RSGNN: A Model-agnostic Approach for Enhancing the Robustness of Signed Graph Neural Networks

Zeyu Zhang
The University of Auckland
Auckland, New Zealand
zzha669@aucklanduni.ac.nz

Jiamou Liu*
The University of Auckland
Auckland, New Zealand
jiamou.liu@auckland.ac.nz

Xianda Zheng
The University of Auckland
Auckland, New Zealand
xzhe162@aucklanduni.ac.nz

Yifei Wang
The University of Auckland
Auckland, New Zealand
wany107@aucklanduni.ac.nz

Pengqian Han
The University of Auckland
Auckland, New Zealand
phan635@aucklanduni.ac.nz

Yupan Wang
The University of Auckland
Auckland, New Zealand
ywan980@aucklanduni.ac.nz

Kaiqi Zhao
The University of Auckland
Auckland, New Zealand
kaiqi.zhao@auckland.ac.nz

Zijian Zhang*
Beijing Institute of Technology
Beijing, China
zhangzijian@bit.edu.cn

ABSTRACT

Signed graphs model complex relations using both positive and negative edges. Signed graph neural networks (SGNN) are powerful tools to analyze signed graphs. We address the vulnerability of SGNN to potential edge noise in the input graph. Our goal is to strengthen existing SGNN allowing them to withstand edge noises by extracting robust representations for signed graphs. First, we analyze the expressiveness of SGNN using an extended Weisfeiler-Lehman (WL) graph isomorphism test and identify the limitations to SGNN over triangles that are unbalanced. Then, we design some structure-based regularizers to be used in conjunction with an SGNN that highlight intrinsic properties of a signed graph. The tools and insights above allow us to propose a novel framework, Robust Signed Graph Neural Network (RSGNN), which adopts a dual architecture that simultaneously denoises the graph while learning node representations. We validate the performance of our model empirically on four real-world signed graph datasets, i.e., Bitcoin_OTC, Bitcoin_Alpha, Epinion and Slashdot, RSGNN can clearly improve the robustness of popular SGNN models. When the signed graphs are affected by random noise, our method outperforms baselines by up to 9.35% Binary-F1 for link sign prediction. **Our implementation is available in PyTorch**[1].

*Corresponding Author
[1]https://github.com/Alex-Zeyu/RSGNN

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583221

CCS CONCEPTS

• **Information systems → World Wide Web**; *Information retrieval*; Information systems applications.

KEYWORDS

Signed Graph, Signed Graph Neural Networks, Signed Graph, Robustness

ACM Reference Format:
Zeyu Zhang, Jiamou Liu, Xianda Zheng, Yifei Wang, Pengqian Han, Yupan Wang, Kaiqi Zhao, and Zijian Zhang. 2023. RSGNN: A Model-agnostic Approach for Enhancing the Robustness of Signed Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3543507.3583221

1 INTRODUCTION

The prevalence of online social, business, and cryptocurrency platforms has accumulated a large amount of graph datasets. Graph representation learning methods, in particular those that are based on graph neural networks (GNN), are popular tools for the analysis of these graph datasets [14, 25, 40]. A GNN produces expressive representations of nodes in a graph through a message-passing mechansim that aggregates information along the edges. Yet, real-world edge relations between nodes are often not limited to expressing positive ties such as friendship, trust, and agreement, but they could also reflect negative ties such as enmity, mistrust, and disagreement. For instance, Slashdot, a tech-related news website, allows users to tag other users as either 'friends' or 'foes'. Such a situation can be naturally modeled as a *signed graph*, i.e., a graph that contains both positive and negative edges. However, the presence of negative edges invalidates the standard message-passing mechanism, and thus the need arises to design new GNN models for signed graphs, i.e., signed graph neural networks (SGNN).

Recent years have witnessed a growing interest in SGNN, with *link sign prediction* as the focal task [10, 18, 20, 27, 30, 31, 36]. Other tasks include signed graph node clustering [16, 26, 31] and signed

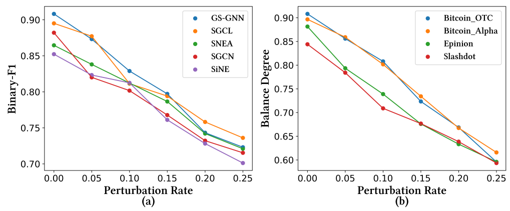

Figure 1: (a) Effects of random noise on link sign prediction performance of popular SGNN models; and (b) Effects of random noise on balance degree of four real-world datasets

recommendations [35]. Yet *no* existing study on SGNNs has addressed the issue of robustness. Indeed, noise may be introduced to a graph dataset during the data collection phase in an intentional or unintentional way. For example, in a Bitcoin trading platform where users can rate other users, a wrong rating may be provided by mistake. Moreover, in an e-commerce site where customers are provided incentives (in the form of rewards) to rate their purchased items, a customer may give random ratings simply to reap the rewards. In these situations, the signed edges may misrepresent the actual relations between nodes in the graph. We now evaluate the impact of such noise to SGNN by flipping the signs of random edges (i.e., hence creating *noisy edges*) in the Slashdot dataset. As is shown in Figure 1(a), the link sign prediction accuracy of popular SGNN models [10, 30, 31, 36, 42] all drops sharply as more noisy edges are introduced. Thus there is a need for a *robust representation learning* framework, which trains GNNs to withstand noise in the input dataset [9, 47, 53], on signed graphs.

We argue that existing robust GNN frameworks, which were designed for unsigned graphs, do not lend well to signed graphs: *(i)* Unlike unsigned graph datasets, real-world signed graph datasets lack node label and attribute information which are critical to most robust GNN models (such as [8, 46]). *(ii)* Existing robust GNN methods rely on certain intrinsic properties of unsigned graph to handle noise [11, 21] which are not directly applicable to signed graphs. For example, feature smoothness, i.e., that connected nodes share similar features, is a generally recognized property for unsigned graphs. However, it is not applicable when negative edges are present.

In this paper, we initiate the study of robust representation learning for signed graphs. We aim to develop a framework that is able to reduce the impact of noisy edges and restore the intended edge relations. We face two major obstacles in our work:

(1) **Understanding the impact of noisy edges**. So far no insight is provided on how noisy edges reduce the performance of SGNNs. Without having a good grasp of this, it is difficult to design a proper denoising strategy.
(2) **Intrinsic properties for signed graphs**. It is still an open question what the intrinsic properties of a "denoised" signed graph. Without these intrinsic properties, it is difficult to design criteria useful for the denoising of signed graph.

To overcome the first obstacle, we invoke *balance theory*, a classical statistical law regarding edge layouts within signed graphs [4, 17]. Balance theory lays out generally expected edge patterns among signed triangles (groups of three nodes that are mutually linked by edges). The theory asserts conditions for a triangle to be "structurally balanced" in the sense that the relations among three individuals have reached a form of stability. Indeed, numerous studies [18, 28, 29] have confirmed that most triangles in real-world signed graphs indeed satisfy these conditions. Aref et al. [1] rephrased this theory using the notion of *(partial) balance degree* that quantifies the extent to which triangles in a signed graph are structurally balanced. Balance degree bridges the gap between noisy edges and SGNN models: First, a signed graph with many noisy edges tends to have a low balance degree, as illustrated in Figure 1(b). Then, when a node belongs to an unbalanced triangle, the message-passing mechanism of an SGNN model cannot learn a proper representation for the node. To make this fact precise, we extend Weisfeiler-Lehman (WL) graph isomorphism test [45] to signed graphs and draw a connection with the expressibility of SGNN. See Section 4.

To overcome the second obstacle, we design three *intrinsic property regularizers*. The most important regularizer is *balance degree loss* which is based on our theoretical analysis above. In particular, our analysis implies that unbalanced triangles are likely consequences of noisy edges and harm model performance. One of the intrinsic properties for a denoised signed graph is thus that it has a high balance degree. The other two intrinsic property regularizers are sparsity loss and feature loss (See Section 5.1). We then propose a novel learning framework Robust Signed Graph Neural Network (RSGNN). The framework is *model-agnostic* in the sense that it may be used in combination with a given SGNN model to enhance the model's robustness. In particular, RSGNN adopts a dual architecture: Given an input signed graph $\mathcal{G}$, RSGNN alternatively (a) updates the signed graph structure $\mathcal{G}$ hoping to down-weight (or remove) the noisy edges, and (b) learns node representations of the signed graph $\mathcal{G}$ using the SGNN model for a downstream task (in our case the task is chosen to be *link sign prediction*).

