MULTIGRAPH MESSAGE PASSING WITH BI-DIRECTIONAL MULTI-EDGE AGGREGATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) have seen significant advances in recent years, yet their application to multigraphs, where parallel edges exist between the same pair of nodes, remains under-explored. Standard GNNs, designed for simple graphs, compute node representations by combining all connected edges at once, without distinguishing between edges from different neighbors. There are some GNN architectures proposed specifically for multigraph tasks, yet these architectures perform only node-level aggregation in their message-passing layers, which limits their expressive power. Furthermore, these approaches either lack permutation equivariance when a natural edge ordering is absent, or fail to preserve the topological structure of the multigraph. To address all these shortcomings, we propose MEGA-GNN, a unified framework for message passing on multigraphs that can effectively perform diverse graph learning tasks. Our approach introduces a two-stage aggregation process in the message passing layers: first, parallel edges are aggregated, followed by a node-level aggregation that operates on aggregated messages from distinct neighbors. We show that MEGA-GNN supports permutation equivariance and invariance properties. We also show that MEGA-GNN is universal when the edges are consistently ordered. Experiments on synthetic and real-world financial transaction datasets demonstrate that MEGA-GNN either significantly outperforms or is on par with the accuracy of state-of-the-art solutions.

1 INTRODUCTION

Graph Neural Networks (GNNs) (Xu et al. (2019); Gilmer et al. (2017); Veličković et al. (2018); Corso et al. (2020); Hamilton et al. (2017)) have become Swiss Army knives for learning on graph-structured data. However, their widespread adoption has primarily focused on simple graphs, where only a single edge can connect a given pair of nodes. This simplification overlooks a crucial aspect of many real-world scenarios, where multigraphs, graphs that feature parallel edges between the same pair of nodes, are common. For instance, financial transaction networks, communication networks and transportation systems are often modeled as multigraphs, allowing multiple different interactions between the same two nodes. Existing GNN architectures are inherently limited in handling such multigraph structures since they aggregate messages only at the node level, combining information from all incoming edges at once without distinguishing between parallel edges and their sources. Although such an aggregation mechanism is sufficient to construct provably powerful GNN models on simple graphs (Maron et al. (2019); Xu et al. (2019)), the expressivity of such models are theoretically and empirically limited on directed multigraphs as shown by Egressy et al. (2024).

In the literature, only two GNN-based solutions, namely Multi-GNN (Egressy et al. (2024)) and ADAMM (Sotiropoulos et al. (2023)), have been proposed for multigraphs, yet both exhibit critical limitations as summarized in Table 1. Multi-GNN, while addressing expressivity concerns, fails to preserve permutation equivariance. ADAMM, on the other hand, fails to effectively support diverse graph learning tasks, thus limiting its practical applicability. To address these limitations, we propose **M**ulti-**E**d**G**e **A**ggregation GNNs, henceforth referred to as **MEGA-GNN**. MEGA-GNN is a unified message passing framework specifically designed for multigraphs. It employs a two-stage message aggregation process: first, parallel edges between the same two nodes are aggregated, and then, the aggregated messages from distinct neighbors are further combined at the node level.

Table 1: Closely-related work vs. our method. MP stands for message passing.

Properties & Capabilities	Multi-GNN	ADAMM	MEGA-GNN (ours)
Proof of Universality	✓		✓
Bi-directional MP	✓		✓
Edge Embeddings	✓		✓
Node Embeddings	✓	✓	✓
Permutation Equivariance		✓	✓
Multi-Edge Aggregations		✓	✓
Multi-Edge Aggregations in MP			✓

Our theoretical analyses show that MEGA-GNN is provably powerful, meaning it can detect any directed subgraph pattern within multi-graphs if a consistent ordering of the edges is possible.

Our main contributions are three fold: **(1)** We introduce a generic bi-directional message passing framework for multigraphs that incorporates artificial nodes between pairs of nodes to aggregate parallel edges before message passing, which makes it effective across diverse multigraph learning tasks. **(2)** We prove that our framework supports essential properties such as permutation equivariance, injectivity, and universality. **(3)** We validate our framework on financial transaction datasets, covering detection of illicit transactions and phishing accounts. We outperform the state-of-the-art for illicit transaction detection and match the state-of-the-art results for phishing account detection.

2 Limitations of the Existing Solutions

In the literature, two key works specifically address multigraphs: Multi-GNN Egressy et al. (2024) and ADAMM Sotiropoulos et al. (2023).

Multi-GNN introduced a provably powerful GNN architecture with simple adaptations for directed multigraphs. A notable contribution of Multi-GNN was the multigraph port ID assignment on edges, which allows the model to differentiate between edges from the same neighbor and edges from different neighbors, an essential feature for handling multigraph structures (see Figure 1). In addition to multigraph port numbering, Multi-GNN incorporates ego-IDs (You et al. (2021)) and bi-directional message passing (Jaume et al. (2019)). The authors also proved that with these three adaptations, it is possible to assign unique node IDs in connected directed multigraphs, making their solution universal.

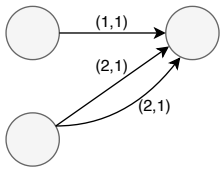

Figure 1: Directed multigraph port numbering of Multi-GNN (Egressy et al. (2024)).

The main drawback of Multi-GNN is that augmenting edge features with multigraph port numbers leads to the loss of permutation invariance and equivariance properties (see Proposition 2.1 and the proof provided in Appendix A.1). These properties are critical for many graph learning tasks to ensure that the model's predictions remain consistent under arbitrary node or edge permutations.

Proposition 2.1. *The multigraph port numbering scheme of Egressy et al. (2024), is not permutation-equivariant in the absence of a natural or contextually-driven ordering of edges.*

ADAMM proposed aggregating all parallel edges between two nodes to a single undirected super-edge, thereby transforming the multigraph into a simple graph before message passing. An initial embedding for this super-edge is computed using DeepSet and the subsequent message passing layers operate on this embedding. However, this approach results in a loss of critical structural information inherent in the multigraph. Specifically, by collapsing multiple edges into a single one, the formulation can not generate embeddings for individual edges. Consequently, it cannot perform graph learning tasks that require edge embeddings.

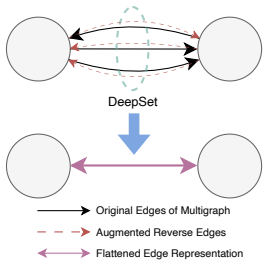

Figure 2: Illustration of ADAMM Sotiropoulos et al. (2023)

Notably, because this approach does not compute embeddings for the original edges, it cannot perform multi-edge aggregations repeatedly in several message passing layers (see Figure 2).

