CoSG: Counterfactual Explainers for Signed Graphs

Anonymous authors
Paper under double-blind review

Abstract

Signed graph neural networks (SGNNs) have shown strong performance in modeling positive and negative relations in signed graphs. However, their black-box nature limits the understanding of prediction behavior and raises concerns about model trustworthiness. Counterfactual explanations (CEs) provide actionable recourse and reveal causal dependencies by identifying minimal input perturbations that would alter model predictions. Despite their promise, CEs remain largely unexplored for signed graphs, where predictions depend jointly on graph structure and relation polarity. To bridge this gap, we first study the problem of generating counterfactual explanations for signed graphs. This problem is particularly challenging because it requires optimizing over a discrete and heterogeneous perturbation space to identify sparse yet effective explanations while enabling the trained explainer to generalize to unseen instances. To address these challenges, we propose CoSG, a novel model-agnostic explainer that generates counterfactual explanations for link sign prediction on signed graphs. Specifically, CoSG decouples edge polarity from edge existence to enable differentiable sign-aware perturbations and trains the explainer across instances under a unified objective that balances counterfactual validity and perturbation sparsity, allowing CoSG to efficiently generate concise, actionable explanations for unseen instances. Extensive experiments on real-world signed graphs demonstrate the superiority of CoSG over state-of-the-art graph explainers in generating counterfactual explanations.

1 Introduction

Many real-world networks exhibit polarized interactions between entities, such as friend-foe ties in social networks (Bonchi et al., 2019; Tzeng et al., 2020) and cooperative-antagonistic relations in biological networks (Ou-Yang et al., 2015). Signed graphs provide a natural representation for such networks by encoding the existence of relationships and their polarity through positive and negative edges. In recent years, signed graph neural networks (SGNNs) (Derr et al., 2018; Li et al., 2020; Nakis et al., 2024; Wang et al., 2025) have been developed to capture these polarized interactions through sign-aware message passing, enabling effective node representations for applications such as recommendation systems (Lian et al., 2026), fraud detection (Xue et al., 2026), and drug discovery (Jin et al., 2024). However, despite their effectiveness, SGNNs operate as black boxes, providing limited insight into how signed relations lead to model predictions and raising concerns about the transparency and trustworthiness of model decisions (Cheong, 2024).

To facilitate model interpretability, many explainable graph neural network (XGNN) methods (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2020; Gao et al., 2026) have been developed to identify a small subset of inputs (e.g., nodes, edges, or features) that are most influential in determining model predictions. However, most existing XGNN methods (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2020) are designed for unsigned graphs and thus do not model the distinctive interactions between positive and negative edges, leading to explanations that may fail to capture important signed structures. Moreover, these methods primarily target node- or graph-level predictions, whereas signed graph analysis often centers on link sign prediction (Fang et al., 2024), which requires explaining the predicted polarity of a potential edge between two nodes. These misalignments in graph semantics and explanation objectives limit the effectiveness of existing XGNN methods on signed graphs. Among existing studies, the closest related work (Gao et al., 2026) explains link sign prediction by restricting the explanatory structure to balanced paths that support the predicted edge and its sign.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

However, while these factual explanation methods (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2020; Gao et al., 2026) identify sufficient input evidence that supports the current predictions, they do not guarantee the necessity of such evidence, i.e., removing such evidence from the input may still leave the prediction unchanged.

Counterfactual explanations (CEs) (Lucic et al., 2022; Tan et al., 2022; Qu et al., 2024; Ma et al., 2022), another class of graph explainability methods, directly address this limitation by identifying minimal perturbations to the input graph that would change the model prediction. Compared with factual explanations, CEs provide a more actionable view of model behavior by revealing possible recourse, exposing causal dependencies, and clarifying decision boundaries (Qu et al., 2024; Prado-Romero et al., 2024). For example, Fig. 1 shows a financial risk assessment model built on a signed network, where positive and negative edges denote compliant and suspicious relationships. For a prospective business relationship predicted as high-risk in the original graph (Fig. 1(a)), removing the identified factual evidence from the input does not necessarily change the negative prediction (Fig. 1(b)), because the remaining signed path may still provide alternative support for the same high-risk outcome. In contrast, a counterfactual explanation (Fig. 1(c)) reveals which minimal changes to the network, such as edge removal or sign flip, lead to a different model prediction. These perturbations may correspond to realistic compliance interventions, such as deleting outdated records or resolving an investigation. However, when extended to signed graphs, existing CE methods (Lucic et al., 2022; Tan et al., 2022; Qu et al., 2024; Ma et al., 2022) still do not account for the distinctive sign interactions and task-specific explanation objectives of signed graphs discussed above. To the best of our knowledge, no prior work has investigated counterfactual explanations for signed graphs.

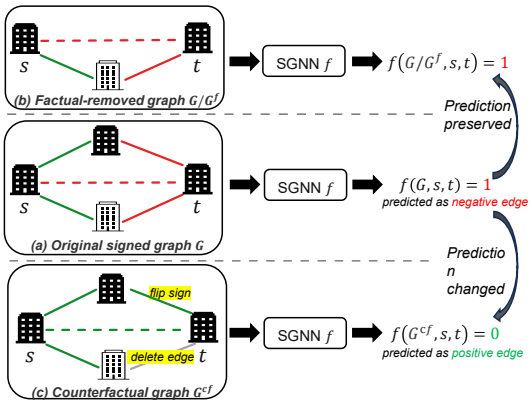


Figure 1: An example of CEs on signed graphs. Green/red lines denote positive/negative edges, and dotted lines denote predicted links.

To fill this gap, we first study the problem of generating counterfactual explanations for signed graphs. Specifically, we focus on explaining link sign prediction in a post-hoc setting, where SGNNs are pre-trained and fixed, without access to model internals during explanation. This problem presents three main challenges: 1) **Complex Perturbation.** Since SGNN predictions depend on both edge existence and edge polarity, CE generation on real signed graphs requires searching over a discrete, large, and heterogeneous perturbation space involving edge removals and sign flips, making the optimization combinatorial and difficult to solve. 2) **Sparse yet Effective Explanation.** A desirable counterfactual explanation should be concise yet effective, requiring a careful trade-off between minimizing the number of perturbations and ensuring that the resulting CEs lead to a different model prediction. 3) **Generalization.** An effective explainer should generalize to unseen instances for scalable explanation generation in practice. However, the discrete topology and polarity-aware edge semantics of signed graphs make it challenging for CE methods to generalize to unseen instances.

To address these challenges, we propose **CoSG**, a novel Counterfactual explainer for Signed Graphs. CoSG is a model-agnostic explainer compatible with popular SGNNs, designed to generate CEs that alter model predictions with just slight perturbations to the input graph. Specifically, to promote generalization on new instances, we employ a multilayer perceptron (MLP)-based explainer that learns a perturbation policy by capturing instance-level patterns and the decision boundary of the target model during training. Once trained, CoSG can efficiently generate CEs for unseen instances without performing instance-specific optimization from scratch. To enable differentiable perturbations on signed graphs, CoSG decouples edge polarity from edge existence and relaxes the corresponding binary variables into continuous-valued masks. Rather than directly optimizing the discrete input graph, CoSG trains an explainer via gradient descent to output these perturbation masks, enabling end-to-end differentiable counterfactual generation. To encourage sparse yet effective counterfactual explanations, CoSG introduces a learning objective that balances counterfactual effectiveness with perturbation sparsity. Trained under this unified objective, CoSG generates concise and actionable explanations that alter model predictions with minimal graph perturbations.

The main contributions of this work can be summarized as follows: 1) **Problem.** We investigate an underexplored problem: generating counterfactual explanations for signed graphs. To the best of our knowledge, this is the first work to study this problem. 2) **Method.** We propose CoSG, a novel counterfactual explainer for signed graphs, which generates concise and actionable counterfactual explanations through minimal sign-aware perturbations. 3) **Experiments.** We conduct extensive experiments on real-world signed graphs to demonstrate the superiority of CoSG over state-of-the-art graph explainers on CE generation. The implementation of CoSG is available at `https://anonymous.4open.science/r/ICLR_CoSG`.

2 Preliminaries and Problem Formulation

Signed Graph and SGNN. A signed graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$, where $\mathcal{V}$ denotes the node set, and $\mathcal{E}^+$ and $\mathcal{E}^-$ denote the sets of positive and negative edges, respectively. Link sign prediction is one of the most fundamental tasks in signed graph analysis and serves as the primary design objective for most SGNN architectures (Derr et al., 2018; Li et al., 2020). Accordingly, SGNNs are commonly formulated to learn node representations that support predicting the relationship type between a target node pair. Given a signed graph together with node features X, an SGNN first encodes the graph into node embeddings $\mathcal{Z}$ via $\mathcal{Z} = f_{\text{enc}}(\mathcal{G}, X)$. For a target node pair (s, t), a prediction head maps the concatenated node embeddings to a three-class logit vector, $\boldsymbol{\ell} = f_{\text{pred}}([z_s \| z_t]) \in \mathbb{R}^3$, where the three logits correspond to the positive, negative, and non-existent edge classes, respectively. For notational simplicity, we denote a target pair by $i = (s, t)$ and the overall SGNN model by $f = f_{\text{pred}} \circ f_{\text{enc}}$. The final predicted label is then given by $\hat{y}_i = \arg\max_c \boldsymbol{\ell}_c$. In this paper, we study link sign prediction, following the dominant task setting for SGNNs, as it directly captures the polarity information that characterizes signed graphs.

Counterfactual Explanation for SGNN. In this work, we consider a post-hoc explanation setting for a trained SGNN, where the trained model parameters are fixed and we can access the prediction as well as the encoder-produced node embeddings for target instances. For a target instance $i = (s, t)$, an $\mathcal{L}$-layer SGNN $f(\cdot)$ computes the embeddings of s and t by aggregating messages from their $\mathcal{L}$-hop neighbors. Thus, under the standard message-passing assumption, the prediction $\hat{y}_i = f(\mathcal{G}, (s, t))$ is fully determined by the $\mathcal{L}$-hop ego-graph induced by the target pair (s, t), denoted as $\mathcal{G}_i$. Equivalently, we have $\hat{y}_i = f(\mathcal{G}, (s, t)) \equiv f(\mathcal{G}_i, (s, t))$. Therefore, we restrict the counterfactual search space to the computation graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i^+, \mathcal{E}_i^-)$, where $\mathcal{V}_i = \{v \in \mathcal{V} \mid \text{dist}(v, s) \leq \mathcal{L} \vee \text{dist}(v, t) \leq \mathcal{L}\}$, and $\mathcal{E}_i^+$ and $\mathcal{E}_i^-$ denote the positive and negative edges induced by $\mathcal{V}_i$, respectively.