To evaluate the effectiveness of RSGNN, we perform extensive experiments on four real-world datasets, i.e., Bitcoin_OTC, Bitcoin_Alpha, Epinions, and Slashdot. We verify that our proposed framework RSGNN can enhance model robustness using two common SGNN models (SGCN [10] and SNEA [30]) as the encoding module. Under random noise, RSGNN improves the link sign prediction accuracy of the base model (SGNN or SNEA) by up to 7.19% in terms of AUC and 13.66% in terms of Binary-F1. Compared with state-of-the-art link sign prediction methods, the proposed framework with SGCN as the encoding module boosts the performances by up to 9.35% on metric Binary-F1 with 25% perturbed edges. These experimental results demonstrate the effectiveness of RSGNN.

Overall, our contributions are summarized as follows:

- We analyze the insufficiencies of existing SGNNs in getting proper representations for nodes in an unbalanced triangle.
- We propose the first novel model-agnostic robust learning framework RSGNN for signed graph.
- Extensive experiments on real-world datasets demonstrate the robustness of our framework.

2 RELATED WORK

Graph representation learning [6] has made great advancements in a wide range of graph data analysis tasks, such as *node classification* [25, 33, 40, 41], *node clustering* [43], *link prediction* [13], *recommender systems* [15, 44], *network visualization* [3, 38]. Graph neural networks (GNN) [25, 40] have become the predominant paradigm in this field. The message-passing mechanism is the main mechanism of GNN models in which nodes aggregate information from their neighbors.

To our best knowledge, no work has directly discussed robustness of signed graph representation learning. Below we briefly introduce existing literature on signed graph representation learning. Problems over signed graphs include node classification [37], node ranking [22], community detection [2] and visualization [42]. Yet *link sign prediction* is a task unique to signed graph and is the focal interest of this field. Early signed graph embedding methods, such as SNE [49], SIDE [24], SGDN [23] are based on random walks. Other methods utilise signed Laplacian embedding [5, 26, 50] and matrix factorization [37].

In recently years, neural-based methods are increasingly being applied to signed graph representation learning. The first work that employs a deep learning framework is SiNE [42] which extracts structural information using triangles with both positive and negative edges, and optimizes an objective function designed inspired from balance theory to learn node embeddings. SGCN, the first SGNN model, [10] generalizes GCN [25] to signed graphs while using balance theory to determine the positive and negative relationships between nodes that are multi-hop apart. Other models such as SiGAT [19], SNEA [30], SDGNN [20], and SGCL [36], are based on GAT [39].

Apart from the above, a few other models for signed graphs are claimed to be equipped with certain noise-resistant properties. SGCL [36] adopts contrastive learning in SGNN, and its encoder adopts an attention layer similar to SNEA. SGCL employs graph augmentations to proactively add a small amount of random perturbation, which can help to enhance the robustness of the model. GS-GNN [31] extends the balance theory assumption (which implies nodes should be divided into two controversial groups) to k-group theory which improves the flexibility of the model to tolerate certain random noises. Different from these methods, we start with a theoretical analysis of the impact of noisy edges on SGNNs. Then motivated by the findings from our analysis, we design a robust learning framework for SGNNs that denoises and learns the node representations.

3 PROBLEM STATEMENT

A *signed graph* is $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$, where $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ denotes the set of nodes, $\mathcal{E}^+$ and $\mathcal{E}^-$ denote the positive and negative edges, respectively. Each edge $e_{ij} \in \mathcal{E}^+ \cup \mathcal{E}^-$ between two nodes v_i and v_j can be either positive or negative, but not both, i.e., $\mathcal{E}^+ \cap \mathcal{E}^- = \varnothing$. Let $\sigma(e_{ij}) \in \{+, -\}$ denote the *sign* of e_{ij}. The structure of $\mathcal{G}$ is represented by the *adjacency matrix* $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where each entry $A_{ij} \in \{1, -1, 0\}$ denotes the sign of edge e_{ij}. Note that for signed graphs, a node is normally not given a feature. This is different from an unsigned graph dataset which typically contains a feature vector x_i for each node v_i.

Positive and *negative neighbors* of v_i are denoted as $\mathcal{N}_i^+ = \{v_j \mid A_{ij} > 0\}$ and $\mathcal{N}_i^- = \{v_j \mid A_{ij} < 0\}$, respectively. Let $\mathcal{N}_i = \mathcal{N}_i^+ \cup \mathcal{N}_i^-$ be the set of neighbors of node v_i. O_3 represents the set of *triangles* in the signed graph, i.e., $O_3 = \{\{v_i, v_j, v_k\} \mid A_{ij}A_{jk}A_{ik} \neq 0\}$. A triangle $\{v_i, v_j, v_k\}$ is called *balanced* if $A_{ij}A_{jk}A_{ik} > 0$, and is called *unbalanced* otherwise.

The extended structural balance theory [7, 34] forms the basis for learning representations of a signed graph. Intuitively, the theory suggests that an individual should bear a higher resemblance with those neighbors who are connected with positive edges than those who are connected with negative edges. Therefore, the goal of an SGNN is to learn a *embedding function* $f_\theta : \mathcal{V} \to H$ that maps the nodes of a signed graph to a latent vector space H where $f_\theta(v_i)$ and $f_\theta(v_j)$ are close if $e_{ij} \in \mathcal{E}^+$ and distant if $e_{ij} \in \mathcal{E}^-$ (See Def. 4.2). Furthermore, we choose *link sign prediction* as the downstream task of SGNN following mainstream studies. The task seeks to infer $\sigma(e_{ij})$ given nodes v_i and v_j [28]. With the aforementioned notations, we formulate our robust signed graph representation learning problem as:

Robust signed graph representation learning. *Given a signed graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$ being influenced by noisy edges, simultaneously learn a denoised graph structure $S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and the SGNN parameters θ to facilitate link sign prediction task.*

4 THEORETICAL ANALYSIS

In this section, we analyze the impact of noisy edges on signed graph neural networks (SGNNs). Our analysis consists of two parts: First, we investigate real-world datasets and empirically verifies the impact of noisy edges on the balance degree of a signed graph. Then, we aim to show an intrinsic limitation with existing SGNN models in the presence of noisy edges. Namely, *an SGNN fails to learn proper representations for nodes in an unbalanced triangle.* For this, we (i) extend the classical Weisfeiler-Lehman graph isomorphism test (WL-test) [45] to the signed graph settings. In particular, the extended WL-test characterizes the expressiveness of SGNN using (k-hop) ego trees; we then (ii) we show, through the means of ego trees, that an SGNN would violate conditions of proper representations for nodes in an unbalanced triangle.

4.1 Noisy Edges and Balance Degree

The notion of balance degree captures the extent to which triangles in a signed graph are balanced.

Definition 4.1. The *balance degree* [1] of a signed graph is defined by:

$$D_3(\mathcal{G}) = \frac{O_3^+}{O_3} = \frac{\mathrm{Tr}\left(A^3\right) + \mathrm{Tr}\left(|A|^3\right)}{2\,\mathrm{Tr}\left(|A|^3\right)} \tag{1}$$

where $|A|$ is the element-wise absolute value of the adjacency matrix A and $\mathrm{Tr}(A)$ denotes the trace of A.

Our goal is to demonstrate the relationship between noisy edges and the balance degree of signed graphs. For this we choose four real-world datasets: Epinions, Slashdot, Bitcoin_Alpha and Bitcoin_OTC (See Sec 6 for details). To simulate noisy edges on signed graphs, we select a set of edges at random from the input signed graph and flip their signs. The ratio of edges we select indicates the level of noise introduced to the graph. We then compute the balance degree in the resulting signed graph and plot the result in Fig. 1(b). It is then apparent that the balance degree of all four datasets drops drastically as more noisy edges are introduced.

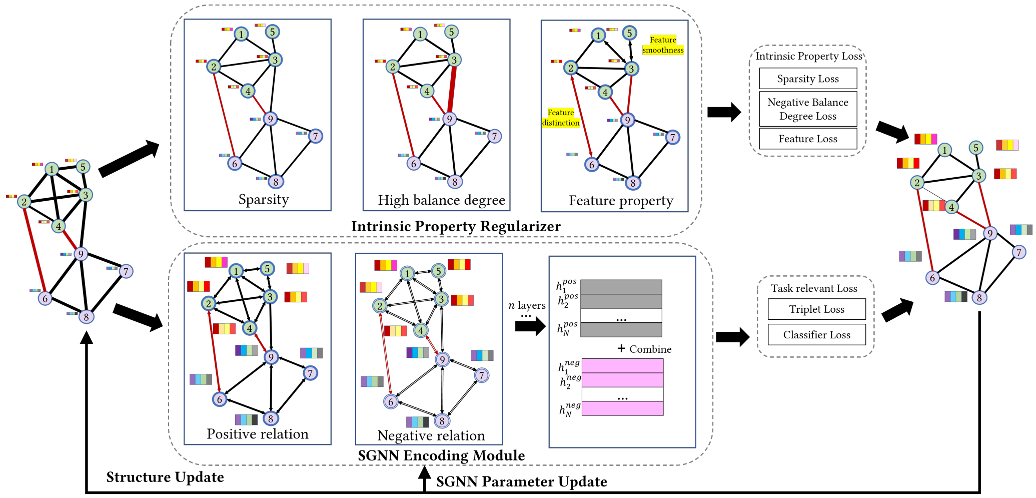

Figure 2: The overall architecture of RSGNN. Black lines represent positive edges and red lines represent negative edges.

