Binary Message Passing for Generalizable Semi-Supervised Graph Anomaly Detection

Jingyuan Zhang[1, 2], **Xin Wang**[1, 2], **Lei Yu**[1, 2], **Li Yang**[1,*], **Fengjun Zhang**[1,*]
[1]Institute of Software Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{zhangjingyuan2023, wangxin, yulei2022, yangli2017, fengjun}@iscas.ac.cn

Abstract

Graph Neural Networks (GNNs) have achieved impressive performance in semi-supervised graph anomaly detection (GAD). While many GNN variants have been developed for this task, they largely focus on advanced message aggregation schemes, leaving the message routing aspect underexplored. We argue that the commonly used broadcast-based routing can also hinder generalization, particularly in the presence of rare and structurally challenging (vertices with a high-degree) anomalies. To address this, we propose Binary Message Passing (BMP), a novel routing paradigm that models the message flow of each vertex as a binary tree (*BMP tree*), where vanilla graph convolution is decoupled by its left and right subtrees. Each vertex recursively gathers information from neighbors with higher anomaly probabilities within each subtree, thereby amplifying the propagation of anomaly information across the topology. The anomaly probabilities are estimated and updated by the model itself, enabling adaptive, self-supervised routing over iterations. Furthermore, combining multiple BMP trees into a *BMP forest* provides multi-scale structural context, enhancing the expressiveness of final vertex embeddings. Extensive experiments show that BMP improves detection performance under limited supervision while exhibiting better generalization across structurally diverse anomalies.

Code — https://github.com/Thankstaro/BMP
Extended version — https://github.com/Thankstaro/BMP

Introduction

In the era of the Web, graph data is ubiquitous. Similarly to general anomaly detection tasks, there exist anomalies in graphs that deviate significantly from most of the objects (Han, Kamber, and Pei 2011). Capturing these anomaly patterns is of great value (e.g., identifying fraudulent social accounts (Adewole et al. 2017), spam (Rao, Verma, and Bhatia 2021), fake reviews (Paul and Nikolaev 2021), financial fraud (Ngai et al. 2011), etc.), which has led to widespread attention on graph anomaly detection (GAD). In real-world scenarios, the limited availability of labeled samples for modeling—due to the large-scale nature of data and high labeling costs—combined with the complex interactions (i.e., graph topology) between samples, makes semi-supervised GAD more challenging than anomaly detection on traditional Euclidean data.

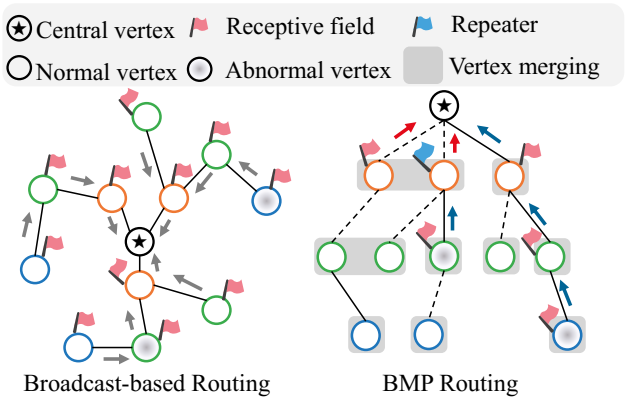


Figure 1: Comparison between broadcast-based routing and BMP routing. orange, green, and blue represent 1-hop, 2-hop, and 3-hop neighbors, respectively. Arrows indicate message passing. In BMP routing, the repeater forwards its right child information to its parent node.

Currently, graph neural networks (GNNs) have achieved remarkable success in semi-supervised graph mining tasks, primarily due to their message-passing mechanisms (Wu et al. 2020; Gilmer et al. 2017). In this framework, all vertices on the graph recursively receive information from their neighbors, allowing limited label information to propagate throughout the graph. However, in GAD, there is a common phenomenon of heterophily (Barranco, Lozares, and Muntanyola-Saura 2019; Zhu et al. 2020) (i.e., connections between normal and anomalous vertices), where anomalous vertices can receive information from normal ones through heterophilous edges. Moreover, as most neighbors of anomalous vertices are benign (Qiao and Pang 2023), this increases the risk of erroneous information injection.

While recent methods aim to mitigate this through advanced aggregation strategies (Liu et al. 2021; Shi et al. 2022; Tang et al. 2022; Chien et al. 2021), attribute adjustment (Duan et al. 2025; Dong et al. 2025) or graph sparsifi-

*Corresponding authors.

cation (Gao et al. 2023; Gong et al. 2023), they largely preserve broadcast-based routing. We observed several state-of-the-art methods and found that their generalization to high-degree anomalous vertices—those with a large number of neighbors—is relatively limited (in Figure 3). Such high-degree anomalies often exhibit strong heterophily, which in real-world scenarios may correspond to more destructive behaviors—for example, large-scale telecom fraud in financial transaction networks or coordinated cyberbullying on social platforms. This reflects the current insufficiency in addressing the heterophily issue. It is well known that broadcast-based routing can fully leverage vertex degree information, as vertices with higher degrees exhibit greater propagation efficiency (Newman 2003). This property makes it easy to discover communities or dense connections in graphs. However, for GAD, our focus is on the few anomalous vertices. This leads to a fundamental yet overlooked question: *is such broadcast-based message passing suitable for GAD?* Since most vertices are benign, anomalous vertex messages are easily drowned out in broadcast-based routing, which hinders the model's ability to learn anomaly patterns. The motivation of this work is to specifically enhance the propagation of anomalous information within the topology. Given that anomaly information originates from sparsely distributed anomalous vertices, a major challenge lies in effectively forwarding it under such class-imbalanced conditions.

Based on the above analysis, we propose a novel message-passing routing — Binary Message Passing (BMP). As shown in Figure 1, the neighborhood of each vertex is divided into two parts: dashed lines indicate connected vertices with anomaly probabilities lower than its own, while solid lines indicate the opposite. During message passing, the process follows the solid lines first (blue arrows) and then the dashed lines (red arrows). From top to bottom, we iteratively merge the vertices from the two sub-neighborhoods, as indicated by the shaded areas in the figure. In this way, the message flow is modeled as a binary tree, where the central vertex serves as the root node, and the vertex sets connected by dashed and solid lines are the left and right child nodes of their parent node. Based on this binary tree, the vanilla graph convolution is decoupled into two aforementioned steps, enabling the extraction of multi-order anomaly information from the left and right subtrees. In this second step (red arrows), the vertices with lower anomaly probabilities act as repeaters, helping two high-anomaly-probability vertices exchange messages.

