

ORIGINAL RESEARCH OPEN ACCESS

Cross-Domain Graph Anomaly Detection via Graph Transfer and Graph Decouple

Changqin Huang^{1,2} | Xinxing Shi² | Chengling Gao² | Qintai Hu³ | Xiaodi Huang⁴ | Qionghao Huang² | Ali Anaissi⁵

¹College of Education, Zhejiang University, Hangzhou, China | ²Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, China | ³School of Computer Science and Technology, GuangDong University of Technology, Guangzhou, China | ⁴School of Computing, Mathematics and Engineering, Charles Sturt University, Albury, Australia | ⁵TD School, University of Technology Sydney, Ultimo, Australia

Correspondence: Chengling Gao (chl_gao@zjnu.edu.cn) | Qintai Hu (huqt8@gdut.edu.cn)

Received: 19 June 2024 | **Revised:** 27 October 2024 | **Accepted:** 13 November 2024

Funding: This research was supported by the National Nature Science Foundation of China, Grant/Award Numbers: 62337001, 62037001; “Pioneer” and “Leading Goose” R&D Program of Zhejiang, Grant/Award Number: 2022C03106.

Keywords: anomaly detection | attributed graphs | domain adaptation | graph neural networks

ABSTRACT

Cross-domain graph anomaly detection (CD-GAD) is a promising task that leverages knowledge from a labelled source graph to guide anomaly detection on an unlabelled target graph. CD-GAD classifies anomalies as unique or common based on their presence in both the source and target graphs. However, existing models often fail to fully explore domain-unique knowledge of the target graph for detecting unique anomalies. Additionally, they tend to focus solely on node-level differences, overlooking structural-level differences that provide complementary information for common anomaly detection. To address these issues, we propose a novel method, Synthetic Graph Anomaly Detection via Graph Transfer and Graph Decouple (GTGD), which effectively detects common and unique anomalies in the target graph. Specifically, our approach ensures deeper learning of domain-unique knowledge by decoupling the reconstruction graphs of common and unique features. Moreover, we simultaneously consider node-level and structural-level differences by transferring node and edge information from the source graph to the target graph, enabling comprehensive domain-common knowledge representation. Anomalies are detected using both common and unique features, with their synthetic score serving as the final result. Extensive experiments demonstrate the effectiveness of our approach, improving an average performance by 12.6% on the AUC-PR compared to state-of-the-art methods.

1 | Introduction

Graph anomaly detection (GAD), particularly the detection of anomalous nodes, is a critical task in graph data mining [1, 2]. The purpose of GAD is to identify nodes in graph data whose attributes or structures significantly deviate from the majority of nodes. Given its vast potential applications in various domains — from detecting fraudulent individuals in financial transaction networks to prevent economic losses [3, 4], to identifying intrusion viruses in network security systems to safeguard

information and property [5, 6], — an increasing number of researchers have been dedicating their efforts to this field. Many deep learning methods are now being applied to GAD tasks, but these approaches are completely unsupervised and often suffer from high rates of false positives because they lack sufficient knowledge about the anomalies of interest. In this context, Cross-domain graph anomaly detection (CD-GAD), with its ability to leverage knowledge from a labelled source graph to guide anomaly detection on an unlabelled target graph, is garnering attention from researchers.

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2025 The Author(s). *CAAI Transactions on Intelligence Technology* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology and Chongqing University of Technology.

In the field of CD-GAD, the target graph contains not only common anomalies similar to those in the source graph but also unique anomalies absent in the source graph [7]. Figure 1 illustrates a simple example of the CD-GAD context. It can be observed that on the target graph, not only are there common anomalies (user A and user B) similar in semantics to those on the source graph, but there are also unique anomalies with significant semantic differences from the common anomalies (such as user C, distinct from user A). Existing CD-GAD models use different approaches to detect the above anomalies. Among them, COMMANDER detects common anomalies using generative adversarial networks and employs traditional graph auto-encoder to detect unique anomalies [7]. ACT uses off-the-shelf anomaly detection models to generate pseudo-labels to guide the model in detecting anomalous nodes of both types after aligning the distributions of source and target graphs [8]. However, they still suffer from the following two problems:

1. Existing models fail to fully explore deep domain-unique knowledge of target graphs because they overlook the semantic differences between domain-unique knowledge and domain-common knowledge (as illustrated by user C and user A in Figure 1). This limitation hampers the models' performance in detecting unique anomalies.
2. Previous models learnt flawed domain-common knowledge by only considering node level distribution differences (known as marginal distribution shifts) while neglecting structural level differences (called structural domain shifts) between source and target graphs [9]. This oversight results in the low-quality detection of common anomalies.

To address the first problem, our model needs to learn both domain-common and domain-unique knowledge. Our approach begins with learning the domain-common knowledge between the source graph and the target graph. We then separately extract the unique features and common features from the target graph to ensure that the model can effectively learn domain-unique knowledge and domain-common knowledge.

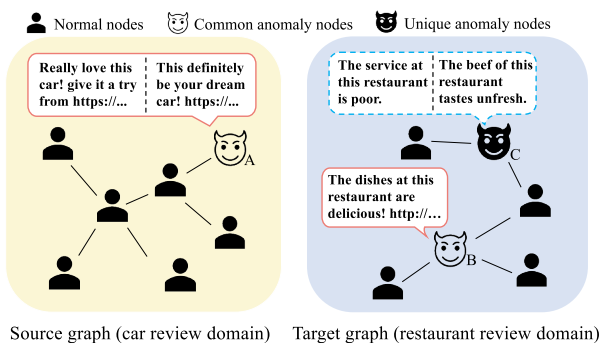


FIGURE 1 | A simple example of cross-domain graph anomaly detection. The source graph from the car review domain and the target graph from the restaurant review domain have users A and B with similar behaviours. They continuously post promotional advertisements about products, which is referred to as a common anomaly. Additionally, user C in the target graph posts malicious negative reviews about restaurants, which is distinct from user A in the source graph and is termed a unique anomaly.

Specifically, we decouple the reconstruction graphs of common features and unique features in terms of structure, ensuring significant semantic separation between them. This process enables us to mine deep domain-unique knowledge, which is beneficial for unique anomaly detection.

To address the second problem, we draw inspiration from refs. [10, 11]. We simultaneously reduce marginal distribution shifts and structural domain shifts by transferring the node and edge information from the source graph to the target graph. This transfer helps the model extract more effective domain-common knowledge. The learnt domain-unique knowledge and domain-common knowledge are then used together to score the nodes in the target graph.

In summary, we propose a novel method named Synthetic Graph Anomaly Detection via Graph Transfer and Graph Decouple (GTGD) to tackle the aforementioned problems. Experimental results on multiple real-world cross-domain graph anomaly detection datasets demonstrate that our proposed GTGD surpasses existing state-of-the-art methods. In this paper, our main contributions can be summarised as follows:

- To explore domain-unique knowledge of the target graph, we propose a novel module, Graph Decouple, which helps the models detect unique anomalies within the target graph.
- To acquire more comprehensive domain-common knowledge, we introduce graph transfer to concurrently mitigate marginal distribution shifts and structural domain shifts between the source and target graphs, enhancing the performance of common anomaly detection.
- We conducted extensive empirical evaluations of GTGD and 11 state-of-the-art competing methods on 4 real CD-GAD datasets to demonstrate the superiority of GTGD. Remarkably, our approach achieved a substantial 12.6% improvement in average performance in terms of the AUC-PR metric.

The rest of this paper is organised as follows: Section 2 reviews related work on graph anomaly detection and domain adaptation methods. Section 3 introduces the preliminary work involved in this study. Section 4 details our proposed model. Section 5 presents the experimental results and performance analysis of our approach. Finally, Section 6 concludes this paper and outlines future research directions.

2 | Related Work

2.1 | Graph Anomaly Detection

The initial studies typically employ nondeep approaches, aiming to identify anomalous information by examining either node features or network structure [12–14]. Nevertheless, their performance cannot experience continuous enhancement without delving into more profound information. In recent years, graph neural networks (GNNs) have garnered significant attention from researchers due to their superior performance in handling complex relationships [15]. As a result, GAD based on

GNNs has become a focal point of research interest. DOMINANT [6] and GUIDE [16] utilise graph auto-encoders to reconstruct both attribute and structural information of the graph, assigning anomaly scores based on the reconstruction quality. CoLA [4] and GRADATE [17] employ contrastive learning to construct instance pairs from different perspectives, aiming to increase the distance between normal and abnormal nodes. In addition, spectral methods [18], hypergraph theory [19], and other approaches have also been applied to GAD. Such methods exhibit good performance, but all of those are completely unsupervised and often suffer from high rates of false positives because they lack sufficient knowledge about the anomalies of interest. Therefore, the CD-GAD methods emerged, which leverages knowledge from a source graph to guide anomaly detection in a target graph. Research on CD-GAD is still in its early stages. COMMANDER [7] combines adversarial networks to align the distributions of the source and target graphs, and utilises source graph labels and reconstruction loss to detect anomalies. ACT [8] reduces the marginal distribution shifts between the target and source graphs and uses pre-trained model detection results as pseudo-labels to guide anomaly detection. Although both approaches have made pioneering progress, their performance is limited because they do not deeply explore domain-unique features in the target graph, and the domain adaptation process only considers marginal distribution shifts, ignoring structural domain shifts between source and target graphs.