Remark. We focus on *random noise*, instead of adversarial noise, in our work for two reasons. First, our intention is to enhance SGNN to defend against random noise introduced during the data collection process which is very different from adversarial noise introduced by a malicious attacker. Then, for signed graph there has not been any established attack model that is introduced. Existing adversarial graph attack methods such as nettack [52] and metattack [53] require node labels and node features, as they mostly target node classification tasks, and cannot be applied here.

4.2 SGNN and WL-test

We next demonstrate how SGNNs may fail to learn a proper representation for nodes in unbalanced triangles. The following definition is consistent with the intended goal of SGNN described above.

Definition 4.2 (Proper representations of nodes). Given a signed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$, an SGNN model $f_\theta \colon \mathcal{V} \to H$ and any non-negative distance metric $dist \colon H \times H \to \mathbb{R}^+$, we call $h_i = f_\theta(v_i)$ a proper representation of any node $v_i \in \mathcal{V}$ if the following conditions all satisfy:

(a) There exist $\epsilon > 0$ such that for any $v_j \in \mathcal{N}_i^-$ and $h_j = f_\theta(v_j)$, $dist(h_i, h_j) > \epsilon$;
(b) For any $v_j \in \mathcal{N}_i^+$, $v_k \in \mathcal{N}_i^-$ and $h_j = f_\theta(v_j)$, $h_k = f_\theta(v_k)$, $dist(h_i, h_j) < dist(h_i, h_k)$,

We now give a concise introduction to the aggregation mechanism for SGNNs. Essentially, mainstream SGNN models such as SGCN [10] and SNEA [30] adopt the following mechanism. The representation of a node v_i at a given layer ℓ is defined as

$$h_i^{(\ell)} = [h_i^{pos(\ell)}, h_i^{neg(\ell)}]$$

where $h_i^{pos(\ell)}$ and $h_i^{neg(\ell)}$ respectively denote the positive and negative representation vectors of node $v_i \in \mathcal{V}$ at the ℓth layer, and $[,]$ denotes the concatenation operation. The updating process at layer $\ell = 1$ could be written as:

$$\begin{aligned} a_i^{pos(1)} &= \text{AGGREGATE}^{(1)}\left(\left\{h_j^{(0)} : v_j \in \mathcal{N}_i^+\right\}\right) \\ h_i^{pos(1)} &= \text{COMBINE}^{(1)}\left(h_i^{(0)}, a_i^{pos(\ell)}\right) \\ a_i^{neg(1)} &= \text{AGGREGATE}^{(1)}\left(\left\{h_j^{(0)} : v_j \in \mathcal{N}_i^-\right\}\right) \\ h_i^{neg(1)} &= \text{COMBINE}^{(1)}\left(h_i^{(0)}, a_i^{neg(\ell)}\right) \end{aligned} \tag{2}$$

And for $\ell > 1$ layer, we have:

$$\begin{aligned} a_i^{pos(\ell)} &= \text{AGGREGATE}^{(\ell)}\left(\left\{h_j^{pos(\ell-1)} : v_j \in \mathcal{N}_i^+\right\}, \left\{h_j^{neg(\ell-1)} : v_j \in \mathcal{N}_i^-\right\}\right) \\ h_i^{pos(\ell)} &= \text{COMBINE}^{(\ell)}\left(h_i^{pos(\ell-1)}, a_i^{pos(\ell)}\right) \\ a_i^{neg(\ell)} &= \text{AGGREGATE}^{(\ell)}\left(\left\{h_j^{neg(\ell-1)} : v_j \in \mathcal{N}_i^+\right\}, \left\{h_j^{pos(\ell-1)} : v_j \in \mathcal{N}_i^-\right\}\right) \\ h_i^{neg(\ell)} &= \text{COMBINE}^{(\ell)}\left(h_i^{neg(\ell-1)}, a_i^{neg(\ell)}\right), \end{aligned} \tag{3}$$

Different from GNNs, SGNNs accommodate positive and negative edges using a two-part representation, and a more involved aggregation scheme. For example, when $\ell > 1$, the positive part of the representation for node v_i could aggregate information from the positive-representation of its positive neighbors and the negative-representation of its negative neighbors. Next, we will prove that based on the above message-passing mechanism of SGNN, nodes in signed graphs with *isomorphic ego trees* will have the same representation.

Definition 4.3 (Signed graph isomorphism). Two signed graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are *isomorphic*, denoted by $\mathcal{G}_1 \cong \mathcal{G}_2$, if there exists a bijection $\psi \colon \mathcal{V}_{\mathcal{G}_1} \to \mathcal{V}_{\mathcal{G}_2}$ such that, for every pair of vertices $v_i, v_j \in \mathcal{V}_{\mathcal{G}_1}$, $e_{ij} \in \mathcal{E}_1$, if and only if $e_{\psi(v_i),\psi(v_j)} \in \mathcal{E}_2$ and $\sigma(e_{ij}) = \sigma(e_{\psi(v_i),\psi(v_j)})$.

We further define the balanced and unbalanced reach set of a node, following similar notions in [10].

Definition 4.4 (Balanced / Unbalanced reach set). For a node v_i, its *ℓ-balanced (unbalanced) reach set $\mathcal{B}_i(\ell)$ ($\mathcal{U}_i(\ell)$))* is defined as a set of nodes that have even (odd) negative edges along a path that connects v_i, where ℓ refers to length of this path. More details are in Appendix A.

Weisfeiler-Lehman (WL) graph isomorphism test [45] is a powerful tool to check if two unsigned graphs are isomorphic. A WL test recursively collects the connectivity information of the graph and maps it to the feature space. If two graphs are isomorphic, they will be mapped to the same element in the feature space. A multiset generalizes a set by allowing multiple instances for its elements. During the WL-test, a multiset is used to aggregate labels from neighbors of a node in an unsigned graph. More precisely, for a node v_i, in the ℓth iteration, the node feature is the collection of node neighbors $\left\{\left(X_i^{(\ell)}, \{X_j^{(\ell)} : v_j \in \mathcal{N}_i\}\right)\right\}$, where $X_i^{(\ell)}$ denotes the feature (label) of node v_i [48].

We now extend WL test to signed graph: For a node v_i in a signed graph, we use two multisets to aggregate information from v_i's balanced reach set and unbalanced reach set separately. In this way, each node in a signed graph has two features $X_i(\mathcal{B})$ and $X_i(\mathcal{U})$ aside from the initial features unlike in an unsigned graph.

Definition 4.5 (Extended WL-test For Signed Graph). Based on the message-passing mechanism of SGNNs in Equations 2 and 3, the process of extended WL-test for signed graph can be defined as below. For the first-iteration update, i.e. $\ell = 1$, the *WL node label* of a node v_i is $(X_i^{(1)}(\mathcal{B}),\ X_i^{(1)}(\mathcal{U}))$ where:

$$
\begin{aligned}
X_i^{(1)}(\mathcal{B}) &= \varphi\left(\left\{\left(X_i^{(0)}, \{X_j^{(0)} : v_j \in \mathcal{N}_i^+\}\right)\right\}\right) \\
X_i^{(1)}(\mathcal{U}) &= \varphi\left(\left\{\left(X_i^{(0)}, \{X_j^{(0)} : v_j \in \mathcal{N}_i^-\}\right)\right\}\right)
\end{aligned}
\tag{4}
$$

For $\ell > 1$, the *WL node label* of v_i is $(X_i^{(\ell)}(\mathcal{B}), X_i^{(\ell)}(\mathcal{U}))$ where:

$$
\begin{aligned}
X_i^{(\ell)}(\mathcal{B}) = \varphi\Big(\Big\{(X_i^{(\ell-1)}(\mathcal{B}), &\{X_j^{(\ell-1)}(\mathcal{B}) : v_j \in \mathcal{N}_i^+\}, \\
&\{X_j^{(\ell-1)}(\mathcal{U}) : v_j \in \mathcal{N}_i^-\})\Big\}\Big) \\
X_i^{(\ell)}(\mathcal{U}) = \varphi\Big(\Big\{(X_i^{(\ell-1)}(\mathcal{U}), &\{X_j^{(\ell-1)}(\mathcal{U}) : v_j \in \mathcal{N}_i^+\}, \\
&\{X_j^{(\ell-1)}(\mathcal{B}) : v_j \in \mathcal{N}_i^-\})\Big\}\Big)
\end{aligned}
\tag{5}
$$

where φ is an injective function.

The extended WL-test above is defined with a similar aggregation and update process as a SGNN, and thus can be used to capture the expressibility of the SGNN.

