ADAMM: Anomaly Detection of Attributed Multi-graphs with Metadata: A Unified Neural Network Approach

Konstantinos Sotiropoulos[*]
Heinz College
Carnegie Mellon University
ksotirop@andrew.cmu.edu

Lingxiao Zhao[*]
Heinz College
Carnegie Mellon University
lingxia1@andrew.cmu.edu

Pierre Jinghong Liang
Tepper School of Business
Carnegie Mellon University
liangj@andrew.cmu.edu

Leman Akoglu
Heinz College
Carnegie Mellon University
lakoglu@andrew.cmu.edu

***Abstract*—Given a complex graph database of node- and edge-attributed multi-graphs as well as associated metadata for each graph, how can we spot the anomalous instances? Many real-world problems can be cast as graph inference tasks where the graph representation could capture complex relational phenomena (e.g., transactions among financial accounts in a journal entry), along with metadata reflecting tabular features (e.g. approver, effective date, etc.). While numerous anomaly detectors based on Graph Neural Networks (GNNs) have been proposed, none are capable of directly handling directed graphs with multi-edges and self-loops. Furthermore, the simultaneous handling of relational and tabular features remains an unexplored area. In this work we propose ADAMM, a novel graph neural network model that handles directed multi-graphs, providing a unified end-to-end architecture that fuses metadata and graph-level representation learning through an unsupervised anomaly detection objective. Experiments on datasets from two different domains, namely, general-ledger journal entries from different firms (accounting) as well as human GPS trajectories from thousands of individuals (urban mobility), validate ADAMM's generality and detection effectiveness of expert-guided and ground-truth anomalies. Notably, ADAMM outperforms existing baselines that handle the two data modalities (graph and metadata) separately with post hoc synthesis efforts.**

***Index Terms*—anomaly detection, complex graphs, graph neural networks, multi-edges, node and edge attributes, metadata**

I. Introduction

Anomaly detection finds numerous practical applications in finance, manufacturing, monitoring, etc. as anomalies are typically indicators of faults, inefficiencies, malicious behavior, etc. in various real-world systems. One of the key challenges in real world settings is the complexity of the data, which exhibit multiple different modalities and heterogeneity—requiring new data representations and novel modeling designs.

This work is motivated by anomaly detection problems in two different real-world domains. The first is from business management and particularly accounting/auditing, where the goal is to identify abnormalities (errors or fraud) among annual general-ledger journal entries from a given firm. Each entry consists of a series of line items of credit or debit transactions of various amounts between accounts with the total debited dollar amount equal to the total credited amount, following the double-entry bookkeeping rules. Accordingly, debits and credits within an entry create directed and weighted links between accounts, and multiple transactions may occur between the same pair of accounts or even within a single general-ledger account. Besides the relational information, each journal entry is also associated with meta-features, such as the approver, entry and effective dates, etc. This poses a multi-modal (relational and tabular) data problem setting. A second example arises from communication networks, where the problem is detecting significant events within a company based on e-mails exchanged between different entities.

[*]Equal contribution. Research supported by the Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number 140D0423C0033 and the PwC Digital Transformation and Innovation Center at Carnegie Mellon University Intelligence. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, the U.S. Government, or the other funding parties.

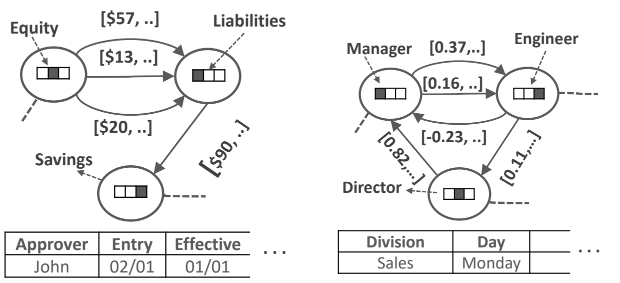

Fig. 1. Modeling complex data. (left) E.g. from **accounting**: a journal entry's attributed *multi-graph* (multiple transactions between two accounts), with edge directions (credit/debit), edge features (e.g. $ amount), and node features (account type; e.g. equity, savings, etc.) plus aux. *meta-features* (approver, entry date, etc.); (right) E.g. from **communication networks**: a daily activity *multi-graph* (multiple e-mails between two company employees), with edge directions (to/from), edge features (e.g. text embedding) and node features (role in the company.) plus aux. *meta-features* (division, day, etc.).

Our goal is to design a novel solution that not only unifies the data modalities under a single, flexible model capable of managing complex relations based on directed multi-graphs, but also offers broad applicability across the domains mentioned earlier and potentially beyond. To this end, we represent the relational information with a node- and edge-attributed directed multi-graph, and the auxiliary or metadata as tabular meta-features. (See Figure 1.) Our proposed solution, ADAMM (for <u>A</u>nomaly <u>D</u>etection of <u>A</u>ttributed <u>M</u>ulti-graphs with <u>M</u>etadata), is a unified neural network framework that learns an expressive graph-level representation for directed and attributed multi-graphs and then fuses it with the meta-features within a shared embedding space before feeding the joint embedding to an unsupervised anomaly detection objective. Notably, our objective is crafted to handle data heterogeneity; where for example, the journal entries may form multiple clusters (e.g. purchases vs. interest gains), and individual behaviors can reflect socio-demographic groups (e.g. single vs. married-with-children). Specifically, we replace the classic SVDD objective that aims to learn embeddings tightly centered around a single centroid [1], and instead employ an unsupervised loss to accommodate multiple centroids.

The literature is abound with anomaly detection techniques [2]–[7], where a vast body focuses on uni-modal data. Numerous prior work address outliers in tabular data [2], [3], possible due to its wide presence in industry and its efficient storage in databases. However, molding real world anomaly detection problems to tabular outlier detection requires "flattening" data from all modalities into manually-extracted features via laborious and often costly domain-expertise [8].

On the other hand, graph anomaly detection has been studied mainly on a single graph for detecting node/edge-level anomalies, with much less emphasis on graph-level anomalies [6], [7]. Few existing traditional approaches to attributed multi-graph anomalies [9], [10] that are not neural network based are not learnable, restricted to handling single-value edge features, not scalable for larger graphs, and do not take auxiliary meta-data into account. Similarly, the more recent neural network based models [11]–[14] are not designed to accommodate directed multi-graphs or graphs with metadata as in this work. (See related work in V for details.) Finally, we argue that a straightforward two-stage approach is naïve and nontrivial; the reasons are first, treating data modalities/sources separately misses the opportunity to capture inter-dependencies and second, the problem of how to combine multiple anomaly rankings/scores open many possibilities without a principled way to choose in the absence of any labels.

We summarize our main contributions as follows.