2.2 | Domain Adaptation

Domain adaptation methods are frequently utilised to tackle cross-domain problems. The main goal of domain adaptation is to facilitate model training in the target domain by reducing distribution discrepancies between the target and source domains, ultimately enhancing the transferability of knowledge from the source to the target domain [20]. One widely adopted strategy involves minimising domain discrepancies, quantified using predefined metrics such as MMD [21, 22], Wasserstein distance [23] and Gromov–Wasserstein distance [24]. Domain adversarial neural network [25] is another common domain adaptation method. It achieves knowledge transfer by maximising the error of a domain classifier, making the feature representation indistinguishable across domains [26]. In addition, there are now domain adaptation methods specifically designed for graph representation learning. CoCo [27] and ALEX [28] employ graph contrastive learning to facilitate knowledge transfer across different domains, providing effective support for unsupervised domain adaptation graph classification [29]. GALA [30] achieves knowledge transfer from the source graph to the target graph without accessing source graph data. Furthermore, it is important to note that, besides the marginal distribution shifts similar to those in traditional data domains, the complex interactive nature of graph structures leads to additional structural domain shifts [9]. GraphAE combines node degree distributions with other domain adaptation methods to reduce the impact of structural domain shifts [31]. JHGDA focuses on hierarchical graph structures, mitigating structural domain shifts more comprehensively by aggregating

differences among hierarchical structures [32]. However, these methods cannot be directly applied to GAD tasks.

3 | Preliminary

Graph Neural Networks. Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where \mathcal{V} represents the node set, \mathcal{E} represents the edge set, and $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbf{R}^{N \times D}$ is a feature matrix associated with the nodes. The objective of GNNs is to learn a local aggregation function that transforms the node features \mathbf{X} into low dimensional representations $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_N\} \in \mathbf{R}^{N \times D'}$. There are lots of methods such as GCN [33], GAT [34], GraphSage [35]. In our method, we choose GraphSage as our encoder to satisfy the need for large-scale datasets. The transformation used to generate the representation \mathbf{h}_v^k of a given target node v at the k -th layer can be expressed as follows:

$$\mathbf{h}_v^k = \sigma(\mathbf{W}^k \cdot AGGR_k(v, \mathcal{N}(v))), \quad (1)$$

where \mathbf{W}^k is the weights, σ denotes the activate function and $AGGR_k(v, \mathcal{N}(v))$ stands for the aggregation function of the first-order neighbouring nodes $\mathcal{N}(v)$ of v for message passing. In our implementation, we utilise the mean aggregation function:

$$AGGR_k(v, \mathcal{N}(v)) = MEAN\left(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right), \quad (2)$$

where \mathbf{h}_v^{k-1} and \mathbf{h}_u^{k-1} denote the latent representations of the node v and its immediate neighbours from the previous layer, respectively.

Cross Domain Graph Anomaly Detection. We explore CD-GAD on attributed graphs. Provided with a source graph $\mathcal{G}_{src} = (\mathcal{V}_{src}, \mathcal{E}_{src}, \mathbf{X}_{src})$ with label $\mathbf{Y} \in \mathbf{R}^N$ from the source domain and an unlabelled target graph $\mathcal{G}_{tar} = (\mathcal{V}_{tar}, \mathcal{E}_{tar}, \mathbf{X}_{tar})$ from the target domain, we adopt the same feature space assumption as prior studies, with no shared nodes or edges between \mathcal{G}_{src} and \mathcal{G}_{tar} . The aim is to train an anomaly detection model capable of transferring knowledge from the labelled source graph \mathcal{G}_{src} to identify anomalies in the unlabelled target graph \mathcal{G}_{tar} . Ideally, the scoring of anomalous nodes should be significantly higher than normal nodes.

4 | GTGD: The Proposed Approach

In this section, we introduce the proposed framework, Graph transfer and graph decouple (GTGD). As shown in Figure 2, the model consists of three components: (1) Common Anomaly Nodes Detection, (2) Unique Anomaly Nodes Detection and (3) Anomaly Score Fusion. Notably, our model involves three different encoders: the source feature encoder Enc_s , which is used to extract features $\mathbf{H}_s = \{\mathbf{h}_s^1, \mathbf{h}_s^1, \dots, \mathbf{h}_s^n\}$ from the source graph; the common feature encoder Enc_c , which is used to extract common features $\mathbf{H}_c = \{\mathbf{h}_c^1, \mathbf{h}_c^1, \dots, \mathbf{h}_c^n\}$ from the target graph; the unique feature encoder Enc_u , which is used to extract unique features $\mathbf{H}_u = \{\mathbf{h}_u^1, \mathbf{h}_u^1, \dots, \mathbf{h}_u^n\}$ from the target graph. We will provide detailed explanations for Common Anomaly Nodes

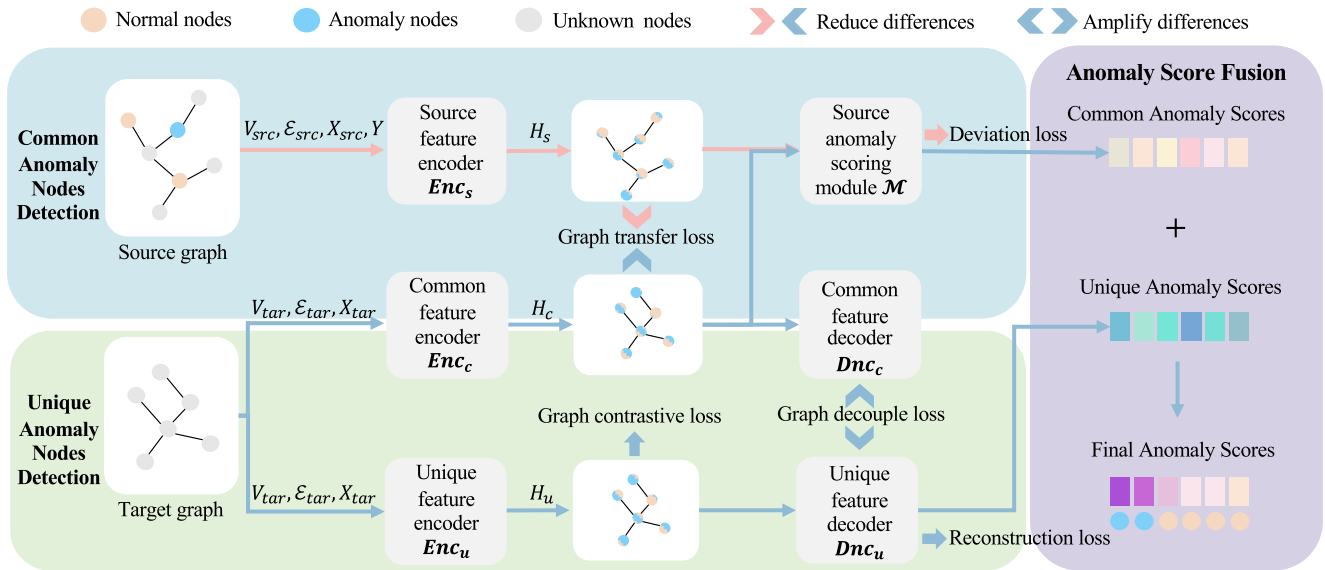


FIGURE 2 | Overview of the GTGD framework. It consists of three parts: (1). Common Anomaly Nodes Detection. We apply deviation loss to the source graph, training the source feature encoder Enc_s to separate normal and abnormal nodes and obtain a source anomaly scoring module \mathcal{M} . Then, we minimise Graph Transfer loss to reduce the marginal distribution shifts and structural domain shifts between the feature distributions of the source and target graphs. Subsequently, we feed the common feature \mathbf{H}_c into \mathcal{M} to obtain the common anomaly scores. (2). Unique Anomaly Nodes Detection. In this stage, the target graph data is input to both Enc_c and Enc_u . We minimise Graph Decouple loss to drive Enc_u to learn the unique features \mathbf{H}_u . At the same time, we utilise graph contrastive learning to ensure that the unique features do not deviate from the semantics of the target graph. Then, we can calculate the unique anomaly scores using the reconstruction loss. (3). Anomaly Score Fusion. We use Z-Score normalisation to standardise the two anomaly scores, then fuse them with weighted summation to get the final anomaly scores.

Detection, Unique Anomaly Nodes Detection, and Anomaly Score Fusion in subsections 4.1, 4.2, and 4.3, respectively.

$$\mathcal{L}_{dev} = \sum_{i=1}^N (1 - y_s^i) |dev(\mathbf{h}_s^i)| + y_s^i \max(0, a - dev(\mathbf{h}_s^i)), \quad (3)$$

where N is the number of nodes for each sampling batch and $y_s^i = 0$ if v_s^i represents a normal node and $y_s^i = 1$ otherwise; and $dev(\mathbf{h}_s^i)$ is a Z-Score-based deviation function:

$$dev(\mathbf{h}_s^i) = \frac{\mathcal{M}(\mathbf{h}_s^i) - \mu_s}{\sigma_s}, \quad (4)$$

where μ_s , σ_s , and a respectively represent the mean, standard deviation, and deviation of anomaly scores from μ_s , following the approach in the original method [36]. In the end, the anomaly score of normal nodes $\mathcal{M}(\mathbf{h}_s^{nor})$ is significantly lower than the anomaly score of anomalous nodes $\mathcal{M}(\mathbf{h}_s^{abn})$. It is important to note that we first train Enc_c and \mathcal{M} , and then freeze the parameters of those modules in the subsequent process.