Definition 4.6. A *(rooted) k-hop ego-tree* is a tree built from a root node v_i (level-0) in $\mathcal{G}$ inductively for k levels: From any node v_j at level $\ell \geq 0$, create a copy of each neighbor $v_p \in \mathcal{N}_j$ at level $\ell + 1$ and connect v_j and v_p with a new tree edge whose sign is $\sigma(e_{j,p})$.

Theorem 4.7. *Suppose two ego-trees τ_1 and τ_2 are isomorphic. An SGNN $\mathcal{A}$ applied to τ_1 and τ_2 will produce the same node embedding for the roots of these ego-trees.*

Proof. Suppose ego-tree τ_1 and τ_2 are two isomorphic signed graphs. After ℓ iterations, we have $\mathcal{A}(root(\tau_1)) \neq \mathcal{A}(root(\tau_2))$, where $root(\tau)$ represents the root of τ. As τ_1 and τ_2 are isomorphic, they have the same extended WL node labels for iteration ℓ for any $\ell = 0, \ldots, k-1$, i.e. $X_1^{(\ell)}(\mathcal{B}) = X_2^{(\ell)}(\mathcal{B})$ and $X_1^{(\ell)}(\mathcal{U}) = X_2^{(\ell)}(\mathcal{U})$,

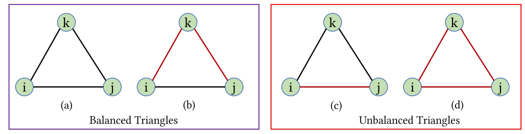

Figure 3: Four isomorphism types of triangles. Black and red lines represent positive and negative edges, resp.

as well as the same collection of neighbor labels, i.e.

$$
\begin{aligned}
&\left(X_1^{(\ell)}(\mathcal{B}), \{X_j^{(\ell)}(\mathcal{B}) : v_j \in \mathcal{N}_1^+\}, \{X_1^{(\ell)}(\mathcal{U}) : v_j \in \mathcal{N}_1^-\}\right) = \\
&\qquad \left(X_2^{(\ell)}(\mathcal{B}), \{X_j^{(\ell)}(\mathcal{B}) : v_j \in \mathcal{N}_2^+\}, \{X_2^{(\ell)}(\mathcal{U}) : v_j \in \mathcal{N}_2^-\}\right) \\
&\left(X_1^{(\ell)}(\mathcal{U}), \{X_j^{(\ell)}(\mathcal{U}) : v_j \in \mathcal{N}_1^+\}, \{X_1^{(\ell)}(\mathcal{B}) : v_j \in \mathcal{N}_1^-\}\right) = \\
&\qquad \left(X_2^{(\ell)}(\mathcal{U}), \{X_j^{(\ell)}(\mathcal{U}) : v_j \in \mathcal{N}_2^+\}, \{X_2^{(\ell)}(\mathcal{B}) : v_j \in \mathcal{N}_2^-\}\right)
\end{aligned}
\tag{6}
$$

Otherwise, the extended WL test should have obtained different node labels for τ_1 and τ_2 at iteration $\ell + 1$. As the ψ is an injective function, the extended WL test always relabels different extended multisets into different labels. As the SGNN and extended WL test follow the similar aggregation and rebel process, if $X_1^{(\ell)} = X_2^{(\ell)}$, we can have $h_1^{(\ell)} = h_2^{(\ell)}$. Thus, $\mathcal{A}(root(\tau_1)) = \mathcal{A}(root(\tau_2))$, we have reached a contradiction. □

We now turn our attention to triangles. Figure 3 shows all four isomorphism types of triangles. Note that if ego-trees τ_1 and τ_2 of two triangles are not isomorphic, the WL-test node label of $root(\tau_1)$ and $root(\tau_2)$ will be different. And thus these two roots will be mapped to different embeddings by an SGNN.

Theorem 4.8. *An SGNN cannot learn proper representations for nodes from unbalanced triangles.*

Proof. We only need to consider triangles (c) and (d) in Figure 3. For simplicity, we only discuss (c). The other unbalanced situation (d) follows a similar argument (See Appendix B).

For (c), we construct the 2-hop ego-trees of node v_i, v_j and v_k in Figure 4, where places the positive neighbors to the left side and negative neighbors to the right side. It is clear to see τ_i and τ_j are isomorphic. Based on Theorem 4.7, they will be mapped to the same embeddings. On the contrary, as τ_i and τ_k are not isomorphic, they will be mapped to different embeddings. Therefore, we can get $dist(h_i, h_j) \leq dist(h_i, h_k)$, where $dist$ is a distance metric, which means the representation of nodes connected with negative edges are closer than nodes connected with positive edges. Based on Definition 4.2, the learned h_i, h_j, h_k are not proper representation for v_i, v_j and v_k. □

The theorem above gives us an intuitive explanation on how noisy edges deteriorates the performances of an SGNN: As noisy edges increases the amount of unbalanced triangles, and the SGNN would fail to distinguish nodes connected by negative and positive edges in an unbalanced triangle, the produced embedding would not reflect the intended meaning.

5 PROPOSED METHOD

Our analysis above confirms that noisy edges harms the performance of SGNN by creating unbalanced triangles. We now describe

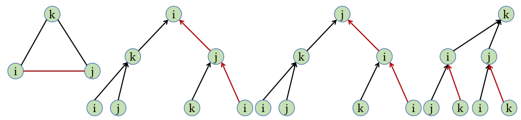

Figure 4: Ego-trees of situation (c) in Fig. 3

our RSGNN framework for defending against this negative impact. As shown in Figure 2, RSGNN adopts a dual architecture that alternatively denoises the graph and learns the node representations.

5.1 Intrinsic Property Regularizers

In particular, the RSGNN may be applied on any SGNN by incorporating three intrinstic property regularizers. These properties describe intrinsic properties of a denoised signed graph: (1) high balance degree; (2) feature smoothness (distinction); and (3) sparsity.

Sparsity. We assume the random noise is unnoticeable and minor perturbations to graphs, the denoised adjacency matrix S should be close to and as sparse as the original adjacency A. Then, we can formulate the above analysis as a structure learning problem [21]:

$$\arg\min_{\mathrm{S}\in\mathcal{S}} \mathcal{L}_0 = \|A - \mathrm{S}\|_F^2 + \mu\|\mathrm{S}\|_1, \text{ s.t., } \mathrm{S} = \mathrm{S}^\top. \tag{7}$$

The first term $\|\mathrm{A} - \mathrm{S}\|_F^2$ enforces the new adjacency S is close to A, and ℓ_1 norm ensures S is sparse. Since in this paper, we focus on undirected signed graphs, the learned adjacency S should be symmetric, i.e., $S = S^\top$. Hyper-parameter μ is used to control the contribution of the sparsity constraint.

High balance degree. Based on the above analysis, random noise can increase the rate of unbalanced triangles, and then worsen the performance of SGNNs. Thus, to alleviate this situation, one potential way is to learn a new adjacency matrix S with a higher triangle index which is calculated by Equation 1. Thus, this learning process can be written as:

$$\arg\min_{\mathrm{S}\in\mathcal{S}} \mathcal{L}_{\text{deg}} = -\beta T(\mathrm{S}), \tag{8}$$

where β is a hyper-parameter to control the importance of triangle index $T(\mathrm{S})$.

Feature smoothness. Feature smoothness is an intrinsic property for unsigned graphs, which implies connected nodes should share similar features [25]. But for signed graphs, edges have more complex semantic information. We extend it to signed graphs and naturally claim nodes connected with positive edges should share similar features (feature smoothness) and nodes connected with negative edges should have distinctive features (feature distinction). We name this property for signed graphs as *feature property* and formulate this process as:

$$\mathcal{L}_f = \frac{1}{2}\sum_{\ell=1}^{L}\sum_{i,j=1}^{|\mathcal{V}|} e_{ij}\left(h_i^{(\ell)} - h_j^{(\ell)}\right)^2, \tag{9}$$

where L represents number of SGNN layers, $h_i^{(\ell)}$ denotes the vector representation for node v_i at the ℓth layer, e_{ij} represents the edge weight between node v_i and v_j. For simplicity, we use the vector representation output by the last (i.e., the Lth) layer as the final representations of nodes. The feature property constraint $\mathcal{L}_f$ can be also written in matrix form:

$$\mathcal{L}_f = \gamma_1 \operatorname{Tr}\left(H^\top \mathbf{L}_{pos} H\right) - \gamma_2 \operatorname{Tr}\left(H^\top \mathbf{L}_{neg} H\right), \tag{10}$$

where $\mathbf{L}_{pos}$ ($\mathbf{L}_{neg}$) indicates the laplacian matrix for positive (negative) denoised adjacency matrix S_{pos} (S_{neg}). $S_{pos} = (|S| + S)/2$ and $S_{neg} = (|S| - S)/2$, where $|S|$ represents the element-wise absolute value of matrix S. Then, $\mathbf{L}_{pos} = D_{pos} - S_{pos}$ and $\mathbf{L}_{neg} = D_{neg} - S_{neg}$. $D_{pos}(D_{neg})$ indicates the degree matrix of positive (negative) adjacency matrix $S_{pos}(S_{neg})$. H represents the final node representation matrix with each row denoting the representation of a node. γ_1, γ_2 are predefined hyper-parameters.