- **Anomaly Detection in Real-World Settings with Complex Data:** We formulate anomaly detection under data complexity/variety, exhibiting relational as well as auxiliary information, in an elegant framework that can jointly handle complex graphs with node/edge attributes, edge multiplicities, directions and self-loops, meta-features, as well as data heterogeneity. The formulation is driven by anomaly detection problems from two different real-world domains, namely accounting and human mobility, yet is general to apply to possibly other domains.
- **A Unified Detection Model:** We introduce ADAMM, a novel neural network architecture that can digest the aforementioned multi-modal data toward anomaly detection in a unified fashion. It tackles edge multiplicities through set representation learning, employs expressive graph-level embedding that is fused with meta-features in a learned shared embedding space, and finally, optimizes an unsupervised anomaly loss that can accommodate heterogeneous data with multiple latent underlying clusters.
- **Generality and Applications:** ADAMM offers a general framework, where the architecture can be extended to several other domains with data variety, using the idea of learning joint/shared-space embeddings and end-to-end anomaly loss optimization. Besides addressing the data variety challenge of big data, our ADAMM also targets business and societal value, as it is applied to two high-stakes domains; accounting (finance) and human mobility (urban). Through extensive experiments, we show that two-stage solutions are blind-sided and that ADAMM outperforms those as well as other existing baselines significantly on accounting data from three different firms, as well as human GPS trajectory simulations.

Reproducibility. To foster future work on anomaly detection on complex multi-graphs with metadata as well as for practical applications, we open-source the code for ADAMM at https://github.com/konsotirop/ADAMM.

II. Preliminaries

We consider anomaly detection on a large database $\mathcal{G} = \{(G_1, M_1), \ldots, (G_n, M_n)\}$ of n pairs of directed, node/edge attributed, multi-graphs (multiple edges may exist between two endpoints), and their associated metadata-level features.

Definition 1: (Directed, attributed, multi-graph). A graph $G_i = (V_i, E_i, \tau)$ is a directed, attributed, multi-graph, endowed with a function $\tau : V_i \mapsto \mathbb{R}^d$ that assigns a real-valued feature vector to every node in G_i. Moreover, E_i is a multi-set, where an element $e_t = (u, v, \mathbf{f}_t)$ is a directed edge between nodes u and v associated with an edge-feature vector $\mathbf{f}_t \in \mathbb{R}^k$.

Definition 2: (Metadata). Each graph G_i is associated with a vector $\mathbf{Z}_{\mathbf{M_i}} \in \mathbb{R}^{d_M}$ reflecting tabular features.

Usually, we operate on sets (or multi-sets) of variable lengths, where there is no specific order of the elements. In such cases, we need functions that are permutation invariant.

Definition 3: (Set-function). A function f acting on sets is called a **set function** if it is permutation invariant to the order of objects in the set. That is, for any permutation π : $f(\{x_1, \ldots, x_n\}) = f(\{x_{\pi(1)}, \ldots, x_{\pi(n)}\}$.

Neural network architectures, like DeepSet [15], can implement arbitrary set functions, while the work of Xu et al. [16] extends such functions for multi-sets.

Graph Neural Network (GNN) model: We use the provably expressive GIN model of [16], where the embedding of a node v is updated during the l^{th} layer/iteration using the following aggregation function:

$$\mathbf{x}_v^{(l)} = MLP^{(l)}((1+\epsilon) \cdot \mathbf{x}_v^{(l-1)} + \sum_{u \in \mathcal{N}(v)} \text{ReLU}(\mathbf{x}_u + \mathbf{f}_{vu}); \boldsymbol{\theta}_l) \tag{1}$$

where MLP is a multi-layer perceptron, ϵ a learnable parameter, $\mathcal{N}(v)$ the neighborhood of node v, $\mathbf{f}_{uv}$ is the feature vector of edge (u, v) and $\boldsymbol{\theta}_l$ a vector of trainable parameters.

To obtain a graph-level representation $\mathbf{Z}_G$ for the whole graph G we can use a permutation-invariant function READOUT that aggregates node embeddings after the final layer/iteration L, i.e.,

$$\mathbf{Z}_G = READOUT(\{\mathbf{x}_v^{(L)} | v \in V\}) \tag{2}$$

Our problem can be defined (informally) as follows:

Problem 1: (Anomaly Detection of Attributed Multi-graphs with Metadata (ADAMM).) Given a database $\mathcal{G} = \{(G_i, M_i)\}_{i=1}^n$ of n node- and edge-attributed multi-graphs and their associated metadata; the goal is to identify the abnormal (graph, metadata) pairs that differ significantly from the majority in the database.

III. ADAMM for Multi-modal Anomaly Detection of (Multi-)Graphs with Metadata

A. Data Representation

The input to ADAMM is a database comprising of pairs of graphs and their associated metadata vectors. In what follows, we describe the capabilities of it in representing complex graph data, the fusion of them with metadata vectors, as well as two concrete examples from the accounting and human mobility domains where this unified representation can be used to model real-world scenarios.

Graph representation. ADAMM is designed to handle complex graph data of virtually any type. Specific design choices allow the model to be able to represent:

1) *Node attributes (or Node labels):* Nodes can have attributes that are updated after each graph convolution step. When nodes do not have attribute vectors but categorical labels, we can learn representations for these node labels using an embedding layer.
2) *Edge features:* Edges can have features containing important information about the link between two nodes in the graph. Thus, node representations are updated taking into account not only the embeddings of the neighboring nodes, but also those of incident edges (see also Eq. (1)).
3) *Edge direction:* In various domains as with transactions (accounting) and trips (mobility), edge direction is semantically important. Thus, we enhance edge features with an encoding of the direction of the edge. Specifically, for each multi-edge $(u, v, \mathbf{f}_t)$, between nodes u and v, we encode it using label "1" if the edge is present in the graph, i.e. $(u, v, \mathbf{f}_t, \text{“1”})$, and in the meanwhile, augment another reversed edge $(v, u, \mathbf{f}_t, \text{“2”})$ with label "2" into the graph. Also, we reserve label "0" for self-loop edges, i.e. $(u, u, \mathbf{f}_t)$ becomes $(u, u, \mathbf{f}_t, \text{“0”})$. These edge direction labels are then applied as input to an embedding layer that produces the edge direction representation vector $\mathbf{d}_t$. The final representation vector $\mathbf{f}'_t$ for the edge e_t is then obtained as the sum of the edge features vector $\mathbf{f}_t$ and of the edge direction vector, that is, $\mathbf{f}'_t = \mathbf{f}_t + \mathbf{d}_t$.
4) *Multiple edges:* Currently, GNNs are not able to handle multi-edges. Instead, they assume there exists a unique edge between two nodes, as in Eq. (1) for GIN convolution. Multi-edges, however, model the multiple interactions that can occur between two nodes in a network and each has its own feature vector. ADAMM is designed to handle multi-edges by learning a single edge representation from the multi-set of edges. More precisely, we treat the edge features of the multi-edges $F = \{\mathbf{f}'_1, \ldots, \mathbf{f}'_T\}$ as a multi-set and we use a permutation-invariant multi-set function $f : F \mapsto \mathbb{R}^{d_e}$ to learn an edge-level d_e-dimensional representation vector.