Moreover, BMP operates in a self-supervised fashion, dynamically updating routing paths during training. By combining multiple such trees into the BMP forest, our approach further captures multi-scale structural context. BMP enables the receptive fields of vertices to become asynchronous, adapting to the complex interaction relationships on the graph. The receptive field of a vertex depends on the network depth, which can be adjusted by controlling the growth of the trees. Extensive experiments on real-world datasets validate the effectiveness of our method in detecting anomalous vertices with diverse structural characteristics, including both low- and high-degree cases. This work offers a new perspective for the architectural design of GAD models.

Related Work

Heterophilous Graph Neural Networks

For general vertex classification tasks, graph heterophily has garnered significant attention, including aspects such as heterophily measurement, GNN architecture design, and scalability. H2GCN (Zhu et al. 2020) separates the embeddings of the central vertex from its neighbors. CPGNN (Zhu et al. 2021) learns a compatibility matrix to model homophily and heterophily in the graph. GPRGNN (Chien et al. 2021) leverages multi-layer embeddings and assigns corresponding weights to different layers. GloGNN (Li et al. 2022) adopts a global perspective for neighbor aggregation, enhancing GNN performance on heterophilous graphs. ACM (Luan et al. 2022) adaptively utilizes multiple channels to extract richer local information. LRGNN (Liang et al. 2024) reformulates the label relationship matrix prediction as a low-rank matrix approximation problem. LD2 (Liao et al. 2024) decouples the graph propagation process, ensuring the scalability of GNNs on heterophilous graphs. CSBM-H (Luan et al. 2024) argues that the distinguishability of intra-class and inter-class nodes is key to GNN performance rather than homophily. (Platonov et al. 2023) introduces the concept of label informativeness, enabling a finer distinction among different types of heterophily.

GNN-based Graph Anomaly Detection

Anomalous vertex detection remains a vertex classification task, which not only involves heterophily but also faces the challenge of class imbalance, with the primary focus on the minority class. PC-GNN (Liu et al. 2021) addresses the class imbalance problem through a label-balanced sampler and a neighborhood sampler. From a spectral perspective, BWGNN (Tang et al. 2022) addresses the phenomenon of spectral "right shift" caused by anomalous disturbances. GHRN (Gao et al. 2023) treats labels as graph signals and prunes heterophilous edges by emphasizing high-frequency components. In the spatial domain, H2-FDetector (Shi et al. 2022) models homophilous and heterophilous connections to attract or repel neighbors. SparseGAD (Gong et al. 2023) sparsifies the original topology to enhance the robustness of GNNs. GTAN (Xiang et al. 2023) models anomaly patterns through risk propagation. PMP (Zhuo et al. 2024) proposes a node-level adaptive grouping aggregation strategy, while DiG-In-GNN (Zhang et al. 2024) mitigates the inconsistency of features and topology caused by fraudsters' camouflage. GAAP (Duan et al. 2025) and SpaceGNN (Dong et al. 2025) primarily focus on node attributes. GAAP mitigates improper attribute fusion by employing a binning strategy, whereas SpaceGNN projects node attributes into a non-Euclidean space. Both methods subsequently perform message aggregation to incorporate structural context. The aforementioned GAD methods often adopt complex message aggregation strategies, which, from the perspective of this paper, compensate for the use of simple broadcast-based message routing. However, with appropriate adjustments to message routing, simple aggregation strategies can also achieve desirable results.

Preliminaries

In this section, we first introduce the broadcast-based message passing routing on graphs, which is employed by the three most popular foundational GNNs. Then, we describe the task of graph anomaly detection, which is the focus of this paper. *In this paper, we refer to entities on the input graph as vertices, and those on the BMP trees as nodes, to avoid ambiguity.*

Message Passing Routing

Define an undirected attribute graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, Y)$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of vertices and edges, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the vertex attributes matrix. $Y \in \{0,1\}^n$ is the ground-truth label, where $Y_i = 0$ if v_i is benign and $Y_i = 1$ otherwise. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix. $n = |\mathcal{V}|$ is the number of vertices.

For a message-passing GNN (Gilmer et al. 2017), the l-th layer representation of vertex v is formualated as:

$$h_v^l = \mathrm{UPDATE}(\mathrm{AGG}(\boxed{\left\{h_u^{l-1} \mid u \in \mathcal{N}(v) \cup \{v\}\right\}})), \quad (1)$$

where $\mathcal{N}(v)$ is the set of neighbor vertices of v. The above formula (Luo, Shi, and Wu 2024) consists of three steps: message passing (the boxed $\boxed{\text{part}}$), message aggregation (AGG), and vertex updating (UPDATE).

$$\begin{aligned} h_v^l &= \sigma(\textstyle\sum_{\boxed{u \in \mathcal{N}(v) \cup \{v\}}} h_u \tilde{\mathbf{A}}_{vu} \boxed{h_u^{l-1}} \mathbf{W}^l), \\ h_v^l &= \sigma(\boxed{h_v^{l-1}} \mathbf{W}_1^l + (\tfrac{1}{|\mathcal{N}(v)|} \textstyle\sum_{\boxed{u \in \mathcal{N}(v)}} \boxed{h_u^{l-1}}) \mathbf{W}_2^l), \\ h_v^l &= \sigma(\textstyle\sum_{\boxed{u \in \mathcal{N}(v)}} \alpha_{vu}^l \boxed{h_u^{l-1}} \mathbf{W}^l), \end{aligned} \quad (2)$$

where α_{vu}^l is the attention of edge (v, u) in l-th layer, $\mathbf{W}$ is a trainable weight matrix, and σ is the activation function. The above formulas correspond sequentially to the three classic GNNs: Graph Convolutional Networks (GCN) (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and Graph Attention Networks (GAT) (Veličković et al. 2018). GCN aggregates neighbor messages by $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-1/2}$, while GraphSAGE separately initializes the learnable weight matrices for the central vertex's embedding and the mean of the neighbor embeddings, then performs summation. GAT computes attention for each edge and performs a weighted sum over the neighboring vertices.

The differences between these GNNs are mainly in AGG, while message passing is broadcasted according to the topology in all cases. After multiple layers of stacking, the broadcast range gradually increases until it covers the entire graph. Even if the topology is adjusted, broadcasting still occurs on the new topology. The focus of this paper is on the massage passing, where we no longer follow the broadcast-based routing to address the challenges of semi-supervised GAD.

Graph-based Anomaly Detection

In this paper, we focus on graph anomaly vertex detection in a semi-supervised setting. Specifically, given the attribute graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, Y)$ described above, we use the known vertices in the training set to learn an anomaly detector. The process involves encoding the vertices to be predicted, followed by classification to identify anomalous vertices. Therefore, GAD is essentially a binary classification task for vertices. In a semi-supervised setting, there may be a large number of unknown vertices, and effectively utilizing these unknown vertices can significantly improve performance.