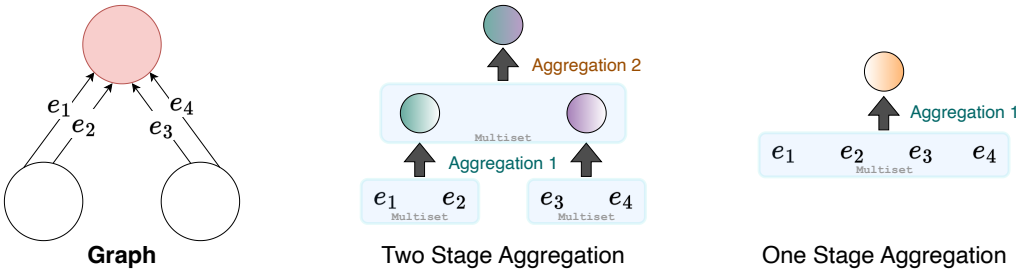

Figure 3: **Comparison of two-stage and single-stage aggregation strategies for multigraphs**. The left panel illustrates a sample multi-graph. The center panel depicts the two-stage aggregation process, wherein the first stage aggregates parallel edges, followed by a second stage node-level aggregation. The right panel shows the single-stage aggregation scheme commonly employed by standard GNNs, which would aggregate all edges, including the parallel edges, at once.

Another major limitation of ADAMM is its lack of support for bi-directional message passing, which has been shown to lead to significant accuracy gains for directed multigraphs(Egressy et al. (2024)).

In summary, a general framework for message passing on multigraphs that supports permutation invariance and a general set of graph learning tasks has yet to be proposed. This paper addresses this gap by introducing a novel message passing architecture that is explicitly designed for multigraphs.

3 Multigraph Message Passing

This section introduces a novel message-passing scheme for multigraphs which combines node-level aggregation with *multi-edge aggregation* and supports a general set of graph learning tasks such as node, edge, and graph classification. Additionally, the resulting models are permutation invariant or equivariant depending on the chosen multi-edge aggregation function and the downstream task.

3.1 Motivation for Multi-Edge Aggregations

We argue that a two-stage aggregation approach is more powerful than a single-stage aggregation in a multigraph setting, wherein aggregation of the parallel edges between the same source and destination nodes is performed before a node-level aggregation. We first give a motivating example, which shows that two-stage aggregation functions are more expressive than their single-stage counterparts.

Consider the example graph depicted in Figure 3, with four edges e_1, e_2, e_3 and e_4. Suppose that we have an aggregation function that operates on multisets and can simultaneously compute both the sum and the maximum of the elements included in the set. To demonstrate that two-stage aggregation approach is more powerful than the single-stage one, we will show that two-stage approach can perform any computation that the single-stage approach can, but not vice versa.

A single-stage scheme, can compute only $(\text{SUM}\{\!\{e_1, e_2, e_3, e_4\}\!\}$ and $\text{MAX}\{\!\{e_1, e_2, e_3, e_4\}\!\})$. The two-stage approach can also compute these results by aggregating the edges in pairs first: $(\text{SUM}\{\!\{\text{SUM}\{\!\{e_1, e_2\}\!\}, \text{SUM}\{\!\{e_3, e_4\}\!\}\}\!\}, \text{MAX}\{\!\{\text{MAX}\{\!\{e_1, e_2\}\!\}, \text{MAX}\{\!\{e_3, e_4\}\!\}\}\!\})$. However, the two-stage aggregation allows for more nuanced computations that the single-stage approach cannot perform. For example, the two-stage approach can compute $\text{SUM}\{\!\{\text{MAX}\{\!\{e_1, e_2\}\!\}, \text{MAX}\{\!\{e_3, e_4\}\!\}\}\!\}$ and $\text{MAX}\{\!\{\text{SUM}\{\!\{e_1, e_2\}\!\}, \text{SUM}\{\!\{e_3, e_4\}\!\}\}\!\}$, which the single-stage approach cannot compute.

This distinction is significant; the two-stage aggregation enables us to capture the statistics of edges originating from different neighbors. Consider a financial application, in which a receiver may receive the highest total payment from one sender (Sender 1), while the highest single payment may originate from another sender (Sender 2). Using the two-stage scheme, we can detect the maximum of the sums of payments from each sender, allowing us to distinguish between their contributions. In contrast, the single-stage scheme can only compute the maximum among all payments without differentiating between identical and distinct senders, which can obscure critical insights.

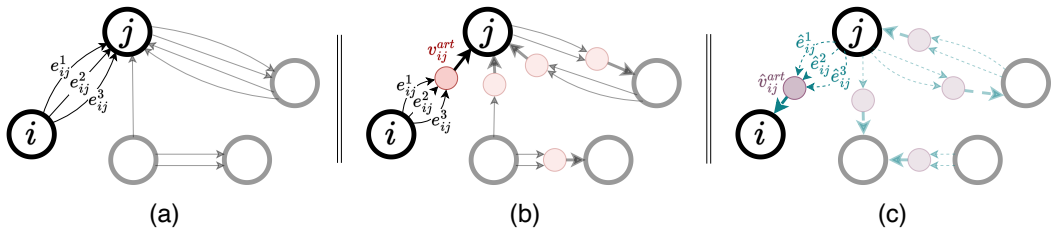

Figure 4: **Illustration of Multi-Edge Aggregation using artificial nodes in a multigraph**. (a) A multigraph with parallel edges $e_{ij}^1, e_{ij}^2, e_{ij}^3$ between the nodes i and j. (b) Artificial nodes positioned between adjacent pairs of nodes to handle the aggregation of parallel edges. First, the information from the *parallel* edges are aggregated into some embedding vectors in the artificial nodes, and then the destination node performs a node-level aggregation on these embeddings. (c) Illustration of the reverse message passing mechanism. In directed multigraphs, reverse edges are created in the reverse direction of the original edges, and then additional artificial nodes are introduced to handle the aggregation of these reverse edges. Separate message computations are performed to handle the original and reversed edges. The result is a bi-directional multigraph message passing solution.

3.2 Preliminaries

Notation: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed multigraph with $|\mathcal{V}| = v$ nodes and $|\mathcal{E}| = e$ edges. The edges $\mathcal{E} = \{e_{ij} = (i, j) \mid i, j \in \mathcal{V}\}$ form an ordered multi-set of pairs of nodes, where each pair (i, j) represents a directed edge from node i to node j. In a multigraph, multiple edges may exists between between the same source node i and the same destination node j, those are referred to as parallel edges. The incoming and outgoing neighbors of node $i \in \mathcal{V}$ are denoted by $N_{in}(i) = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$ and $N_{out}(i) = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$, respectively.

Moreover, we consider attributed graphs such that the nodes and the edges of the graph $\mathcal{G}$ are associated with feature vectors $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{e}_{ij} \in \mathbb{R}^K$, respectively.

Message-Passing GNNs: In the standard Message Passing Neural Networks (MPNNs) proposed by Gilmer et al. (2017), the information on the graph is propagated using an iterative application of local message passing operations, as described by the following formula:

$$\mathbf{x}_i^{(l)} = g_v^{(l-1)}\left(\mathbf{x}_i^{(l-1)}, \text{AGG}\{f^{(l-1)}(\mathbf{x}_j^{(l-1)}, \mathbf{e}_{ji}^{(l-1)}) \mid j \in N_{in}(i)\}\right), \tag{1}$$

where $f^{(l)}$ is a function that constructs the messages from the incoming neighbors of node i. These messages are then aggregated using the AGG function, which is typically a permutation-invariant operation such as sum, mean, or max. After the message aggregation step, the hidden representation of node i is updated using the update function $g_v^{(l)}$. Additionally, since edge features are considered, the edge features e_{ij} are also updated according to the following formula:

$$\mathbf{e}_{ji}^{(l)} = g_e^{(l-1)}\big(\mathbf{x}_j^{(l-1)}, \mathbf{e}_{ji}^{(l-1)}, \mathbf{x}_i^{(l-1)}\big), \tag{2}$$

where $g_e^{(l)}$ is the update function for the edges. Note that the equations 1 and 2 are general and not specifically formulated for multigraphs.