The versatility of ADAMM in handling complex graph-data allows it to be used in a wide variety of domains. We present two exemplar ones as follows.

(i) Bookkeeping Graphs [17]: Each graph is a representation of a *journal entry*: a detailed transaction record. Every account present in the entry is associated with a node, with its label being the account type (e.g. equity, revenue, etc.). A directed edge represents monetary flow from a credited account to a debited account and the feature of an edge is the monetary value associated with this transaction. Directed multi-edges capture multiple credit/debit flows that can take place between two accounts.
(ii) Human Mobility (or Activity) Graphs [18]: These represent the mobility or activity behavior of an agent within a time-frame (e.g., a day of the week). Nodes represent visited locations, while node labels represent the Points Of Interest (POI) type in that location (school, restaurant, etc.). Directed multi-edges stand for the trips between those locations and their features capture information about the trip (duration, distance, etc.).

Metadata representation. ADAMM is able to fuse the graph-level representation with associated metadata vectors that contain auxiliary information. We give examples of such information in the two aforementioned domains below.

(i) Metadata for Bookkeeping Graphs: The metadata vector contains information regarding the ID of the user that created the specific journal entry, the approver of this entry, total credit amount, a binary indicator of whether it is a reversal, the date transactions took effect, or the date transactions were recorded in the journal, etc.
(ii) Metadata for Human Mobility Graphs: For activity graphs the metadata vector could contain information about the day of the week this activity took place (e.g., Tuesday), a vector representation of the agent it describes (or simply a unique ID), or other information that could contain GPS related information, like speed-limit violations, etc.

B. A Unified Neural Network Architecture

ADAMM provides a unified architecture for anomaly detection in a database of graphs and their associated metadata features. Figure 2 presents an overview of our model and the steps it involves. The input is two-pronged: a directed, node/edge attributed multi-graph and its associated metadata vector. Following the set representation learning of the multi-edges, a GNN is employed to learn a graph-level embedding, which is then fused with the metadata vector to obtain the final joint embedding. A parameter estimation network decides on the (soft) membership of the final embeddings to one of K clusters. ADAMM is trained in an end-to-end fashion, i.e. all of its parameters are optimized jointly with respect to a suitable objective function that minimizes the weighted distance of

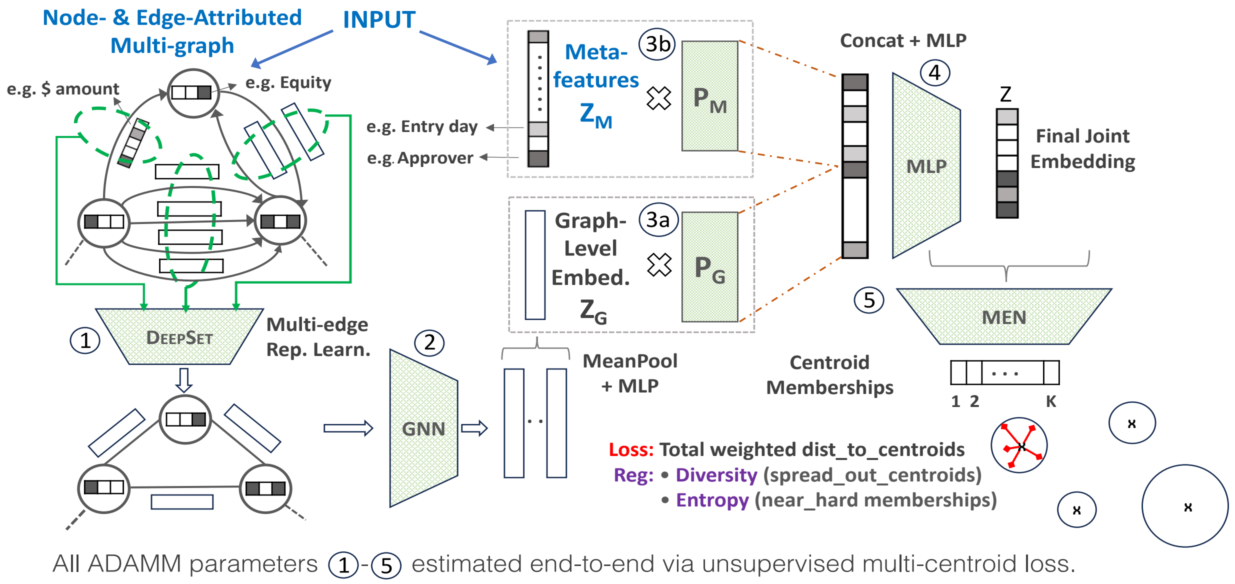
Node- & Edge-Attributed Multi-graph
INPUT
e.g. $ amount
e.g. Equity
e.g. Entry day
e.g. Approver
Meta-features Z_M
3b
P_M
Graph-Level Embed. Z_G
3a
P_G
Concat + MLP
4
MLP
Z
Final Joint Embedding
5
MEN
1
DeepSet
Multi-edge Rep. Learn.
2
GNN
MeanPool + MLP
Centroid Memberships
1 2 K
Loss: Total weighted dist_to_centroids
Reg: • Diversity (spread_out_centroids)
• Entropy (near_hard memberships)
All ADAMM parameters 1-5 estimated end-to-end via unsupervised multi-centroid loss.

Fig. 2. A workflow overview of the ADAMM architecture. Given two-pronged input (in blue), i.e. attributed multi-graph and metadata, ADAMM first processes the former by ① learning a multi-set representation of the multi-edges, and ② flattening the resulting graph via GNN into node representations that are pooled into a graph-level embedding. Then, ③ graph-level embed. and meta-features are projected, ④ followed with a joint embedding learning. Finally, ⑤ the output layer employs an unsupervised regularized multi-centroid anomaly loss where soft assignments are learned via a membership estimation network (MEN). It is notable that ADAMM provides a unified multi-modal framework where parameters of all the modules (in green) are estimated end-to-end.

embeddings to the K centroids of the clusters. Additional regularization terms are introduced to spread-out the centroids as well as to nudge the estimation network toward more confident assignments of cluster memberships. We describe each of these steps in greater detail next.