5.2 Encoding Module

As a model agnostic framework, RSGNN can adopt any SGNN model, e.g., e.g., SGCN [10], SNEA [30] and GS-GNN [31], to extract node embeddings in the encoding module. In our experiments (Section 6), we will test our model using two implementations which are based on SGCN and SNEA, respectively. More details are in Appendix C.

5.3 Objective Function of RSGNN

Based on the description above, the final loss function of RSGNN is as follows:

$$\begin{aligned}\arg\min_{\mathrm{S}\in\mathcal{S},\theta} \mathcal{L} =& \mathcal{L}_0 + \mathcal{L}_{\text{deg}} + \mathcal{L}_f + \mathcal{L}_{\text{task}} \\ =& \|A - \mathrm{S}\|_F^2 + \mu\|\mathrm{S}\|_1 - \beta T(\mathrm{S}) \\ &+ \gamma_1 \operatorname{Tr}\left(H^\top \mathbf{L}_{\text{pos}} H\right) - \gamma_2 \operatorname{Tr}\left(H^\top \mathbf{L}_{\text{neg}} H\right) \\ &+ \zeta \mathcal{L}_{SGNN}(\theta, \mathrm{S}, H^{(0)}) \\ &\text{s.t., } \mathrm{S} = \mathrm{S}^\top\end{aligned} \tag{11}$$

We adopt a similar optimization strategy as in [21], an alternating schema to iteratively update θ and S. As $S_{ij} \in [-1, 1]$, we clamp $S_{ij} < -1$ to -1 and $S_{ij} > 1$ to 1. We denote this operation as $\text{Clamp}(S)$, the optimization algorithm is shown in Algorithm 1.

Algorithm 1: RSGNN

Data: Adjacency matrix A, Node attribute matrix X, Hyper-parameters η, η'
Result: Learned adjacency matrix S, SGNN parameters θ

```
Initialized S ← A
Randomly initialize θ
while Stopping condition is not met do
    L_0 ← ||A − S||_F^2 + μ||S||_1
    L_deg ← −βT(S)
    L_f ← γ1 Tr(H^T L_pos H) − γ2 Tr(H^T L_neg H)
    L_task ← ζ L_SGNN(θ, S, H^(0))
    L ← L_0 + L_deg + L_f + L_task
    S ← S − η ∇_S(L)
    S ← Clamp(S)
    for i = 1 to m do
        θ ← θ − η' ∂L_task(θ,S,X)/∂θ
return S, θ
```

6 EXPERIMENTS

In this section, we conduct experiments on real-world datasets to demonstrate the effectiveness of RSGNN to increase the robustness of SGNNs against random noise in link sign prediction and compare it with state-of-the-art methods in unsigned graph representation methods and signed graph representation methods. We will answer the following questions:

- **Q1**: Can RSGNN increase the robustness of SGNNs?

Table 1: Link sign prediction results with AUC and Binary-F1 between SGCN and RSGNN+SGCN.

Dataset	Ptb(%)	SGCN		RSGNN + SGCN	
		AUC	F1	AUC	F1
Bitcoin _OTC	0	0.8456	**0.9417**	**0.8501**	0.9369
	10	0.7843	0.8195	**0.8200**	**0.8996**
	20	0.7433	0.7856	**0.7777**	**0.8496**
	25	0.6965	0.7348	**0.7466**	**0.8060**
Bitcoin _Alpha	0	**0.8430**	0.9443	0.8347	**0.9457**
	10	0.7549	0.8258	**0.7768**	**0.8811**
	20	0.7157	0.7621	**0.7215**	**0.8525**
	25	0.6957	0.7334	**0.7082**	**0.8336**

Table 2: Link sign prediction results with AUC and Binary-F1 between SNEA and RSGNN+SNEA.

Dataset	Ptb(%)	SNEA		RSGNN + SNEA	
		AUC	F1	AUC	F1
Bitcoin _OTC	0	0.8610	0.9142	**0.8732**	**0.9235**
	10	0.7732	0.8695	**0.8032**	**0.8921**
	20	0.7172	0.7581	**0.7569**	**0.8326**
	25	0.6832	0.7131	**0.7326**	**0.8020**
Bitcoin _Alpha	0	**0.8510**	0.9245	0.8347	**0.9334**
	10	0.7492	0.8133	**0.7827**	**0.8623**
	20	0.7037	0.7422	**0.7269**	**0.8324**
	25	0.6732	0.7125	**0.7033**	**0.8032**

- **Q2**: How does RSGNN perform compared with the state-of-the-art methods with noisy edges?
- **Q3**: How would the different intrinsic properties affect the performance of RSGNN?

6.1 Experimental Settings

Datasets. We conduct experiments on four public real-world datasets, i.e., Epinions, Slashdot, Bitcoin_Alpha and Bitcoin_OTC. See Appendix D for detailed statistics. It is noticeable that all datasets are very sparse. More details are in Appendix E.

Baselines. To validate the effectiveness of RSGNN against noisy edges, we compare it with state-of-the-art methods in the field of unsigned graph representation learning (i.e., GCN and GAT), signed graph representation without noise-tolerant properties (i.e., SiNE, SGCN and SNEA) and signed graph representation methods with noise-tolerant properties (i.e., SGCL and GS-GNN). As existing signed graph datasets lack node labels and features and the most robust GNN methods [8, 21] target at node classification task, they cannot be applied to signed graph representation learning, thus, in this paper, we do not use them as baselines. More details about the baselines are in Appendix F.

Hyper-parameters setting. We follow the hyper-parameter setting suggestions by those papers and set the embedding dimension to 64 for all the baselines to achieve a fair comparison. The proposed model RSGNN is implemented by PyTorch [32] and PYG [12]. As signed graph datasets do not collect node features, we use the top 64-dimensional vectors corresponding to the smallest eigenvalues of the signed Laplacian matrix as node features proposed in [26]. The value of β which controls the negative balance degree is set to 1. The value of γ_1 and γ_2 which control the contribution of feature smoothness and feature distinction are both set to 1 and 2. The value of μ is set 5e-4 and value of ζ is set 1. λ is set 5 as that in [10]. Set ω_s as $[1, 1, 1]$. m is set 2.

Random noise. Following the remark in Section 4.1, we only focus on random noise. We introduce random noisy edges by randomly flipping a certain proportion of edge signs. In the experiment, the perturbation rate is from 0.05 to 0.25.

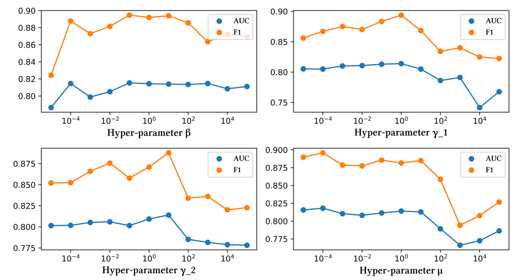

Figure 5: Parameter analysis on Bitcoin_OTC dataset

Task and evaluation metrics. Following the previous works on signed graph representation learning [10, 19, 30, 31], we adopt link sign prediction as our task, i.e., predicting the sign of unseen edges. We randomly split the edges into a training set and a testing set with a ratio 8:2. For evaluation metrics, we adopt the following two metrics: area under the curve (AUC) and binary average F1 score (Binary-F1) as [30]. Higher values represent better performance.

6.2 Experiment Results

Performance of RSGNN-enhanced SGNNs (Q1). To answer **Q1**, i.e., verifying whether RSGNN can enhance the SGNN models as the encoding module. Put succinctly, we test RSGNN with different encoding modules, namely SGCN [10] (graph convolutional network [25] based) and SNEA [30] (graph attention network [40] based), respectively. Specifically, we inject random noise into the training set according to Section 6.1 and compare SGCN and SNEA with and without using the proposed RSGNN to improve the robustness. The results are shown in Table 1 (SGCN) and Table 2 (SNEA). More results are in Appendix G.

We make the following observations from the results. First, RSGNN clearly improves the performance of SGCN and SNEA with random noise injected. As more noise is injected, the improvement becomes more obvious. This means that RSGNN can indeed enhance the robustness of SGNNs. Second, RSGNN has a competitive performance even with no or small amount of noise injected. Our framework performs slightly worse than the corresponding SGNN on some datasets (e.g., Bitcoin_OTC) when no noise is injected. This is because SGNN works properly without noisy edges so that our robust learning framework does not help improve the performance. Overall, this experiment demonstrates the effectiveness of our method, which enhances the robustness of SGNNs with a structure learning framework based on the intrinsic features.