3.3 Multigraph Message Passing with Multi-Edge Aggregations

Commonly used GNN architectures Kipf & Welling (2017); Veličković et al. (2018); Hamilton et al. (2017) are not inherently designed to handle multigraphs. We argue that in the context of multigraphs, performing message aggregation solely at the node level can limit the expressive power of these models on graph learning tasks. To address this limitation, we propose a two-stage aggregation process, wherein the first stage, termed *multi-edge aggregation*, aggregates parallel edges, followed by a second stage node-level aggregation. To enable effective multi-edge aggregation, we introduce novel *artificial nodes* into the multigraph.

Let $\mathcal{E}^{unique} \subseteq \mathcal{E}$ denote the set of unique edges obtained by removing duplicated entries from the edge multi-set $\mathcal{E}$. We define the set of *artificial nodes*, positioned between each unique pair of nodes in the multigraph as shown in Figure 4 (b) as follows:

$$\mathcal{V}^{art} = \{v_{ij}^{art} \mid (i,j) \in \mathcal{E}^{unique}\}. \tag{3}$$

Similarly the incoming neighbors of the *artificial node* v_{ij}^{art} is denoted by $N_{in}(v_{ij}^{art}) = \{e_{ij} | (i,j) \in \mathcal{E}\}$, corresponding to all parallel edges between nodes i and j in the original multigraph $\mathcal{G}$. Let $h_{ij} \in \mathbb{R}^d$ be the latent representation of the *artificial node* v_{ij}^{art}.

The message-passing algorithm for multigraphs is adapted to perform information aggregation in two stages. First, parallel edges are aggregated at the artificial nodes, thereby reducing the multigraph to an equivalent simple graph:

$$\mathbf{h}_{ji}^{(l-1)} = \text{EdgeAgg}\{\mathbf{e}_{ji}^{(l-1)} \mid e_{ji} \in N_{in}(v_{ji}^{art})\}. \tag{4}$$

In the second stage, aggregation occurs at the original graph nodes:

$$\begin{aligned} \mathbf{a}_i^{(l)} &= \text{AGG}\{f^{(l-1)}(\mathbf{x}_j^{(l-1)}, \mathbf{h}_{ji}^{(l-1)}) \mid j \in N'_{in}(i)\} \\ \mathbf{x}_i^{(l)} &= g_v^{(l-1)}\left(\mathbf{x}_i^{(l-1)}, \mathbf{a}_i^{(l)}\right), \end{aligned} \tag{5}$$

where $N'_{in}(i) = \{j \in \mathcal{V} | (j,i) \in \mathcal{E}^{unique}\}$. As in 2, the latent representations of each individual edge can be updated without loss of generality:

$$\mathbf{e}_{ji}^{(l)} = g_e^{(l-1)}\big(\mathbf{x}_j^{(l-1)}, \mathbf{e}_{ji}^{(l-1)}, \mathbf{h}_{ji}^{(l-1)}\big). \tag{6}$$

This two-stage aggregation process, enabled by our innovative use of artificial nodes, preserves the topology of the original multigraph. Equations 4, 5, and 6 can be applied iteratively, allowing for effective propagation and updates of both node and edge features in each message-passing layer. Unlike previous methods such as ADAMM Sotiropoulos et al. (2023), which collapses parallel edges into a single edge prior to message passing—thereby preventing updates on the original multigraph edges—our approach maintains distinct edges through individual updates, enabling richer feature propagation in multigraph learning tasks.

3.4 Bi-directional Multigraph Message Passing

Bi-directional message passing (Egressy et al. (2024)) has been shown to significantly improve model expressivity, especially for directed graphs. In standard message passing, a node does not receive information from its outgoing neighbors unless those neighbors are also incoming. Consequently, the node is unable to account for its outgoing edges. The empirical results of (Egressy et al. (2024)) further highlight the importance of bi-directional message passing in multi-graphs, demonstrating its critical role in achieving improved performance across various datasets

We ensure that our proposed model, MEGA-GNN, also supports bi-directional message passing for multigraphs. In MEGA-GNN, a node is able to receive messages from its outgoing neighbors by using the reversed versions of their outgoing edges. Formally, the reversed edge set is defined as $\hat{\mathcal{E}} = \{\hat{e}_{ji} = (j,i) \mid (i,j) \in \mathcal{E}\}$, resulting in the construction of a *reversed* multigraph, denoted as $\hat{\mathcal{G}}$ (illustrated in Figure 4(c)). This process is analogous to utilizing a relational GNN as described in Schlichtkrull et al. (2018), with two distinct edge types, where the reverse edges are added to the original multigraph $\mathcal{G}$. Notably, the node representations remain unchanged during this process. In the reversed multigraph, *artificial nodes* $\hat{\mathcal{V}}^{art}$ are placed between each unique pair of nodes, in a manner that mirrors their positioning in the original multigraph:

$$\hat{\mathcal{V}}^{art} = \{\hat{v}_{ij}^{art} \mid (i,j) \in \hat{\mathcal{E}}^{unique}\}. \tag{7}$$

The incoming neighbors of an *artificial node* $\hat{v}_{ij}^{art}$ are defined as $N_{in}(\hat{v}_{ij}^{art}) = \{\hat{e}_{ij} \mid (i,j) \in \hat{\mathcal{E}}\}$, which corresponds to all parallel edges between nodes i and j in the reversed multigraph $\hat{\mathcal{G}}$. Let $\mathbf{h}_{ij} \in \mathbb{R}^d$ and $\hat{\mathbf{h}}_{ij} \in \mathbb{R}^d$ be the latent representation of the *artificial node* in the original and reversed multigraphs, respectively.

$$\hat{\mathbf{h}}_{ji}^{(l-1)} = \text{EdgeAgg}\{\hat{\mathbf{e}}_{ji}^{(l-1)} \mid e_{ji} \in N_{in}(\hat{v}_{ji}^{art})\}, \tag{8}$$

$\hat{\mathbf{h}}_{ji}^{(l)}$ is the aggregation of the incoming edges from node j to node i in the reversed multigraph,

$$\begin{aligned}\hat{\mathbf{a}}_i^{(l)} &= \text{AGG}\{\hat{f}^{(l-1)}(\mathbf{x}_j^{(l-1)}, \hat{\mathbf{h}}_{ji}^{(l-1)}) \mid j \in \hat{N}'_{in}(i)\} \\ \mathbf{x}_i^{(l)} &= g_v^{(l-1)}\left(\mathbf{x}_i^{(l-1)}, \mathbf{a}_i^{(l)}, \hat{\mathbf{a}}_i^{(l)}\right),\end{aligned} \tag{9}$$

$\hat{N}'_{in}(i) = \{j \in \mathcal{V} | (i,j) \in \hat{\mathcal{E}}^{unique}\}$ and $\hat{f}^{(l)}$ is a function that constructs the messages for the reversed edges. As in 2, the latent representations of each individual edge can be updated without loss of generality:

$$\hat{\mathbf{e}}_{ji}^{(l)} = \hat{g}_e^{(l-1)}\big(\mathbf{x}_j^{(l-1)}, \hat{\mathbf{h}}_{ji}^{(l-1)}, \mathbf{x}_i^{(l-1)}\big), \tag{10}$$

where $\hat{g}_e^{(l)}$ is the update function for the reversed edges. To compute $a_i^{(l)}$ and $e_{ji}^{(l)}$ we use the same equations given in the previous subsection.