We focus on *structural* perturbations and do not consider feature perturbations, because signed graph datasets typically lack informative node features, and the synthetic features commonly used in prior work (Derr et al., 2018; Li et al., 2020) do not support meaningful feature-level explanations. Unlike unsigned graphs, structural perturbations on signed graphs involve not only edge removal but also edge sign flipping. Therefore, a counterfactual explanation should identify sparse structural changes within $\mathcal{G}_i$ that are sufficient to alter the original prediction. Formally, a valid counterfactual explanation $\mathcal{G}_i^{cf}$ should remain close to the original computation graph $\mathcal{G}_i$ while changing the model prediction, i.e., $\hat{y}_i^{cf} = f(\mathcal{G}_i^{cf}, (s, t))$ and $\hat{y}_i^{cf} \neq \hat{y}_i$. In this work, we consider *untargeted* counterfactual explanations, where the perturbed prediction only needs to differ from the original one. The framework can also be extended to the targeted setting by specifying a desired label. Based on the above setting, we define the problem of *counterfactual explanation generation for signed graphs* as follows:

Problem 1 (Counterfactual Explanation Generation for Signed Graphs)**.** *Given a trained SGNN model $f(\cdot)$, a set of target instances $\mathcal{I}$, and the corresponding computation graph $\mathcal{G}_i$ for each instance $i = (s, t) \in \mathcal{I}$, the goal is to identify counterfactual graph $\mathcal{G}_i^{cf}$ for each target instance by applying minimal structural perturbations to $\mathcal{G}_i$ such that the model prediction changes:*

$$\mathcal{G}_i^{cf} = \arg \min_{\tilde{\mathcal{G}}_i \in \Omega(\mathcal{G}_i)} \mathcal{P}(\tilde{\mathcal{G}}_i, \mathcal{G}_i) \quad \textit{s.t.} \quad f(\mathcal{G}_i^{cf}, (s, t)) \neq f(\mathcal{G}_i, (s, t)),$$

where $\tilde{\mathcal{G}}_i$ denotes a candidate perturbed graph of $\mathcal{G}_i$, $\Omega(\mathcal{G}_i)$ denotes the feasible structural perturbation space over $\mathcal{G}_i$, induced by allowable edge removals and sign flips, and $\mathcal{P}(\tilde{\mathcal{G}}_i, \mathcal{G}_i)$ measures the perturbation cost, e.g., the number of edge removals and/or sign flips.

The resulting counterfactual explanation consists of the perturbation set and its induced counterfactual graph, which together highlight the critical signed structural evidence underlying the model prediction.

3 THE COSG EXPLAINER

In this section, we present CoSG, a post-hoc counterfactual explanation framework for SGNNs. For a target link instance, CoSG aims to identify minimal sign-aware perturbations within the local computation graph that change the model's original prediction. To support CEs on signed graphs, CoSG explicitly models edge existence and edge polarity, allowing each candidate edge to be removed, sign-flipped, or preserved. Fig. 2 provides an overview of the proposed framework, and the detailed training and inference procedure is summarized in Algorithm 1 in Appendix B.

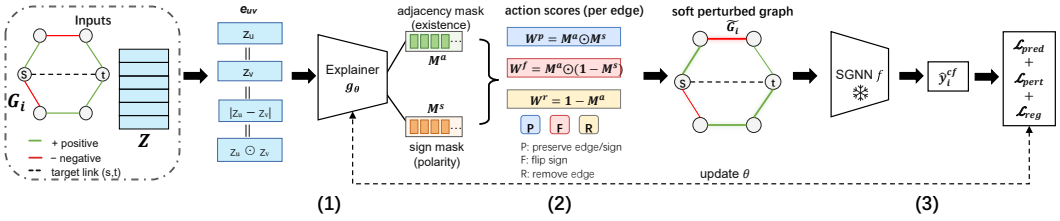


Figure 2: Overview of the proposed CoSG explainer. (1) Explanation network: Given the local computation graph $\mathcal{G}_i$ of a target link instance $i = (s, t)$, CoSG queries the frozen SGNN for node embeddings Z, constructs edge representations $\mathbf{e}_{uv}$, and feeds them into an MLP-based explainer g_θ to produce an adjacency mask $\mathbf{M}^a$ for edge existence and a sign mask $\mathbf{M}^s$ for sign polarity. (2) Soft perturbation: The two masks are converted into soft action scores for preserving, sign-flipping, or removing edges, inducing a differentiable perturbed graph $\widetilde{\mathcal{G}}_i$, which is fed into SGNN to get the perturbed prediction. (3) Learning objective: The explainer is optimized with a counterfactual objective that encourages prediction change while promoting perturbation sparsity and mask discreteness.

3.1 SIGN-AWARE COUNTERFACTUAL EXPLANATION GENERATION

Dual-mask Explainer Network. For each target instance $i = (s, t)$, let $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i^+, \mathcal{E}_i^-)$ denote the computation graph used by the SGNN to predict the target link. CoSG restricts the counterfactual search space to the observed signed edges, i.e., $\mathcal{E}_i = \mathcal{E}_i^+ \cup \mathcal{E}_i^-$, which lie within the receptive field of the target prediction and directly contribute to the SGNN decision. Each candidate edge in $\mathcal{E}_i$ can be removed, sign-flipped, or retained.

To efficiently search this structured perturbation space, CoSG uses an MLP-based explainer g_θ to learn an amortized perturbation policy across target instances. Given $\mathcal{G}_i$, CoSG first queries the frozen SGNN to obtain the node embeddings $\mathbf{Z}$ and the original predicted label $\hat{y}_i$, where $\hat{y}_i$ is used as the reference class in the counterfactual objective. For each candidate edge $(u, v) \in \mathcal{E}_i$, CoSG construct edge representations as:

$$\mathbf{e}_{uv} = [\mathbf{z}_u \,\|\, \mathbf{z}_v \,\|\, |\mathbf{z}_u - \mathbf{z}_v| \,\|\, \mathbf{z}_u \odot \mathbf{z}_v], \qquad \mathbf{R}_i = \{\mathbf{e}_{uv} \mid (u, v) \in \mathcal{E}_i\} \in \mathbb{R}^{|\mathcal{E}_i| \times d_e},$$

where $\|$ denotes concatenation, $\odot$ denotes element-wise multiplication, $\mathbf{R}_i$ is the edge-representation matrix for all candidate edges, and d_e denotes the dimension of each edge representation.

The edge-representation matrix $\mathbf{R}_i$ is passed into g_θ, followed by two sigmoid heads, to produce two continuous masks over candidate edges: an adjacency mask $\mathbf{M}_i^a$ for edge retention/removal and a sign mask $\mathbf{M}_i^s$ for sign preservation/flipping:

$$\mathbf{H}_i = g_\theta(\mathbf{R}_i), \qquad \mathbf{M}_i^a = \sigma(\mathbf{H}_i \mathbf{w}_a + b_a), \qquad \mathbf{M}_i^s = \sigma(\mathbf{H}_i \mathbf{w}_s + b_s),$$

where $\mathbf{H}_i \in \mathbb{R}^{|\mathcal{E}_i| \times d_h}$ denotes the hidden edge representations, d_h is the hidden dimension, $\mathbf{w}_a, \mathbf{w}_s \in \mathbb{R}^{d_h}$, $b_a, b_s \in \mathbb{R}$, $\sigma(\cdot)$ denotes the sigmoid function, and $\mathbf{M}_i^a, \mathbf{M}_i^s \in [0, 1]^{|\mathcal{E}_i|}$. Larger values in $\mathbf{M}_i^a$ indicate higher tendencies to retain the corresponding edges, whereas larger values in $\mathbf{M}_i^s$ indicate strong tendencies to preserve their original signs rather than flip them.

Soft Signed Graph Perturbation. To make counterfactual perturbations differentiable, CoSG introduces a soft perturbation mechanism that relaxes the discrete actions of edge preservation, sign flipping, and removal into continuous action scores. Given the adjacency mask $\mathbf{M}_i^a$ and sign mask $\mathbf{M}_i^s$, CoSG defines three soft action-score vectors:

$$\mathbf{W}_i^p = \mathbf{M}_i^a \odot \mathbf{M}_i^s, \qquad \mathbf{W}_i^f = \mathbf{M}_i^a \odot (1 - \mathbf{M}_i^s), \qquad \mathbf{W}_i^r = 1 - \mathbf{M}_i^a. \tag{1}$$

where $\mathbf{W}_i^p$, $\mathbf{W}_i^f$, and $\mathbf{W}_i^r$ correspond to edge preservation with the original sign, edge-sign flipping, and edge removal, respectively. This formulation provides a differentiable relaxation of the discrete perturbations and enables gradient-based optimization of the explainer.

We then use these action scores to construct the soft perturbed computation graph $\widetilde{\mathcal{G}}_i = (\mathcal{V}_i, \widetilde{\mathcal{E}}_i^+, \widetilde{\mathcal{E}}_i^-)$, where $\widetilde{\mathcal{E}}_i^+$ and $\widetilde{\mathcal{E}}_i^-$ denote the weighted positive and negative edge sets, respectively:

$$\begin{aligned} \widetilde{\mathcal{E}}_i^+ &= \left\{((u,v), W_{i,uv}^p) \mid (u,v) \in \mathcal{E}_i^+\right\} \cup \left\{((u,v), W_{i,uv}^f) \mid (u,v) \in \mathcal{E}_i^-\right\}, \\ \widetilde{\mathcal{E}}_i^- &= \left\{((u,v), W_{i,uv}^p) \mid (u,v) \in \mathcal{E}_i^-\right\} \cup \left\{((u,v), W_{i,uv}^f) \mid (u,v) \in \mathcal{E}_i^+\right\}. \end{aligned} \tag{2}$$

Here, $W_{i,uv}^p$ and $W_{i,uv}^f$ denote the entries of $\mathbf{W}_i^p$ and $\mathbf{W}_i^f$ associated with edge (u,v), representing the soft weights for preserving edges or flipping the original sign, respectively. Under this construction, preserving an edge keeps it in its original signed edge set, whereas flipping an edge moves it to the opposite signed edge set. During explainer training, the frozen SGNN performs message passing on the soft graph $\widetilde{\mathcal{G}}_i$, using $\mathbf{W}_i^p$ and $\mathbf{W}_i^f$ as differentiable edge weights for the corresponding edge sets. Therefore, the soft graph perturbation is differentiable with respect to the explainer outputs, enabling CoSG to optimize the explainer parameters via gradient descent.