Methodology

In this section, we will elaborate on Binary Message Passing (BMP) in detail. In short, the vertex anomaly probability is randomly initialized first, and then updated by the model in each iteration. A BMP tree is constructed based on the abnormal probability, then, by combining multiple BMP trees, we obtain the backbone — the Binary Message Passing Forest (BMP forest). Finally, we describe the optimization objective of BMP and its computational complexity.

Binary Message Passing Tree

Compared to broadcast-based routing, we model message routing using a single binary tree. This sparse routing aligns well with GAD, a binary classification task, while also achieving our goal of reinforcing anomalous messages in the routing. Let $(\mathbf{A}+\mathbf{I})^{up}$ and $(\mathbf{A}+\mathbf{I})^{low}$ be the upper and lower triangular matrices (including the main diagonal) of $(\mathbf{A}+\mathbf{I})$, respectively. $\tilde{\mathbf{A}}^{up} = (\mathbf{D}^{up})^{-1/2}(\mathbf{A}+\mathbf{I})^{up}(\mathbf{D}^{up})^{-1/2}$, $\tilde{\mathbf{A}}^{low} = (\mathbf{D}^{low})^{-1/2}(\mathbf{A}+\mathbf{I})^{low}(\mathbf{D}^{low})^{-1/2}$, $\mathbf{D}^{up}$ and $\mathbf{D}^{low}$ are the degree matrices of $(\mathbf{A}+\mathbf{I})^{up}$ and $(\mathbf{A}+\mathbf{I})^{low}$, respectively. For a p layers BMP, its q-th BMP tree $\mathcal{T}_{p,q}^*$ decouples the vanilla graph convolution into two independent operations: the right graph convolution $\mathcal{T}_{right}^*$ and the left graph convolution $\mathcal{T}_{left}^*$. For the central vertex v as the root node on the tree, we first perform p layers of $\mathcal{T}_{right}^*$ on the right subtree to obtain $\hat{h}_v$. Then, $\hat{h}_v$ is subjected to q layers of $\mathcal{T}_{left}^*$ on the left subtree. For $\mathcal{T}_{right}^*$ and $\mathcal{T}_{left}^*$, the l-th layer representations h_v^l and $\hat{h}_v^l$ are expressed as follows:

$$\begin{cases} \mathcal{T}_{right}^*(h_v^{l-1}) = \sigma(\textstyle\sum_{\boxed{u \in \mathcal{N}_{up}(v) \cup \{v\}}} \tilde{\mathbf{A}}_{vu}^{up} \boxed{h_u^{l-1}} \mathbf{W}_1^l) \\ \mathcal{T}_{left}^*(\hat{h}_v^{l-1}) = \sigma(\textstyle\sum_{\boxed{u \in \mathcal{N}_{low}(v) \cup \{v\}}} \tilde{\mathbf{A}}_{vu}^{low} \boxed{\hat{h}_u^{l-1}} \mathbf{W}_2^l), \end{cases} \quad (3)$$

where $\mathcal{N}_{up}(v)$ and $\mathcal{N}_{low}(v)$ represent the neighborhoods of v in $\tilde{\mathbf{A}}^{up}$ and $\tilde{\mathbf{A}}^{low}$, respectively. Since the vertices are sorted in ascending order according to anomaly probabilities P, the larger the anomaly probability, the larger the index. In $\tilde{\mathbf{A}}^{up}$, the index of the central vertex is less than or equal to that of its neighbors, while in $\tilde{\mathbf{A}}^{low}$, it is the opposite. Therefore, $\mathcal{T}_{right}^*$ is able to capture information from vertices with higher anomaly probabilities. Then, based on the results of $\mathcal{T}_{right}^*$, performing $\mathcal{T}_{left}^*$ can capture information from vertices with higher anomaly probabilities in the left subtree. p and q determine the height of the BMP tree (i.e., the receptive field of central vertex v). At this point, the message routing is no longer broadcast-based, but has been sparsified into a binary tree structure. We also highlight the message-passing component in Equation (3). Compared to

Algorithm 1: The Message Passing Paradigm of the Binary Message Passing Tree

Input: $\mathbf{X}'$, $\tilde{\mathbf{A}}^{up}$, $\tilde{\mathbf{A}}^{low}$, p, q, the trainable parameters $\{\mathbf{W}_1^1, \mathbf{W}_1^2, ..., \mathbf{W}_1^p\}$ and $\{\mathbf{W}_2^1, \mathbf{W}_2^2, ..., \mathbf{W}_2^q\}$
Output: $\mathcal{T}^*_{p,q}(\mathbf{X}')$

Let $\mathbf{H}^0 = \mathbf{X}'$.
for $i = 1$ to p **do**
  $\mathbf{H}^i = \sigma(\tilde{\mathbf{A}}^{up}\mathbf{H}^{i-1}\mathbf{W}_1^i)$. ▷ $\mathcal{T}^*_{right}$
end for
Let $\hat{\mathbf{H}}^0 = \mathbf{H}^p$.
for $i = 1$ to q **do**
  $\hat{\mathbf{H}}^i = \sigma(\tilde{\mathbf{A}}^{low}\hat{\mathbf{H}}^{i-1}\mathbf{W}_2^i)$. ▷ $\mathcal{T}^*_{left}$
end for
Let $\mathcal{T}^*_{p,q}(\mathbf{X}')$ = $\hat{\mathbf{H}}^q$.
return $\mathcal{T}^*_{p,q}(\mathbf{X}')$

existing works, the message aggregation is consistent with GCN, but the message passing follows a different routing. The specific message passing paradigm of the BMP tree can be seen in Algorithm 1.

For ease of analyzing message-passing routings, we temporarily omit the nonlinear activation units. When the number of network layers is L, broadcast-based routing can be represented as $\tilde{\mathbf{A}}^L$. In contrast, the message-passing routing of a single BMP tree is represented as $(\tilde{\mathbf{A}}^{low})^q(\tilde{\mathbf{A}}^{up})^p$. Taking Figure 1 as an example, these two routings are $\tilde{\mathbf{A}}^3$ and $(\tilde{\mathbf{A}}^{low})^1(\tilde{\mathbf{A}}^{up})^3$, respectively. Essentially, we designed a novel message-passing routing by decoupling vanilla graph convolutions and executing them separately on two subneighborhoods. Combined with the adjustment of vertex indices, this routing dynamically evolves.

Proposition 1. *If $\mathcal{G}$ is a connected graph with radius not greater than L, $\tilde{\mathbf{A}}^L$ ensures that all vertices have a receptive field of L. In contrast, for $(\tilde{\mathbf{A}}^{low})^q(\tilde{\mathbf{A}}^{up})^p$, when $p+q>1$, the receptive fields of vertices are not necessarily equal, with a maximum of $p+q$ and a minimum of 1.*

Detailed proof can be found in **Extended Version**. Define a triplet of vertex (v, u, w), where u is a common neighbor of v and w. As stated in Proposition 1, in BMP, when $P_v > P_u$ and $P_w > P_u$, v and w can communicate with each other, with u acting as a repeater. Otherwise, the two vertices cannot communicate. Proposition 1 shows that broadcast-based routing keeps the receptive fields of vertices **synchronized**, whereas in BMP, the receptive fields of vertices are **asynchronous**.