3.5 Properties

In this section, we establish several key theoretical properties of our proposed method, proving its generalization capabilities and expressiveness.

Theorem 3.1. ***Permutation Equivariance.*** *Given permutation-invariant EdgeAgg and AGG functions, the message-passing framework defined by Equations 4, 5, and 6 is permutation-equivariant.*

The proof of Theorem 3.1 is provided in Appendix A.2. For neural networks operating on graph-structured data, permutation equivariance is a crucial property, ensuring that the model's output remains consistent regardless of the ordering of nodes and edges. Our model is explicitly designed to maintain permutation equivariance at both the node and edge levels. Achieving permutation invariance for graph-level outputs is straightforward by using a permutation-invariant *graph readout* function.

Corollary 3.1. ***Injectivity.*** *Given injective edge and node aggregation functions EdgeAgg and AGG, respectively, the message-passing framework defined by Equations 4, 5, and 6 is also injective.*

The proof of Corollary 3.1 follows Lemma 5 of Xu et al. (2019), which shows that there exists aggregation functions that are injective over multisets and that these functions can be parameterized by neural networks. By extending this result to our two-stage aggregation mechanism—where aggregation occurs first over parallel edges and then over unique neighbors—our framework ensures injectivity. This is due to the composition of injective functions remains injective, which ensures that our method preserves distinguishing power across different graph structures.

Theorem 3.2. ***Universality.*** *The message-passing framework defined by Equations 4, 5, and 6 can compute unique node IDs in connected multigraphs when there is a consistent ordering of the edges.*

The proof of the Theorem 3.2 can be found in the Appendix A.3. It is known that when unique node IDs are available, sufficiently-powerful standard MPNNs are universal and, thus, capable of detecting any subgraph pattern (Loukas (2020); Abboud et al. (2021)). Theorem 3.2 states that, when there is a consistent ordering among the edges, MEGA-GNN can compute unique node IDs in multigraphs. Consequently, our method is universal and capable of detecting any directed subgraph pattern in multigraphs.

All the above claims remain valid when the bi-directional message passing mechanism of Section 3.4 is used. Extending Theorem 3.1 and Corollary 3.1 to the bi-directional case is straightforward, and the proof of Theorem 3.2 in Appendix A.3 already incorporates bi-directional message passing.

4 Experimental Results

In this section, we present our experiments designed to evaluate the performance of MEGA-GNN. We focus on financial transaction graphs, which are multigraphs, and address two key applications in financial crime analysis: illicit transaction detection through edge classification and phishing account detection in Ethereum transactions through node classification. We provide detailed descriptions of the datasets used in our experiments and the baselines against which we compare our method.

Anti-Money Laundering (AML): Given the concerns over privacy and legal regulations, there is a scarcity of publicly available financial transaction data. Recently, Altman et al. (2023) release realistic synthetic financial transaction data for money laundering detection. In this synthetic data, a multi-agent world simulates transactions between entities such as banks, individuals, and companies. We use Small_LI, Small_HI, Medium_LI, and Medium_HI datasets, where LI indicates low-illicit and HI indicates high-illicit, referring to the ratios of illicit transactions in the datasets. The task is to perform binary classification on the transactions, labeling them as either illicit or non-illicit.

Ethereum Phishing Transaction Network: Since access to real financial transaction data from banks is limited due to privacy concerns, cryptocurrencies provide an alternative data source. In our study, we use the Ethereum transaction dataset (Chen et al. (2021)), where accounts are treated as nodes and transactions as edges. Each node has a label indicating whether it is a phishing node or not. The edges contain attributes such as transaction amount and timestamp.

Implementation: We implement our solution using PyTorch Geometric Fey & Lenssen (2019). We apply temporal splits of 64-19-17 (AML-Small) and 61-17-22 (AML-Medium) on the AML datasets and temporal splits of 65-15-20 on the ETH dataset across the edges (transactions). We adopt Ego-IDs You et al. (2021) and incorporate bi-directional message passing (see Section 3.4 for details). To ensure the statistical significance of our results, each experiment is repeated five times with different random seeds, and the mean and the standard deviation of these runs are reported. Further details about the models and the hyperparameters are provided in Appendix B.2 and Appendix B.1, respectively.

Baselines We select several baselines for comparison. In particular, we include GFP+LightGBM and GFP+XGBoost Blanuša et al. (2024), as well as Multi-GNN Egressy et al. (2024), as the state-of-the-art approaches. Additionally, within the general multigraph message passing framework (see Section 3.3), we explore different aggregation methods such as GIN (Xu et al. (2019)), PNA (Corso et al. (2020)) and GenAgg (Kortvelesy et al. (2023)), can be applied to both multi-edge aggregation (see Appendix B.2 for details) and node-level aggregation. The flexibility of our proposed model, allowing the use of different aggregation functions for multi-edges and nodes, enables the exploration of various combinations.

4.1 Edge Classification

Table 2: Minority-class F1 scores (%) on AML edge classification task. GIN and PNA baselines are designed for simple graphs. GFP performs a combinatorial search to extract discriminative subgraph patterns from multigraphs. Multi-GNNs are specifically designed to mine patterns in multigraphs.

Method	AML Small HI	AML Small LI	AML Medium HI	AML Medium LI
GIN Xu et al. (2019)	28.70 ± 1.13	7.90 ± 2.78	42.20 ± 0.44	3.86 ± 3.62
PNA Corso et al. (2020)	56.77 ± 2.41	14.85 ± 1.46	59.71 ± 1.91	27.73 ± 1.65
GFP+LightGBM Blanuša et al. (2024)	62.86 ± 0.25	20.83 ± 1.50	59.48 ± 0.15	24.74 ± 0.46
GFP+XGBoost Blanuša et al. (2024)	63.23 ± 0.17	27.28 ± 0.69	65.70 ± 0.26	31.03 ± 0.22
Multi-GIN Egressy et al. (2024)	64.79 ± 1.22	26.88 ± 6.63	58.92 ± 1.83	16.30 ± 4.73
Multi-PNA Egressy et al. (2024)	68.16 ± 2.65	33.07 ± 2.63	66.48 ± 1.63	36.07 ± 1.17
Multi-GenAgg	64.92 ± 3.85	36.35 ± 4.07	66.44 ± 1.30	37.72 ± 0.72
MEGA-GIN	70.83 ± 2.18	43.66 ± 0.54	68.83 ± 1.66	39.03 ± 1.88
MEGA-PNA	74.01 ± 1.55	45.07 ± 2.26	78.26 ± 0.11	49.40 ± 0.54
MEGA-GenAgg	74.88 ± 0.38	46.29 ± 0.41	76.69 ± 0.30	44.89 ± 0.06
MEGA(PNA)-GIN	74.69 ± 0.50	41.36 ± 0.73	72.63 ± 0.60	42.23 ± 0.24
MEGA(GenAgg)-GIN	72.24 ± 3.42	45.73 ± 0.47	71.34 ± 1.78	45.33 ± 2.08
MEGA(GenAgg)-PNA	71.95 ± 1.67	41.94 ± 2.02	76.90 ± 0.70	47.69 ± 0.48

We evaluate our proposed MEGA-GNN models across four different synthetic AML datasets, as shown in Table 2. Six distinct MEGA-GNN models are employed, using different combinations of aggregation functions across the two stages. For clarity, we adopt the following naming convention: MEGA-GIN refers to the model that applies GIN Xu et al. (2019) aggregation at both stages, while MEGA(GenAgg)-GIN applies GenAgg (Kortvelesy et al. (2023)) for multi-edge aggregation and GIN for node-level aggregation. The results consistently prove the effectiveness of our multi-edge

aggregation mechanism in improving edge classification performance across datasets with varying sizes and levels of class imbalance, regardless of the aggregation function combinations used.