Discrete Explanation Generation. After training, CoSG generates counterfactual explanations for unseen target links by discretizing the learned soft action scores. Given a target instance $i = (s,t)$, CoSG queries the frozen SGNN for the prediction label and performs a single forward pass through the trained explainer to obtain the action-score vectors $\mathbf{W}_i^p$, $\mathbf{W}_i^f$, and $\mathbf{W}_i^r$. The discrete perturbation-action vector $\mathbf{P}_i$ is obtained by selecting the highest-scoring action for each candidate edge:

$$\mathbf{P}_i = \left\{ P_{i,uv} = \arg\max_{a \in \{p,f,r\}} W_{i,uv}^a \mid (u,v) \in \mathcal{E}_i \right\}, \qquad \mathcal{G}_i^{cf} = \mathbf{P}_i \odot \mathcal{G}_i.$$

Here, $W_{i,uv}^a$ denotes the entry of $\mathbf{W}_i^a$ associated with edge (u,v), and $\odot$ denotes applying the decoded perturbation actions to the corresponding candidate edges. This decoding strategy avoids threshold tuning, and the final counterfactual explanation consists of the decoded perturbation-action vector $\mathbf{P}_i$ and its induced counterfactual graph $\mathcal{G}_i^{cf}$.

3.2 The Learning Objective

The desirable counterfactual explanations should be concise yet effective, requiring a careful trade-off between minimizing the number of perturbations and ensuring that the resulting CEs lead to different model prediction. In this subsection, we formulate the counterfactual objective of CoSG, which encourages the perturbed graph to change the original prediction while promoting sparse and discrete perturbations. Given a batch $\mathcal{B}$ of target instances, the overall objective is defined as

$$\mathcal{L}_{\text{batch}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{L}^{(i)}, \qquad \mathcal{L}^{(i)} = \mathcal{L}_{\text{pred}}^{(i)} + \mathcal{L}_{\text{pert}}^{(i)} + \mathcal{L}_{\text{reg}}^{(i)}$$

Prediction Loss. The prediction loss encourages the soft perturbed graph to alter the SGNN's original prediction. Given the soft perturbed graph $\widetilde{\mathcal{G}}_i$, CoSG minimizes the log-probability assigned to the original predicted class $\hat{y}_i$:

$$\mathcal{L}_{\text{pred}}^{(i)} = \log \frac{\exp(f_{\hat{y}_i}(\widetilde{\mathcal{G}}_i, (s,t)))}{\sum_c \exp(f_c(\widetilde{\mathcal{G}}_i, (s,t)))},$$

where $f_c(\tilde{\mathcal{G}}_i, (s,t))$ denotes the output logit of class c for target link (s,t) under the perturbed computation graph $\tilde{\mathcal{G}}_i$. Minimizing this loss reduces the frozen SGNN's confidence in its original prediction and thereby encourages the perturbed graph to cross the decision boundary.

Perturbation Loss. The perturbation loss encourages CoSG to generate sparse and interpretable perturbations by penalizing deviations from the original graph, namely edge removal and sign flipping:

$$\mathcal{L}_{\text{pert}}^{(i)} = \lambda_{\text{rem}} \frac{1}{|\mathcal{E}_i|} \sum (1 - \mathbf{M}_i^a) \; + \; \lambda_{\text{flip}} \frac{1}{|\mathcal{E}_i|} \sum (\mathbf{M}_i^a \odot (1 - \mathbf{M}_i^s)),$$

where $\sum(\cdot)$ denotes summation over all entries of the corresponding vector. The first term penalizes edge removal, while the second term penalizes sign flipping among retained edges. The hyperparameters λ_{rem} and λ_{flip} control the relative costs of edge removal and edge-sign flipping.

Regularization Loss. Since CoSG produces continuous masks during training but decodes them into discrete perturbation actions at inference, we use entropy regularization to encourage binary mask values. The regularization loss for instance i is defined as

$$\mathcal{L}_{\text{reg}}^{(i)} = \lambda_{\text{ent},a} \frac{1}{|\mathcal{E}_i|} \sum H(\mathbf{M}_i^a) + \lambda_{\text{ent},s} \frac{1}{|\mathcal{E}_i|} \sum H(\mathbf{M}_i^s),$$

where $H(\cdot)$ denotes the element-wise binary entropy function. The hyperparameters $\lambda_{\text{ent},a}$ and $\lambda_{\text{ent},s}$ control the strengths of entropy regularization for the adjacency and sign masks, respectively.

4 EXPERIMENT

In this section, we conduct experiments to answer the following research questions: **RQ1:** How effective is CoSG in generating high-quality counterfactual explanations compared with state-of-the-art baselines? **RQ2:** How efficiently does CoSG generate explanations compared with existing methods? **RQ3:** How do different components of CoSG contribute to its explanation performance? **RQ4:** How sensitive is CoSG to hyperparameter settings? **RQ5:** How can the counterfactual explanations generated by CoSG be qualitatively interpreted? Due to space limitations, we include the qualitative case study for **RQ5** in Appendix D.4.

4.1 EXPERIMENTAL SETUP

Datasets and SGNNs. We conduct experiments on three real-world signed networks: **BitcoinOTC**, **Twitter** and **Epinions**. Since these original datasets do not provide node features, we follow prior work (Derr et al., 2018; Li et al., 2020) and use spectral features for initialization. We use two popular SGNN models as explanation targets: **SGCN** (Derr et al., 2018) and **SNEA** (Li et al., 2020). For both models, we adopt a 2-layer message-passing architecture followed by a single-layer prediction head for link sign prediction, and train each model using the loss function and hyperparameter settings recommended in its original paper. All trained models achieve over 85% accuracy on every dataset, as shown in Table 5, providing a reliable basis for explanation evaluation. For SGNN training, we split positive and negative edges separately into train/validation/test sets with 70/10/20%. For explanation, CoSG and SGExplainer are trained on sampled target node pairs from the training split, while all methods are evaluated on the same sampled target node pairs from the test split. Detailed dataset statistics and model training settings are provided in Appendix C.1 and Appendix C.2.

Baseline Explainers. We adopt the following baselines for comparison: **1) Random**: for each instance, it randomly perturbs the local computational graph for at most T steps and stops once the prediction changes. **2) RandRM**: following the same procedure as Random, it only removes edges; **3) RandFlip**: following the same procedure as Random, it only flips edge signs. **4) GNNExplainer** (Ying et al., 2019): it learns an edge mask to identify important edges for the original prediction. We adapt it to CE generation by removing important edges with high mask scores. **5) CF-GNNExplainer** (Lucic et al., 2022): it is a counterfactual explainer originally proposed for node classification, where it searches for edge removals that change the model prediction. We adapt CF-GNNExplainer to explaining link sign prediction. **6) CF^2** (Tan et al., 2022): it jointly optimizes factual and counterfactual objectives to identify important structures as explanations. and **7) SGExplainer** (Gao et al., 2026): it is a factual explainer tailored to link sign prediction, which restricts explanations to balanced paths. We adapt it to CE generation by removing the identified important paths. Detailed implementation settings are provided in Appendix C.3.

Evaluation Metrics. We evaluate explanations in terms of effectiveness, sparsity, and efficiency. For effectiveness, **Validity** measures the proportion of instances whose predicted labels are changed by the counterfactual graph, while **Fidelity** measures the average decrease in the model confidence for the original prediction. **Sparsity** measures the average proportion of the local computational graph that remains unchanged after perturbation. For efficiency, **Avg. Time** measures the average time required to generate CEs. Formal definitions are provided in Appendix C.4.

4.2 RQ1: COUNTERFACTUAL EXPLANATION QUALITY

Table 1 compares CoSG with state-of-the-art baselines for explaining link sign predictions of SGCN and SNEA. We have the following observations: 1) CoSG achieves the strongest counterfactual effectiveness. CoSG consistently obtains the highest Validity and Fidelity across all datasets and

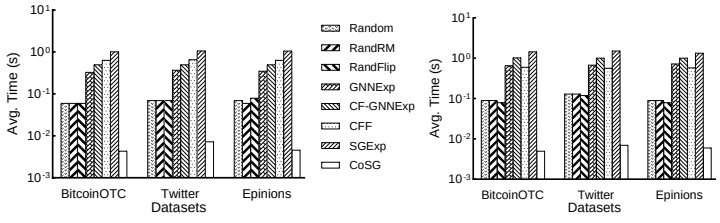
Table 1: The performance of different methods for explaining SGCN and SNEA model, reported as mean±std over 10 runs. Bold indicates the best result, and underlining indicates the runner-up.

