multilayer network-based approach. *Information Processing & Management*, 59(6):103095, 2022.
- [12] Gianluca Bonifazi, Francesco Cauteruccio, Enrico Corradini, Michele Marchetti, Giorgio Terracina, Domenico Ursino, and Luca Virgili. A framework for investigating the dynamics of user and community sentiments in a social platform. *Data & Knowledge Engineering*, 146:102183, 2023.
- [13] Federico Baldassarre and Hossein Azizpour. Explainability techniques for graph convolutional networks. In *International Conference on Machine Learning (ICML) Workshops, 2019 Workshop on Learning and Reasoning with Graph-Structured Representations*, 2019.
- [14] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10772–10781, 2019.
- [15] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMIR, 2017.
- [16] Hao Yuan, Haiyang Yu, Shirui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5782–5799, 2022.
- [17] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, pages 9240–9251, 2019.
- [18] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM Web Conference 2022, WWW ’22*, page 1018–1027, New York, NY, USA, 2022. Association for Computing Machinery.

TABLE III

COMPARISON OF DIFFERENT METHODS, WHERE *Acc*, *Pre*, *Re* AND *F₁* DENOTE THE DATASET-LEVEL METRICS *Accuracy*, *Precision*, *Recall* AND *F₁*, RESPECTIVELY, IN %.

Models	BA-Shapes				Tree-Cycles				Mutag ₀			
	<i>Acc</i> %	<i>Pr</i> %	<i>Re</i> %	<i>F₁</i> %	<i>Acc</i> %	<i>Pr</i> %	<i>Re</i> %	<i>F₁</i> %	<i>Acc</i> %	<i>Pr</i> %	<i>Re</i> %	<i>F₁</i> %
GNNExplainer	95.25	60.08	60.08	60.08	92.78	68.06	68.06	68.06	96.96	59.71	85.17	68.85
CF-GNNExplainer	94.39	67.19	54.11	56.79	90.27	87.40	47.45	59.10	96.91	66.09	39.46	47.39
CF ²	96.37	73.15	68.18	66.61	93.26	84.92	73.84	75.69	97.34	65.28	88.59	72.56
Connected-CF ²	95.22	56.73	81.51	62.00	93.76	68.71	97.45	78.60	96.54	49.87	96.30	63.45
CRoCEXplainer	94.99	58.33	80.14	62.28	93.79	69.37	96.53	78.62	96.53	49.7	95.37	63.18

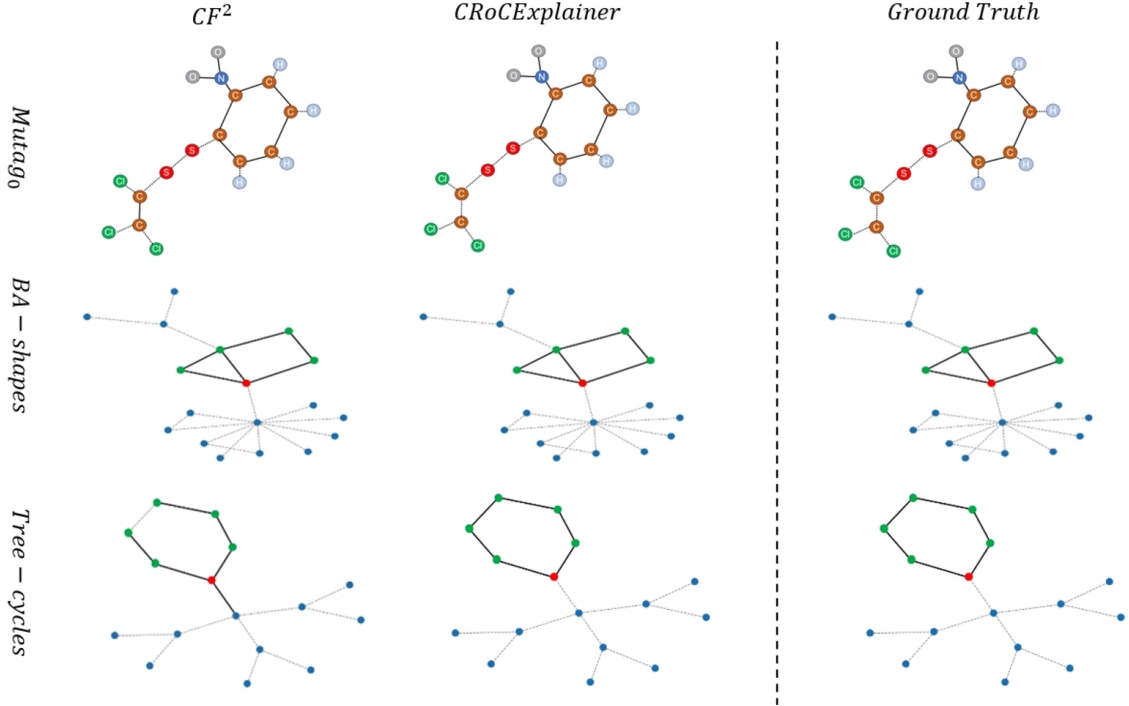


Fig. 8. An example of the explanations (in bold edges) tested on the Mutag₀, BA-shapes, and Tree-cycles datasets. While CF² produces a more disconnected subgraph, CRoCEXplainer tends to find the connected one.

TABLE IV
RUNNING TIME(S) COMPARISON BETWEEN CROCEXPLAINER AND CONNECTED-CF² ON DIFFERENT DATASETS.

Datasets	BA-Shapes	Tree-Cycles	Mutag ₀
Connected-CF ²	13572.9	1330.4	1647.7
CRoCEXplainer	3799.7	1051.1	480.7

- [19] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pages 12241–12252. PMLR, 2021.
- [20] Yanghepu Li. Connected-cf2: Learning the explainability of graph neural network on counterfactual and factual reasoning via connected component. In *International Conference on AI-generated Content*, pages 83–94. Springer, 2023.
- [21] William N Anderson Jr and Thomas D Morley. Eigenvalues of the laplacian of a graph. *Linear and multilinear algebra*, 18(2):141–145, 1985.
- [22] Dragoš M Cvetković, Peter Rowlinson, and Slobodan Simic. *Eigenspaces of graphs*. Number 66. Cambridge University Press, 1997.
- [23] Türker Bıyıkoglu, Josef Leydold, and Peter F Stadler. Laplacian eigenvectors of graphs. *Lecture notes in mathematics*, 1915, 2007.
- [24] Gilbert Strang. *Linear Algebra and Its Applications*. Academic Press, Inc., Orlando, FL, 2nd edition, 1980.
- [25] Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci.*, 5(1):17–60, 1960.
- [26] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.
- [27] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
- [28] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.
- [29] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.
- [30] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4499–4511. PMLR, 2022.
- [31] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR '17*, 2017.