1) Graph-level Embedding: We learn a graph-level embedding in two steps:

- Multi-edge Representation Learning: As noted, GNNs (including GIN) can **not** readily handle multi-edges. For this reason, we "flatten" all directed multi-edges between two nodes to a single undirected edge and its associated feature vector by learning a permutation-invariant multi-set function based on a DeepSet [15] architecture. This procedure is depicted in Figure 2 (see step ①), where the input attributed multi-graph is transformed using a learnable multi-set function to an attributed graph with single edges among its pairs of nodes.
- Node Embeddings: The transformed graph, where all multi-edges have been replaced by an attributed single edge, is used as input to a GNN model (step ② in Figure 2). In ADAMM we opt to use GIN [16] as a provably expressive GNN. GIN learns node embeddings by performing the graph convolution of Eq. (1), also incorporating the edge features.

To obtain a graph-level embedding we use a READOUT function on the node embeddings (see Eq. 2). We implement this function by performing mean pooling over the node embeddings followed by a multi-layer perceptron (MLP) to learn the final graph-level embedding as

$$\mathbf{Z}_G = MLP\left(\frac{1}{|V|}\sum_{v\in V}\mathbf{x}_v^{(L)};\boldsymbol{\theta}_G\right) \ . \tag{3}$$

2) A Unifying Embedding Space for Graph and Metadata: After having obtained a graph-level embedding, we learn a joint representation of the graph and its metadata in a unifying embedding space. We are motivated by the CLIP-style latents [19] that learn a shared embedding space for images and their associated text captions, analogous to our graph and metadata pairs. More precisely, we first linearly project the graph-level embedding vector $\mathbf{Z}_G$ as well as the metadata vector $\mathbf{Z}_M$ by learning two projection functions (with separate parameters) $P_G(\mathbf{Z}_G;\boldsymbol{\theta}_G) : \mathbb{R}^{d_G} \mapsto \mathbb{R}^{d_P}$ and $P_M(\mathbf{Z}_M;\boldsymbol{\theta}_M) : \mathbb{R}^{d_M} \mapsto \mathbb{R}^{d_P}$ to obtain two new vectors $\mathbf{Z}'_G$ and $\mathbf{Z}'_M$ of the same length d_P as they share the same space. We normalize both vectors to have unit l_2 norm and then concatenate them. Finally, we employ an MLP to obtain the final joint embedding, denoted $\mathbf{Z} \in \mathbb{R}^d$ (see steps ③ - ④ in Figure 2) as

$$\mathbf{Z} = MLP\left(\text{CONCAT}(\mathbf{Z}'_G, \mathbf{Z}'_M);\boldsymbol{\theta}_J\right) \ . \tag{4}$$

C. Anomaly Detection Loss

Objective functions used in anomaly detection, like One-Class DeepSVDD [20], make the somewhat strong assumption that all of the normal instances come from the same distribution. Hence, their objective is to use a deep neural network to map all normal instances as close to the center of a *single* hypersphere, with anomalous instances identified as those mapped farther from this centroid. However, this objective does not take into account the multiple modalities or heterogeneities that may exist in real world data. For this reason, we introduce a new objective function that accommodates *multiple* clusters of the input samples. ADAMM estimates the (soft) cluster membership of each sample using a membership estimation network and tries to minimize the total average weighted distance from the K centroids, where hyperparameter K is carefully tuned (as discussed later in III-D). Contrary to One-Class DeepSVDD, where the center of the hypersphere is fixed during the training process, the K centroids in our case are inferred from the membership estimation network and the final embedding vectors.

Membership Estimation Network (MEN). The MEN is an MLP with a softmax activation function (step ⑤ of Fig. 2) that gives the membership predictions for the final embedding vectors $\mathbf{Z}$ in Eq. (4). That is,

$$\widehat{\boldsymbol{\gamma}} = \text{softmax}(\text{MLP}(\mathbf{Z}; \boldsymbol{\theta}_{MEN})) \ , \tag{5}$$

where $\widehat{\boldsymbol{\gamma}}$ is a K-dimensional vector depicting the soft membership probability predictions of $\mathbf{Z}$.

In what follows, and for a batch of N pairs of (graph, metadata) samples, where $\mathbf{Z}_i$ is the embedding of the i^{th} sample, we denote by $\widehat{\mathbf{\Gamma}}$ the $N \times K$ matrix of cluster membership estimations from Eq. (5) and by $\widehat{\gamma}_{ik}$ each entry of this matrix.

Then, the cluster centroids $\widehat{\boldsymbol{c}}_k \in \mathbb{R}^d$ can be calculated using the cluster membership estimations from Eq. (5) and embedding vectors $\mathbf{Z}_i$, for $i = 1, \ldots, N$, by

$$\widehat{\boldsymbol{c}}_k = \frac{\sum_{i=1}^{N} \widehat{\gamma}_{ik} \mathbf{Z}_i}{\sum_{i=1}^{N} \widehat{\gamma}_{ik}} \ . \tag{6}$$

Loss Function and Anomaly Score. Having estimated the embedding vectors and cluster membership estimations, the anomaly score of a sample $T_i = (G_i, M_i)$ is then defined as the weighted sum of the Euclidean distance between the final embedding vector $\mathbf{Z}_i$ and the cluster centroids $\widehat{\boldsymbol{c}}_k$'s, i.e.

$$\text{score}(T_i) = \sum_{k=1}^{K} \widehat{\gamma}_{ik} \| \mathbf{Z}_i - \widehat{\boldsymbol{c}}_k \|^2 \ , \tag{7}$$

which also serves as the *anomaly score* of a sample T_i (the higher, the farther and the more anomalous).

ADAMM is then trained in an end-to-end fashion to optimize the following unsupervised objective.

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \widehat{\gamma}_{ik} \| \mathbf{Z}_i - \widehat{\boldsymbol{c}}_k \|^2 + \lambda_1 \cdot H(\widehat{\mathbf{\Gamma}}) + \lambda_2 \cdot D(\widehat{\mathbf{C}}) \tag{8}$$

where $\boldsymbol{\theta}$ depicts all ADAMM parameters collectively and λ_1, λ_2 are hyperparameters that aim to strike a balance between the distance-to-centroids anomaly loss (first term) and two regularization terms, respectively, Entropy and Diversity, which we describe as follows.