Table 2 shows that our MEGA-GNN models consistently outperform both traditional GNN baselines, such as GIN and PNA, and state-of-the-art models, such as Multi-GNN and those based on GFP, yielding significant performance benefits across all datasets. On average, our models achieve a 9.25% increase in minority-class F1 score on the HI datasets and a 13.31% improvement on the LI datasets over the state-of-the-art. A clear trend of enhanced performance with the incorporation of multi-edge aggregation is visible across all these datasets. For instance, on the AML Medium HI dataset, MEGA-GIN boosts performance from 58.92% (Multi-GIN) to 68.83%, and similarly, MEGA-PNA advances the performance from 66.48% (Multi-PNA) to 78.26%. These gains of nearly 10% are consistent across all the AML datasets, underscoring the power of multi-edge aggregation.

4.2 Node Classification

Table 3: Minority-class F1 scores ($\%$) on ETH node classification task. Both ADAMM and Multi-GNN are specifically designed for multigraphs. ADAMM does not support edge classification.

Method	F1
ADAMM-GIN Sotiropoulos et al. (2023)	34.73 ± 15.75
ADAMM-PNA Sotiropoulos et al. (2023)	37.99 ± 5.41
ADAMM-GenAgg Sotiropoulos et al. (2023)	33.10 ± 19.32
Multi-GIN Egressy et al. (2024)	51.34 ± 3.92
Multi-PNA Egressy et al. (2024)	64.61 ± 1.40
Multi-GenAgg	44.60 ± 21.50
MEGA-GIN	55.19 ± 2.33
MEGA-PNA	60.02 ± 5.10
MEGA-GenAgg	56.28 ± 5.77
MEGA(PNA)-GIN	56.41 ± 3.35
MEGA(GenAgg)-GIN	56.67 ± 1.81
MEGA(GenAgg)-PNA	63.71 ± 1.23

We further evaluate our proposed models on real-world financial transactions using the ETH dataset Chen et al. (2021), which focuses on the identification of phishing accounts. The results, summarized in Table 3, highlight the effectiveness of our MEGA-GNN variants. Notably, MEGA(GenAgg)-PNA achieves competitive performance compared to state-of-the-art methods.

Our method performs competitively with existing approaches, demonstrating significant improvements over most baseline models. For instance, MEGA-GIN improves the F1 score by 3.85% over the Multi-GIN model, while MEGA-GenAgg surpasses GenAgg-based GNN by 11.68%. The best-performing model, MEGA(GenAgg)-PNA, achieves a near state-of-the-art result, with only a marginal 0.9% difference in F1 score compared to Multi-PNA (Egressy et al. (2024)) on the minority F1 score, underscoring the robustness of our approach.

When compared to the ADAMM (Sotiropoulos et al. (2023)) baseline, our MEGA-GNN variants exhibit a striking improvement, consistently delivering over 20% higher performance across base architectures (GIN, PNA, GenAgg). These results highlight the power of our multi-edge aggregation mechanism as well as the bi-directional message passing capabilities we contributed in Section 3.4.

5 Related Work

The expressive power of GNNs is crucial in evaluating their capabilities, and is mainly discussed in the context of simple graphs, which can have only a single edge between two graph nodes. Xu et al. (2019) demonstrated that the expressive power of standard message-passing GNNs on simple graphs is fundamentally limited by the 1-WL (Weisfeiler-Lehman) test for distinguishing isomorphic graphs. To address this limitation, they introduced the Graph Isomorphism Network (GIN), which is provably the most expressive among standard message-passing GNNs. Recent research on enhancing GNN expressivity has pursued several directions. Higher-order GNNs were proposed by

Maron et al. (2019); Morris et al. (2019), Bouritsas et al. (2023); Barceló et al. (2021), leverage precomputed local structural features around target nodes, while Bevilacqua et al. (2022; 2024); Frasca et al. (2022) focus on increasing expressivity through subgraph-based aggregation techniques.

The expressiveness of GNNs is closely linked to their universality, as more expressive models can approximate a broader class of functions. Sato et al. (2021); Loukas (2020) demonstrated that incorporating unique node identifiers can make GNNs universal, though this comes at the cost of losing permutation invariance. However, Abboud et al. (2021) showed that partially randomized node features can achieve universality while preserving permutation invariance. In a related line of work, set functions, such as those proposed in Zaheer et al. (2017), have been shown to be universal under certain input constraints, and Fuchs* & Veličković* (2023) made the connection between universal set and graph functions. While not theoretically more expressive than GIN, Corso et al. (2020) demonstrated that combining arbitrary aggregation functions leads to better empirical performance. In Kortvelesy et al. (2023), a learnable aggregation functions were proposed that improve sample efficiency compared to both PNA and DeepSet-based aggregations on multiset neighborhoods.

There has been relatively little work on GNNs for multigraphs. Two notable prior works are Sotiropoulos et al. (2023) and Egressy et al. (2024). The former (ADAMM) transforms a multigraph into a simple graph using a DeepSet-based approach Zaheer et al. (2017), while the latter (Multi-GNN) uses simple adaptations on top of baseline GNN methods to achieve universality.

6 Discussion and Conclusion

We contribute the first message-passing framework explicitly designed for multigraphs, enabling a two-stage aggregation scheme in every message-passing layer. Unlike the method of Egressy et al. (2024), our framework guarantees both permutation equivariance and injectivity. Our framework also achieves universality when we make the same assumptions made by Egressy et al. (2024). Furthermore, unlike the method of Sotiropoulos et al. (2023), we do not transform a multigraph into a simple graph before the message passing layers. Thus, our message passing framework supports a more general set of multigraph learning tasks including the classification of multigraph edges.

Our experiments on financial transaction datasets show that MEGA-GNN significantly outperforms or matches the accuracy of state-of-the-art multigraph GNN models. Notably, when detecting illicit transactions on the synthetic AML datasets of Altman et al. (2023), we achieve improvements of up to 13.31% in minority-class F1 scores over the method of Egressy et al. (2024), demonstrating the effectiveness of MEGA-GNN. When detecting phishing accounts on the Ethereum transaction dataset of (Chen et al. (2021)), we achieve improvements of around 20% in minority-class F1 scores over the method of Sotiropoulos et al. (2023), while matching the results of Egressy et al. (2024).