Binary Message Passing Forest

Based on the above BMP tree, we construct the BMP forest by combining multiple BMP trees with different values of q. The difference between each BMP tree lies in the range of perception of its left subtree. As q increases, the perception range of the left subtree grows larger, covering a wider range of vertices with higher anomaly probabilities. Finally, the vertex representations from all BMP trees are concatenated to obtain the final vertex embeddings. When p is fixed,

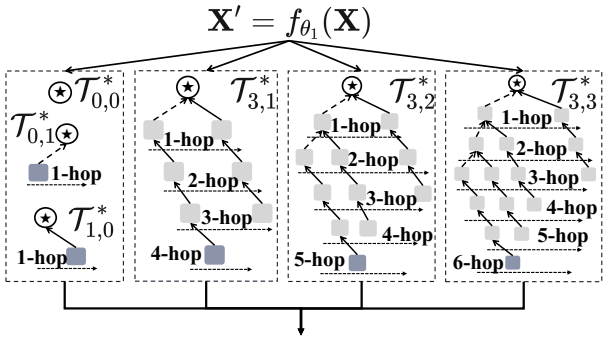


Figure 2: BMP Forest and Bagging Encoding ($p = 3$). $\mathcal{F}^*_0 = \{\mathcal{T}^*_{0,0}, \mathcal{T}^*_{0,1}, \mathcal{T}^*_{1,0}\}$. Each BMP tree has a different receptive field and can be regarded as a weak encoder, with the final embedding obtained through the bagging strategy. The farthest neighbors are marked in darker colors in each tree.

for the q-th BMP tree $\mathcal{T}^*_{p,q}$, it contains p layers of $\mathcal{T}^*_{right}$ and q layers of $\mathcal{T}^*_{left}$. The p-order BMP forest can then be represented as:

$$\mathcal{F}^*_p = \left\{\mathcal{T}^*_{p,1}, \mathcal{T}^*_{p,2}, ..., \mathcal{T}^*_{p,p}\right\}. \tag{4}$$

As shown in Figure 1, BMP trees rely on modeling each vertex's first-order neighbors, where an accurate bipartition of the neighborhood by the central vertex contributes to constructing reliable routing paths. Therefore, we additionally introduce three trees: $\mathcal{T}^*_{0,0}$, $\mathcal{T}^*_{0,1}$, and $\mathcal{T}^*_{1,0}$. $\mathcal{T}^*_{0,0}$ indicates the features without message passing and can be considered as information from 0-order neighbors (the central vertex itself). $\mathcal{T}^*_{0,1}$ and $\mathcal{T}^*_{1,0}$ cover the complete 1-order neighbor information. At this point, the complete BMP forest is represented as:

$$\begin{aligned}\mathcal{F}^* &= \left\{\mathcal{T}^*_{0,0}, \mathcal{T}^*_{0,1}, \mathcal{T}^*_{1,0}\right\} \cup \mathcal{F}^*_p \\ &= \left\{\mathcal{T}^*_{0,0}, \mathcal{T}^*_{0,1}, \mathcal{T}^*_{1,0}, \mathcal{T}^*_{p,1}, \mathcal{T}^*_{p,2}, ..., \mathcal{T}^*_{p,p}\right\}.\end{aligned} \tag{5}$$

Since the message routing differs across the BMP trees, the convolution operations within these trees do not share parameters. They independently perform the encoding operations, and then their respective encoded representations are concatenated to obtain the final representation $\mathbf{E}$.

$$\mathbf{E} = \mathrm{Concat}(\{\mathbf{X}', \mathcal{T}^*_{1,0}(\mathbf{X}'), \mathcal{T}^*_{0,1}(\mathbf{X}'), ..., \mathcal{T}^*_{p,p}(\mathbf{X}')\}), \tag{6}$$

where $\mathbf{X}' = f_{\theta_1}(\mathbf{X}) = \mathcal{T}^*_{0,0}(\mathbf{X}')$, f_{θ_1} is a multi-layer perceptron (MLP). Figure 2 illustrates the BMP forest with $p = 3$.

Proposition 2. *If $p+q \leq 1$, the message routing of the BMP is equivalent to 1-hop broadcast-based routing.*

The proof can be found in **Extended Version**. Proposition 2 illustrates the relationship between the two routings. When both have a receptive field of 1, they are equivalent. However, the two differ as the receptive field expands.

Optimization Objective

In BMP, the optimization objective is divided into two parts: The first part is the supervised loss, which calculates the

cross-entropy between the predicted probabilities of known vertices and their labels. The second part is the consistency loss (Wang et al. 2020; Feng et al. 2020), which aims to obtain more robust predicted probabilities P. Since the binary message routing is based on the sorted results of the predicted probabilities, robust predictions enhance the confidence of the sorting results. Let $P = f_{\theta_2}(\mathbf{E})$ be the predicted probabilities, the supervised loss $\mathcal{L}_{sup}$ is given by:

$$\mathcal{L}_{sup} = -\sum_{v \in \mathcal{V}_{train}} [Y_v \log P_v + (1 - Y_v) \log(1 - P_v)], \quad (7)$$

where $\mathcal{V}_{train}$ is the set of known vertices. The consistency loss is a loss function applied to the prediction results. It introduces noise to the encoded representations and then enforces that the noisy predictions are close to the original predictions. This regularizes both the encoder and the predictor, thereby improving the robustness of the predictions. In this work, we introduce masked noise by using a trainable module to mask out redundant features. The calculation of the masked representation $\tilde{\mathbf{E}}$ is as follows:

$$\begin{aligned} \mathbf{M} &= \text{sigmoid}(f_{\theta_3}(\mathbf{E})) \\ \tilde{\mathbf{E}} &= \mathbf{M} \odot \mathbf{E}, \end{aligned} \quad (8)$$

where $\odot$ denotes the element-wise multiplication, $\mathbf{M}$ is the masked noise and f_{θ_3} is the masking module, parameterized as a multilayer perceptron. Let $\tilde{P} = f_{\theta_2}(\tilde{\mathbf{E}})$ be the predicted result of $\tilde{\mathbf{E}}$, and the consistency loss $\mathcal{L}_{consis}$is then:

$$\mathcal{L}_{consis} = -\sum_{v \in \mathcal{V}'} [Y'_v \log \tilde{P}_v + (1 - Y'_v) \log(1 - \tilde{P}_v)], \quad (9)$$

where Y' is the pseudo-label. During the validation phase, the optimal classification threshold is obtained from the training set to derive Y'. The vertex set involved $\mathcal{V}'$ in $\mathcal{L}_{consis}$ is typically the set of unknown vertices, but in practice, known vertices are also included to improve the utilization of the data. Finally, the total loss of BMP is:

$$\mathcal{L} = \mathcal{L}_{sup} + \mathcal{L}_{consis} + \lambda \left\| \mathbf{M} \right\|_1, \quad (10)$$

where λ is the regularization coefficient. The third term of $\mathcal{L}$ serves as the regularization term. By adjusting λ, we can control the sparsity of $\mathbf{M}$. A larger λ results in sparser $\mathbf{M}$ and stronger regularization. As described above, BMP requires assigning new indices to vertices based on P before performing message passing. During iterations, to avoid frequent updates, we introduce a hyperparameter T, where indices are updated once every T iterations. The complete training paradigm of BMP is outlined in Algorithm 2.