Method	BitcoinOTC			Twitter			Epinions		
	Validity (↑)	Fidelity (↑)	Sparsity (↑)	Validity (↑)	Fidelity (↑)	Sparsity (↑)	Validity (↑)	Fidelity (↑)	Sparsity (↑)
SGCN									
Random	0.19±0.02	0.27±0.03	0.59±0.02	0.10±0.02	0.17±0.02	0.77±0.02	0.15±0.04	0.16±0.01	0.73±0.02
RandRM	0.15±0.00	0.24±0.02	0.58±0.02	0.08±0.03	0.16±0.02	0.76±0.01	0.14±0.05	0.16±0.01	0.73±0.02
RandFlip	0.25±0.03	0.31±0.03	0.60±0.02	0.12±0.02	0.19±0.02	0.77±0.01	0.18±0.04	0.17±0.01	0.74±0.02
GNNExplainer	0.20±0.01	0.35±0.03	0.89±0.01	0.11±0.02	0.31±0.02	0.94±0.01	0.24±0.06	0.40±0.02	0.93±0.01
CF-GNNExp	0.15±0.02	0.30±0.03	0.79±0.02	0.08±0.02	0.29±0.02	0.83±0.01	0.22±0.06	0.35±0.02	0.85±0.02
CF^2	0.18±0.02	0.33±0.03	0.87±0.01	0.08±0.02	0.31±0.02	0.94±0.00	0.23±0.06	0.37±0.02	0.91±0.01
SGExplainer	0.35±0.02	0.48±0.02	0.88±0.02	0.37±0.03	0.52±0.04	0.93±0.01	0.32±0.05	0.49±0.03	0.90±0.01
CoSG	**0.68±0.06**	**0.95±0.06**	**0.94±0.01**	**0.58±0.05**	**0.68±0.07**	**0.96±0.00**	**0.52±0.04**	**0.59±0.05**	**0.95±0.01**
SNEA									
Random	0.39±0.05	0.50±0.04	0.66±0.04	0.21±0.03	0.38±0.03	0.79±0.02	0.50±0.05	0.59±0.03	0.81±0.02
RandRM	0.33±0.03	0.50±0.03	0.65±0.04	0.17±0.01	0.35±0.04	0.78±0.01	0.44±0.03	0.59±0.03	0.79±0.02
RandFlip	0.44±0.03	0.51±0.03	0.68±0.04	0.25±0.02	0.44±0.02	0.80±0.02	0.52±0.04	0.58±0.04	0.81±0.02
GNNExplainer	0.40±0.03	0.57±0.05	0.90±0.01	0.27±0.02	0.43±0.04	**0.95±0.00**	0.56±0.04	0.67±0.04	**0.93±0.00**
CF-GNNExp	0.34±0.05	0.26±0.04	0.77±0.01	0.18±0.02	0.14±0.02	0.88±0.01	0.40±0.04	0.13±0.00	0.81±0.01
CF^2	0.32±0.04	0.26±0.04	0.88±0.01	0.23±0.02	0.14±0.00	0.92±0.00	0.40±0.04	0.16±0.02	0.90±0.01
SGExplainer	0.53±0.03	0.62±0.04	0.91±0.03	0.45±0.03	0.68±0.05	0.92±0.02	0.66±0.02	0.75±0.03	0.90±0.02
CoSG	**0.82±0.03**	**0.79±0.03**	**0.94±0.02**	**0.77±0.02**	**0.82±0.02**	0.93±0.01	**0.78±0.03**	**0.80±0.02**	0.91±0.02

both target SGNNs, showing that its generated counterfactual graphs are more likely to change the target prediction and reduce the confidence in the original class. 2) CoSG maintains a favorable effectiveness-sparsity trade-off. On SGCN, CoSG achieves the highest Sparsity across all datasets; on SNEA, it remains competitive and achieves the best or second-best Sparsity. For example, on Twitter with SGCN, CoSG reaches a Sparsity of 0.96, indicating that only about 4% of local edges are perturbed on average. Thus, CoSG does not obtain high Validity and Fidelity through excessive graph modifications; instead, it identifies concise perturbations sufficient to change the prediction. 3) Sign-aware counterfactual modeling is important. RandFlip generally outperforms RandRM, suggesting that edge polarity plays a key role in SGNN predictions and that sign flipping can be more effective than edge removal. Meanwhile, counterfactual baselines such as CF-GNNExplainer and CF^2 underperform because they rely on only edge-removal perturbations. SGExplainer achieves the second-best performance in most cases, showing the benefit of incorporating sign information; however, it is a factual explainer that does not directly optimize both edge removal and sign flipping for CE generation. In contrast, CoSG explicitly models edge existence and edge polarity, leading to more effective signed counterfactual search. 4) CoSG is compatible with different SGNN backbones. CoSG consistently outperforms baselines on both SGCN and SNEA, indicating that it is robust across different SGNN architecture. CoSG obtains particularly strong results on SNEA, and a similar trend is observed for several baselines. This suggests that SNEA may be more amenable to counterfactual perturbation. One possible reason is that SNEA learns embeddings by weighting neighbors, making predictions more sensitive to perturbations on influential edges.

4.3 RQ2: Explanation Efficiency

Fig. 3 reports the average per-instance explanation generation time of different methods. CoSG consistently achieves the lowest runtime across all datasets and both SGNN backbones, requiring only a few milliseconds per explanation. This is substantially faster than random baselines and optimization-based explainers, which require iterative perturbation trials or instance-specific mask optimization for each target instance. SGExplainer incurs the highest runtime among all methods due to its computationally expensive balanced-path search during final explanation generation, although



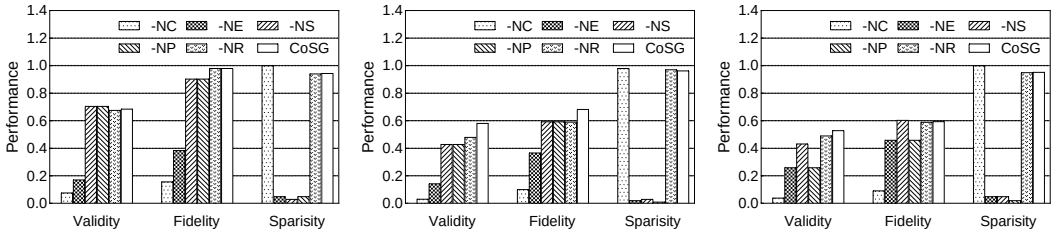
(a) SGCN (b) SNEA

Figure 3: Runtime comparison of baseline methods on two backbones.

this procedure may also contribute to its competitive effectiveness observed in RQ1. The efficiency advantage of CoSG mainly comes from its inductive design, which avoids repeated instance-specific optimization. CoSG learns an amortized perturbation policy during training and, after training, generates explanations for unseen instances by querying the frozen SGNN for prediction labels and performing a single forward pass through the explainer. Although CoSG requires an offline training phase, in practice, this cost is amortized across a large number of explainee instances. Moreover, CoSG's runtime remains stable across SGCN and SNEA, indicating that it is well suited for scalable explanation generation.

We further evaluate the scalability of CoSG under increasing SGNN depths in Appendix D.1.

4.4 RQ3: Ablation Study



(a) BitcoinOTC (b) Twitter (c) Epinions

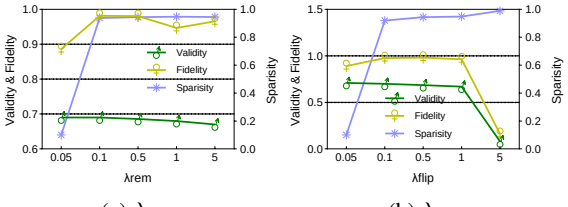
Figure 4: Ablation studies on three datasets.

To assess the contribution of different components of CoSG, we conduct an ablation study with the following variants: (a) **CoSG-NC**, where we remove the counterfactual prediction loss; (b) **CoSG-NE**, where we remove the perturbation loss on the edge mask; (c) **CoSG-NS**, where we remove the perturbation loss on the sign mask; (d) **CoSG-NP**, where we remove the perturbation losses on both masks; (e) **CoSG-NR**, where we remove the regularization losses on both masks. As shown in Fig. 4, we make the following observations: 1) The Validity of CoSG-NC drops dramatically across all datasets after removing the counterfactual prediction loss, indicating that this loss is essential for CE generation. 2) The Sparsity of CoSG-NE, CoSG-NS, and CoSG-NP drops significantly when the perturbation losses are removed, indicating that these losses are important for constraining unnecessary changes and producing compact explanations. Moreover, CoSG-NS often obtains higher Validity than CoSG-NE, suggesting that sign flipping is a particularly influential counterfactual perturbation in signed graphs; however, its low Sparsity also shows that such sign perturbations must be properly regularized. 3) CoSG-NR achieves competitive results but generally underperforms the full CoSG. This suggests that the regularization losses help align the continuous masks learned during training with the discrete perturbation actions used during inference, leading to more stable and effective explanations. 4) Overall, the full CoSG provides the most balanced performance among Validity, Fidelity, and Sparsity, demonstrating that the counterfactual prediction loss, perturbation losses, and regularization losses make complementary contributions to high-quality CEs.

We further provide a perturbation-restricted ablation in Appendix D.2, where edge removal and sign flipping are explicitly disabled to directly examine the contribution of each perturbation type.

4.5 RQ4: Hyperparameter Sensitivity Study

To answer RQ4, we study the sensitivity of CoSG to the key perturbation loss coefficients, including the edge removal penalty λ_{rem} and the sign-flipping penalty λ_{flip}. Specifically, we vary λ_{rem} and λ_{flip} in $\{0.05, 0.1, 0.5, 1, 5\}$. Unless otherwise specified, we vary one hyperparameter at a time and use the default settings reported in Table 7 in Appendix C.5.



(a) λ_{rem} (b) λ_{flip}

Figure 5: Parameter studies on BitcoinOTC.

As shown in Fig. 5, CoSG is generally robust to the perturbation loss coefficients and remains stable over a broad range of moderate values. When λ_{rem} or λ_{flip} is too small, the corresponding

perturbation type is under-penalized, so the model tends to make excessive edge removals or sign flips, leading to poor Sparsity. Once λ_{rem} is sufficiently large, CoSG remains stable, whereas an overly large λ_{flip} suppresses useful sign flips and sharply reduces Validity and Fidelity. This shows that sign flipping is an effective but sensitive perturbation in signed graphs and requires proper regularization. Additional sensitivity results are provided in Appendix D.3.