4.1 | Common Anomaly Nodes Detection

At this stage, we first utilise supervised learning methods on the source graph to train the source feature encoder Enc_s and the source anomaly scoring module \mathcal{M} . Subsequently, we freeze the parameters of source feature encoder Enc_s and reduce the shifts between the source graph features and the target graph features. This makes the common feature encoder Enc_c explore the common features \mathbf{H}_c between the source and target graphs. It maximises the compatibility between \mathcal{M} and the target graph features to detect common anomalies on the target graph similar to those on the source graph.

Source Anomaly Scoring Module. The main idea of this module is to design a well-designed scoring module \mathcal{M} to ensure that $\mathcal{M}(\mathbf{h}^{nor}) \ll \mathcal{M}(\mathbf{h}^{abn})$, where \mathbf{h}^{nor} and \mathbf{h}^{abn} respectively represent embeddings feature of normal nodes and anomaly nodes. That is, the anomaly score for anomalous nodes is greater than the anomaly score for normal nodes. In our method, we use deviation loss [36] to train an MLP as \mathcal{M} . This method enforces a statistically significant deviation between the feature of labelled anomalous nodes and the feature of normal nodes by leveraging labelled anomalies and prior probabilities. Specifically, the deviation loss adapted for our method can be represented as follows:

Graph Transfer. With a well-trained Enc_s and \mathcal{M} , we guide the common feature encoder Enc_c to mine node common features \mathbf{H}_c of nodes in the target graph. Existing CD-GAD methods, when mining domain-common knowledge, only consider the marginal distribution shifts, ignoring the structural domain shifts unique to graph data from different domains, resulting in flawed domain-common knowledge. We address this problem by introducing Graph Transfer. Graph Transfer consists of node transfer and edge transfer, which are respectively used to reduce the marginal distribution shifts and structural domain shifts between the source graph and the target graph. The unique

implementations of node transfer and edge transfer are as follows:

Node transfer. We achieve node transfer by minimising the Wasserstein distance (WD) [23] between the target graph and the source graph. WD is a mathematical distance used in optimal transport to measure the difference between two probability distributions. It is defined as minimising the minimum cost to transform one distribution into another. To compute feasibility, we adopt the approach of Sinkhorn distance [37] and define the following loss function:

$$D_{node}(\mathbf{u}, \mathbf{v}) = \min_{\mathbf{T} \in U(\mathbf{u}, \mathbf{v})} \sum_{i=1}^n \sum_{j=1}^m \mathbf{T}_{ij} \cdot c(\mathbf{h}_c^i, \mathbf{h}_s^j), \quad (5)$$

where \mathbf{u} and \mathbf{v} respectively represent the probability measure of target nodes and source nodes; $U(\mathbf{u}, \mathbf{v})$ represents the feasible space for the transfer from the probability vector \mathbf{u} of the target graph to the probability vector \mathbf{v} of the source graph; \mathbf{T} is the transfer matrix for the target nodes to the source nodes; $c(\mathbf{h}_c^i, \mathbf{h}_s^j)$ is utilised to quantify the transfer cost of v_i node from the target graph to v_j node from the source graph. In this context, we adopt to use the Euclidean distance as the measurement.

Edge transfer. We employ the Groove-Wasserstein distance (GWD) [24] for edge transfer. WD measures the transfer cost by directly comparing the similarity of embeddings (nodes) from two different graphs, while GWD indirectly assesses the transfer cost by comparing the similarity of relationships (edges) between two different graphs [10, 11]. The GWD formula adapted for our task is as follows:

$$D_{edge}(\mathbf{u}, \mathbf{v}) = \min_{\mathbf{T}} \sum_{i,i',j,j'} \mathbf{T}_{i,i',j,j'} \cdot \|\cos(\mathbf{h}_c^i, \mathbf{h}_c^{i'}) - \cos(\mathbf{h}_s^j, \mathbf{h}_s^{j'})\|, \quad (6)$$

where \mathbf{T}' is the edges transfer matrix for the target graph to the source graph. $\cos(\mathbf{h}_c^i, \mathbf{h}_c^{i'})$ and $\cos(\mathbf{h}_s^j, \mathbf{h}_s^{j'})$ represent the edges of the target graph and the source graph. $\|\cos(\mathbf{h}_c^i, \mathbf{h}_c^{i'}) - \cos(\mathbf{h}_s^j, \mathbf{h}_s^{j'})\|$ represent the edge transfer cost for the i th edge of target graph to the j th edge of source graph. We combine node transfer and edge transfer to derive the final Graph Transfer loss:

$$\mathcal{L}_g = \lambda D_{node} + (1 - \lambda) D_{edge}, \quad (7)$$

where λ is a hyperparameter that adjusts the balance between node transfer and edge transfer. We minimise \mathcal{L}_g to eliminate the marginal distribution shifts and structural domain shifts between the target graph and the source graph, enabling the source anomaly scoring module \mathcal{M} to be maximally compatible with the common features \mathbf{H}_c . With the help of edge transfer, the structural domain shifts between the source graph and the target graph will be effectively reduced, providing important supplementary information for learning more comprehensive domain-common knowledge.

Common Anomaly Score Computation. After the Enc_c is well trained, we feed the common feature \mathbf{h}_c^i of the node v_i into \mathcal{M} to obtain the common anomaly score:

$$Score_c(v_i) = \mathcal{M}(\mathbf{h}_c^i). \quad (8)$$

4.2 | Unique Anomaly Nodes Detection

As mentioned earlier, the target graph not only includes common anomalies similar to those in the source graph but may also have its own unique anomalies. At this stage, we will explicitly mine the unique features \mathbf{H}_u of nodes in the target graph and perform unique anomaly detection based on these features.

Graph Decouple. The existing CD-GAD method overlooks the semantic differences between domain-unique knowledge and domain-common knowledge, failing to adequately explore the domain-unique knowledge of the target graph. This limitation restricts the performance of the model in detecting domain-unique anomalies. To explore deeper domain-unique knowledge on the target graph, we employ a unique feature encoder Enc_u , which has the same structure as the Enc_c but with different weights. However, solely using encoders with different weights does not ensure efficient extraction of unique features. Therefore, we increase the reconstruction graph difference to guarantee significant decoupling of common features \mathbf{H}_c and unique features \mathbf{H}_u in structure. We calculate the corresponding adjacency matrix \mathbf{A}_c of the reconstructed graph for common features \mathbf{H}_c based on the inner product as follows:

$$\mathbf{A}_c = \text{Sigmoid}(\mathbf{H}_c \mathbf{H}_c^T), \quad (9)$$

where $\text{Sigmoid}(\ast)$ is the Sigmoid function, and the elements in \mathbf{A}_c signify the relationships between nodes in the reconstructed graph of common features. Similarly, we obtain the adjacency matrix \mathbf{A}_u of the reconstructed graph for unique features \mathbf{H}_u in the same way:

$$\mathbf{A}_u = \text{Sigmoid}(\mathbf{H}_u \mathbf{H}_u^T), \quad (10)$$

the elements in \mathbf{A}_u represent the relationships between nodes in the reconstructed unique graph. Then, we achieve decoupling in structure by enlarging the Euclidean distance between \mathbf{A}_u and \mathbf{A}_c :

$$\mathcal{L}_d = \max(\text{margin} - \|\mathbf{A}_u - \mathbf{A}_c\|_F^2, 0) \quad (11)$$

where margin that controls the difference between \mathbf{A}_u and \mathbf{A}_c . By minimising \mathcal{L}_d , the reconstructed common graph and the reconstructed unique graph will have significant structural differences, ensuring that Enc_u can extract unique features \mathbf{H}_u distinct from the common features \mathbf{H}_c . It's important to note that during this stage of training, we freeze the parameters of the Enc_c to ensure that the common features \mathbf{H}_c remain accurate.

Graph Contrastive Learning. Decoupling may cause the learnt unique features to deviate from the original semantics of the target graph. To address this issue, we first utilise graph contrastive learning to locally capture the original similarities between node pairs, thereby preserving the original semantic features of the nodes. Graph contrastive learning enhances

feature representation by bringing positive pairs closer and pushing negative pairs farther apart [38]. In our approach, we use simple graph contrastive learning to prevent learnt features from losing their original semantics:

$$\mathcal{L}_c = -\log \sum_{t=1}^N \frac{\exp(\mathbf{h}_u^t \mathbf{h}_u^p / \tau)}{\exp(\mathbf{h}_u^t \mathbf{h}_u^p / \tau) + \sum_{\mathbf{h}_u^n \in \mathcal{N}(t)} \exp(\mathbf{h}_u^t \mathbf{h}_u^n / \tau)}, \quad (12)$$

where \mathbf{h}_u^t represents the unique features of the target node, \mathbf{h}_u^p represents the features of the first-order neighbouring nodes of the target node, considered positive samples; and \mathbf{h}_u^n represents the unique features of the nonneighbouring nodes of the target node, considered negative samples. By minimising \mathcal{L}_c , the unique features \mathbf{h}_u^t of the target node learn representations of other nodes in its neighbourhood, preventing the unique features \mathbf{h}_u^t of the target node from deviating too much from the common features \mathbf{h}_c^t and thus preserving the original semantic information.