Computational Complexity

In this section, we analyze the time complexity of BMP. According to Algorithm 2, the costs of f_{θ_1}, f_{θ_2}, and f_{θ_3} are all $\mathcal{O}(ndd')$, where d' is the embedding dimension. Line 12, i.e. Algorithm 1, incurs its primary cost in graph convolution, which requires $\mathcal{O}(m), m = |\mathcal{E}|$. Therefore, for $\mathcal{F}^*$, as per Equation (5), the total cost is $\mathcal{O}(p^2 m)$. The complexity of Line 6 is $\mathcal{O}(n \log n)$. In summary, the total cost of BMP is $\mathcal{O}(ndd' + p^2 m + n \log n)$.

Algorithm 2: The Training Paradigm of the BMP

Input: $\mathbf{X}$, $\mathbf{A}$, p, T, λ, a MLP f_{θ_1}, the predictor f_{θ_2}, the masking module f_{θ_3}, the trainable parameters of $\mathcal{F}^*$
Output: f_{θ_1}, f_{θ_2}, f_{θ_3} and $\mathcal{F}^*$

1: Initialize the vertex indices randomly.
2: Update $\mathbf{A}$ and $\mathbf{X}$ based on the new indices.
3: Compute $\tilde{\mathbf{A}}^{up}$ and $\tilde{\mathbf{A}}^{low}$ based on $\mathbf{A}$.
4: **for** $e = 1$ to $max\ epochs$ **do**
5: **if** $e\%T == 0$ **then**
6: Assign indices to vertices according to P.
7: Update $\mathbf{A}$ and $\mathbf{X}$ based on the new indices.
8: Compute $\tilde{\mathbf{A}}^{up}$ and $\tilde{\mathbf{A}}^{low}$ based on $\mathbf{A}$.
9: **end if**
10: Let $\mathbf{E} = \{\}$, $\mathbf{X}' = f_{\theta_1}(\mathbf{X})$.
11: **for** $\mathcal{T}^* \in \mathcal{F}^*$ **do**
12: $\mathbf{E} = \mathbf{E} \cup \{\mathcal{T}^*(\mathbf{X}')\}$. ▷ Algorithm 1
13: **end for**
14: Get $\mathbf{E}$ by Equation (6). ▷ Feature concatenation
15: Get $\tilde{\mathbf{E}}$ by Equation (8). ▷ Feature masking
16: Let $P = f_{\theta_2}(\mathbf{E})$, $\tilde{P} = f_{\theta_2}(\tilde{\mathbf{E}})$.
17: Optimize Equation (10) to update parameters.
18: **end for**
19: **return** f_{θ_1}, f_{θ_2}, f_{θ_3} and $\mathcal{F}^*$

Experiments

Datasets and Baselines

We conducted extensive evaluations on four real-world datasets: YelpChi (Rayana and Akoglu 2015), Amazon (McAuley and Leskovec 2013), T-Finance(Tang et al. 2022) and T-Social(Tang et al. 2022). YelpChi and Amazon are two multi-relation graphs used for detecting fraudulent reviews on products and hotels. T-Finance and T-Social are transaction and social graphs, respectively, designed to identify anomalous accounts within them. The **13 baselines** are divided into two categories: the first includes three classical GNNs: GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and GAT (Veličković et al. 2018). The second comprises the current state-of-the-art (SOTA) methods in GAD, including PC-GNN (Liu et al. 2021), BWGNN (Tang et al. 2022), H2-FDetector (Shi et al. 2022), GHRN (Gao et al. 2023), GTAN (Xiang et al. 2023), SparseGAD (Gong et al. 2023), PMP (Zhuo et al. 2024), DiG-In-GNN (Zhang et al. 2024), GAAP (Duan et al. 2025) and SpaceGNN (Dong et al. 2025).

Metrics and Setting

To accurately evaluate the effectiveness of the methods, we selected three commonly used metrics: **AUC-ROC (AUC)** (Davis and Goadrich 2006), **F1-macro**, and **Average Precision (AP)**. AUC provides a comprehensive assessment of model performance. F1-macro and AP focus more on detecting anomalies. Following previous work on GAD (Tang et al. 2022), in the semi-supervised setting, 1% of the data is used as the training set on YelpChi, Amazon, and T-Finance, while only 0.01% is used for T-Social. The validation and test sets are split by 1:2. Further details, including perfor-