5 Related Work

Signed Graph Neural Networks and Their Explainability. In recent years, signed graph neural networks (SGNNs) (Chen et al., 2024; Derr et al., 2018; Li et al., 2020; Liu, 2022; Zhang et al., 2023; Shu et al., 2021; Lee et al., 2020; Chakraborty et al., 2023) have been developed to learn sign-aware node representations. Among these SGNNs, SGCN (Derr et al., 2018) and SNEA (Li et al., 2020) are two popular and representative models, and we use them as backbone SGNNs to evaluate the proposed explainer. SGCN extends graph convolution (Kipf & Welling, 2016) to signed graphs by separately aggregating information from positive and negative neighbors. SNEA introduces graph attention (Veličković et al., 2017) into signed graph representation learning, allowing the model to learn weights to different neighbors. Subsequent studies further improve signed graph learning by modeling directed signed relations (Huang et al., 2021), latent groups (Liu et al., 2021), spectral properties (Singh & Chen, 2022), or signed distance information (Chen et al., 2024). For a comprehensive overview of signed graph learning, please refer to the recent survey (Zhang et al., 2024). Despite the progress of SGNN architectures, their explainability remains largely underexplored (Li et al., 2024). The closest recent work, SGExplainer (Gao et al., 2026), proposes a balanced-path-enforcing framework, aiming to generate path-based explanations that are both faithful to model predictions and aligned with signed-graph domain knowledge. However, SGExplainer provides factual explanations by extracting supportive paths, whereas our work focus on counterfactual explanations that identify minimal structural perturbations sufficient to change the model prediction.

Factual and Counterfactual Explanations for General Graphs. GNN explanation methods aim to reveal model decision-making processes and can be broadly categorized into factual explanations and counterfactual explanations. Factual explanation methods (Ying et al., 2019; Luo et al., 2020; Schlichtkrull et al., 2020; Yuan et al., 2021; Zhang et al., 2022) identify influential inputs that preserves the original prediction. For example, GNNExplainer (Ying et al., 2019), which learns soft masks over edges and node features to maximize the mutual information between the explanation and the model prediction, and PGExplainer (Luo et al., 2020), which parameterizes the explanation generation process to improve efficiency and generalization across instances. These factual explanation methods mainly identify evidence that supports the current prediction, but they do not directly reveal which inputs should be modified to change the current prediction. To respond to this need, counterfactual explanation methods (Lucic et al., 2022; Tan et al., 2022; Qu et al., 2024; Ma et al., 2022; Huang et al., 2023) search for minimal perturbations that alter the model prediction, thereby highlighting decision-critical subsets of the inputs. For example, CF-GNNExplainer (Lucic et al., 2022) formulates CE generation as a graph perturbation problem and searches for sparse edge deletions that change the prediction. CF^2 (Tan et al., 2022) further introduces a model-agnostic framework that combines factual and counterfactual reasoning for learning GNN explanations. However, existing factual and counterfactual GNN explainers are mainly designed for unsigned graphs and target node- or graph-level prediction tasks. As a result, they typically identify or perturb edges/features but do not model edge polarity, which is crucial for link sign prediction. In contrast, our work studies counterfactual explanations for SGNNs by identifying sparse perturbations, including edge removals and sign flips, that change the prediction.

6 Conclusion

In this paper, we study counterfactual explanation generation for signed graph neural networks. We propose CoSG, a model-agnostic explainer for link sign prediction that decouples edge existence from edge polarity, enabling sign-aware counterfactual perturbations such as edge removals and sign flips. CoSG trains an explainer with a unified objective that jointly promotes counterfactual validity, perturbation sparsity, and discrete explanations, allowing efficient generalization to unseen target instances. Experiments on real-world signed graphs demonstrate that CoSG consistently outperforms existing graph explainers in generating effective, compact, and efficient counterfactual explanations.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing and polishing of the manuscript, including improving clarity, readability, and presentation. We also used generative AI tools for retrieval and discovery purposes, such as identifying potentially relevant literature. All retrieved sources and AI-assisted text were independently reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Selahattin Akkas and Ariful Azad. Gnnshap: Scalable and accurate gnn explanation using shapley values. In *Proceedings of the ACM web conference 2024*, pp. 827–838, 2024.

Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. *arXiv preprint arXiv:2206.09677*, 2022.

Francesco Bonchi, Edoardo Galimberti, Aristides Gionis, Bruno Ordozgoiti, and Giancarlo Ruffo. Discovering polarized communities in signed networks. In *Proceedings of the 28th acm international conference on information and knowledge management*, pp. 961–970, 2019.

Roshni Chakraborty, Ritwika Das, and Joydeep Chandra. Siggan: Adversarial model for learning signed relationships in networks. *ACM Transactions on Knowledge Discovery from Data*, 17(1): 1–20, 2023.

Jing Chen, Xinyu Yang, Mingxin Liu, and Miaomiao Liu. Sdegnn: Signed graph neural network for link sign prediction enhanced by signed distance encoding. *Journal of Supercomputing*, 80(13), 2024.

Ben Chester Cheong. Transparency and accountability in ai systems: safeguarding wellbeing in the age of algorithmic decision-making. *Frontiers in Human Dynamics*, 6:1421273, 2024.

Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 929–934. IEEE, 2018.

Alexandre Duval and Fragkiskos D Malliaros. Graphsvx: Shapley value explanations for graph neural networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 302–318. Springer, 2021.

Zhihong Fang, Shaolin Tan, and Yaonan Wang. A signed subgraph encoding approach via linear optimization for link sign prediction. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):14659–14670, 2024. doi: 10.1109/TNNLS.2023.3280924.

Jie Gao, Jia Hu, Geyong Min, and Fei Hao. Sgexplainer: Balanced path-based signed graph neural network explanation for link sign prediction. In *Proceedings of the ACM Web Conference 2026*, pp. 638–649, 2026.

Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. Sdgnn: Learning node representation for signed directed networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 196–203, 2021.

Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Global counterfactual explainer for graph neural networks. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pp. 141–149, 2023.

Shuyi Jin, Mengji Zhang, Meijie Wang, and Lun Yu. A deep graph model for the signed interaction prediction in biological network. *arXiv preprint arXiv:2407.07357*, 2024.

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

Yeon-Chang Lee, Nayoun Seo, Kyungsik Han, and Sang-Wook Kim. Asine: Adversarial signed network embedding. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval*, pp. 609–618, 2020.

Lu Li, Jiale Liu, Xingyu Ji, Maojun Wang, and Zeyu Zhang. Se-sgformer: A self-explainable signed graph transformer for link sign prediction. *arXiv preprint arXiv:2408.08754*, 2024.

Yu Li, Yuan Tian, Jiawei Zhang, and Yi Chang. Learning signed network embedding via graph attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4772–4779, 2020.

Yahong Lian, Haotian Tian, Chunyao Song, and Tingjian Ge. Sign-aware multimodal graph recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15225–15233, 2026.

Haoxin Liu. Lightsgcn: Powering signed graph convolution network for link sign prediction with simplified architecture design. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2680–2685, 2022.

Haoxin Liu, Ziwei Zhang, Peng Cui, Yafeng Zhang, Qiang Cui, Jiashuo Liu, and Wenwu Zhu. Signed graph neural network with latent groups. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 1066–1075, 2021.

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*, pp. 4499–4511. PMLR, 2022.

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.

Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. Clear: Generative counterfactual explanations on graphs. *Advances in neural information processing systems*, 35: 25895–25907, 2022.

Nikolaos Nakis, Chrysoula Kosma, Giannis Nikolentzos, Michalis Chatzianastasis, Iakovos Evdaimon, and Michalis Vazirgiannis. Signed graph autoencoder for explainable and polarization-aware network embeddings. *arXiv preprint arXiv:2409.10452*, 2024.

Le Ou-Yang, Dao-Qing Dai, and Xiao-Fei Zhang. Detecting protein complexes from signed protein-protein interaction networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 12(6):1333–1344, 2015.

Mario Alfonso Prado-Romero, Bardh Prenkaj, Giovanni Stilo, and Fosca Giannotti. A survey on graph counterfactual explanations: definitions, methods, evaluation, and research challenges. *ACM Computing Surveys*, 56(7):1–37, 2024.

Zhan Qu, Daniel Gomm, and Michael Färber. Cody: Counterfactual explainers for dynamic graphs. *arXiv preprint arXiv:2403.16846*, 2024.

Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for nlp with differentiable edge masking. *arXiv preprint arXiv:2010.00577*, 2020.

Lin Shu, Erxin Du, Yaomin Chang, Chuan Chen, Zibin Zheng, Xingxing Xing, and Shaofeng Shen. Sgcl: Contrastive representation learning for signed graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pp. 1671–1680, 2021.

Rahul Singh and Yongxin Chen. Signed graph neural networks: A frequency perspective. *arXiv preprint arXiv:2208.07323*, 2022.

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*, pp. 1018–1027, 2022.

Ruo-Chun Tzeng, Bruno Ordozgoiti, and Aristides Gionis. Discovering conflicting groups in signed networks. *Advances in Neural Information Processing Systems*, 33:10974–10985, 2020.

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

Jiaqi Wang, Xinyi Wu, James Cheng, and Yifei Wang. A signed graph approach to understanding and mitigating oversmoothing in gnns. *arXiv preprint arXiv:2502.11394*, 2025.

Chang Xue, Fang Liu, Jiaye Wang, Jinming Xing, and Chen Yang. Tas-gnn: A status-aware signed graph neural network for anomaly detection in bitcoin trust systems. *arXiv preprint arXiv:2603.13290*, 2026.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 430–438, 2020.

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*, pp. 12241–12252. PMLR, 2021.

Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. *Advances in neural information processing systems*, 35: 19810–19823, 2022.

Zeyu Zhang, Jiamou Liu, Xianda Zheng, Yifei Wang, Pengqian Han, Yupan Wang, Kaiqi Zhao, and Zijian Zhang. Rsgnn: A model-agnostic approach for enhancing the robustness of signed graph neural networks. In *Proceedings of the ACM web conference 2023*, pp. 60–70, 2023.

Zeyu Zhang, Peiyao Zhao, Xin Li, Jiamou Liu, Xinrui Zhang, Junjie Huang, and Xiaofeng Zhu. Signed graph representation learning: A survey. *arXiv preprint arXiv:2402.15980*, 2024.

A NOTATIONS

Table 2 summarizes the main notations used in the paper for reference.

Table 2: Notations.