- The first is an entropy regularization that forces the network to be more confident on the cluster to which it estimates an input sample to belong. More specifically, we aim to minimize the average entropy over the rows of the $\widehat{\mathbf{\Gamma}}$ membership estimation matrix as

$$H(\widehat{\mathbf{\Gamma}}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{i=1}^{K} -\widehat{\gamma}_{ik} \log(\widehat{\gamma}_{ik}) \ . \tag{9}$$

- The second is a diversity term that promotes separation between the cluster centroids to avoid the undesired mode-collapse solutions where the network collapses all centroids to the same point. Letting $\widehat{\mathbf{C}}$ depict the $K \times d$ matrix containing the cluster centroids $\widehat{\boldsymbol{c}}_k$'s as its rows;

$$D(\widehat{\mathbf{C}}) = -\log(\det(\text{Cov}(\widehat{\mathbf{C}})) \ , \tag{10}$$

where $\det(\text{Cov}(\widehat{\mathbf{C}}))$ is the determinant of the covariance matrix of $\widehat{\mathbf{C}}$. In effect, the larger the determinant of the covariance matrix, the more the centroids are dispersed, promoting separation between and diversity among the cluster centroids. A similar term has also been used in [21] for selecting diverse features in regression settings.

D. Model Selection

ADAMM, as with other deep neural networks based models, is configured with a set of hyperparameters (HPs), such as the number of layers, weight decay and learning rates, number of training epochs, among others. In addition, our multi-centroid anomaly objective incurs the number of centroids K, and the λ_1 and λ_2 terms from Eq. (8). Each different configuration of these results in a different model, with potentially drastic differences in anomaly detection effectiveness. The challenge is that anomaly detection is an unsupervised task, where we usually lack ground-truth labels of whether a sample is an anomaly. As a result, we do not have a labeled validation set for hyperparameter tuning. For this reason, we devise an unsupervised validation score, *without using any labels*, toward selecting an effective model that performs better in anomaly detection than what we would have obtained by picking at random (in absence of any other guidance).

Given a family of models, $\mathcal{M}$, we opt to choose the model m that minimizes the sum of the weighted distances of the N samples in the training set from the K centroids as our model selection criterion, specifically,

$$\sum_{i=1}^{N} \sum_{k=1}^{K} \widehat{\gamma}_{ik} \| \mathbf{Z}_i - \widehat{\boldsymbol{c}}_k \|^2 \ . \tag{11}$$

This rule favors the model that succeeds into learning a tight representation of the training instances into each of the K clusters by better extracting their shared patterns, which in effect helps reveal the anomalies that deviate from these patterns. In experiments, we compare the effectiveness of our model selection criterion against random picking (i.e. average/expected performance over possible HP configurations).

IV. Experiments

A. Experimental Setup

Datasets. For evaluation we use four datasets from two different domains, each containing a large database of graphs and their associated metadata. Those include annual general-ledger journal entries from three different firms, in collaboration with PwC. The fourth dataset involves simulated human GPS trajectories. Summary of datasets is given in Table I.

- *Accounting Datasets*: Three datasets from accounting consist of all annual journal entries from different firms anonymized as SH, HW, and KD. Each dataset contains

TABLE I
DATASET SUMMARY STATISTICS

Name	Graphs	Nodes	Multi-edges	Node-attr.	Edge-attr.	Meta-feat.
SH	39,011	[1,15]	[1,338]	11	1	11
KD	152,105	[1,91]	[1,774]	10	1	9
HW	90,274	[1,25]	[1,897]	11	1	7
MobiNet	140,000	[1,22]	[1,59]	41	4	9

tens of thousands of bookkeeping graphs [17] capturing itemized transactions between impacted accounts along with dollar amounts, and metadata entries capturing auxiliary journal information including entry and effective date, requester, approver, reversal indicator, and so on.

- *Human Mobility Dataset*: The fourth dataset, referred to as MobiNet, contains the trajectories of $10,000$ simulated agents over a period of two weeks. We use these trajectories to extract daily activity graphs [18] of the places (i.e. POI) visited by each agent and the trips between them, as described in Section III-A. The metadata contains information about individual trips and are transformed to a single vector using a DEEPSET architecture during the end-to-end training of ADAMM.

Baselines. For comparison, we use as baselines existing graph-level anomaly detectors and tabular data outlier detectors. Unlike ADAMM, existing graph-level anomaly detectors can not handle multi-edges. Therefore, we collapse all multi-edges to a single edge and use the average representation of their feature vectors. To the best of our knowledge, there is also no prior work that fuses graphs and metadata and provides a single anomaly score. For this reason, we employ two-stage baselines: First, we create a ranking of the samples with respect to their anomaly score as obtained by a graph-level anomaly detector. Then, we create a second ranking by using a tabular data outlier detector. We combine these two rankings to obtain a single graph&metadata anomaly ranking using two well-established aggregation methods detailed as follows.

(a) *Graph-level Anomaly Detectors*: We first aim to detect graph-level anomalies using the following baselines:
   (1) Weisfeiler-Lehman (WL) graph kernel [22], followed by the OCSVM outlier detector [23] that can admit a kernel matrix as input.
   (2) graph2vec [24], for graph-level embedding, followed by the OCSVM detector.
   (3) DOMINANT [25], a GNN-based node anomaly detector, from which we average the scores to obtain a graph-level anomaly score.

(b) *Tabular Data Outlier Detectors*: We use the tree-ensemble based Isolation Forest algorithm [26] to score outlierness on the meta-features, which is the state-of-the-art tabular data outlier detector [27].

After having obtained a ranking from (a) the graph-level anomaly detectors and (b) the tabular data anomaly detectors, we create a unique ranking for pairs of graphs and metadata, by using: (i) a BFS-style aggregation that first sorts the results of each stage in descending order of their anomaly score, and then selects the next object that has the highest anomaly score by visiting the lists in a BFS fashion [28]; and (ii) the Inverse Rank (IR) aggregation method, in which we score each sample by $\frac{1}{r_a} + \frac{1}{r_b}$, where r_a is the rank by the graph-level anomaly detector and r_b by the tabular data outlier detector.

Overall we construct 6 baselines based on (1)–(3) $\times$ (i)–(ii).

Labeled Anomalies. Our datasets do not come with any ground truth anomalies, therefore, we use the guidance of experts in the fields of accounting and human mobility to simulate anomalies that are typically present in these datasets and of interest in detecting them. We create two types of graph-level anomalies, as well two types of metadata-level anomalies.

1) Graph anomalies involve small perturbations in nodes and edges as follows.

- *Label change* (**GA1**): We change the label of a node to a randomly chosen new label. This injection corresponds to entry-error in an accounting dataset, or an unusual visit to a new POI in human mobility behavior.
- *Path injection* (**GA2**): We delete an edge between nodes u and v and rewire through an intermediary, creating a path u-z-v. This injection mimics money-laundering in finance, where funds are passed through an intermediate account instead of being transferred directly. For human mobility, this corresponds to an unusual stop.