Performance of RSGNN against other baselines under random noise (Q2). To answer **Q2**, we compare our method RSGNN with other state-of-the-art graph representation methods. For the sake of fairness, we choose three different classes of graph representation learning methods, namely unsigned graph network embeddings (unsigned NE), signed graph embeddings (SNE) and noise-tolerant SNE. As currently public signed graph datasets lack node label and attribute information which are relied on by most robust GNN models [8, 21] and most robust unsigned GNN methods [51] only target at node classification methods, thus, we cannot directly apply them to link sign prediction task. Specifically, as there are no related methods directly targeting robust learning of SGNN, we choose two noise-tolerant SGNN models, i.e., SGCL [36]

Table 3: Link sign prediction performance with Binary-F1 under random noise effect.

Dataset	Ptb(%)	unsigned NE		SNE			Noise-tolerant SNE		Ours
		GCN	GAT	SiNE	SGCN	SNEA	SGCL	GS-GNN	RSGNN
Bitcoin_OTC	0	0.8432 ± 0.0429	0.8621 ± 0.0325	0.8836 ± 0.0427	0.9417 ± 0.0201	0.9142 ± 0.0107	0.9518 ± 0.0089	**0.9523** ± 0.0150	0.9369 ± 0.0082
	10	0.7901 ± 0.0532	0.7922 ± 0.0410	0.8101 ± 0.0351	0.8195 ± 0.0210	0.8695 ± 0.0256	0.8821 ± 0.0116	0.8755 ± 0.0275	**0.8996** ± 0.0052
	20	0.7163 ± 0.0515	0.7348 ± 0.0420	0.7531 ± 0.0377	0.7856 ± 0.0238	0.7581 ± 0.0341	0.8077 ± 0.0209	0.7988 ± 0.0134	**0.8496** ± 0.0076
	25	0.7022 ± 0.0501	0.7182 ± 0.0344	0.7339 ± 0.0511	0.7348 ± 0.0231	0.7131 ± 0.0227	0.7533 ± 0.0109	0.7432 ± 0.0191	**0.8060** ± 0.0071
Bitcoin_Alpha	0	0.8717 ± 0.0331	0.8894 ± 0.0329	0.9307 ± 0.0378	0.9443 ± 0.0309	0.9245 ± 0.0216	**0.9622** ± 0.0230	0.9570 ± 0.0197	0.9457 ± 0.0102
	10	0.7842 ± 0.0430	0.8021 ± 0.0352	0.8201 ± 0.0315	0.8258 ± 0.0278	0.8133 ± 0.0251	0.8723 ± 0.0209	0.8409 ± 0.0252	**0.8811** ± 0.0116
	20	0.7018 ± 0.0269	0.7389 ± 0.0310	0.7542 ± 0.0326	0.7621 ± 0.0231	0.7422 ± 0.0220	0.7921 ± 0.0259	0.7881 ± 0.0189	**0.8525** ± 0.0059
	25	0.6847 ± 0.0310	0.7092 ± 0.0348	0.7133 ± 0.0271	0.7334 ± 0.0220	0.7125 ± 0.0278	0.7622 ± 0.0185	0.7623 ± 0.0192	**0.8336** ± 0.0074
Epinion	0	0.9009 ± 0.0442	0.9032 ± 0.0381	0.9127 ± 0.0355	0.9243 ± 0.0320	0.9227 ± 0.0229	0.9322 ± 0.0211	**0.9544** ± 0.0199	0.9351 ± 0.0085
	10	0.8102 ± 0.0433	0.8133 ± 0.0389	0.8272 ± 0.0356	0.8419 ± 0.0419	0.8322 ± 0.0240	0.8749 ± 0.0251	0.8537 ± 0.0109	**0.8831** ± 0.0065
	20	0.7289 ± 0.0326	0.7301 ± 0.0301	0.7544 ± 0.0334	0.7754 ± 0.0247	0.7832 ± 0.0201	0.8010 ± 0.0132	0.7842 ± 0.0102	**0.8365** ± 0.0123
	25	0.6981 ± 0.0310	0.7013 ± 0.0312	0.7219 ± 0.0285	0.7288 ± 0.0243	0.7311 ± 0.0298	0.7633 ± 0.0201	0.7522 ± 0.0203	**0.8149** ± 0.0104
Slashdot	0	0.8533 ± 0.0342	0.8629 ± 0.0365	0.8523 ± 0.0316	0.8821 ± 0.0305	0.8646 ± 0.0347	0.8951 ± 0.0201	**0.9082** ± 0.0207	0.8932 ± 0.0113
	10	0.8014 ± 0.0361	0.8033 ± 0.0325	0.8125 ± 0.0374	0.8017 ± 0.0268	0.8122 ± 0.0219	0.8114 ± 0.0209	0.8289 ± 0.0209	**0.8537** ± 0.0159
	20	0.7115 ± 0.0429	0.7231 ± 0.0429	0.7282 ± 0.0379	0.7322 ± 0.0321	0.7421 ± 0.0353	0.7582 ± 0.0209	0.7433 ± 0.0250	**0.8087** ± 0.0162
	25	0.6882 ± 0.0228	0.6939 ± 0.0247	0.7012 ± 0.0216	0.7154 ± 0.0261	0.7210 ± 0.0250	0.7361 ± 0.0194	0.7231 ± 0.0163	**0.7981** ± 0.0091

Table 4: The ablation study results of using different intrinsic properties of RSGNN.

Metric	Ptb(%)	$\mathcal{L}_{pos}$	$\mathcal{L}_{neg}$	$\mathcal{L}_{pos} + \mathcal{L}_{neg}$	$\mathcal{L}_{deg}$	All
AUC	10	0.7908	0.7898	0.7968	0.8008	**0.8200**
	15	0.7710	0.7682	0.7820	0.7841	**0.7857**
	20	0.7461	0.7637	0.7527	0.7664	**0.7777**
	25	0.7237	0.7317	0.7228	0.7306	**0.7466**
F1	10	0.8431	0.8494	0.8416	0.8536	**0.8996**
	15	0.8182	0.8044	0.8119	0.8578	**0.8703**
	20	0.7783	0.8081	0.7944	0.8416	**0.8496**
	25	0.7713	0.7517	0.7533	0.7981	**0.8060**

and GS-GNN [31] as our baselines. We only report the result using Binary-F1 as the metric, since other metrics show similar trends.

The results are shown in Table 3. In general, with the addition of noise, our method is significantly better than other baselines. Specifically, we have the following observations. **First**, the performance of the SGNN model declines quickly with more random noise. The experimental results prove our guess, i.e., GNNs are found to be very vulnerable to structure noises and SGNNs have the same problem. **Second**, SGCL and GS-GNN outperform our methods with slight perturbation, which really proves they are noise-tolerant. In terms of SGCL, proactively adding minor perturbation force the model to learn invariant and robust representations. But when the rate of noise increases, obviously this mechanism fails.

6.3 Ablation Study

The key part of RSGNN is the intrinsic property regularizer where high balance degree and feature properties are two exclusive properties of signed graphs. To answer **Q3**, we perform ablation studies to verify the effect of different property loss, i.e., $\mathcal{L}_{pos}$ (feature smoothness loss), $\mathcal{L}_{neg}$ (feature distinction loss) and $\mathcal{L}_{deg}$ (negative balance degree loss). We choose Bitcoin_OTC as our experimental dataset and randomly select 80% edges as the training set and 20% edges as the testing set. Then, we inject random noise to the training part which is varied from $\{0.1, 0.2, 0.25\}$. We report the results using AUC and Binary-F1. The results are shown in Table 4. From the results, we can see the $\mathcal{L}_{deg}$ plays a more important role than the other two losses, which verifies the negative impact of increasing unbalanced triangles on the performance. Overall, these property losses all contribute to our proposed RSGNN.

6.4 Parameter Analysis

In this subsection, we explore the sensitivity of hyper-parameters $\mu, \gamma_1, \gamma_2, \eta$, which corresponds to three parts of intrinsic properties constraints, i.e., high balance degree, feature properties, and sparsity. In the experiments, we alter the value of $\mu, \gamma_1, \gamma_2, \eta$ to see how they can affect the performance of RSGNN. More specifically, we vary the hyper-parameter from 1e-5 to 1e5. We report the results with metric AUC and Binary-F1 on the Bitcoin_OTC dataset with a random noise rate of 10%. The performance of RSGNN is illustrated in Figure 5. From the result, we can see proper settings of γ_1, γ_2, and η can boost the performance of RSGNN, large values will greatly hurt the model performance. As if we pay more attention to sparsity and feature properties, we will lose too much structure information which will result in inferior graph structure learning results. For hyper-parameter μ which controls the contribution of high balance degree, the chosen value from 0.1 to 100 is proper.