Dataset(Anomaly Ratio)	**YelpChi**(14.53%)			**Amazon**(6.87%)			**T-Finance**(4.58%)		
Vertices / Edges	45,954 / 3,846,979			11,944 / 4,398,392			39,357 / 21,222,543		
Metric	AUC↑	F1-macro↑	AP↑	AUC↑	F1-macro↑	AP↑	AUC↑	F1-macro↑	AP↑
GCN$_{\text{(ICLR'17)}}$	$53.25_{\pm 0.4}$	$32.54_{\pm 8.7}$	$19.54_{\pm 0.3}$	$86.32_{\pm 0.5}$	$70.32_{\pm 2.7}$	$47.54_{\pm 2.3}$	$88.94_{\pm 0.2}$	$79.32_{\pm 2.2}$	$52.24_{\pm 2.7}$
GraphSAGE$_{\text{(NeurIPS'17)}}$	$71.55_{\pm 0.4}$	$60.32_{\pm 0.6}$	$32.44_{\pm 0.6}$	$88.67_{\pm 0.4}$	$85.21_{\pm 2.2}$	$74.42_{\pm 3.3}$	$88.44_{\pm 1.2}$	$78.32_{\pm 1.8}$	$48.57_{\pm 5.7}$
GAT$_{\text{(ICLR'18)}}$	$72.31_{\pm 1.4}$	$62.56_{\pm 1.7}$	$29.34_{\pm 1.7}$	$82.37_{\pm 3.2}$	$65.51_{\pm 8.4}$	$44.62_{\pm 9.3}$	$86.21_{\pm 3.9}$	$77.22_{\pm 4.9}$	$44.97_{\pm 8.9}$
PC-GNN$_{\text{(WWW'21)}}$	$73.21_{\pm 0.6}$	$63.44_{\pm 0.4}$	$36.14_{\pm 0.5}$	$90.17_{\pm 0.5}$	$85.64_{\pm 1.2}$	$75.92_{\pm 1.6}$	$91.05_{\pm 0.9}$	$83.72_{\pm 0.5}$	$72.39_{\pm 0.9}$
BWGNN$_{\text{(ICML'22)}}$	$73.07_{\pm 0.6}$	$61.54_{\pm 0.4}$	$31.24_{\pm 1.2}$	$89.34_{\pm 0.7}$	$88.92_{\pm 0.9}$	$78.69_{\pm 1.3}$	$92.04_{\pm 1.8}$	$87.02_{\pm 1.7}$	$76.35_{\pm 2.9}$
H2-FDetector$_{\text{(WWW'22)}}$	$73.05_{\pm 1.1}$	$62.88_{\pm 0.8}$	$32.66_{\pm 2.7}$	$84.21_{\pm 1.9}$	$71.52_{\pm 3.8}$	$48.59_{\pm 4.1}$	OOM	OOM	OOM
GHRN$_{\text{(WWW'23)}}$	$72.11_{\pm 0.9}$	$60.34_{\pm 0.3}$	$30.32_{\pm 1.5}$	$88.29_{\pm 1.8}$	$86.64_{\pm 2.1}$	$75.10_{\pm 3.8}$	$91.85_{\pm 0.9}$	$80.11_{\pm 3.4}$	$66.24_{\pm 4.0}$
GTAN$_{\text{(AAAI'23)}}$	$72.28_{\pm 1.3}$	$57.44_{\pm 2.9}$	$30.99_{\pm 2.0}$	$84.71_{\pm 8.3}$	$81.24_{\pm 7.8}$	$66.57_{\pm 9.2}$	$93.14_{\pm 0.6}$	$86.37_{\pm 0.4}$	$78.91_{\pm 0.6}$
SparseGAD$_{\text{(IJCAI'23)}}$	$73.84_{\pm 0.3}$	$55.91_{\pm 2.3}$	$35.57_{\pm 0.9}$	$92.00_{\pm 1.0}$	$81.09_{\pm 0.8}$	$\underline{81.43}_{\pm 0.8}$	$93.71_{\pm 0.5}$	$87.31_{\pm 1.0}$	$80.65_{\pm 0.5}$
PMP$_{\text{(ICLR'24)}}$	$\underline{74.68}_{\pm 0.5}$	$61.82_{\pm 1.0}$	$34.83_{\pm 1.7}$	$90.12_{\pm 1.1}$	$86.40_{\pm 1.8}$	$76.91_{\pm 2.6}$	$93.74_{\pm 0.9}$	$86.97_{\pm 1.2}$	$77.58_{\pm 2.1}$
DiG-In-GNN$_{\text{(AAAI'24)}}$	$74.37_{\pm 0.7}$	$63.12_{\pm 0.5}$	$\underline{36.63}_{\pm 0.6}$	$90.22_{\pm 0.3}$	$82.31_{\pm 1.3}$	$70.68_{\pm 1.3}$	$92.82_{\pm 0.9}$	$\underline{87.61}_{\pm 0.2}$	$80.45_{\pm 0.4}$
GAAP$_{\text{(AAAI'25)}}$	$71.21_{\pm 0.6}$	$62.92_{\pm 0.4}$	$33.64_{\pm 0.7}$	$84.53_{\pm 0.1}$	$74.08_{\pm 0.8}$	$43.25_{\pm 0.9}$	$88.30_{\pm 0.9}$	$81.45_{\pm 0.7}$	$63.86_{\pm 0.3}$
SpaceGNN$_{\text{(ICLR'25)}}$	$73.33_{\pm 1.5}$	$\underline{63.47}_{\pm 1.3}$	$36.01_{\pm 2.2}$	$\underline{93.10}_{\pm 1.0}$	$\mathbf{89.52}_{\pm 0.3}$	$80.87_{\pm 2.3}$	$\underline{94.24}_{\pm 0.1}$	$86.96_{\pm 0.2}$	$\underline{80.91}_{\pm 0.2}$
BMP	$\mathbf{81.59}_{\pm 0.4}$	$\mathbf{68.12}_{\pm 0.4}$	$\mathbf{44.46}_{\pm 0.7}$	$\mathbf{94.74}_{\pm 0.4}$	$\underline{89.38}_{\pm 0.3}$	$\mathbf{85.26}_{\pm 0.7}$	$\mathbf{95.99}_{\pm 0.2}$	$\mathbf{90.46}_{\pm 0.3}$	$\mathbf{86.51}_{\pm 0.4}$

Table 1: Comparison of detection performance in the semi-supervised scenario (%) on YelpChi, Amazon and T-Finance, where we show $avg_{\pm std}$ of each. OOM means the out of memory.

Dataset(Anomaly Ratio)	**T-Social**(3.01%)		
Vertices / Edges	5,781,065 / 73,105,508		
Metric	AUC↑	F1-macro↑	AP↑
GCN$_{\text{(ICLR'17)}}$	$82.45_{\pm 0.9}$	$64.45_{\pm 1.2}$	$23.67_{\pm 1.4}$
GraphSAGE$_{\text{(NeurIPS'17)}}$	$70.59_{\pm 1.9}$	$57.34_{\pm 0.8}$	$9.41_{\pm 0.9}$
GAT$_{\text{(ICLR'18)}}$	$74.24_{\pm 2.5}$	$62.36_{\pm 2.1}$	$17.47_{\pm 2.7}$
PC-GNN$_{\text{(WWW'21)}}$	$65.31_{\pm 0.5}$	$47.39_{\pm 0.2}$	$6.20_{\pm 0.1}$
BWGNN$_{\text{(ICML'22)}}$	$84.23_{\pm 2.6}$	$75.59_{\pm 1.5}$	$50.12_{\pm 2.8}$
H2-FDetector$_{\text{(WWW'22)}}$	OOM	OOM	OOM
GHRN$_{\text{(WWW'23)}}$	$84.39_{\pm 3.0}$	$72.21_{\pm 3.3}$	$38.12_{\pm 9.8}$
GTAN$_{\text{(AAAI'23)}}$	$86.53_{\pm 5.3}$	$69.31_{\pm 5.1}$	$34.79_{\pm 3.0}$
SparseGAD$_{\text{(IJCAI'23)}}$	OOM	OOM	OOM
PMP$_{\text{(ICLR'24)}}$	$93.42_{\pm 0.9}$	$79.61_{\pm 0.4}$	$55.73_{\pm 1.3}$
DiG-In-GNN$_{\text{(AAAI'24)}}$	OOT	OOT	OOT
GAAP$_{\text{(AAAI'25)}}$	$80.75_{\pm 1.1}$	$65.51_{\pm 7.5}$	$23.65_{\pm 9.8}$
SpaceGNN$_{\text{(ICLR'25)}}$	$\underline{94.83}_{\pm 0.6}$	$\underline{86.78}_{\pm 0.4}$	$\underline{77.84}_{\pm 2.4}$
BMP	$\mathbf{96.18}_{\pm 0.3}$	$\mathbf{91.96}_{\pm 0.4}$	$\mathbf{87.13}_{\pm 0.8}$