Limitations and Future Work. Our work does not consider dynamically evolving multigraphs with several different snapshots. Future work could investigate strategies to support such setups. Other interesting research directions include improving the scalability and GPU performance of our methods, and building federated multigraph learning solutions with differential privacy guarantees.

7 Reproducibility Statement

To ensure the reproducibility of our research, we will open-source our code for the various node and edge aggregation techniques and different multigraph GNNs. The code will be available in a publicly accessible repository. Furthermore, all experiments conducted as part of this study utilized publicly available datasets.

8 Ethics Statement

The UN estimates 2-5% of global GDP or $0.8-$2.0 trillion dollars are laundered globally each year. We aim to address this critical issue by improving the accuracy of detecting illicit financial activities. Notably, our work supports the global fight against financial crime by contributing provably powerful GNN-based methods for analysing financial transaction networks. Moreover, the datasets used in our experiments do not involve any personally identifiable information. Thus, our work does not lead to any discrimination/bias/fairness concerns or privacy, security, and legal compliance issues.

REFERENCES

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization, 2021. URL https://arxiv.org/abs/2010.01179.

Erik Altman, Jovan Blanuša, Luc Von Niederhäusern, Beni Egressy, Andreea Anghel, and Kubilay Atasu. Realistic synthetic financial transactions for anti-money laundering models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Pablo Barceló, Floris Geerts, Juan Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25280–25293. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d4d8d1ac7e00e9105775a6b660dd3cbb-Paper.pdf.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dFbKQaRk15w.

Beatrice Bevilacqua, Moshe Eliasof, Eli Meirom, Bruno Ribeiro, and Haggai Maron. Efficient subgraph GNNs by learning effective selection policies. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gppLqZLQeY.

Jovan Blanuša, Maximo Cravero Baraja, Andreea Anghel, Luc von Niederhäusern, Erik Altman, Haris Pozidis, and Kubilay Atasu. Graph feature preprocessor: Real-time extraction of subgraph-based features from transaction graphs, 2024. URL https://arxiv.org/abs/2402.08593.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2023. doi: 10.1109/TPAMI.2022.3154319.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Liang Chen, Jiaying Peng, Yang Liu, Jintang Li, Fenfang Xie, and Zibin Zheng. Phishing scams detection in Ethereum transaction network. *ACM Trans. Internet Technol.*, 21(1):1–16, Feb. 2021. doi: 10.1145/3398071.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13260–13271. Curran Associates, Inc., 2020.

Béni Egressy, Luc von Niederhäusern, Jovan Blanusa, Erik Altman, Roger Wattenhofer, and Kubilay Atasu. Provably powerful graph neural networks for directed multigraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. Underline Science Inc., 2024. doi: 10.48448/p6r4-c103.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and extending subgraph GNNs by rethinking their symmetries. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=sc7bBHAmcN.

Fabian B. Fuchs* and Petar Veličković*. Universality of neural networks on sets vs. graphs. In *ICLR Blogposts 2023*, 2023. URL https://iclr-blogposts.github.io/2023/blog/2023/sets-and-graphs/. https://iclr-blogposts.github.io/2023/blog/2023/sets-and-graphs/.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272. JMLR.org, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

Guillaume Jaume, An phi Nguyen, María Rodríguez Martínez, Jean-Philippe Thiran, and Maria Gabrani. edgnn: a simple and powerful gnn for directed labeled graphs, 2019. URL https://arxiv.org/abs/1904.08745.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Ryan Kortvelesy, Steven Morad, and Amanda Prorok. Generalised f-mean aggregation for graph neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JMrIeKjTAe.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1l2bp4YwS.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bb04af0f7ecaee4aae62035497da1387-Paper.pdf.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: higher-order graph neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019. doi: 10.1609/aaai.v33i01.33014602. URL https://doi.org/10.1609/aaai.v33i01.33014602.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM*, 2021.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pp. 593–607. Springer International Publishing, 2018. ISBN 978-3-319-93417-4.

Konstantinos Sotiropoulos, Lingxiao Zhao, Pierre Jinghong Liang, and Leman Akoglu. Adamm: Anomaly detection of attributed multi-graphs with metadata: A unified neural network approach. In *2023 IEEE International Conference on Big Data (BigData)*, 2023. doi: 10.1109/BigData59044.2023.10386245.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

A Appendix

A.1 Proof of Proposition 2.1

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a multigraph, the nodes of which are associated with feature vectors $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{e}_{ij} \in \mathbb{R}^K$. Assume that each edge $e \in \mathcal{E}$ is assigned a port number $p(e)$ by a given port numbering scheme, and these port numbers are used as edge features as proposed by Egressy et al. (2024).

Consider a scenario where there is no inherent or contextually-driven ordering of the edges (e.g., no timestamps or other distinguishing features that naturally dictate how the ports are numbered). As a result, the assignment of port numbers to edges is arbitrary. For a node v with d incoming edges, there are $d!$ possible ways to assign port numbers to the incoming edges, corresponding to all possible permutations.

The latent representation of the node i at layer l is computed as follows, where ϕ is the message function, and $p(j, i)$ is the port number assigned to the edge between node j and node i"

$$\mathbf{x}_i^{(l)} = \sum_{j \in N(i)} \phi\big(\mathbf{x}_j^{(l-1)}, [\mathbf{e}_{ji}^{(l-1)} \,||\, p(j,i)]\big). \tag{11}$$

Now, assume that the GNN with port numbering is permutation equivariant. If our assumption is correct, permuting the port numbers assigned to edges will not affect the final node embeddings. Let σ be a permutation of the port numbers. Applying this permutation to the port numbers of the edges yields a new port assignment $p_\sigma(e)$. Under this permutation, the node representation would be:

$$\hat{\mathbf{x}}_i^{(l)} = \sum_{j \in N(i)} \phi\big(\mathbf{x}_j^{(l-1)}, [\mathbf{e}_{ji}^{(l-1)} \,||\, p_\sigma(j,i)]\big). \tag{12}$$

Since we are assuming the GNN is permutation-equivariant, the node embeddings should remain the same after permuting the port numbers:

$$\mathbf{x}_i^{(l)}(p) = \hat{\mathbf{x}}_i^{(l)}(p_\sigma) \tag{13}$$

This implies that for all possible permutations σ, the output embeddings of the nodes should be invariant. Let the GNN, f, mimics the Algorithm 1 from Egressy et al. (2024), which computes unique node IDs based on port numbers. If we apply this GNN, f, to a *Star Graph* with order $n > 3$ with different port number permutations, resulting node embeddings must be distinct. Formally, applying f to a the graph with port numbering p gives the node embeddings $\mathbf{X}^{(\mathbf{l})}$, and applying f to the graph with a permuted port numbering p_σ, gives a different set of node embeddings $\hat{\mathbf{X}}^{(l)}$:

$$\mathbf{X}^{(l)} = f(\mathcal{G}(\mathcal{X}, p) \neq f(\mathcal{G}(\mathcal{X}, p_\sigma)) = \hat{\mathbf{X}}^{(l)}. \tag{14}$$

This result directly contradicts our earlier assumption that the model is permutation-equivariant with respect to node ordering, preserves the identity of node outputs. Specifically, if the GNN were permutation-equivariant, we would expect $\mathbf{X}^{(l)} = \hat{\mathbf{X}}^{(l)}$, but instead the permutation of port numbers leads to different output. Illustration of this contradictory example is given in Figure 5.