7 CONCLUSION

In this paper, we provide a theoretical explanation for the influence of noise on SGNNs and prove that current SGNN models cannot learn *proper* representations for nodes in unbalanced triangles which is a general limitation for SGNN. Then, we explore the properties of real-world signed graph to defend the negative effect of noise and propose a novel framework RSGNN which adopts a dual architecture that simultaneously denoises the graph and learns the node representations. We empirically validate our model on a number of signed graph benchmarks and demonstrate that our model achieves state-of-the-art performance. This is the first trial of robust learning in signed graph representation learning, we believe that there will be more research in the future.

ACKNOWLEDGMENTS

This research is supported by the Marsden Fund Council from Government funding (MFP-UOA2123), administered by the Royal Society of New Zealand and National Natural Science Foundation of China (NSFC) under the grant No. 62172040. The first author, fourth author and fifth author are supported by a PhD scholarship from China Scholarship Council.

REFERENCES

[1] Samin Aref and Mark C Wilson. 2018. Measuring partial balance in signed networks. *Journal of Complex Networks* 6, 4 (2018), 566–595.

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

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 891–900.

[4] Dorwin Cartwright and Frank Harary. 1956. Structural balance: a generalization of Heider's theory. *Psychological review* 63, 5 (1956), 277.

[5] Kai-Yang Chiang, Joyce Jiyoung Whang, and Inderjit S Dhillon. 2012. Scalable clustering of signed networks using balance normalized cut. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 615–624.

[6] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE transactions on knowledge and data engineering* 31, 5 (2018), 833–852.

[7] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. 2012. Sitting closer to friends than enemies, revisited. In *International Symposium on Mathematical Foundations of Computer Science*. Springer, 296–307.

[8] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 181–191.

[9] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

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

[11] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 2020. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 169–177.

[12] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[16] Yixuan He, Gesine Reinert, Songchao Wang, and Mihai Cucuringu. 2022. SSSNET: Semi-Supervised Signed Network Clustering. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 244–252.

[17] Fritz Heider. 1946. Attitudes and cognitive organization. *The Journal of psychology* 21, 1 (1946), 107–112.

[18] Junjie Huang, Huawei Shen, Qi Cao, Shuchang Tao, and Xueqi Cheng. 2021. Signed Bipartite Graph Neural Networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 740–749.

[19] Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. 2019. Signed graph attention networks. In *International Conference on Artificial Neural Networks*. Springer, 566–577.

[20] Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. 2021. SDGNN: Learning Node Representation for Signed Directed Networks. *arXiv preprint arXiv:2101.02390* (2021).

[21] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 66–74.

[22] Jinhong Jung, Woojeong Jin, Lee Sael, and U Kang. 2016. Personalized ranking in signed networks using signed random walk with restart. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 973–978.

[23] Jinhong Jung, Jaemin Yoo, and U Kang. 2020. Signed Graph Diffusion Network. *arXiv preprint arXiv:2012.14191* (2020).

[24] Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U Kang. 2018. Side: representation learning in signed directed networks. In *Proceedings of the 2018 World Wide Web Conference*. 509–518.

[25] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[26] Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, Jürgen Lerner, Ernesto W De Luca, and Sahin Albayrak. 2010. Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proceedings of the 2010 SIAM international conference on data mining*. SIAM, 559–570.

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

[28] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*. 641–650.

[29] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. 2010. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1361–1370.

[30] Yu Li, Yuan Tian, Jiawei Zhang, and Yi Chang. 2020. Learning signed network embedding via graph attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 4772–4779.

[31] Haoxin Liu, Ziwei Zhang, Peng Cui, Yafeng Zhang, Qiang Cui, Jiashuo Liu, and Wenwu Zhu. 2021. Signed Graph Neural Network with Latent Groups. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1066–1075.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[33] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[34] Yi Qian and Sibel Adali. 2014. Foundations of trust and distrust in networks: Extended structural balance theory. *ACM Transactions on the Web (TWEB)* 8, 3 (2014), 1–33.

[35] Changwon Seo, Kyeong-Joong Jeong, Sungsu Lim, and Won-Yong Shin. 2022. SiReN: Sign-Aware Recommendation Using Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[36] Lin Shu, Erxin Du, Yaomin Chang, Chuan Chen, Zibin Zheng, Xingxing Xing, and Shaofeng Shen. 2021. SGCL: Contrastive Representation Learning for Signed Graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1671–1680.

[37] Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Node classification in signed social networks. In *Proceedings of the 2016 SIAM international conference on data mining*. SIAM, 54–62.

[38] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

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

[41] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[42] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. 2017. Signed network embedding in social media. In *Proceedings of the 2017 SIAM international conference on data mining*. SIAM, 327–335.

[43] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Thirty-first AAAI conference on artificial intelligence*.

[44] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[45] Boris Weisfeiler and Andrei Leman. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series* 2, 9 (1968), 12–16.

[46] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610* (2019).

[47] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* 17, 2 (2020), 151–178.

[48] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.

[49] Shuhan Yuan, Xintao Wu, and Yang Xiang. 2017. SNE: signed network embedding. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 183–195.

[50] Quan Zheng and David B Skillicorn. 2015. Spectral embedding of signed networks. In *Proceedings of the 2015 SIAM international conference on data mining*. SIAM, 55–63.

[51] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1399–1407.

[52] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.

[53] Daniel Zügner and Stephan Günnemann. 2019. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412* (2019).

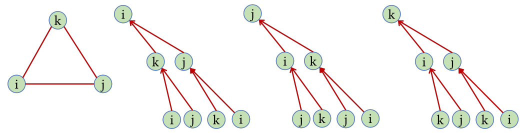

Figure 6: Ego-trees of situation (d)

APPENDIX

A DETAILS OF DEFINITION 4.4

The balanced (unbalanced) reach set extends positive (negative) neighbors from one-hop to multi-hop paths. In particular, the *balanced reach set* $\mathcal{B}_i(\ell)$ and the *unbalanced reach set* $\mathcal{U}_i(\ell)$) of a node v_i with path length $\ell = 1$ are defined as:

$$\begin{aligned}\mathcal{B}_i(\ell) &= \left\{v_j \mid v_j \in \mathcal{N}_i^+\right\}\\ \mathcal{U}_i(\ell) &= \left\{v_j \mid v_j \in \mathcal{N}_i^-\right\}\end{aligned} \tag{12}$$

For the path length $\ell > 1$:

$$\begin{aligned}\mathcal{B}_i(\ell) =& \left\{v_j \mid v_k \in \mathcal{B}_i(\ell-1) \text{ and } v_j \in \mathcal{N}_k^+\right\}\\ &\cup \left\{v_j \mid v_k \in \mathcal{U}_i(\ell-1) \text{ and } v_j \in \mathcal{N}_k^-\right\}\\ \mathcal{U}_i(\ell) =& \left\{v_j \mid v_k \in \mathcal{U}_i(\ell-1) \text{ and } v_j \in \mathcal{N}_k^+\right\}\\ &\cup \left\{v_j \mid v_k \in \mathcal{B}_i(\ell-1) \text{ and } v_j \in \mathcal{N}_k^-\right\}.\end{aligned} \tag{13}$$

B MORE DETAILS OF PROOF THEOREM 4.8

Proof. We further consider triangles (d) in Figure 3.

For situation (d) which is a unbalanced triangle, as is shown in Figure 6, apparently, the 2-hop ego-trees τ_i, τ_j and τ_k of node v_i, v_j and v_k are isomorphic, thus, they will be mapped to the same embeddings, i.e., $h_i = h_j = h_k$. $dis(h_i, h_j) = 0$ and $dist(h_i, h_k) = 0$, which do not conform the Definition 4.2. Thus, h_i, h_j and h_k are not proper representations. □

C DETAILS OF ENCODING MODULE

For completeness, we describe the implementation of the encoding module based on SGCN [10]. Here, the node representations are updated by aggregating information from different types of neighbors as follows. For the first aggregation layer $\ell = 1$:

$$\begin{aligned}H^{pos(1)} &= \sigma\left(\mathbf{W}^{pos(1)}\left[\mathcal{S}^+H^{(0)}, H^{(0)}\right]\right)\\ H^{neg(1)} &= \sigma\left(\mathbf{W}^{neg(1)}\left[\mathcal{S}^-H^{(0)}, H^{(0)}\right]\right)\end{aligned} \tag{14}$$

For the aggregation layer $\ell > 1$:

$$\begin{aligned}H^{pos(\ell)} &= \sigma\left(\mathbf{W}^{pos(\ell)}\left[\mathcal{S}^+H^{pos(\ell-1)}, \mathcal{S}^-H^{neg(\ell-1)}, H^{pos(\ell-1)}\right]\right)\\ H^{neg(\ell)} &= \sigma\left(\mathbf{W}^{neg(\ell)}\left[\mathcal{S}^+H^{neg(\ell-1)}, \mathcal{S}^-H^{pos(\ell-1)}, H^{neg(\ell-1)}\right]\right),\end{aligned} \tag{15}$$

where $H^{pos(\ell)}(H^{neg(\ell)})$ are positive (negative) part of representation matrix at the ℓth layer. $\mathcal{S}^+(\mathcal{S}^-)$ are the row normalized matrix of positive (negative) part of the denoised adjacency matrix S. $\mathbf{W}^{pos(\ell)}(\mathbf{W}^{neg(\ell)})$ are learnable parameters of positive (negative) part, and $\sigma(\cdot)$ is the activation function. [.] is the concatenation operation. After conducting message-passing for L layers, the final node representation matrix is $H^{(L)} = \left[H^{pos(L)}, H^{neg(L)}\right]$, which will be used in the downstream task.