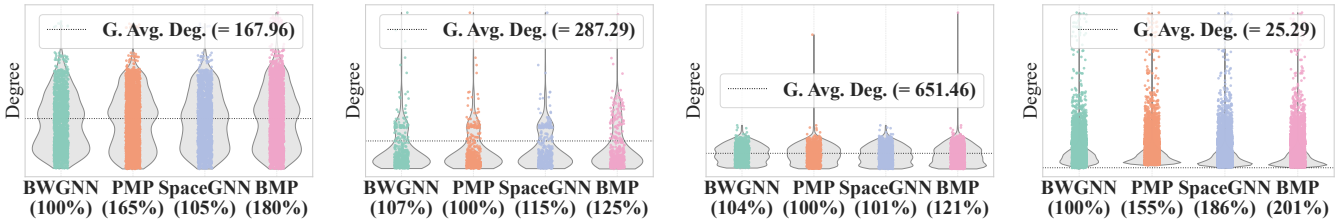
Table 2: Detection performance in the semi-supervised scenario (%) on T-Social. OOT means the out of time ($>$ 24h).

mance under varying training ratios, runtime efficiency, hyperparameter configurations and other specific experimental settings, are provided in the **Extended version**.

Performance

The results of the performance comparison experiments are reported in **Table 1 and 2**. The best and second-best performances are highlighted using bold and underline, respectively. Overall, BMP demonstrates a significant advantage over other baselines, achieving improvements ranging from 4.7% to 21.38% across four datasets. H2-FDetector, SparseGAD and DiG-In-GNN exhibit a certain level of detection capability but suffer from low scalability. PMP's group aggregation strategy demonstrates strong adaptability; however, it is constrained by the broadcast-based routing, which is particularly evident on the large graph (T-Social). As the best-performing baseline overall, SpaceGNN reveals the potential of mapping vertex attributes into multiple feature spaces. The AP score reflects the confidence of the prediction probabilities, which in turn demonstrates the reliability of the binary message-passing routing and its positive impact on classification performance. This provides evidence for the effectiveness of the proposed method.

In Figure 3, we present the detection preferences of models toward anomalies with different structural characteristics (high-degree or low-degree). We use the average degree of anomalous vertices in the dataset as a threshold (i.e. Global Avg Degree), treating those with degrees above this value as high-degree anomalies. We report the relative percentage increase in the number of high-degree anomalies, where 100% denotes the lowest number of high-degree anomalies observed across methods on this dataset. Compared to state-of-the-arts, BMP achieves a 10%–17% relative increase in the number of high-degree anomalies detected, while maintaining overall detection performance. On YelpChi and T-Finance, BMP captures anomalous vertices with higher maximum degrees. On Amazon, BMP significantly improves the detection of anomalies with degrees in the range of 600 to 800. For the large-scale graph T-Social, BMP also achieves the highest relative improvement, on par with that observed on YelpChi. *Notably, SpaceGNN—despite its focus on vertex attributes—never ranks as the worst-performing model. Combined with BMP's performance, this further sug-*



(a) YelpChi (b) Amazon (c) T-Finance (d) T-Social

Figure 3: The degree distribution of the detected anomaly. G. Avg. Deg. is the global average degree of the abnormal vertices.

Dataset	YelpChi			Amazon			T-Finance			T-Social		
Metric	AUC↑	F1-macro↑	AP↑	AUC↑	F1-macro↑	AP↑	AUC↑	F1-macro↑	AP↑	AUC↑	F1-macro↑	AP↑
$\text{BMP}_{\text{w/o MS}}$	81.23	67.22	42.55	94.31	89.01	83.18	95.91	89.87	85.06	86.85	68.83	32.55
$\text{BMP}_{\text{w/o consis}}$	81.00	67.53	42.32	87.27	84.90	72.79	95.33	89.03	84.78	93.88	88.06	82.40
BMP	**81.59**	**68.12**	**44.46**	**94.74**	**89.38**	**85.26**	**95.99**	**90.46**	**86.51**	**96.18**	**91.96**	**87.13**

Table 3: Results of ablation experiments for BMP (%).

gests that the key to advancing semi-supervised GAD may lie beyond the design of message aggregation mechanisms.

Ablation Study

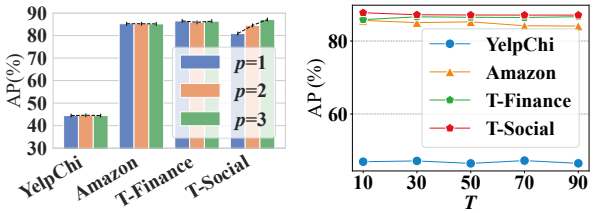
We conducted ablation studies on the multi-scale trees ensemble strategy and the consistency loss, the experimental results are reported in Table 3. $\text{BMP}_{\text{w/o MS}}$ denotes the variant where the forest $\mathcal{F}^*$ contains only $\mathcal{T}^*_{p,p}$, while $\text{BMP}_{\text{w/o consis}}$ removes the consistency loss component.

Multi-scale. Integrating BMP trees with different receptive fields enables the model to capture information from first-order and higher-order neighbors, leading to varying degrees of improvement across all four datasets. The gains are particularly evident in F1-macro and AP scores, highlighting the direct impact of multi-scale structural context on anomaly detection.On T-Social, the AP metric increased by nearly threefold. For large-scale graphs, the number of vertices involved in the binary tree is relatively large, low-order, even first-order, neighborhood information of vertices helps mitigate the cumulative modeling error in message routing. Therefore, incorporating such trees are recommended.

Consistency loss. Consistency loss helps further improve performance under effective modeling, as it contributes to preventing overfitting and enhancing the propagation of limited labels. It regularizes the prediction outcomes, enabling nodes to be assigned more stable indices, which in turn facilitates more reliable routing construction. However, if there are issues with the modeling, consistency loss alone is insufficient to compensate (e.g. $\text{BMP}_{\text{w/o MS}}$ on T-Social).

Sensitivity Analysis

The p in BMP. According to Proposition 1, p controls the receptive field of vertices. Figure 4a shows the AP scores as p varies from 1 to 3. For smaller graphs, shallow networks are often sufficient. For T-Social, deeper networks improve performance. One possible reason is that T-Social is sparser



(a) (b)

Figure 4: The impact of different p and T on performance.

compared to other graphs, and capturing complex anomaly patterns requires a larger receptive field.