Unique Graph Reconstruction. In addition to leveraging contrastive learning for the local preservation of the original semantics in the target graph, ensuring that the semantics of the reconstructed subgraph are consistent with those of the original target graph at a global level is equally important for detecting unique anomalous nodes. Inspired by previous work [12], we adopt a structural reconstruction loss to address this issue. The structural reconstruction loss ensures structural consistency by minimising the difference between the reconstructed adjacency matrix and the original adjacency matrix. We use the previously calculated reconstructed adjacency matrix \mathbf{A}_u and the original adjacency matrix \mathbf{A}_{tar} of the target graph to compute the reconstruction loss:

$$\mathcal{L}_r = \|\mathbf{A}_{tar} - \mathbf{A}_u\|_F^2. \quad (13)$$

After integrating the functionalities of various modules, the complete objective function for the unique reconstruction loss can be represented as follows:

$$\mathcal{L}_u = \beta \mathcal{L}_r + \gamma \mathcal{L}_c + \delta \mathcal{L}_d, \quad (14)$$

where β , γ and δ are the hyperparameter to balance the contribution of each module.

Unique Anomaly Score Computation. After training \mathcal{L}_u to convergence, we use the reconstruction error between the original and reconstructed graphs as a key indicator of node anomalies. The intuition is that anomalies often deviate from normal patterns, making them harder to reconstruct and leading to higher reconstruction errors. Specifically, the anomaly score of node v_i can be formulated as follows:

$$Score_u(v_i) = \|\mathbf{A}_{tar}^i - \mathbf{A}_u^i\|_F^2. \quad (15)$$

4.3 | Anomaly Score Fusion

A straightforward way to integrate anomaly scores is by taking the weighted sum of the two scores. However, since the calculation methods of the two scores may not be consistent, the

resulting score distributions may differ significantly, leading to suboptimal performance when using a simple weighted sum. To address this issue, we choose to normalise both scores using Z-Score normalisation:

$$Score_c^{std}(v_i) = \frac{Score_c(v_i) - \mu_c}{\sigma_c}, \quad (16)$$

where μ_c and σ_c respectively represent the mean and standard deviation of common scores for all nodes in the target graph. Similarly, we have standardised unique anomaly scores as follows:

$$Score_u^{std}(v_i) = \frac{Score_u(v_i) - \mu_u}{\sigma_u}, \quad (17)$$

where μ_u and σ_u respectively represent the mean and standard deviation of unique scores for all nodes in the target graph. Finally, we compute the weighted sum of the normalised scores from the two components to obtain the final fused anomaly score for a node v_i in the target graph:

$$Score(v_i) = \omega Score_u^{std}(v_i) + (1 - \omega) Score_c^{std}(v_i), \quad (18)$$

where ω is the balancing hyperparameter between common anomaly scores and unique anomaly scores.

4.4 | Algorithm Description

Generally, the overall procedures of GTGD are shown in Algorithm 1. Given a labelled source graph $\mathcal{G}_{src} = (\mathcal{V}_{src}, \mathcal{E}_{src}, \mathbf{X}_{src}, \mathbf{Y})$ and an unlabelled target graph $\mathcal{G}_{tar} = (\mathcal{V}_{tar}, \mathcal{E}_{tar}, \mathbf{X}_{tar})$, our goal is to obtain the fused anomaly scores for each node in \mathcal{G}_{tar} .

Our algorithm is divided into three stages: Common Anomaly Nodes Detection (Lines 1–7), Unique Anomaly Nodes Detection (Lines 8–13), and Anomaly Score Fusion (Lines 14–15). Specifically, we first use deviation loss to train a source feature encoder Enc_s and a common anomaly scoring module \mathcal{M} (Line 1). Then, we fix the learnt Enc_s and \mathcal{M} , and train the common feature encoder Enc_c of the target graph through Graph Transfer (Lines 2–6). Next, the output of the trained Enc_c is used for common anomaly scoring (Line 7). After training Enc_c , we use Graph Decouple to train our unique feature encoder Enc_u to obtain explicit unique features (Lines 8–12). Then, we use this feature to compute unique anomaly scores (Line 13). Finally, we use Z-Score normalisation to standardise the distributions of common anomaly scores and unique anomaly scores, and then fuse them with weighted summation to get the final anomaly scores (Lines 14–15).

ALGORITHM 1 | Overall Process of GTGD.

Input: $\mathcal{G}_{tar}, \mathcal{G}_{src}$.

Parameter: $N, epoch_{GT}, epoch_{GD}, \mu, \sigma, \lambda, \beta, \gamma, \delta, \mu_s, \sigma_s, \omega$.

Output: Anomaly scores of all nodes in \mathcal{G}_{tar} .

//Common Anomaly Nodes Detection

1: Training for source feature encoder Enc_s and source anomaly scoring module \mathcal{M} according to Equations (3) and (4);

2: **while** $i < epoch_{GT}$ **do**

```

3:   Sample  $N$  nodes from  $\mathcal{G}_{src}$  and  $\mathcal{G}_{tar}$ ;
4:   Train the common feature encoder  $Enc_c$  via
Equation (7);
5:   Take gradient steps and update the parameters;
6: end while
7: Compute common anomaly scores for each node in  $\mathcal{G}_{tar}$  via
Equation (8);
   //Unique Anomaly Nodes Detection
8: while  $i < epoch_{GD}$  do
9:   Sample  $N$  nodes from  $\mathcal{G}_{tar}$ ;
10:  Train the unique feature encoder  $Enc_u$  via
Equation (14);
11:  Take gradient steps and update the parameters;
12: end while
13: Compute unique anomaly scores for each node in  $\mathcal{G}_{tar}$  via
Equation (15);
   //Anomaly Score Fusion
14: Normalise common anomaly scores and unique anomaly
scores via Equations (16) and (17);
15: Compute the fused anomaly scores for each node in  $\mathcal{G}_{tar}$  via
Equation (18).

```

4.5 | Complexity Analysis

We analyse the main computational costs of the model in two stages. (1). Common Anomaly Nodes Detection. The computational cost in this stage primarily consists of the training of the source anomaly scoring module, as well as node transfer and edge transfer, with their time complexities being $O(M \cdot D)$, $O(N \cdot M \cdot (D + K))$, and $O(N^2 + M^2 + D \cdot (N + M) + K \cdot N \cdot M \cdot D)$, respectively. Here, N and M represent the number of source graph nodes and target graph nodes obtained in each sampling, D is the dimension of the node features, and K indicates the number of iterations for the Sinkhorn algorithm to estimate the Wasserstein distance and Gromov-Wasserstein distance. Therefore, the main time complexity of this stage is $O(N^2 + M^2)$. (2). Unique Anomaly Nodes Detection. The computational cost in this stage mainly consists of the decoupling loss, contrastive learning loss, and reconstruction loss, all of which have a time complexity of $O(N^2 \cdot D)$. Thus, the overall time complexity of GTGD is $O(N^2 \cdot D + M^2)$.

5 | Experiments

In this section, we conduct experiments on various benchmark datasets to validate the outstanding efficacy of GTGD. We begin

by introducing the related datasets, followed by a description of the comparison methods and evaluation metrics, implementation details, an ablation study, and a parameter study, in that order.

5.1 | Datasets

Our experiments involve a total of six real-world CD-GAD datasets, including *YelpHotel* (HTL), *YelpRes* (RES), *YelpNYC* (NYC), *Amazon* (AMZ), *Elliptic* (ELL) and *Dgraph-Fin* (DGP). The dataset statistics are listed in Table 1 and we summarise the details of these datasets as follows:

Accommodation Businesses. *YelpHotel* is a dataset of online reviews on Yelp accommodations in the Chicago area [39]. A node symbolises a reviewer, and an edge signifies that two reviewers have evaluated the same business. Reviewers are regarded as anomalies if they have posted fraudulent reviews.

Dining Businesses. *YelpRes* and *YelpNYC* are datasets of dining businesses review graphs for the Chicago and New York areas [39], respectively. Similar to *YelpHotel*, nodes in *YelpRes* and *YelpNYC* also represent reviewers, and anomalous nodes denote reviewers who have posted fraudulent reviews.

E-commerce. *Amazon* is also a review graph dataset but for e-commerce [40]. In this dataset, a user is marked as a fraudulent user if they have reviewed two or more products that have been identified through crowdsourcing efforts [40]; otherwise, the user is regarded as legitimate.

Bitcoin Transaction. *Elliptic* describes a Bitcoin transaction network of real entities, where legitimate transaction users are normal nodes, such as miners and wallet providers, while illegal transaction users are abnormal nodes, such as ransomware and terrorists [41].

Financial Transaction. *Dgraph-Fin*, a real financial transaction graph is recorded, where regular users are labelled as normal nodes, and fraudsters are labelled as anomalous nodes [42].

We conducted multiple CD-GAD experiments for the datasets mentioned above (see Table 2), such as: NYC \rightarrow HTL, RES \rightarrow HTL etc. It is important to note that in the notation A \rightarrow B, A represents the source graph from which adaptability knowledge is extracted, typically where obtainable labels are present, to assist in the GAD task in the target graph B.

TABLE 1 | Statistics of the real-world datasets.