Therefore, we conclude that the arbitrary assignment of port numbers is not permutation-equivariant with respect to node orderings.

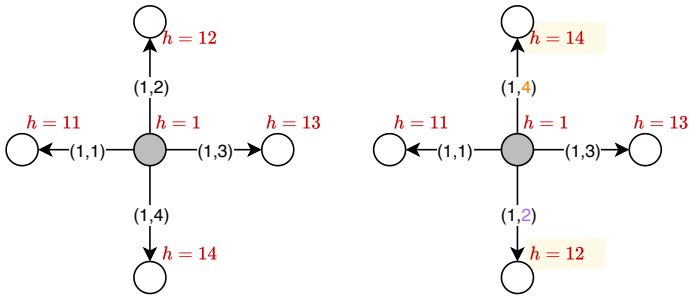

Figure 5: Counterexample for the permutation-equivariance of Multi-GNN Egressy et al. (2024). The left panel shows the graph $\mathcal{G}$ with one permutation of the port numbering, while the right panel illustrates a different permutation of the assigned port numbers. Notice how the node embeddings change when computed using Algorithm 1 from Egressy et al. (2024), highlighting that different port number assignments lead to different embeddings, thus violating permutation-equivariance

.

A.2 Proof of Theorem 3.1

We adopt the notation and terminology introduced in Section 3.3 of this paper to ensure consistency and ease of reference.

Proof: The proposed message passing layer performs two aggregations over the neighborhood of a target node i. The first is the multi-edge aggregation, in which the features of the parallel edges are aggregated at artificial nodes. The one-hop neighborhood of artificial node v_{ji}^{art} forms a multi-set:

$$\mathbf{E}_{N_{in}(v_{ji}^{art})} = \{\!\{\mathbf{e}_{ji} \mid (j,i) \in N_{in}(v_{ji}^{art})\}\!\}. \tag{15}$$

Let's define a permutation-invariant multi-edge aggregation function ψ, which operates on the neighbourhood of an artificial node (the parallel edges):

$$\mathbf{h}_{ji} = \psi(\mathbf{E}_{N_{in}(v_{ji}^{art})}) \quad \text{for} \quad e_{ji} \in \mathcal{E}^{unique}. \tag{16}$$

Since ψ is a permutation-invariant function, for any permutation function ρ acting on the neighboring edges (parallel edges) of the artificial node, we have $\psi(\rho \cdot \mathbf{E}_{N_{in}(v_{ji}^{art})}) = \rho \cdot \psi(\mathbf{E}_{N_{in}(v_{ji}^{art})})$.

The second aggregation is then performed over the neighborhood of the target nodes, all of which happen to be artificial nodes associated with distinct neighbors in the original graph (see Figure 4). This neighborhood also forms a multi-set:

$$\mathbf{X}_{N_{in}(v_i)} = \{\!\{\mathbf{h}_{ji} \mid (j,i) \in N_{in}(v_i)\}\!\}. \tag{17}$$

Similarly, let's define a permutation-invariant node-level aggregation function ϕ, which operates over the neighborhood of the target nodes:

$$\mathbf{x}_i = \phi(\mathbf{x}_i, \mathbf{X}_{N_{in}(v_i)}) \quad \text{for} \quad v_i \in \mathcal{V}. \tag{18}$$

Since ϕ is a permutation-invariant function, for any permutation function π acting on the neighboring edges of a target node i, we have $\phi(\pi \cdot (\mathbf{x}_i, \mathbf{X}_{N_{in}(v_i)})) = \pi \cdot \phi((\mathbf{x}_i, \mathbf{X}_{N_{in}(v_i)}))$.

In our framework, ψ performs multi-edge aggregation, handling parallel edges, while ϕ is responsible for node-level aggregation, combining messages from distinct neighbors of the target node. The two-stage aggregation scheme integrates ψ and ϕ within a single message passing layer. Since the composition of permutation-invariant functions remains permutation-invariant, our message passing layer $(\psi \circ \phi)$ is invariant to the permutations of neighboring nodes and edges $(\rho \circ \pi)$. Unlike simple graphs, node permutations do not directly imply edge permutations due to the presence of parallel edges. Thus, we explicitly define the permutation of parallel edges, ρ, ensuring that our message passing layer remains permutation-invariant to both nodes and edges in the neighborhood of the target node.

Finally, as demonstrated by Bronstein et al. (2021), the composition of permutation-invariant layers $(f = \psi \circ \phi \circ \psi \circ \phi \cdots)$ allows the construction of functions f that are equivariant to symmetry

Algorithm 1 BFS Node ID Assignment

Input: Connected directed multigraph $G = (V, E)$ with n nodes and m edges, diameter D, and root node $r \in V$. Active nodes $X \subseteq V$ and finished nodes $F \subseteq V$. Edge Labeling $L : E \to [1, m]$

Output: Unique node IDs $h(v)$ for all $v \in V$ (in base $2n$)

$h(r) \leftarrow 1$; $h(v) \leftarrow 0$ for all $v \in V \setminus \{r\}$
$F \leftarrow \emptyset$; $X \leftarrow \{r\}$
for $k \leftarrow 1$ **to** D **do**
  for $v \in V$ **do**
    if $v \in X$ **then**
      send $h(v) \parallel \min\{\!\{L((v,u))_{\text{out}}\}\!\}$ to $u \in N_{\text{out}}(v)$
      send $h(v) \parallel m + \min\{\!\{L((u,v))_{\text{in}}\}\!\}$ to $u \in N_{\text{in}}(v)$
      $F \leftarrow F \cup \{v\}$; $X \leftarrow X \setminus \{v\}$
    end if
    if $v \notin F$ **then**
      if Incoming messages $M(v) \neq \emptyset$ **then**
        $h(v) \leftarrow \min\{M(v)\}$
        $X \leftarrow X \cup \{v\}$
      end if
    end if
  end for
end for

group actions. In the multigraph domain, this symmetry group includes permutations of both nodes and edges, as node permutations do not directly induce edge permutations due to the presence of parallel edges. The overall permutation equivariance of the MEGA-GNN model follows from the fact that each permutation-invariant message passing layer operates independently on each node's neighborhood, regardless of the ordering of nodes or edges. Specifically, for any permutation $g \in \sum_n$ acting on the set of node and edge indices, the model's output satisfies $f(g \cdot X) = g \cdot f(X)$.

A.3 Proof of Theorem 3.2

Proof: Given a graph $G(V, E)$ with n nodes and m edges, assume that there is a consistent ordering among the graph edges represented by an edge labelling function $L : E \to \mathbb{N} \cap [1, m]$, which assigns unique labels to the edges. We will prove that MEGA-GNN can compute unique node IDs under these assumptions.