Notation	Meaning/Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$	signed graph with nodes, positive edges, and negative edges
X	node feature matrix
$f = f_{\text{pred}} \circ f_{\text{enc}}$	trained SGNN with encoder and prediction head
$\mathbf{Z}$	node embeddings produced by the frozen SGNN encoder
$i = (s, t)$	target link instance corresponding to node pair (s, t)
$\mathcal{I}$	set of target instances to be explained
$\hat{y}_i$	original predicted label for target instance i
$\mathcal{L}$	number of SGNN layers
$\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i^+, \mathcal{E}_i^-)$	local computation graph of target instance i
$\mathcal{E}_i = \mathcal{E}_i^+ \cup \mathcal{E}_i^-$	candidate edge set for perturbation
$\Omega(\mathcal{G}_i)$	feasible perturbation space over $\mathcal{G}_i$
$\mathcal{P}(\widetilde{\mathcal{G}}_i, \mathcal{G}_i)$	perturbation cost between perturbed and original graphs
g_θ	MLP-based explainer shared across target instances
$\mathbf{R}_i$	edge-representation matrix for candidate edges in $\mathcal{E}_i$
$\mathbf{M}_i^a, \mathbf{M}_i^s$	adjacency and sign mask vectors
$\mathbf{W}_i^p, \mathbf{W}_i^f, \mathbf{W}_i^r$	action scores for preserving, flipping, and removing edges
$\widetilde{\mathcal{G}}_i$	soft candidate perturbed graph of $\mathcal{G}_i$
$\mathbf{P}_i$	decoded perturbation-action vector
$\mathcal{G}_i^{cf}$	discrete counterfactual graph induced by $\mathbf{P}_i$
$\mathcal{L}_{\text{pred}}^{(i)}, \mathcal{L}_{\text{pert}}^{(i)}, \mathcal{L}_{\text{reg}}^{(i)}$	prediction, perturbation, and regularization losses

Algorithm 1: Counterfactual Explainer CoSG for Signed Graph

Input: signed graph $\mathcal{G}$, frozen SGNN f, explainer g_θ, training instances $\mathcal{I}_t$, target instances $\mathcal{I}$, training epochs T, and learning rate η.

Output: counterfactual explanations $\{(\mathbf{P}_i, \mathcal{G}_i^{cf}) \mid i \in \mathcal{I}\}$.

/* Training Explainer */

for *each epoch* $1, \ldots, T$ **do**
  sample a batch of instances $\mathcal{B} \subseteq \mathcal{I}_t$;
  for *each instance* $i = (s, t) \in \mathcal{B}$ **do**
    $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i^+, \mathcal{E}_i^-) \leftarrow$ extract computation graph from $\mathcal{G}$;
    $\mathbf{Z}_i, \hat{y}_i \leftarrow f(\mathcal{G}_i, (s, t))$;
    $\mathbf{R}_i = \{\mathbf{e}_{uv} \mid (u, v) \in \mathcal{E}_i\}, \quad \mathbf{M}_i^a, \mathbf{M}_i^s \leftarrow g_\theta(\mathbf{R}_i)$;
    $\mathbf{W}_i^p, \mathbf{W}_i^f, \mathbf{W}_i^r \leftarrow$ compute action-score vectors using Eq. equation 1;
    $\widetilde{\mathcal{G}}_i \leftarrow$ construct soft perturbed graph using Eq. equation 2;
    $\mathcal{L}^{(i)} \leftarrow \mathcal{L}_{\text{pred}}^{(i)}(f, \widetilde{\mathcal{G}}_i, \hat{y}_i) + \mathcal{L}_{\text{pert}}^{(i)}(\mathbf{M}_i^a, \mathbf{M}_i^s) + \mathcal{L}_{\text{reg}}^{(i)}(\mathbf{M}_i^a, \mathbf{M}_i^s)$;
  $\mathcal{L}_{\text{batch}} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathcal{L}^{(i)}$;
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{batch}}$;

/* Generating Explanations */

for *each instance* $i = (s, t) \in \mathcal{I}$ **do**
  $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i^+, \mathcal{E}_i^-) \leftarrow$ extract computation graph from $\mathcal{G}$;
  $\mathbf{Z}_i \leftarrow f(\mathcal{G}_i, (s, t))$;
  $\mathbf{R}_i = \{\mathbf{e}_{uv} \mid (u, v) \in \mathcal{E}_i\}, \quad \mathbf{M}_i^a, \mathbf{M}_i^s \leftarrow g_\theta(\mathbf{R}_i)$;
  $\mathbf{W}_i^p, \mathbf{W}_i^f, \mathbf{W}_i^r \leftarrow$ compute action-score vectors using Eq. equation 1;
  $\mathbf{P}_i \leftarrow \{P_{i,uv} = \arg\max_{a \in \{p,f,r\}} W_{i,uv}^a \mid (u, v) \in \mathcal{E}_i\}$;
  $\mathcal{G}_i^{cf} \leftarrow \mathbf{P}_i \odot \mathcal{G}_i$;
return $\{(\mathbf{P}_i, \mathcal{G}_i^{cf}) \mid i \in \mathcal{I}\}$

B ALGORITHMS

Algorithm 1 presents the training and inference procedure of CoSG, following the proposed method in Section 3.

CoSG generates counterfactual explanations by learning dual masks over the local computation graph, which define sign-aware perturbation actions for each candidate edge. Given a signed graph $\mathcal{G}$, a frozen SGNN f, and an explainer g_θ, CoSG first samples a batch of training instances from $\mathcal{I}_t$ at each epoch (Line 2). For each target instance $i = (s, t)$, CoSG extracts its local computation graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i^+, \mathcal{E}_i^-)$ from the input graph (Lines 3–4), where the candidate perturbation space is restricted to the observed signed edges $\mathcal{E}_i = \mathcal{E}_i^+ \cup \mathcal{E}_i^-$. The frozen SGNN is then queried on $\mathcal{G}_i$ to obtain $\mathbf{Z}_i$ and $\hat{y}_i$ (Line 5). Then, CoSG constructs edge representations $\mathbf{R}_i$ and feeds them into the explainer g_θ to produce the adjacency mask $\mathbf{M}_i^a$ and sign mask $\mathbf{M}_i^s$. These two masks decouple edge existence from edge polarity and are transformed into three action-score vectors $\mathbf{W}_i^p$, $\mathbf{W}_i^f$, and $\mathbf{W}_i^r$ (Line 7), corresponding to preserving an edge with its original sign, flipping its sign, and removing it from the computation graph. Based on these soft action scores, CoSG constructs the differentiable perturbed computation graph $\widetilde{\mathcal{G}}_i$ (Line 8). The instance-level objective $\mathcal{L}^{(i)}$ is then computed by combining the prediction loss $\mathcal{L}_{\text{pred}}^{(i)}$, perturbation loss $\mathcal{L}_{\text{pert}}^{(i)}$, and regularization loss $\mathcal{L}_{\text{reg}}^{(i)}$ (Line 9). The batch loss $\mathcal{L}_{\text{batch}}$ is obtained by averaging the instance-level losses over the sampled batch (Line 10) and used to update the explainer parameters θ (Lines 10–11).

During inference, CoSG applies the trained explainer to each target instance in $\mathcal{I}$. It first extracts the computation graph and obtains node embeddings from the frozen SGNN (Lines 12–14). Then, the explainer generates the dual masks and action-score vectors following the same procedure as training (Lines 15–16). CoSG decodes the soft action scores into a discrete perturbation-action vector $\mathbf{P}_i$ by selecting the highest-scoring action from $\{p, f, r\}$ for each candidate edge (Line 17). Finally, the decoded actions are applied to the computation graph $\mathcal{G}_i$ to obtain the counterfactual graph $\mathcal{G}_i^{cf}$ (Line 18). The final explanation is returned as the pair $(\mathbf{P}_i, \mathcal{G}_i^{cf})$ for each target instance (Line 19).

C REPRODUCIBILITY

In this section, we provide additional details of the experimental setup to facilitate reproducibility.

Table 3: Detailed statistics of the datasets.

Dataset	**BitcoinOTC**	**Twitter**	**Epinions**
#Nodes	6,006	10,884	131,580
#Positive edges	32,029	238,612	589,888
#Negative edges	3,563	12,794	121,322
Avg out-degree	5.92	23.09	5.40
Feature dimension	64	64	64

C.1 DATASETS AND EXPERIMENTAL PROTOCOL

We adopt three widely-used signed network datasets [1] in our experiments: BitcoinOTC, Twitter and Epinions. The dataset statistics are summarized in Table 3. **BitcoinOTC** is a trust network derived from the Bitcoin-OTC platform, where users rate others as trust or distrust to build reputation and prevent fraud, corresponding to positive and negative edges, respectively. Following prior practice, we binarize these ratings into -1 and +1. **Twitter** is a signed network constructed from Twitter discussions of the 2016 Italian constitutional referendum, where users with the same stance are connected by positive edges while users with opposing stances are connected by negative edges. **Epinions** is a signed review network sourced from the trading platform Epinions where users express satisfaction or dissatisfaction based on their transaction experiences, corresponding to positive and negative edges, respectively.

[1] https://github.com/NDS-VU/signed-network-datasets

For SGNN training, we split positive and negative edges separately into train/validation/test sets with ratios of 0.7/0.1/0.2. Because the original datasets do not include node features, following prior practice, we initialize spectral node features using truncated SVD computed only from the training edges to avoid information leakage. The target SGNN is trained only on the training relations, with validation relations used for model selection and test relations held out.

After the target SGNN is trained and frozen, the parametrized explainers, CoSG and SGExplainer, are trained using 300 target node pairs in the training split. Other baselines do not require a separate explainer-training stage and are directly applied at inference time according to their original procedures. Test node pairs are used only for inference and evaluation after the target SGNN and, where applicable, the explainer have been trained and frozen. All methods are evaluated on the same 1,000 target node pairs corresponding to edges in the test split. Following prior GNN explanation studies on real-world graphs (Akkas & Azad, 2024; Duval & Malliaros, 2021; Gao et al., 2026; Amara et al., 2022), we adopt a fixed evaluation budget; for example, GraphFramEx (Amara et al., 2022) evaluates 100 testing nodes, while SGExplainer (Gao et al., 2026) evaluates 1,000 sampled instances. This choice is also motivated by the substantial cost of exhaustive explanation evaluation: a single run across the three datasets requires approximately 20 hours for GNNExplainer with SGCN and 77 hours for SGExplainer with SNEA when evaluating all test node pairs. Extending such evaluations across multiple backbones and repeated runs would further multiply this cost. To assess whether the sampled evaluation is representative of the full test set, we additionally evaluate CoSG on all available test node pairs. As shown in Table 4, the full-test results remain close to the corresponding results in Table 1 across all datasets and both SGNN backbones.