Dataset	#Nodes	#Edges	#Attributes	#Anomalies
YelpHotel	5196	171,743	8000	250
YelpRes	5102	239,738	8000	275
YelpNYC	21,040	303,949	10,000	10,000
Amazon	18,601	274,458	10,000	750
Elliptic	203,769	234,355	166	4545
Dgraph-Fin	3,700,550	4,300,999	17	15,509

TABLE 2 | (Continued)

		NYC→HTL	RES→HTL	HTL→HTL	HTL→RES	NYC→RES	RES→RES	RES→NYC	HTL→NYC	AMZ→NYC	NYC→NYC	NYC→AMZ
Domain adaptation	ADDA	0.227 ± 0.028	0.171 ± 0.062	0.260 ± 0.126	0.254 ± 0.140	0.254 ± 0.140	0.254 ± 0.140	0.181 ± 0.021	0.239 ± 0.110	0.051 ± 0.002	0.177 ± 0.057	
	COMMANDER (s)	0.210 ± 0.007	N/A	0.268 ± 0.006	N/A	N/A	N/A	N/A	0.145 ± 0.001	0.242 ± 0.019	0.216 ± 0.008	
	COMMANDER (u)	0.216 ± 0.008	0.207 ± 0.009	0.253 ± 0.025	0.267 ± 0.015	0.144 ± 0.003	0.144 ± 0.003	0.144 ± 0.002	0.145 ± 0.001	0.220 ± 0.024	0.209 ± 0.014	
Ours	ACT	<u>0.287 ± 0.006</u>	<u>0.284 ± 0.010</u>	<u>0.330 ± 0.018</u>	<u>0.477 ± 0.065</u>	0.249 ± 0.012	0.249 ± 0.012	0.241 ± 0.009	0.243 ± 0.003	<u>0.497 ± 0.020</u>	<u>0.358 ± 0.002</u>	
	GTGD	0.379 ± 0.020	0.368 ± 0.032	0.563 ± 0.024	0.776 ± 0.031	0.388 ± 0.017	0.388 ± 0.017	0.380 ± 0.012	0.371 ± 0.011	0.649 ± 0.016	0.484 ± 0.020	

Note: The boldfaced values indicate the best results and the underlined values indicate the second-best results.

5.2 | The Comparison Methods and Evaluation Metrics

Comparison methods. We have listed 11 state-of-the-art methods in the field of GAD. These methods can be categorised into unsupervised methods and domain adaptation methods based on their training approaches. In the unsupervised training category, two unsupervised methods are formed by combining LOF [43] or IF [44] with DGI [45]. In addition, the remaining methods employ unsupervised GAD methods published in recent years, including ANOMALOUS [14], DOMINANT [1], ADONE [46], GAAN [47] and COLA [4]. The experimental results of these methods all demonstrate direct efficacy in the target graph. We include them to investigate whether GTGD can leverage source graph information to enhance unsupervised GAD performance on the target graph. For CD-GAD methods, we primarily compared four models, namely ADDA [20], COMMANDER(s) [7], COMMANDER(u) and ACT [8]. As shown in Table 2, GTGD and the aforementioned CD-GAD methods are experimented under the same source graph and target graph backgrounds to evaluate the effectiveness of GTGD.

Evaluation metrics. In GAD, the Area Under the Receiver Operating Characteristic curve (AUC-ROC) and the Area Under the Precision-Recall curve (AUC-PR) are crucial metrics. AUC-ROC evaluates the model's classification performance for positive and negative samples across different thresholds, while AUC-PR focuses on the classification of positive samples, making it particularly suitable for imbalanced datasets. Combining these metrics provides a comprehensive evaluation of model performance.

5.3 | Implementation Details

The three encoders Enc_s , Enc_c , and Enc_u involved in our model all consist of two layers of GraphSage [35], with the number of sampled neighbours set to 25 and 10 for each layer. Additionally, we set the feature dimensions for each layer to 256 and 64, ensuring that features from the source graph and the target graph are mapped to the same feature space. For the training stage of Common Anomaly Nodes Detection, \mathcal{M} is trained 50 epochs using a learning rate of 10^{-3} . Enc_c also requires training for 50 epochs, but the values of Graph Transfer balance hyperparameter λ and the learning rate α_c are not exactly the same for each experiment, as detailed in Table 3. For the training stage of Unique Anomaly Nodes Detection, Enc_u is trained for 30 epochs with fixed $\beta = 1$, $\gamma = 1$ and $\omega = 0.15$, and the learning rate α_u and δ can also be seen in Table 3. It is important to note that the sampling batch size is 128 for both stages. The results of GTGD are averaged over 5 independent runs with random seeds in the Table 2, and other results follow the works of ACT [8]. The experimental setup utilises the NVIDIA A6000 GPU with a VRAM size of 48 GB, and the CUDA version installed is 470.74.

5.4 | Results Analysis

Comparison Experiment. Experimental results demonstrate the effectiveness of our method by listing 11 existing state-of-

the-art approaches. As shown in Table 2, we can clearly see that compared to existing state-of-the-art methods, GTGD shows a significant performance improvement. Its average AUC-ROC and AUC-PR reach 0.894 and 0.484, respectively. Compared to the best baseline, they are noticeably higher by 2.6% and 12.6%, respectively. Specifically, in our eight settings, GTGD's AUC-ROC significantly surpasses existing state-of-the-art models in seven settings. In HTL \rightarrow RES and RES \rightarrow HTL, it outperforms the best baseline by a significant 6.7% and 6.6%, respectively. Moreover, in terms of AUC-PR, GTGD achieves improvement in all eight settings. In NYC \rightarrow RES and HTL \rightarrow RES, compared to the best baseline, GTGD's performance is enhanced by 29.9% and 23.3%, respectively. This indicates that our method can detect more anomalies in imbalanced data distributions, thus validating the effectiveness and robustness of our approach.

It can be observed that, through comparison, the performance of unsupervised methods is not competitive. Specifically, the performance of DGI + IF and DGI + LOF is limited because their designs were not originally intended for node anomaly detection. As for other unsupervised GAD models, their inability to utilise information from the source graph for anomaly node detection results in poor performance.

Next, we compare the performance of GTGD with other domain adaptation methods. Since the ADDA model was not specifically designed for GAD, its performance is not satisfactory. As for COMMANDER, its failure to decouple common features and unique features when mining unique anomalies in the graph leads to the model's inability to better learn unique features. Consequently, it cannot further explore unique anomalies. We will demonstrate this conclusion in subsequent ablation experiments. ACT only considers marginal distribution shifts when transferring knowledge from the source graph, neglecting the

unique structural domain shifts in graph domain adaptation. Additionally, it relies on existing anomaly detection models for fine-tuning, which makes its performance dependent on the choice of these models. In contrast, our model addresses both marginal distribution shifts and structural domain shifts simultaneously, learning unique features to identify distinctive anomalies, thus ensuring superior performance.

Visual Analysis of Anomaly Score Distribution. To provide a more intuitive view of our model's performance, we present the node anomaly score distributions for RES and HTL in Figure 3. The figure shows that the score distribution ranges for normal and abnormal nodes are largely distinct, with the former being noticeably narrower than the latter. Using the first two plots as examples, we can see that the anomaly scores for normal nodes are primarily concentrated between -0.4 and 0 , with the interquartile range being relatively narrow at $[-0.2, 0.2]$. In contrast, the anomaly scores for abnormal nodes are distributed around 0 to 4 , with the interquartile range being $[0.5, 2.0]$, which is noticeably wider than that of the normal nodes. This indicates that the anomaly scores for normal data are more concentrated, while the anomaly scores for abnormal nodes are more dispersed. This aligns with our intuition that normal nodes exhibit consistency among themselves, whereas abnormal nodes exhibit diversity. This result further validates the effectiveness and reliability of our model in anomaly detection tasks.

5.5 | Ablation Study

Graph Transfer. Graph Transfer consists mainly of two parts: node transfer, which deals with the marginal distribution shifts between domains, and edge transfer, which addresses the

TABLE 3 | Related hyperparameters of GTGD.

Hyperparameter	Dataset							
	NYC \rightarrow HTL	RES \rightarrow HTL	HTL \rightarrow RES	NYC \rightarrow RES	RES \rightarrow NYC	HTL \rightarrow NYC	AMZ \rightarrow NYC	NYC \rightarrow AMZ
λ	0.5	0.7	0.2	0.4	0.8	0.9	0.4	0.7
α_c	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-5}	10^{-5}	10^{-6}	10^{-4}
α_u	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-3}	10^{-4}
δ	1.5	1	1	1	0.5	0.5	1	0.1

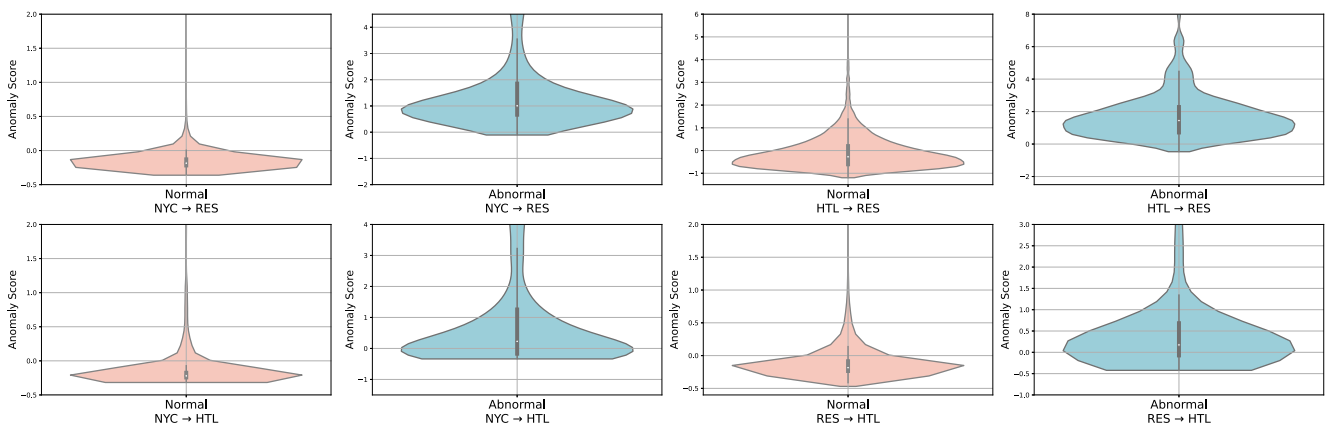


FIGURE 3 | Visualisation of node anomaly score distribution.