Egressy et al. (2024) showed that a GNN can mimic a Breadth-First Search (BFS) algorithm to compute unique node IDs given pre-computed port IDs for the edges. We follow the same BFS-based approach and derive unique node ids without relying on pre-computed port IDs. Instead of the pre-computed port ids, we use the unique edge labels provided by $L(e)$ to guide the node ID assignment process. As in Egressy et al. (2024), we use the Universal Approximation Theorem Hornik et al. (1989) for MLPs, to avoid explicit construction of the MEGA-GNN layers. We also assume that the MEGA-GNN aggregates the multi-edges by computing their minimum, which is followed by a node-level aggregation, where an MLP is applied element-wise to the incoming messages, followed by another minimum computation.

Following the approach of Egressy et al. (2024), the node ID assignment algorithm starts from a root node (also called the ego node) and assigns IDs to all the other nodes connected to it via message passing . We are not going to reiterate the setup of the entire proof and focus on the differences. Namely, instead of pre-computing port-IDs, we use an edge labeling function, which assumes a consistent ordering among the edges (e.g., a lexicographic ordering) based on the original edge features. In addition, the multi-edges are aggregated by selecting the edge with the minimum label.

Our MEGA-GNN model, which mimics Algorithm 1, assigns ids to each node connected to the root node. What remains to be shown is that those assigned node IDs are unique. First, note that nodes at different distances from the root cannot end up with the same node ID. A node at distance k will receive its first proposal in round k and, therefore, it will have an ID with exactly $k + 1$ digits.

Furthermore, an inductive argument shows that active nodes (nodes at the same distance) cannot have the same node IDs. Certainly, this is true at the start when $X = \{r\}$. Now assuming all active nodes from the previous round $(k-1)$ had distinct node IDs, then the only way two active nodes (in round k) can have the same ID is if they accept a proposal from the same neighboring node. This is because, based on the induction hypothesis, proposals from different nodes will already differ in their first $k-1$ digits. If two active nodes accepted a proposal from the same node, then they would have received different edge labels —a consistent ordering among the edges enables assignment of distinct edge labels. In addition, because m is added to all incoming labels, incoming labels cannot be the same as the outgoing labels. Therefore the active nodes always accept unique proposals.

B Implementation Details

B.1 Hyperparameters

For each base GNN model and dataset, we utilized a distinct set of hyperparameters, as detailed in Table 4. The MEGA-GenAgg and Multi-GenAgg models employed the aggregation function proposed by Kortvelesy et al. (2023). In all experiments involving GenAgg, we adopted the default layer sizes of $(1, 2, 2, 4)$, and both the a and b parameters were made learnable, allowing the model to tailor the aggregation function to the specific downstream task. Additionally, for the GenAgg experiments, we applied the hyperparameters configured for GIN-based models as shown in Table 4.

For the AML dataset, the model was operated on neighborhoods constructed around the seed edges, while for the ETH dataset, the neighborhoods were selected around the seed nodes. In both datasets, we sampled 2-hop neighborhoods, selecting 100 neighbors per hop.

Table 4: Hyperparameter settings for AML and ETH

	GIN		**PNA**	
	AML	**ETH**	**AML**	**ETH**
lr	0.003	0.006	0.0008	0.0008
h	64	32	20	20
bs	8192	4096	8192	4096
do	0.1	0.1	0.28	0.1
w_ce1, w_ce1	1, 6.27	1, 6.27	1, 7	1, 3

B.2 Multi-Edge Aggregation

In this section, we provide a detailed explanation of the GIN, PNA, and GenAgg base multi-edge aggregations in our study. The following aggregations are used to aggregate parallel edges between node j to node i.

$$\mathbf{h}_{ji}^{(l)} = \text{MLP}\left(\text{SUM}\{\!\{\mathbf{e}_{ji}^{(l-1)} \mid e_{ji} \in N_{in}(v_{ji}^{art})\}\!\}\right). \tag{19}$$

The formulation in Equation 19 employs a sum-based aggregation over the multiset of incoming edges, then applies an MLP. While this approach is similar to DeepSet (Zaheer et al. (2017)), we have opted to use the term GIN-based multi-edge aggregation for consistency.

In the PNA aggregation we combined ['mean', 'min', 'max', 'std'] statistics in combination with scalars ['identity', 'amplification', 'attenuation'] around the neighborhood of each artificial node, $N_{in}(v_{ij}^{art}) = \{e_{ij} | (i,j) \in \mathcal{E}\}$.

$$\mathbf{h}_{ji}^{(l)} = \text{MLP}\left(\text{PNA}\{\!\{\mathbf{e}_{ji}^{(l-1)} \mid e_{ji} \in N_{in}(v_{ji}^{art})\}\!\}\right). \tag{20}$$

GenAgg is a learnable permutation-invariant aggregator which is provably capable of representing all "standard" aggregators (see Kortvelesy et al. (2023) for details.) Different from DeepSet (Zaheer et al. (2017)), GenAgg is a scalar-valued functions applied element-wise rather than fully connected functions over the feature dimension.

$$\mathbf{h}_{ji}^{(l)} = \text{GenAgg}\left(\text{SUM}\{\!\{\mathbf{e}_{ji}^{(l-1)} \mid e_{ji} \in N_{in}(v_{ji}^{art})\}\!\}\right). \tag{21}$$

C Additional Performance Metrics for AML Results

In this section, we present additional results utilizing various metrics for the AML edge classification task on both small and medium datasets. These results are provided for future reference and illustrate the performance of our proposed method across key metrics, including F1, Precision, Recall, and the area under the precision-recall curve (PR-AUC).

Table 5: AML edge classification task results for Small datasets.

	Small_LI				Small_HI			
Method	**F1**	**Precision**	**Recall**	**PR-AUC**	**F1**	**Precision**	**Recall**	**PR-AUC**
MEGA-GIN	43,66 ± 0.54	63.95 ± 4.29	33.25 ± 0.87	34.94 ± 2.76	70.83 ± 2.18	70.11 ± 4.23	**71.74 ± 1.64**	72.69 ± 0.83
MEGA-PNA	45.07 ± 2.26	62.05 ± 9.97	**35.79 ± 0.60**	**35.07 ± 4.76**	74.01 ± 1.55	76.90 ± 4.05	71.48 ± 1.32	73.91 ± 1.05
MEGA-GenAgg	**46.29 ± 0.41**	**66.16 ± 1.22**	35.62 ± 0.50	19.69 ± 5.80	**74.88 ± 0.38**	**78.28 ± 2.46**	71.14 ± 1.72	**74.39 ± 1.93**

Table 6: AML edge classification task results for Medium datasets.

	Medium_LI				Medium_HI			
Method	**F1**	**PR-AUC**	**Precision**	**Recall**	**F1**	**PR-AUC**	**Precision**	**Recall**
MEGA-GIN	39.03 ± 1.88	67.32 ± 11.24	28.18 ± 3.52	12.06 ± 14.93	68.83 ± 1.66	70.97 ± 2.82	66.87 ± 0.90	68.82 ± 3.06
MEGA-PNA	**49.40 ± 0.54**	**75.74 ± 2.75**	**36.69 ± 0.72**	27.85 ± 19.38	**78.26 ± 0.11**	84.62 ± 0.62	**73.07 ± 0.39**	**75.51 ± 3.58**
MEGA-GenAgg	44.89 ± 0.06	72.09 ± 0.68	32.60 ± 0.18	**36.50 ± 0.40**	76.69 ± 0.30	**85.57 ± 1.29**	69.53 ± 1.40	73.37 ± 4.19