2) Metadata anomalies perturb feature values and reflect different semantics in both domains.

For Accounting Datasets:

- *Unusual back-dating* (**MA1**): We pick a subset of entries with effective date close to the entry date (up to 3 days before), and change the entry date randomly to one of $\{7, 14, 21\}$ days after the effective date. This corresponds to an unusual late entry date. We lack information about entry date in HW, hence our experiments do not use this type of anomaly for this dataset.
- *Combination of unrelated transactions* (**MA2**): We merge two unrelated transactions by creating a new one with a unique Journal ID. We set the metadata entries to a randomly chosen value from the two initial transactions. Note that this injection also modifies the graph structure into the representation of the merged journals.

For Human Mobility Dataset:

- *Unusual start time* (**MA3**): We change the start time of a trip to a very early (or late one). This corresponds to a trip occurring in an unusual time.
- *Unusual trip duration* (**MA4**): We change the duration of a trip to an unusually long one.

3) Potpourri anomalies involve a combination of graph and metadata level anomaly injections, where we pick one graph level anomaly and one metadata level anomaly from above and inject both to a sample.

We inject anomalies on 5% of the samples in each dataset, where we use half of the original dataset for training, and the remaining half with the injected anomalies for testing. Note that the labeled anomalies are used only for evaluation purposes and not during model training or model selection.

Model Selection. Anomaly detection is typically a fully unsupervised task, where we lack ground truth labels of which

TABLE II

Anomaly Detection Results for all methods across all datasets based on AUROC. For baseline methods we run the experiments over a grid of hyperparameters and report the average performance, along with the std. dev. ADAMM employs a model selection criterion and outputs a unique ranking. Last row reports significance test results, where (**) and (***) denote that ADAMM is significantly better than baselines w.r.t. the Wilcoxon Signed Rank Test at $p = 0.05$ and $p = 0.01$, respectively.

Dataset	Anomaly Type	ADAMM	WL+BFS	WL+IR	G2V+BFS	G2V+IR	DOM.+BFS	DOM.+IR
SH	GA1	**0.992**	0.925 ± 0.01	0.922 ± 0.01	0.839 ± 0.09	0.833 ± 0.09	0.824 ± 0.01	0.821 ± 0.01
	GA2	**0.968**	0.827 ± 0.02	0.829 ± 0.02	0.854 ± 0.02	0.854 ± 0.02	0.834 ± 0.01	0.837 ± 0.01
	MA1	**0.846**	0.591 ± 0.02	0.610 ± 0.03	0.586 ± 0.01	0.592 ± 0.01	0.602 ± 0.02	0.613 ± 0.02
	MA2	**0.918**	0.638 ± 0.02	0.642± 0.02	0.614 ± 0.01	0.618 ± 0.01	0.615 ± 0.01	0.618 ± 0.01
	GA1 + MA1	**0.955**	0.899 ± 0.01	0.897 ± 0.02	0.807 ± 0.01	0.800± 0.01	0.811 ± 0.01	0.807 ± 0.02
	GA2 + MA1	**0.977**	0.841 ± 0.03	0.840 ± 0.01	0.871 ± 0.01	0.869±0.02	0.836 ± 0.02	0.838 ± 0.01
KD	GA1	**0.928**	0.885 ± 0.01	0.880 ± 0.01	0.835 ± 0.04	0.828 ± 0.04	0.462 ± 0.01	0.450 ± 0.01
	GA2	**0.939**	0.825 ± 0.03	0.828 ± 0.04	0.820 ± 0.01	0.824 ± 0.01	0.543 ± 0.01	0.528 ± 0.01
	MA1	**0.841**	0.727 ± 0.01	0.716 ± 0.03	0.729 ± 0.01	0.718 ± 0.01	0.610 ± 0.01	0.588 ± 0.01
	MA2	**0.854**	0.738 ± 0.02	0.743 ± 0.02	0.736 ± 0.02	0.741 ± 0.02	0.518 ± 0.01	0.505 ± 0.01
	GA1 + MA1	**0.933**	0.901 ± 0.01	0.895 ± 0.01	0.814 ± 0.07	0.805 ± 0.01	0.458 ± 0.01	0.448 ± 0.01
	GA2 + MA1	**0.916**	0.849 ± 0.03	0.849 ±0.04	0.818 ± 0.01	0.805 ± 0.07	0.537 ± 0.01	0.522 ± 0.01
HW	GA1	**0.973**	0.922 ± 0.01	0.916 ± 0.01	0.922 ±0.03	0.920 ± 0.03	0.713 ± 0.21	0.710 ± 0.21
	GA2	**0.994**	0.895 ± 0.01	0.888 ± 0.01	0.666 ± 0.05	0.660 ± 0.05	0.400 ± 0.07	0.406 ± 0.07
	MA2	**0.967**	0.691 ± 0.02	0.661 ± 0.02	0.706 ± 0.02	0.694 ± 0.01	0.535 ± 0.01	0.527 ± 0.01
MobiNet	GA1	0.526	0.676 ± 0.01	**0.678** ± 0.02	0.466 ± 0.07	0.467 ± 0.05	0.320 ± 0.01	0.323 ± 0.01
	GA2	0.491	0.487 ± 0.02	0.482 ± 0.03	0.499 ± 0.05	**0.505** ± 0.04	0.341± 0.01	0.346 ± 0.01
	MA3	0.441	0.558 ± 0.01	0.563 ± 0.01	0.556 ± 0.02	**0.574** ± 0.03	0.411± 0.01	0.415 ± 0.01
	MA4	0.450	0.492 ± 0.01	0.490 ± 0.01	0.517 ± 0.02	**0.524** ± 0.01	0.349± 0.01	0.353 ± 0.0
	GA1 + MA3	0.563	0.750 ± 0.01	**0.754** ± 0.02	0.451 ± 0.01	0.454 ± 0.02	0.326 ± 0.01	0.329 ± 0.03
	GA1 + MA4	0.678	0.743 ± 0.02	**0.747** ± 0.01	0.457 ± 0.02	0.460 ± 0.01	0.350± 0.01	0.353 ± 0.02
	GA2 + MA3	0.470	0.483 ± 0.01	0.480 ± 0.02	0.505 ± 0.01	**0.510** ± 0.02	0.329 ± 0.04	0.332 ± 0.01
	GA2 + MA4	0.477	0.494 ± 0.02	0.489 ± 0.01	0.520 ± 0.01	**0.526** ± 0.03	0.332 ± 0.01	0.336 ± 0.01
Average AUROC		0.787	0.730	0.728	0.678	0.677	0.524	0.522
Average Rank		2.13	3.08 (**)	2.78 (**)	4.09 (***)	3.74 (***)	6.17 (***)	6 (***)

samples are anomalous. As a result, we do not have a validation set for hyperparameter tuning. For this reason, for each of the baseline methods, we consider a set of hyperparamater configurations, across which we report the average performance. This corresponds to the expected performance of each method if one were to select a configuration at random. For ADAMM we show that our proposed model selection criterion presented in III-D can consistently yield better results than what we would expect when picking hyperparameters at random (in the absence of any other guidance). **Hyperparameter (HP) Configurations:** Detailed HP configurations can be found in the full version of the paper: https://arxiv.org/abs/2311.07355.