structural domain shifts unique to cross-domain graph data. This enables the target graph to more comprehensively learn knowledge from the source graph. To validate the effectiveness of our approach, we conducted ablation studies separately for node transfer and edge transfer. The experimental results are depicted in Tables 4 and 5. Among them, “GTGD w/o Node” refers to GTGD without node transfer, which is used to verify the effect of node-level loss. Correspondingly, “GTGD w/o Edge” refers to GTGD without edge transfer, which is used to verify the effect of structure-level loss. GTGD shows a significant improvement in both AUC-ROC and AUC-PR compared to “GTGD w/o Edge” and “GTGD w/o Node”. Specifically, compared to “GTGD w/o Edge”, there is an average improvement of 2.4% and 7.2% in the two metrics. Similarly, compared to “GTGD w/o Node”, there is an improvement of 7.6% and 13.2% in the two metrics. These results indicate that the Graph Transfer of GTGD can adequately and efficiently handle marginal distribution shifts and structural domain shifts, thus ensuring that the model can better detect common anomalies in both the source and target domains.

Graph Decouple. COMMANDER proposed a unique anomalies concept but relies only on simple unsupervised GAD methods to detect them, without further exploring the target graph’s unique features. This leads to inefficiency in uncovering

unique graph-unique anomalies. In GTGD, the role of Graph Decouple is to ensure that the model learns the explicit unique features of the target graph. By utilising these unique features for reconstruction, GTGD performs unique anomaly node detection, thereby ultimately improving the model’s performance. To thoroughly validate the effectiveness of Graph Decouple, we designed three experiments: removing the unique anomaly detection part and only using Common Anomaly detection (GTGD w/o UAD), joint framework of graph contrastive learning, unsupervised reconstruction, and common anomaly detection (GTGD w/o GD), and the complete model GTGD. The results of the experiment are shown in the Tables 4 and 5. It can be observed that using unsupervised reconstruction loss to detect unique anomalies in “GTGD w/o GD” yields only a few results superior to “GTGD w/o UAD”, which detects only common anomalies. The overall performance even decreases. A reasonable explanation is that simple unsupervised anomaly detection methods can detect unique anomalies to some extent. However, when paired with an efficient common anomaly detection model, they fail to further enhance its performance. This happens because they do not effectively explore the unique features of the target graph and may even have a detrimental effect. In contrast, comparing “GTGD w/o UAD” and GTGD, it can be observed that under the influence of Graph Decouple, the model’s precision is further improved by 1.3% in AUC-ROC

TABLE 4 | The AUC-ROC values of ablation study for Graph transfer and Graph decouple.

Experiments	Graph transfer		Graph decouple		GTGD
	GTGD w/o edge	GTGD w/o node	GTGD w/o UAD	GTGD w/o GD	
NYC→HTL	0.813	0.822	0.829	0.823	0.862
RES→HTL	0.798	0.803	0.832	0.828	0.848
HTL→RES	0.935	0.879	0.951	0.946	0.959
NYC→RES	0.964	0.987	0.991	0.987	0.992
RES→NYC	0.848	0.831	0.847	0.849	0.859
HTL→NYC	0.846	0.834	0.848	0.841	0.861
AMZ→NYC	0.847	0.817	0.850	0.844	0.861
NYC→AMZ	0.910	0.568	0.905	0.908	0.917
Average	0.870	0.818	0.881	0.878	0.894

Note: The bold values indicate the best results.

TABLE 5 | The AUC-PR values of ablation study for Graph transfer and Graph decouple.

Experiments	Graph transfer		Graph decouple		GTGD
	GTGD w/o edge	GTGD w/o node	GTGD w/o UAD	GTGD w/o GD	
NYC→HTL	0.328	0.317	0.332	0.331	0.379
RES→HTL	0.307	0.329	0.357	0.351	0.368
HTL→RES	0.432	0.356	0.533	0.535	0.563
NYC→RES	0.607	0.701	0.754	0.720	0.776
RES→NYC	0.348	0.318	0.347	0.345	0.388
HTL→NYC	0.347	0.327	0.340	0.341	0.380
AMZ→NYC	0.323	0.325	0.347	0.345	0.371
NYC→AMZ	0.607	0.141	0.620	0.636	0.649
Average	0.412	0.352	0.438	0.435	0.484

Note: The bold values indicate the best results.

and 4.6% in AUC-PR. This once again proves the effectiveness of the Graph Decouple proposed by us.

Contrastive Learning and Structural Reconstruction.

Contrastive learning and structural reconstruction ensure that the original semantics of the target graph are preserved during the decoupling process, from local and global perspectives, respectively. This plays a crucial role in detecting unique anomalies in the target graph. To validate their effectiveness, we conducted the ablation study shown in Figure 4. In the figure, “GTGD w/o CL” represents GTGD without contrastive learning, and “GTGD w/o SR” represents GTGD without structural reconstruction. By comparing “GTGD w/o CL” and “GTGD w/o SR” with GTGD, it is evident that both contrastive learning and structural reconstruction enhance the model’s anomaly detection capabilities. This indicates that both contrastive learning and structural reconstruction can ensure that our model does not lose the original semantics of the target graph. Furthermore, the results show that “GTGD w/o CL” performs better overall than “GTGD w/o SR”, suggesting that structural reconstruction plays a more significant role than contrastive learning in preserving the original semantics of the target graph in our method.

5.6 | Parameter Study

In this section, we carry out a series of experiments to study the effectiveness of various hyperparameters in GTGD, including the hyperparameter λ of Graph Transfer module to balance the nodes transfer and edges transfer, and the hyperparameter δ to control the impact of Graph Decouple module. Because of the

limited space, we execute such experiments on four cross-domain settings, which include RES \rightarrow HTL, HTL \rightarrow RES, RES \rightarrow NYC, and NYC \rightarrow AMZ.

Sensitivity Test for Graph Transfer. In these experiments, we discuss the influence of Graph Transfer balance factor λ changes on the GTGD performance. We tune the λ in a range of {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}, and the results are illustrated in Figure 5. Based on the results, the values of AUC-ROC and AUC-PR show a trend of initially increasing and then decreasing. These results suggest that considering both marginal distribution shifts and structural domain shifts simultaneously can improve the quality of learnt domain-common knowledge. This ensures a better fit of the source anomaly score module \mathcal{M} trained on the source graph to the target graph. Additionally, the selection of λ is highly dependent on the cross-domain dataset. For instance, in RES \rightarrow HTL, the optimal performance for AUC-ROC and AUC-PR is achieved when λ is set to 0.7. In contrast, the best AUC-ROC and AUC-PR performance is attained at $\lambda = 0.2$ in HTL \rightarrow RES. This suggests that the weighting relationship between marginal distribution shifts and structural domain shifts needs to be adapted to the different source and target graph settings to learn more comprehensive domain-common knowledge.

Sensitivity Test for Graph Decouple. In this study, we explore the influence of varying the hyperparameter δ of the Graph Decouple module on the performance of GTGD. This hyperparameter controls the degree of differentiation between common and unique features. The experimental outcomes obtained by varying δ are presented in Figure 6. Generally, GTGD

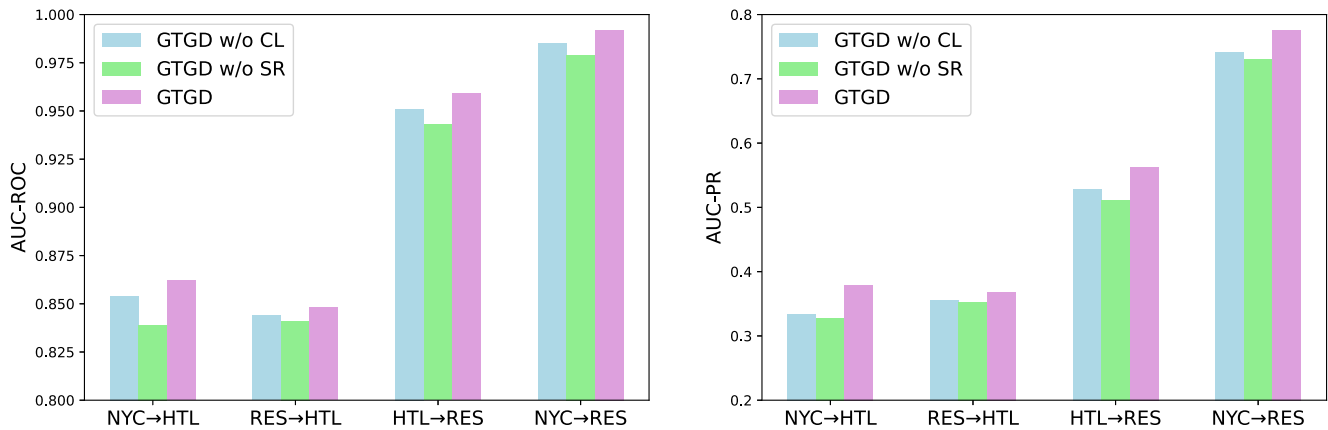


FIGURE 4 | The AUC-ROC and AUC-PR values of ablation study for Contrastive learning and structure reconstruction.