Table 4: Performance of CoSG on the full test sets.

Backbone	Dataset	Validity ↑	Fidelity ↑	Sparsity ↑
SGCN	BitcoinOTC	0.68 ± 0.03	0.94 ± 0.04	0.94 ± 0.01
	Twitter	0.57 ± 0.06	0.67 ± 0.07	0.95 ± 0.01
	Epinions	0.51 ± 0.03	0.59 ± 0.05	0.95 ± 0.01
SNEA	BitcoinOTC	0.80 ± 0.04	0.78 ± 0.02	0.93 ± 0.01
	Twitter	0.75 ± 0.02	0.82 ± 0.03	0.93 ± 0.01
	Epinions	0.78 ± 0.02	0.79 ± 0.02	0.91 ± 0.02

Table 5: Performance of SGNN models on the three datasets.

Model	**Metric**	**BitcoinOTC**	**Twitter**	**Epinions**
SGCN	AUC	0.842±0.000	0.905±0.005	0.813±0.002
	F1-score	0.923±0.014	0.945±0.002	0.903±0.003
	Accuracy	0.869±0.022	0.901±0.004	0.846±0.005
SNEA	AUC	0.843±0.005	0.876±0.008	0.810±0.004
	F1-score	0.931±0.003	0.922±0.003	0.907±0.001
	Accuracy	0.881±0.005	0.862±0.005	0.852±0.002

C.2 Details of SGNN Models

Our experiments involve two representative SGNN models as explanation targets: SGCN (Derr et al., 2018) and SNEA (Li et al., 2020). SGCN separately aggregates messages from positive and negative neighbors within a balance-theory-inspired architecture. SNEA generalizes graph attention to signed graphs by introducing a sign-aware attention mechanism that learns the relative importance of neighbors with different signs. Specifically, each prediction model consists of:

- Two message-passing layers for learning node embeddings.
- A single-layer prediction head that takes concatenated endpoint embeddings and outputs logits.

For link sign prediction, we train each SGNN model using the loss function and hyperparameter settings recommended in its original paper. In all experiments, node features are 64-dimensional

spectral embeddings. In the 2-layer setting, each hidden representation is split into two signed channels, i.e., $64 \rightarrow (32 + 32) \rightarrow (32 + 32)$. For prediction, the two endpoint embeddings are concatenated into a 128-dimensional vector and mapped to three logits for positive, negative, and non-edge classes. We use Adam optimizer with learning rate 0.01, weight decay $5e - 4$, and 200 training epochs. We keep the checkpoint with the best validation AUC and report test results using that checkpoint. Table 5 shows the SGNN models achieve strong performance on the link sign prediction task across datasets. All models achieve over 85% accuracy on every dataset, indicating that they are well-trained SGNNs and thus provide a reliable basis for evaluating explanations.

C.3 Details of Baseline Explainers

Table 6 summarizes the key properties of the baseline explainers. We provide more implementation details for the baseline explainers below:

- **Random/ RandRM/ RandFlip:** For each explainee instance, we randomly perturb its local computational graph for at most 50 steps, with at most one edit per step, and stop once the model prediction changes. We consider three variants: 1) mixed random flip/remove strategy, 2) edge removal only, and 3) sign flipping only.
- **GNNExplainer (Ying et al., 2019):** For each explainee instance, GNNExplainer learns an edge mask that quantifies each local edge's contribution to the original model prediction, where higher mask values indicate greater importance. To generate counterfactual explanations, we threshold the learned mask at 0.5 and remove edges with mask score ≥ 0.5.
- **CF-GNNExplainer (Lucic et al., 2022):** We apply CF-GNNExplainer to each explainee's local subgraph, where it learns a continuous edge mask that quantifies how each edge should be perturbed to induce a counterfactual change. Accordingly, unlike GNNExplainer, edges with mask values ≥ 0.5 are retained as candidate explanations. In addition, CF-GNNExplainer performs iterative counterfactual search during optimization: at each iteration, the current edge mask is thresholded, converted to a hard perturbation, and checked for whether it changes the prediction. We keep the best valid counterfactual found across iterations. The number of optimization iterations is set to 100.
- **CF^2 (Tan et al., 2022):** For each target instance's local subgraph, CF^2 learns an edge mask using an objective jointly captures factual and counterfactual reasoning. To generate counterfactual explanations,, we decode edge removals from the learned mask: edges with mask score ≥ 0.5 are removed.
- **SGExplainer (Gao et al., 2026):** It is a factual explainer tailored to signed graphs and link sign prediction, incorporating sign information during explainer training and restricting final explanations to balanced paths. We adapt it to CE generation by removing important balanced paths identified by SGExplainer. For final path extraction, we set the maximum path length and edge budget to 4 and 50, respectively.

Table 6: Comparison of the properties of baseline explainers and our method CoSG when generalizing to explain link sign prediction. The last column reports the number of explainers required to explain N instances.

Method	Explainability	Signed Graph	Edge Remove	Sign Flip	# explainers
Random	-	-	✓	✓	N
RandRM	-	-	✓	×	N
RandFlip	-	-	×	✓	N
GNNExplainer	Factual	×	✓	×	N
CF-GNNExplainer	Counterfactual	×	✓	×	N
CF^2	Counterfactual+Factual	×	✓	×	N
SGExplainer	Factual	✓	✓	×	1
CoSG (ours)	Counterfactual	✓	✓	✓	1

C.4 Details of Evaluation Metrics

For each explainee instance i, let $\mathcal{G}_i$ denote the original computational graph and $(\mathbf{P}_i, \mathcal{G}_i^{cf})$ denote its counterfactual explanation, where $\mathbf{P}_i$ is the set of perturbation actions and $\mathcal{G}_i^{cf}$ is the counterfactual

graph induced by applying $\mathbf{P}_i$ to $\mathcal{G}_i$. Let $\hat{y}_i$ and $\hat{y}_i^{cf}$ denote the predictions on $\mathcal{G}_i$ and $\mathcal{G}_i^{cf}$, respectively. Let $l_i(\hat{y}_i)$ and $l_i^{cf}(\hat{y}_i)$ denote the logits of the original predicted class before and after applying the counterfactual explanation, respectively.

- **Validity** measures whether the counterfactual graph changes the predicted label:

$$\text{Validity} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left(\hat{y}_i^{cf} \neq \hat{y}_i\right),$$

  where N is the number of explainee instances and $\mathbb{I}(\cdot)$ is the indicator function.
- **Fidelity** measures the change in model confidence for the original predicted class:

$$\text{Fidelity} = \frac{1}{N}\sum_{i=1}^{N} \left| l_i(\hat{y}_i) - l_i^{cf}(\hat{y}_i) \right|.$$

- **Sparsity** measures the proportion of the local computational graph that remains unchanged after perturbation:

$$\text{Sparsity} = \frac{1}{N}\sum_{i=1}^{N} \left(1 - \frac{|\mathbf{P}_i|}{|\mathcal{E}_i|}\right),$$

  where $\mathcal{E}_i$ is the edge set of the original computational graph $\mathcal{G}_i$.
- **Avg. Time** measures the average time required to generate counterfactual explanations for each instance:

$$\text{Avg. Time} = \frac{1}{N}\sum_{i=1}^{N} t_i,$$

  where t_i is the time required to generate counterfactual explanations for instance i.

C.5 Implementation Environment and Settings

The source code for CoSG is available at `https://anonymous.4open.science/r/ICLR_CoSG`. The hyperparameter settings of CoSG are presented in Table 7. Our implementation is built with the following packages:

Python	3.9.16	PyTorch	1.12.1	CUDA 12.3
PyG	2.1.0	torch-scatter	2.0.9	torch-sparse 0.6.15
NumPy	1.23.5	pandas	1.5.3	SciPy 1.10.1
scikit-learn	1.2.2			

Table 7: Hyperparameter settings of CoSG on the three datasets.

Hyperparameter	**BitcoinOTC**	**Twitter**	**Epinions**
Number of layers	3	3	3
Hidden dimensions	128, 64, 32	128, 64, 32	128, 64, 32
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.005	0.005
Epochs	100	50	100
Batch size	16	16	16
λ_{rem}	0.5	0.5	0.5
λ_{flip}	0.5	0.5	0.5
$\lambda_{\text{ent_adj}}$	10^{-3}	10^{-4}	10^{-4}
$\lambda_{\text{ent_sign}}$	10^{-3}	10^{-4}	10^{-4}

D More Experimental Results

D.1 RO2: Scalability with Increasing SGNN Depth

To further investigate the scalability of CoSG as the SGNN receptive field increases, we evaluate CoSG with 2-, 3-, and 4-layer SGCN models on BitcoinOTC, while keeping the explainer architecture and evaluation protocol unchanged. The results are reported in Table 8.

Table 8: Scalability of CoSG under different SGCN depths on BitcoinOTC.

Layers	SGCN Acc.	Validity	Fidelity	Sparsity	Avg. Time (s)
2	0.869 ± 0.022	0.68 ± 0.06	0.95 ± 0.06	0.94 ± 0.01	0.004 ± 0.001
3	0.864 ± 0.020	0.68 ± 0.05	0.94 ± 0.05	0.95 ± 0.01	0.006 ± 0.001
4	0.855 ± 0.023	0.65 ± 0.06	0.92 ± 0.06	0.95 ± 0.01	0.013 ± 0.003

As shown in Table 8, CoSG maintains relatively stable explanation quality as the SGCN depth increases from 2 to 4 layers, while the average explanation time increases with the enlarged receptive field. These results indicate that CoSG remains effective under deeper SGNNs, with additional runtime cost as the computation graph grows.