B. Detection Results

In evaluating proposed ADAMM, we conducted a series of experiments to answer the following questions:

Q1) **Effectiveness:** How effective is ADAMM in detecting graph- and metadata-level anomalies, as compared to the two-stage baseline approaches?

Q2) **Model Selection:** Can our proposed unsupervised model selection criterion for ADAMM select a model (i.e. hyperparameter configuration) that is better than random picking (i.e. avg. performance across config.s)?

Q3) **Ablation:** How important are key components of ADAMM in the detection results?

A1: To answer the first question, we conduct extensive experiments using all four datasets and graph, metadata as well as potpourri anomalies. The results are presented for each method across all datasets and injection types in Table II based on the Area Under the Receiver Operator Characteristic Curve (AUROC), and in Table III based on the Area Under Precision-Recall Curve (AUPRC). We observe that ADAMM succeeds in detecting both graph-level and metadata-level anomalies effectively. It outperforms the baseline methods in 3 out of 4 datasets and across all injection types, where the baselines do not show consistent performance. ADAMM performs consistently well for all types of anomalies, whether they are graph, metadata-level, or even of mixed type. The only exception is the MobiNet dataset, where the nature of metadata information (multiple vectors for a single graph that have to be aggregated) poses significant challenges[1]

The superior performance of ADAMM over baselines is also validated using the Wilcoxon signed rank test. ADAMM not only has the lowest average rank among the competitors, but is also significantly better at p-value $p = 0.05$.

A2: The problem of model selection is an important one in unsupervised anomaly detection. The lack of labels and of a validation set makes it challenging to choose an effective model. For ADAMM we provide a validation criterion tightly connected to its loss function (recall Eq. (11)). In Figure 3 we see that the model ADAMM chooses based on its unsupervised criterion achieves consistently better performance than random picking that corresponds to the average performance across all configurations. This makes ADAMM not only able to spot

[1]For baseline methods, we score all vectors separately and assign the maximum score as the anomaly score of the sample. This gives better results than taking the average as the latter dilutes the signal among multiple vectors.

TABLE III

Anomaly Detection Results for all methods across all datasets based on AUPRC (Area Under Precision-Recall Curve). For baselines, average performance across hyperparameters along with the std. dev. is reported. ADAMM outputs a unique ranking based on a model selection criterion. Last row reports significance test results, where (**) and (***) denote that ADAMM is significantly better than baselines w.r.t. the Wilcoxon Signed Rank Test at $p = 0.05$ and $p = 0.01$, respectively.

Dataset	Anomaly Type	ADAMM	WL+BFS	WL+IR	G2V+BFS	G2V+IR	DOM.+BFS	DOM.+IR
SH	GA1	**0.939**	0.470 ± 0.01	0.461 ± 0.01	0.270 ± 0.01	0.327 ± 0.01	0.327 ± 0.02	0.320 ± 0.01
	GA2	**0.708**	0.214 ± 0.02	0.217 ± 0.02	0.269 ± 0.04	0.252 ± 0.01	0.252 ± 0.01	0.256 ± 0.01
	MA1	**0.280**	0.125 ± 0.01	0.126 ± 0.01	0.118 ± 0.01	0.125 ± 0.01	0.125 ± 0.01	0.126 ± 0.01
	MA2	**0.599**	0.125 ± 0.01	0.125 ± 0.01	0.120 ± 0.01	0.121 ± 0.01	0.121 ± 0.01	0.122 ± 0.01
	GA1 + MA1	**0.877**	0.458± 0.01	0.450 ± 0.01	0.255 ± 0.09	0.246 ± 0.09	0.080 ± 0.01	0.077 ± 0.01
	GA2 + MA1	**0.836**	0.224 ± 0.02	0.225 ± 0.04	0.286 ± 0.03	0.241 ± 0.01	0.097± 0.01	0.093 ± 0.01
KD	GA1	**0.553**	0.363 ± 0.02	0.347 ± 0.01	0.255 ± 0.09	0.246 ± 0.09	0.08 ± 0.01	0.080± 0.01
	GA2	**0.430**	0.284 ± 0.04	0.283 ± 0.04	0.234± 0.014	0.240 ± 0.01	0.102 ± 0.01	0.098 ± 0.01
	MA1	0.141	0.142 ± 0.01	0.136 ± 0.01	**0.148** ± 0.002	0.144 ± 0.01	0.106 ± 0.01	0.101 ± 0.01
	MA2	**0.342**	0.117 ± 0.001	0.104 ± 0.01	0.162 ± 0.01	0.164 ± 0.01	0.089 ± 0.01	0.086 ± 0.01
	GA1 + MA1	**0.677**	0.443 ± 0.01	0.424 ± 0.01	0.256 ± 0.12	0.247 ± 0.12	0.080 ± 0.01	0.077 ± 0.01
	GA2 + MA1	**0.351**	0.261 ± 0.04	0.261 ± 0.04	0.237 ± 0.01	0.240 ± 0.01	0.097 ± 0.01	0.094 ± 0.01
HW	GA1	**0.866**	0.446 ± 0.03	0.438 ± 0.03	0.455 ± 0.10	0.451 ± 0.01	0.314 ± 0.21	0.310 ± 0.21
	GA2	**0.941**	0.292 ± 0.01	0.280 ± 0.02	0.146 ± 0.03	0.144 ± 0.03	0.106 ± 0.04	0.105 ± 0.04
	MA2	**0.815**	0.165 ± 0.01	0.151 ± 0.01	0.175 ± 0.01	0.168 ± 0.01	0.114 ± 0.01	0.110 ± 0.01
MobiNet	GA1	0.050	0.100 ± 0.01	**0.102** ± 0.01	0.046 ± 0.01	0.046 ± 0.01	0.066± 0.01	0.065 ± 0.01
	GA2	0.043	0.047 ± 0.01	0.047 ± 0.03	0.054 ± 0.01	0.054 ± 0.01	**0.069** ± 0.01	**0.069** ± 0.01
	MA3	0.040	0.054 ± 0.01	0.055 ± 0.01	0.055 ± 0.01	0.057 ± 0.01	**0.076** ± 0.01	**0.076**± 0.01
	MA4	0.040	0.047 ± 0.01	0.047 ± 0.01	0.051 ± 0.01	0.051 ± 0.01	0.069 ± 0.01	**0.067** ± 0.01
	GA1 + MA3	0.052	0.165 ± 0.01	**0.166** ± 0.01	0.043 ± 0.01	0.044 ± 0.01	0.067 ± 0.01	0.067 ± 0.01
	GA1 + MA4	0.074	0.157 ± 0.01	**0.158** ± 0.01	0.046 ± 0.01	0.046 ± 0.01	0.069 ± 0.01	0.068 ± 0.01
	GA2 + MA3	0.052	0.044± 0.01	0.044 ± 0.01	0.054 ± 0.01	0.053 ± 0.01	**0.066** ± 0.01	0.065 ± 0.01
	GA2 + MA4	0.041	0.046 ± 0.01	0.046 ± 0.01	0.055 ± 0.01	0.056 ± 0.01	**0.066** ± 0.01	0.066 ± 0.01
Average AUPRC		0.443	0.208	0.204	0.165	0.163	0.131	0.130
Average Rank		2.13	3.91 (***)	4.26 (***)	4.73 (***)	3.95 (***)	4.52 (***)	4.48 (***)