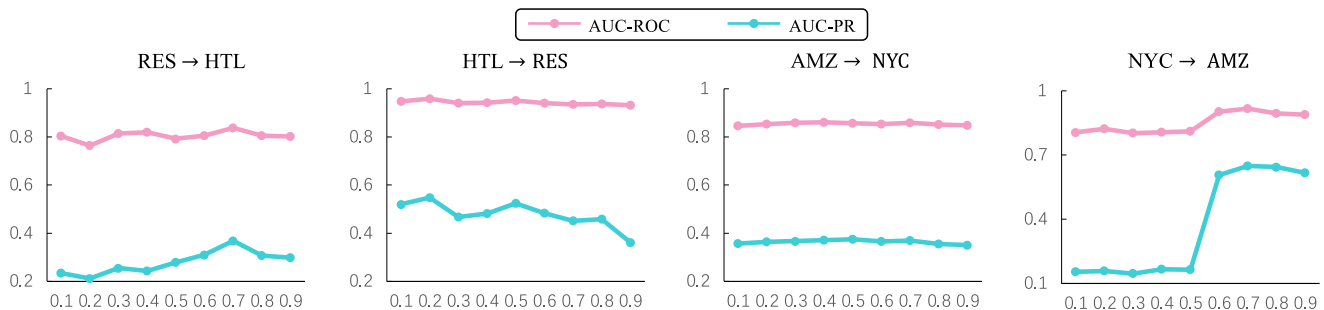


FIGURE 5 | Sensitivity test results w.r.t. λ in the Graph transfer module.

shows consistent performance across a range of δ values, spanning from 0.1 to 2. This indicates robust stability across datasets with diverse characteristics.

5.7 | Scalability Experiments

In this part of the study, we evaluate the anomaly detection performance of the proposed model when handling large-scale complex datasets with significant domain differences. In the cross-domain experiments, the large-scale datasets *Elliptic* and *Dgraph-Fin* (see Table 1) are used as the target domains, while *YelpHotel* and *YelpNYC*, which have significant domain differences from the two, are used as the source domains to meet our design objectives. The specific experimental results are shown in Table 6. We conducted experiments on five models in total, including three unsupervised models: DOMINANT, GCNAE [48], and COLA, and two domain adaptation models: ACT and our proposed GTGD. In terms of experimental design, for the unsupervised models, we directly tested their performance on ELL (*Elliptic*) and DGP (*Dgraph-Fin*). Compared to the out-of-memory issue with DOMINANT, we ensured that GTGD could be applied to large-scale datasets by sampling nodes through GraphSAGE and training the model in batches. In

comparison to the other two unsupervised models, our model achieved higher performance by incorporating label information from other domains. Even when faced with experimental designs involving greater domain differences between the source and target domains, GTGD was still able to effectively utilise the information provided by both the source and target graphs through the proposed graph transfer and decoupling techniques, leading to superior performance. This once again demonstrates the effectiveness of the proposed model.

6 | Conclusion

In this paper, we investigate the problem of Cross Domain Graph Anomaly Detection (CD-GAD). We propose the GTGD model, which consists mainly of three parts: Common Anomaly node detection, Unique Anomaly node detection, and Anomaly Score Fusion. Specifically, Common Anomaly node detection employs Graph Transfer to simultaneously address both marginal distribution shifts and structural domain shifts, thereby comprehensively leveraging domain-common knowledge from the source graph. Building upon this, Unique Anomaly node detection utilises Graph Decouple to decouple the features of the target graph in structure, thereby learning deeper domain-

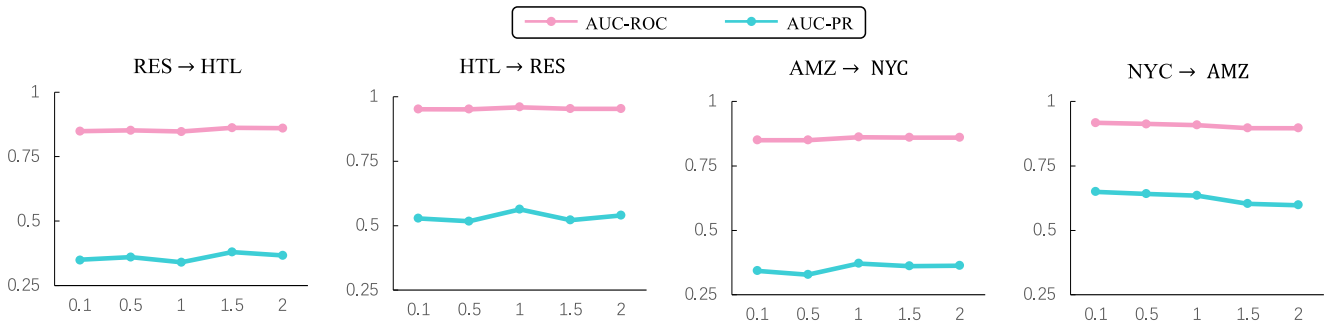


FIGURE 6 | Sensitivity test results w.r.t. δ in the Graph decouple module.

TABLE 6 | Performance comparison for AUC-ROC and AUC-PR (\pm std).

	Type	Method	Dataset				
			ELL	DGP	Average		
AUC-ROC	Unsup	DOMINANT	OOM	OOM	OOM		
		GCNAE	0.433 ± 0.000	0.409 ± 0.000	0.421 ± 0.000		
		COLA	0.532 ± 0.004	0.505 ± 0.002	0.519 ± 0.003		
			HTL→ELL	NYC→ELL	HTL→DGP	NYC→DGP	
	Domain adaptation	ACT	0.595 ± 0.021	0.615 ± 0.012	0.599 ± 0.007	0.616 ± 0.012	0.606 ± 0.013
		GTGD (ours)	0.632 ± 0.032	0.652 ± 0.023	0.615 ± 0.011	0.655 ± 0.016	0.639 ± 0.021
AUC-PR	Unsup	DOMINANT	OOM	OOM	OOM		
		GCNAE	0.060 ± 0.000	0.009 ± 0.000	0.030 ± 0.000		
		COLA	0.072 ± 0.001	0.013 ± 0.002	0.042 ± 0.002		
			HTL→ELL	NYC→ELL	HTL→DGP	NYC→DGP	
	Domain adaptation	ACT	0.231 ± 0.024	0.243 ± 0.026	0.017 ± 0.001	0.018 ± 0.001	0.127 ± 0.013
		GTGD (ours)	0.244 ± 0.037	0.267 ± 0.039	0.019 ± 0.001	0.020 ± 0.002	0.138 ± 0.020

Note: The boldfaced values indicate the best results and the underlined values indicate the second-best results. OOM denotes out of memory.

unique knowledge to effectively detect unique target anomalies. Finally, by integrating common anomaly scores and unique anomaly scores, GTGD computes the final anomaly scores for each node in the target graph. Extensive experiments validate the effectiveness of GTGD.

Despite GTGD demonstrating excellent performance and strong scalability across multiple experiments, it comes at a certain computational cost. This highlights that there is still room for improvement in the model's practical applicability. Therefore, balancing the trade-off between performance and time complexity will be a key focus of our upcoming work. In addition, we plan to extend GTGD to multi-source domain adaptation for graph anomaly detection tasks in future work.

Acknowledgements

This research was supported by the National Nature Science Foundation of China, Grant/Award Numbers: 62337001, 62037001; "Pioneer" and "Leading Goose" R&D Program of Zhejiang, Grant/Award Number: 2022C03106.

Conflicts of Interest

Xiaodi Huang is an editorial board member for the journal, and was not involved in peer review process or the decision to publish this article. The authors declare that they have no conflict of interest.

Data Availability Statement

The data that support the findings of this study are publicly available: YelpHTL: <https://odds.cs.stonybrook.edu/yelpchi-dataset/>; YelpRes: <https://odds.cs.stonybrook.edu/yelpchi-dataset/>; YelpNYC: <https://odds.cs.stonybrook.edu/yelpnyc-dataset/>; Amazon: <https://github.com/QZ-WANG/ACT/tree/main/datasets/Amazon>.

References

1. K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep Anomaly Detection on Attributed Networks," *Proceedings of the 2019 SIAM International Conference on Data Mining* (2019): 594–602, <https://doi.org/10.1137/1.9781611975673.67>.
2. J. Tang, F. Hua, Z. Gao, P. Zhao, and J. Li, "Gadbench: Revisiting and Benchmarking Supervised Graph Anomaly Detection," *Advances in Neural Information Processing Systems* (2024).
3. Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng, "A Deep Multi-View Framework for Anomaly Detection on Attributed Networks," *IEEE Transactions on Knowledge and Data Engineering* 34, no. 6 (2020): 2539–2552.
4. Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning," *Proceedings of IEEE Transactions on Neural Networks and Learning Systems* 33, no. 6 (2021): 2378–2392, <https://doi.org/10.1109/tnnls.2021.3068344>.
5. B. Chen, J. Zhang, X. Zhang, et al., "GCCAD: Graph Contrastive Coding for Anomaly Detection," *IEEE Transactions on Knowledge and Data Engineering* 35, no. 8 (2023): 8037–8051.
6. K. Ding, Q. Zhou, H. Tong, and H. Liu, "Few-shot Network Anomaly Detection via Cross-Network Meta-Learning," *Proceedings of the Web Conference 2021* (2021): 2448–2456.
7. K. Ding, K. Shu, X. Shan, J. Li, and H. Liu, "Cross-domain Graph Anomaly Detection," *Proceedings of IEEE Transactions on Neural*

Networks Learning Systems 33, no. 6 (2021): 2406–2415, <https://doi.org/10.1109/tnnls.2021.3110982>.