The T in BMP. We also report the AP scores for T ranging from 10 to 90 across different datasets in Figure 4b. Overall, the model demonstrates stable performance across different values of T. For certain datasets, such as Amazon, slight fluctuations occur in later stages. As the smallest graph in scale, Amazon may experience overfitting with increasing T due to delayed index updates, which can affect the stability of BMP routing and should be monitored. Through extensive empirical validation, we typically set $T = 50$ in practice.

Conclusion

This paper introduces a novel GAD method from the perspective of message-passing routing: Binary Message-Passing routing (BMP). BMP models the message flow of vertices as a binary tree based on their anomaly probabilities, aiming to enhance the propagation of anomaly information within the topology and address the challenges faced by broadcast-based message routing in GAD. Extensive experiments validate the effectiveness of BMP. In the future, extending the binary tree to an N-ary tree to adapt to multi-class classification could be a promising direction.

Acknowledgments

This work is funded by the National Key Research and Development Program of China (No.2023YFB3307203).

References

Adewole, K. S.; Anuar, N. B.; Kamsin, A.; Varathan, K. D.; and Razak, S. A. 2017. Malicious accounts: Dark of the social networks. *Journal of Network and Computer Applications*, 79: 41–67.

Barranco, O.; Lozares, C.; and Muntanyola-Saura, D. 2019. Heterophily in social groups formation: a social network analysis. *Quality & Quantity*, 53(2): 599–619.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*.

Davis, J.; and Goadrich, M. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, 233–240.

Dong, X.; Zhang, X.; Chen, L.; Yuan, M.; and Wang, S. 2025. SpaceGNN: Multi-Space Graph Neural Network for Node Anomaly Detection with Extremely Limited Labels. In *The Thirteenth International Conference on Learning Representations*.

Duan, M.; He, D.; Zheng, T.; Jia, L.; Song, M.; Wang, X.; and Feng, Z. 2025. Global Attribute-Association Pattern Aggregation for Graph Fraud Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 11616–11624.

Feng, W.; Zhang, J.; Dong, Y.; Han, Y.; Luan, H.; Xu, Q.; Yang, Q.; Kharlamov, E.; and Tang, J. 2020. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33: 22092–22103.

Gao, Y.; Wang, X.; He, X.; Liu, Z.; Feng, H.; and Zhang, Y. 2023. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference 2023*, 1528–1538.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Gong, Z.; Wang, G.; Sun, Y.; Liu, Q.; Ning, Y.; Xiong, H.; and Peng, J. 2023. Beyond Homophily: Robust Graph Anomaly Detection via Neural Sparsification. In *IJCAI*, 2104–2113.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Han, J.; Kamber, M.; and Pei, J. 2011. *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Li, X.; Zhu, R.; Cheng, Y.; Shan, C.; Luo, S.; Li, D.; and Qian, W. 2022. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning*, 13242–13256. PMLR.

Liang, L.; Hu, X.; Xu, Z.; Song, Z.; and King, I. 2024. Predicting global label relationship matrix for graph neural networks under heterophily. *Advances in Neural Information Processing Systems*, 36.

Liao, N.; Luo, S.; Li, X.; and Shi, J. 2024. LD2: Scalable Heterophilous Graph Neural Network with Decoupled Embeddings. *Advances in Neural Information Processing Systems*, 36.

Liu, Y.; Ao, X.; Qin, Z.; Chi, J.; Feng, J.; Yang, H.; and He, Q. 2021. Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection. In *Proceedings of the Web Conference 2021*, WWW '21, 3168–3177. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383127.

Luan, S.; Hua, C.; Lu, Q.; Zhu, J.; Zhao, M.; Zhang, S.; Chang, X.-W.; and Precup, D. 2022. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35: 1362–1375.

Luan, S.; Hua, C.; Xu, M.; Lu, Q.; Zhu, J.; Chang, X.-W.; Fu, J.; Leskovec, J.; and Precup, D. 2024. When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. *Advances in Neural Information Processing Systems*, 36.

Luo, Y.; Shi, L.; and Wu, X.-M. 2024. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

McAuley, J. J.; and Leskovec, J. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, 897–908.

Newman, M. E. 2003. The structure and function of complex networks. *SIAM review*, 45(2): 167–256.

Ngai, E. W.; Hu, Y.; Wong, Y. H.; Chen, Y.; and Sun, X. 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems*, 50(3): 559–569.

Paul, H.; and Nikolaev, A. 2021. Fake review detection on online E-commerce platforms: a systematic literature review. *Data Mining and Knowledge Discovery*, 35(5): 1830–1881.

Platonov, O.; Kuznedelev, D.; Babenko, A.; and Prokhorenkova, L. 2023. Characterizing Graph Datasets for Node Classification: Homophily-Heterophily Dichotomy and Beyond. In *The Second Learning on Graphs Conference*.

Qiao, H.; and Pang, G. 2023. Truncated affinity maximization: One-class homophily modeling for graph anomaly detection. *Advances in Neural Information Processing Systems*, 36.

Rao, S.; Verma, A. K.; and Bhatia, T. 2021. A review on social spam detection: Challenges, open issues, and future directions. *Expert Systems with Applications*, 186: 115742.

Rayana, S.; and Akoglu, L. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, 985–994.

Shi, F.; Cao, Y.; Shang, Y.; Zhou, Y.; Zhou, C.; and Wu, J. 2022. H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. In *Proceedings of the ACM web conference 2022*, 1486–1494.

Tang, J.; Li, J.; Gao, Z.; and Li, J. 2022. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*, 21076–21089. PMLR.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; Liu, J.; and Hooi, B. 2020. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 207–217.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xiang, S.; Zhu, M.; Cheng, D.; Li, E.; Zhao, R.; Ouyang, Y.; Chen, L.; and Zheng, Y. 2023. Semi-supervised credit card fraud detection via attribute-driven graph representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 14557–14565.

Zhang, J.; Xu, Z.; Lv, D.; Shi, Z.; Shen, D.; Jin, J.; and Dong, F. 2024. DiG-In-GNN: Discriminative Feature Guided GNN-Based Fraud Detector against Inconsistencies in Multi-Relation Fraud Graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 9323–9331.

Zhu, J.; Rossi, R. A.; Rao, A.; Mai, T.; Lipka, N.; Ahmed, N. K.; and Koutra, D. 2021. Graph neural networks with heterophily. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11168–11176.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33: 7793–7804.

Zhuo, W.; Liu, Z.; Hooi, B.; He, B.; Tan, G.; Fathony, R.; and Chen, J. 2024. Partitioning message passing for graph fraud detection. In *The Twelfth International Conference on Learning Representations*.