Although there exist several analysis tasks for signed graphs, **link sign prediction** [10, 18, 20, 30] is still the main downstream task. This paper focuses on link sign prediction, but the node representation learned by our framework can be applied to other tasks. Specifically, we follow a similar loss function as SGCN [10] and define the task relevant loss function as:

$$\arg\min_{\mathrm{S}\in S} \mathcal{L}_{\text{task}} = \zeta \mathcal{L}_{\text{SGNN}}(\theta, \mathrm{S}, H^{(0)}), \tag{16}$$

where θ represents all parameters of the SGNN encoding model. ζ is a predefined hyper-parameter. S represents the denoised adjacency matrix and $H^{(0)}$ represents the input node attributes. More specifically, since link sign prediction is a classification problem, we use the weighted multinomial logistic regression (MLG) classifier as [10]. There exist three edge types positive, negative and unobserved, i.e., $s_{ij} \in \{+, -, ?\}$ and we construct trianglets $\mathcal{M}$ of form (v_i, v_j, s_{ij}). Besides, extended structural balance theory is used to constrain node representations, which makes (1) nodes connected with positive edges are closer than those connected with no edges; and (2) nodes connected with no edges are closer than those connected with negative edges in the representation space. The loss function of the SGNN module $\mathcal{L}_{\text{SGNN}}$ can be formalized as:

$$\begin{aligned}&\mathcal{L}_{\text{SGNN}} =\\ &-\frac{1}{\mathcal{M}} \sum_{(v_i,v_j,s)\in\mathcal{M}} \omega_s \log \frac{\exp\left(\left[h_i^{(\ell)}, h_j^{(\ell)}\right]\theta_s^{MLG}\right)}{\sum_{q\in\{+,-,?\}}\exp\left(\left[h_i^{(\ell)}, h_j^{(\ell)}\right]\theta_q^{MLG}\right)}\\ &+\lambda\left[\frac{1}{\left|\mathcal{M}_{(+,?)}\right|}\sum_{\substack{(v_i,v_j,v_k)\\ \in\mathcal{M}}} \max\left(0, \left(\left\|h_i^{(\ell)} - h_j^{(\ell)}\right\|_2^2 - \left\|h_i^{(\ell)} - h_k^{(\ell)}\right\|_2^2\right)\right)\right.\\ &\left.+\frac{1}{\left|\mathcal{M}_{(-,?)}\right|}\sum_{\substack{(v_i,v_j,v_k)\\ \in\mathcal{M}_{(-,?)}}} \max\left(0, \left(\left\|h_i^{(\ell)} - h_k^{(\ell)}\right\|_2^2 - \left\|h_i^{(\ell)} - h_j^{(\ell)}\right\|_2^2\right)\right)\right]\\ &+\mathrm{Reg}(\theta),\end{aligned} \tag{17}$$

where θ^{MLG} represents the parameters of the MLG classifier. ω_s represents the weight associated with the edge type. The term $\mathrm{Reg}(\theta)$ represents the regularization on the parameters θ. λ is the contribution control of the two part.

D KEY STATISTIC OF DATASETS

The key statistic of dataset is shown in Table 5.

Table 5: Key statistic of the Datasets.

Dataset	# Node	# Pos Edges	# Neg Edges	% Density
Epinions	16,992	276,309	50,918	0.2266%
Slashdot	33,586	295,201	100,802	0.0702%
Bitcoin_Alpha	3,784	12,729	1,416	0.1976%
Bitcoin_OTC	5,901	18,390	3,132	0.1236%

E DETAILED INFORMATION FOR DATASETS

- **Epinion**[2] is a who-trust-whom online social network of a general consumer review site Epinions.com. Members of the site can decide whether to *trust* each other.
- **Slashdot**[3] is a technology-related news website known for its specific user community which allows users to tag each other as friends or foes.
- **Bitcoin_alpha**[4] and **Bitcoin_OTC**[5] are who-trusts-who networks of people who using Bitcoin on a platform Bitcoin Alpha and Bitcoin OTC. Since Bitcoin users are anonymous, people give trust or not-trust tags to others in order to enhance security.

In these datasets, users rate others from -10 (completely distrust) to 10 (completely trust). We treat the marks bigger than 0 as positive edges and others as negative edges.

F DETAILED INFORMATION FOR BASELINES

- **GCN** [25], is an initial and representative GNN model designed for unsigned graphs which employs an efficient layer-wise propagation rule.
- **GAT** [40], adopts an attention-based architecture which can learn different weights to neighbors. It is also designed for unsigned graphs.
- **SiNE** [42], is a representative signed graph embedding method based on deep neural networks. The loss function is based on extended structural balance theory which drives nodes connected with positive edges closer than those connected with negative edges
- **SGCN** [10], generalizes GCN to signed graphs by designing a new information aggregator which is based on balance theory. It is also the encoding part of our RSGNN model.
- **SNEA** [30], generalizes GAT to signed graphs which adopts attention-based aggregators in message passing mechanism and is also based on the balance theory
- **SGCL** [36], generalizes graph contrastive learning to signed graphs, which employ graph augmentations to reduce the harm of interaction noise and enhance the model robustness.
- **GS-GNN** [31], beyond the balance theory assumption and adopt a dual GNN architecture to encoder both global and local information which claims to be noise-tolerant.

We use the author's released codes for SiNE[6], SNEA[7], SGCL[8], GS-GNN[9], and leverage the code from PyG[10] for GCN, GAT, SGCN. The unsigned graph embedding methods (i.e., GCN, GAT) are trained by edges without sign information and the same loss function $\mathcal{L}_{task}$ as SGCN [10]. As SiNE [43] is not trained in a end-to-end mode, we first obtain the node representations and train a binary logistic regression as the classifier. For other baselines, we adopt the end-to-end training.

G MORE RESULTS FOR PERFORMANCE OF RSGNN-ENHANCED SGNNS (Q1)

The performance of RSGNN-enhanced SGNNs on Epinion and Slashdot datasets are shown in Table 6 and 7.

Table 6: Link sign prediction results with AUC and Binary-F1 between SGCN and RSGNN+SGCN.

Dataset	Ptb(%)	SGCN		RSGNN+SGCN	
		AUC	F1	AUC	F1
Epinion	0	0.7981	0.9243	**0.8035**	**0.9351**
	10	0.7570	0.8419	**0.7756**	**0.8831**
	20	0.7212	0.7754	**0.7402**	**0.8365**
	25	0.6953	0.7288	**0.7139**	**0.8149**
Slashdot	0	**0.8583**	0.8821	0.8527	**0.8932**
	10	0.7732	0.8017	**0.8209**	**0.8537**
	20	0.7320	0.7322	**0.7531**	**0.8087**
	25	0.7016	0.7154	**0.7360**	**0.7981**

Table 7: Link sign prediction results with AUC and Binary-F1 between SNEA and RSGNN+SNEA.

Dataset	Ptb(%)	SNEA		RSGNN+SNEA	
		AUC	F1	AUC	F1
Epinion	0	**0.8547**	0.9227	0.8533	**0.9278**
	10	0.8066	0.8322	**0.8239**	**0.8832**
	20	0.7217	0.7832	**0.7805**	**0.8462**
	25	0.6891	0.7311	**0.7422**	**0.8049**
Slashdot	0	**0.8012**	0.8646	0.8003	**0.8721**
	10	0.7533	0.8122	**0.7748**	**0.8321**
	20	0.7109	0.7421	**0.7328**	**0.8077**
	25	0.6851	0.7210	**0.7199**	**0.7842**

[2]http://www.epinions.com
[3]http://www.slashdot.com
[4]http://www.btc-alpha.com/
[5]http://www.bitcoin-otc.com

[6]https://faculty.ist.psu.edu/szw494/codes/SiNE.zip
[7]https://github.com/liyu1990/snea
[8]https://github.com/xi0927/SGCL
[9]https://github.com/haoxin1998/GS-GNN
[10]https://github.com/pyg-team/pytorch_geometric