8. Q. Wang, G. Pang, M. Salehi, W. Buntine, and C. Leckie, "Cross-domain Graph Anomaly Detection via Anomaly-Aware Contrastive Alignment," *Proceedings of the AAAI Conference on Artificial Intelligence* 37, no. 4 (2023): 4676–4684, <https://doi.org/10.1609/aaai.v37i4.25591>.
9. B. Shi, Y. Wang, F. Guo, B. Xu, H. Shen, and X. Cheng, Graph Domain Adaptation: Challenges, Progress and Prospects. arXiv preprint arXiv:2402.00904 (2024).
10. T. Vayer, L. Chapel, R. Flamary, R. Tavenard, and N. Courty, "Fused Gromov-Wasserstein Distance for Structured Objects," *Algorithms* 13, no. 9 (2020): 212, <https://doi.org/10.3390/a13090212>.
11. L. Chen, Z. Gan, Y. Cheng, L. Li, L. Carin, and J. Liu, "Graph Optimal Transport for Cross-Domain Alignment," *Proceedings of International Conference on Machine Learning* (2020): 1542–1553.
12. J. Li, H. Dani, X. Hu, and H. Liu, "Radar: Residual Analysis for Anomaly Detection in Attributed Networks," *Proceedings of 26th International Joint Conference on Artificial Intelligence* (2017): 2152–2158, <https://doi.org/10.24963/ijcai.2017/299>.
13. B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller, "Focused Clustering and Outlier Detection in Large Attributed Graphs," *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2014): 1346–1355, <https://doi.org/10.1145/2623330.2623682>.
14. Z. Peng, M. Luo, J. Li, H. Liu, and Q. Zheng, "ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks," *Proceedings of 27th International Joint Conference on Artificial Intelligence* (2018): 3513–3519, <https://doi.org/10.24963/ijcai.2018/488>.
15. W. Ju, Z. Mao, S. Yi, et al., "Hypergraph-enhanced Dual Semi-supervised Graph Classification," *Proceedings of International Conference on Machine Learning* (2024).
16. X. Yuan, N. Zhou, S. Yu, H. Huang, Z. Chen, and F. Xia, "Higher-order Structure Based Anomaly Detection on Attributed Networks," *Proceedings of 2021 IEEE International Conference on Big Data* (2021): 2691–2700, <https://doi.org/10.1109/bigdata52589.2021.9671990>.
17. J. Duan, S. Wang, P. Zhang, et al., "Graph Anomaly Detection via Multi-Scale Contrastive Learning Networks With Augmented View," *Proceedings of the AAAI Conference on Artificial Intelligence* 37, no. 6 (2023): 7459–7467, <https://doi.org/10.1609/aaai.v37i6.25907>.
18. J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking Graph Neural Networks for Anomaly Detection," *Proceedings of International Conference on Machine Learning* (2022): 21076–21089.
19. J. Liu, M. He, X. Shang, J. Shi, B. Cui, and H. Yin, BOURNE: Bootstrapped Self-Supervised Learning Framework for Unified Graph Anomaly Detection. arXiv preprint arXiv:2307.15244 (2023).
20. G. Wilson and D. J. Cook, "A Survey of Unsupervised Deep Domain Adaptation," *ACM Transactions on Intelligent Systems and Technology* 11, no. 5 (2020): 1–46, <https://doi.org/10.1145/3400066>.
21. Y. Zhu, F. Zhuang, and D. Wang, "Aligning Domain-Specific Distribution and Classifier for Cross-Domain Classification From Multiple Sources," *Proceedings of the AAAI Conference on Artificial Intelligence* 33, no. 1 (2019): 5989–5996, <https://doi.org/10.1609/aaai.v33i01.33015989>.
22. Y. Zhu, F. Zhuang, J. Wang, et al., "Multi-representation Adaptation Network for Cross-Domain Image Classification," *Neural Networks* 119 (2019): 214–221, <https://doi.org/10.1016/j.neunet.2019.07.010>.
23. C.-Y. Lee, T. Batra, M. H. Baig, and D. Ulbricht, "Sliced Wasserstein Discrepancy for Unsupervised Domain Adaptation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019): 10285–10295.
24. X. Li, Z. Qiu, X. Zhao, et al., "Gromov-wasserstein Guided Representation Learning for Cross-Domain Recommendation," *Proceedings of the*

- 31st ACM International Conference on Information & Knowledge Management (2022): 1199–1208, <https://doi.org/10.1145/3511808.3557338>.
25. Y. Ganin, E. Ustinova, H. Ajakan, et al., “Domain-adversarial Training of Neural Networks,” *Journal of Machine Learning Research* 17, no. 59 (2016): 1–35.
26. Y. Liu, W. Zhang, and J. Wang, “Source-free Domain Adaptation for Semantic Segmentation,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021): 1215–1224, <https://doi.org/10.1109/cvpr46437.2021.00127>.
27. N. Yin, L. Shen, M. Wang, et al., “CoCo: A Coupled Contrastive Framework for Unsupervised Domain Adaptive Graph Classification,” *Proceedings of International Conference on Machine Learning* (2023): 40040–40053.
28. J. Yuan, X. Luo, Y. Qin, Z. Mao, W. Ju, and M. Zhang, “ALEX: Towards Effective Graph Transfer Learning With Noisy Labels,” *Proceedings of the 31st ACM International Conference on Multimedia* (2023): 3647–3656, <https://doi.org/10.1145/3581783.3612026>.
29. W. Ju, Y. Wang, Y. Qin, et al., Towards Graph Contrastive Learning: A Survey and beyond. arXiv preprint arXiv:2405.11868 (2024).
30. J. Luo, Y. Gu, X. Luo, et al., “GALA: Graph Diffusion-Based Alignment With Jigsaw for Source-Free Domain Adaptation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, no. 12 (2024): 9038–9051, <https://doi.org/10.1109/tpami.2024.3416372>.
31. G. Guo, C. Wang, B. Yan, et al., “Learning Adaptive Node Embeddings Across Graphs,” *IEEE Transactions on Knowledge and Data Engineering* 35, no. 6 (2022): 6028–6042.
32. B. Shi, Y. Wang, F. Guo, J. Shao, H. Shen, and X. Cheng, “Improving Graph Domain Adaptation With Network Hierarchy,” *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (2023): 2249–2258, <https://doi.org/10.1145/3583780.3614928>.
33. T. N. Kipf and M. Welling, “Semi-supervised Classification With Graph Convolutional Networks,” *International Conference on Learning Representations* (2017).
34. P. Velickovic, G. Cucurull, A. Casanova, et al., “Graph Attention Networks,” *ICLR 2017* (2018): 10–48550.
35. W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017): 1025–1035.
36. G. Pang, C. Shen, and A. van den Hengel, “Deep Anomaly Detection With Deviation Networks,” *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019): 353–362, <https://doi.org/10.1145/3292500.3330871>.
37. M. Cuturi, “Sinkhorn Distances: Lightspeed Computation of Optimal Transport,” *Advances in Neural Information Processing Systems* (2013): 2292–2300.
38. W. Ju, Y. Gu, Z. Mao, et al., “GPS: Graph Contrastive Learning via Multi-Scale Augmented Views From Adversarial Pooling,” *Science China Information Sciences* 68, no. 1 (2024): 112101, <https://doi.org/10.1007/s11432-022-3952-3>.
39. S. Rayana and L. Akoglu, “Collective Opinion Spam Detection: Bridging Review Networks and Metadata,” *Proceedings of the 21th Acm Sigkdd International Conference on Knowledge Discovery and Data Mining* (2015): 985–994, <https://doi.org/10.1145/2783258.2783370>.
40. P. Kaghazgaran, J. Caverlee, and A. Squicciarini, “Combating Crowdsourced Review Manipulators: A Neighborhood-Based Approach,” *Proceedings of the eleventh ACM International Conference on Web Search and Data Mining* (2018): 306–314, <https://doi.org/10.1145/3159652.3159726>.
41. M. Weber, G. Domeniconi, J. Chen, et al., Anti-money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. arXiv preprint arXiv:1908.02591 (2019).
42. X. Huang, Y. Yang, Y. Wang, et al., “Dgraph: A Large-Scale Financial Dataset for Graph Anomaly Detection,” *Advances in Neural Information Processing Systems* (2022).
43. M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying Density-Based Local Outliers,” *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (2000): 93–104, <https://doi.org/10.1145/342009.335388>.
44. F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” *Proceedings of IEEE International Conference on Data Mining* (2008): 413–422, <https://doi.org/10.1109/icdm.2008.17>.
45. P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, *Deep Graph Infomax*, Vol. 4 (ICLR (Poster), 2019).
46. S. Bandyopadhyay, L. N. S. V. Vivek, and M. N. Murty, “Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding,” *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020): 25–33.
47. Z. Chen, B. Liu, M. Wang, P. Dai, J. Lv, and L. Bo, “Generative Adversarial Attributed Network Anomaly Detection,” *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020): 1989–1992.
48. T. N. Kipf and M. Welling, Variational Graph Auto-Encoders. arXiv preprint arXiv:1611.07308 (2016).