D.2 RO3: Perturbation-Restricted Ablation Study

To assess the respective contributions of edge removal and sign flipping perturbations, we additionally construct two perturbation-restricted variants that completely disable one perturbation type. **CoSG-RM** removes the sign-mask head and fixes $M_i^s \equiv 1$, allowing only edge preservation and removal. **CoSG-FL** removes the adjacency-mask head and fixes $M_i^a \equiv 1$, allowing only edge preservation and sign flipping. For each variant, only the loss terms associated with the active mask are retained.

Table 9: Performance of perturbation-restricted CoSG variants with SGCN.

Dataset	Method	Validity ↑	Fidelity ↑	Sparsity ↑
BitcoinOTC	CoSG-RM	0.46 ± 0.05	0.81 ± 0.06	0.89 ± 0.02
	CoSG-FL	0.66 ± 0.05	0.93 ± 0.04	$\mathbf{0.95 \pm 0.01}$
	CoSG	$\mathbf{0.68 \pm 0.06}$	$\mathbf{0.95 \pm 0.06}$	0.94 ± 0.01
Twitter	CoSG-RM	0.35 ± 0.04	0.55 ± 0.05	0.93 ± 0.02
	CoSG-FL	0.55 ± 0.05	0.67 ± 0.06	$\mathbf{0.97 \pm 0.01}$
	CoSG	$\mathbf{0.58 \pm 0.05}$	$\mathbf{0.68 \pm 0.07}$	0.96 ± 0.00
Epinions	CoSG-RM	0.38 ± 0.04	0.52 ± 0.04	0.92 ± 0.02
	CoSG-FL	0.50 ± 0.03	0.58 ± 0.04	$\mathbf{0.96 \pm 0.01}$
	CoSG	$\mathbf{0.52 \pm 0.04}$	$\mathbf{0.59 \pm 0.05}$	0.95 ± 0.01

As shown in Table 9, CoSG-FL consistently remains close to the full CoSG, whereas CoSG-RM exhibits substantially lower Validity and Fidelity. This provides direct evidence that sign flipping makes a stronger contribution than edge removal to counterfactual generation on the evaluated signed graphs. Nevertheless, the full CoSG consistently achieves the highest Validity and Fidelity while maintaining high Sparsity, indicating that jointly supporting both perturbation actions improves counterfactual effectiveness without substantially sacrificing explanation compactness.

D.3 RO4: More Hyperparameter Sensitivity Results

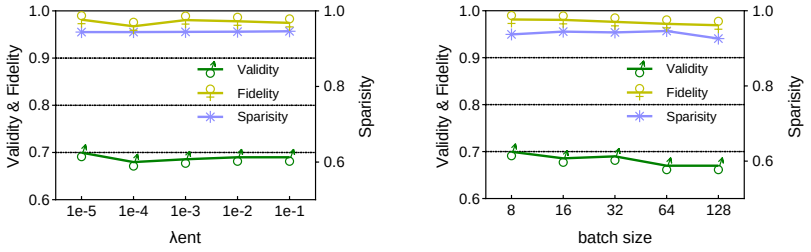
D.3.1 Additional Results on BitcoinOTC

We further evaluate the sensitivity of CoSG to the entropy regularization coefficient and the training batch size. For entropy regularization, we set $\lambda_{\text{ent},a} = \lambda_{\text{ent},s} = \lambda_{\text{ent}}$ and vary the shared coefficient as $\lambda_{\text{ent}} \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. For batch size, we vary it in $\{8, 16, 32, 64, 128\}$. Unless otherwise specified, we vary one hyperparameter at a time while keeping the others fixed to their default values in Table 7.

Fig. 6 shows the sensitivity results on BitcoinOTC with SGCN. CoSG is less sensitive to the entropy regularization coefficient, with Validity, Fidelity, and Sparsity remaining stable across a wide range of values. This suggests that entropy regularization helps encourage discrete masks without dominating the optimization process. For batch size, CoSG also maintains stable performance across different settings, indicating that its explanation quality is not highly sensitive to the batch size.

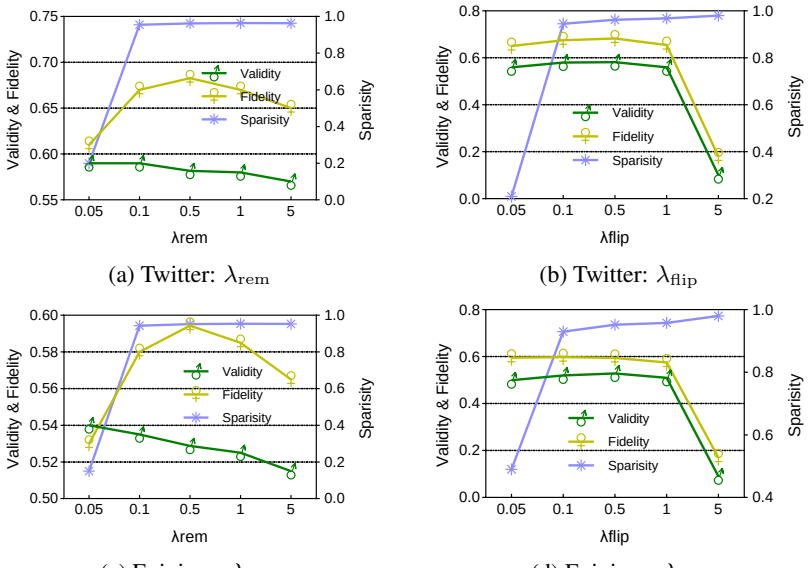
D.3.2 Parameter Study on Other Datasets

We further study the sensitivity of CoSG to the edge removal penalty λ_{rem} and the sign-flipping penalty λ_{flip} on Twitter and Epinions. As shown in Fig. 7, the results are generally consistent with those observed on BitcoinOTC in Section 4.5. CoSG maintains stable performance under moderate hyperparameter settings, whereas overly small perturbation penalties tend to cause excessive edge removals or sign flips, thereby reducing Sparsity.



(a) λ_{ent} (b) Batch size

Figure 6: Additional hyperparameter sensitivity analysis on BitcoinOTC.



(a) Twitter: λ_{rem} (b) Twitter: λ_{flip}

(c) Epinions: λ_{rem} (d) Epinions: λ_{flip}

Figure 7: Hyperparameter sensitivity analysis on Twitter and Epinions.

D.4 RO5: Qualitative Case Study

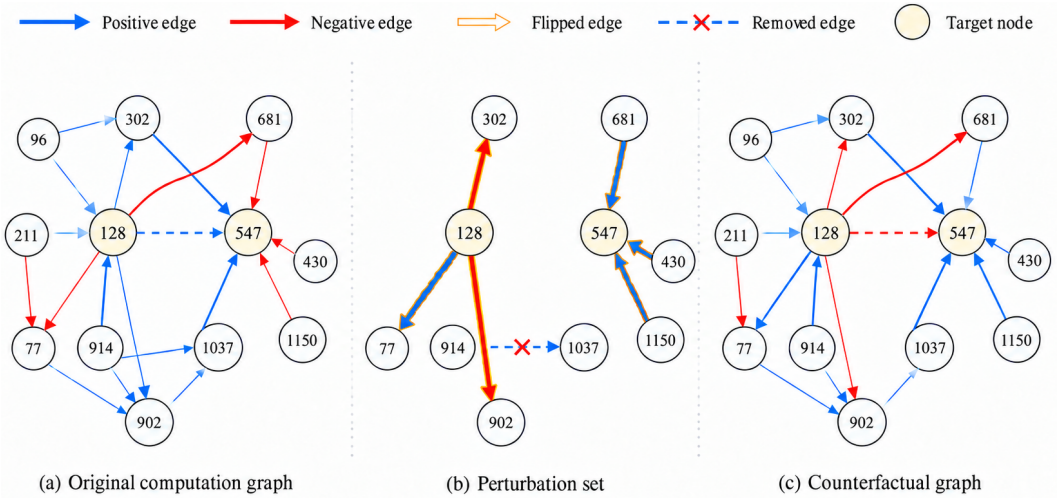


Figure 8: Qualitative case study on BitcoinOTC for the target pair (128, 547). Blue edges denote trust relations and red edges denote distrust relations. (a) Original computation graph used by the frozen SGCN. (b) Perturbation set identified by CoSG, including sign-flipped and removed edges. (c) Counterfactual graph obtained after applying the selected perturbations, which changes the prediction from Trust to Distrust.

To provide a more concrete qualitative analysis of the generated counterfactual explanations, we examine a representative BitcoinOTC instance with target pair $(128, 547)$. Its two-hop computation graph contains 186 signed edges. For visual clarity, only a subset of the edges in the computation graph is shown in Fig. 8, while all perturbation statistics are computed on the complete computation graph. As illustrated in Fig. 8, CoSG changes the frozen SGCN's predicted sign for the target relation $(128, 547)$ from *Trust* in Fig. 8(a) to *Distrust* in Fig. 8(c) by flipping edge signs and removing edge, as highlighted in Fig. 8(b), modifying only 3.76% of the edges in the computation graph.

Among the seven perturbations shown in Fig. 8(b), three representative examples illustrate how CoSG modifies different parts of the computation graph. First, the edge $(128, 302)$ is flipped from *Trust* to *Distrust*. Since this edge is directly adjacent to the source node 128, the perturbation redirects the corresponding relation from the positive to the negative message-passing channel. In the BitcoinOTC context, this can be interpreted as the source user revising a neighboring rating from trust to distrust. Second, the edge $(681, 547)$ is flipped from *Distrust* to *Trust*. This edge is directly adjacent to the target node 547, and the perturbation redirects its contribution from the negative to the positive message-passing channel. Semantically, this may correspond to a neighboring user revising the target user's rating from distrust to trust. Third, the edge $(914, 1037)$ is removed. As an internal edge in the source–target computation graph, its removal eliminates this positive relation from subsequent message passing and removes its contribution to the endpoint representations. This can be interpreted as an existing trust relation being withdrawn, becoming outdated, or no longer being available.

After applying these perturbations, Fig. 8(c) shows the resulting counterfactual graph. Although most of the original graph structure is preserved, the selected perturbations are sufficient to reverse the frozen SGCN's predicted sign for the target relation $(128, 547)$ from *Trust* to *Distrust*. Overall, this example demonstrates that CoSG can generate sparse counterfactual explanations by jointly leveraging sign flipping and edge removal while preserving most of the original computation graph.