graph and metadata level anomalies, but also robust against different hyperparameter choices.

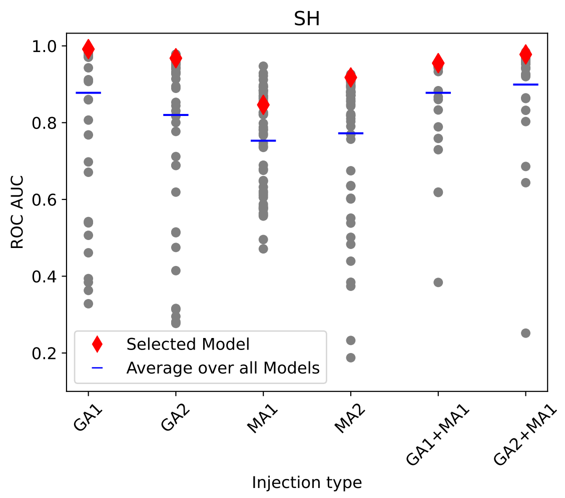

Fig. 3. Model selection for ADAMM over all models with different hyperparameter configurations. The model selected consistently performs better than random picking, i.e. average/expected performance over all models.

A3: ADAMM exhibits three key building blocks; multi-edge representation learning, graph-metadata fusion, and a suitable anomaly detection loss. Accordingly, we perform an *ablation study* and design threevariants of ADAMM, each excluding the respective design component to demonstrate its added benefit.

V1. ADAMM **without Metadata Fusion**: Here we remove the metadata fusion component and instead we input only the graph-level embeddings $\mathbf{Z}_G$ to the membership estimation network. Our goal is to explore if the metadata component interferes with the graph-level component by having a negative influence on graph-level anomaly detection when only such anomalies are present.

V2. ADAMM **without DeepSet**: In this version, we remove the DeepSet component that aims to learn a single representation of the attributed multi-edges. Instead, we simply average the attributes over each multi-edge.

V3. ADAMM **with One-Class DeepSVDD loss**: Finally, we compare ADAMM and its loss function in Eq. (8) with a varint where we replace it with the loss introduced by the One-Class DeepSVDD [20] method which, as we described in III-C, maps all normal instances to a *single* hypersphere centered around a *fixed* centroid.

Results of the ablation study are given in Table IV for SH (as a representative of the transaction datasets) as well as the MobiNet dataset. We see that ADAMM outperforms all of its variants on the SH dataset, demonstrating the importance of the various components in anomaly detection. The improvement is particularly noticeable for the MA2 type anomalies (merge of unrelated transactions), which is of mixed type (both metadata and graph). We note that ADAMM without metadata also performs well for graph-only anomalies of type GA1 and GA2. In fact, excluding metadata lifts the interference on MobiNet, leading to better detection.

TABLE IV
ABLATION STUDY RESULTS - COMPARING ADAMM AGAINST ITS THREE VARIANTS: (I) ADAMM WITHOUT METADATA FUSION, (II) ADAMM WITHOUT DEEPSET & (III) ADAMM WITH ONE-CLASS DEEPSVDD LOSS (OCDL)

Dataset	Anomaly Type	ADAMM		ADAMM **w/o Metadata**		ADAMM **w/o DeepSet**		ADAMM **with OCDL**	
		AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
SH	GA1	**0.992**	**0.938**	0.989	0.920	0.988	0.920	0.986	0.894
	GA2	**0.968**	**0.708**	0.961	0.622	0.965	0.758	0.930	0.489
	MA2	**0.918**	**0.598**	0.898	0.580	0.816	0.427	0.868	0.467
MobiNet	GA1	0.526	0.049	**0.779**	**0.179**	0.588	0.060	0.577	0.061
	GA2	0.491	0.043	**0.678**	**0.079**	0.475	0.042	0.472	0.045

C. Case Studies

Through quantitative experiments in IV-B we showed that ADAMM can successfully spot expert-guided injected anomalies. To further validate the effectiveness of our method, we consider the original SH dataset that contains **no** injected anomalies. That is, we use the whole dataset of $39,011$ graphs with metadata for training and inspect their anomaly scores obtained by Eq. (7). As presented in Figure 4 (left), we see that ADAMM is able to highlight a small fraction of the samples as standing out from the majority.

As ADAMM is unique in handling complex directed graphs with attributes and multi-edges, we take a closer look at two example graphs as shown in Figure 4 (right). Self-loops are the common feature of these graphs: the first has one self-loop with a large dollar amount ($1.5M), while the second contains 38 self-loops in one graph. From an accounting domain perspective, self-loops represent transactions recorded by moving dollars *within* the same general ledger (GL) account. From a bookkeeping standpoint, these within-GL movements indicate the presence of misidentification of the correct sub-ledger account in the recording of a prior transaction. The self-loop in the current journal entry is then designed to correct such a misidentification at a later date. In the first graph A, a total of $\$1.5$M was recorded in a sub-ledger incorrectly, necessitating the current self-loop transaction to correct the cumulative mistakes made earlier. The second graph B (with 38 self-loops) is even more pronounced in terms of the number of corrections involved as well as the presence of errors beyond the simple one illustrated in the first example.

Based on this transaction-level bookkeeping analysis, these two spotted transactions are indeed unusual and worthy of the auditor's attention to examine further. ADAMM's ability to spot these anomalies involving edge-attributes and multi-edges can be of assistance to the accounting/auditing practitioners.

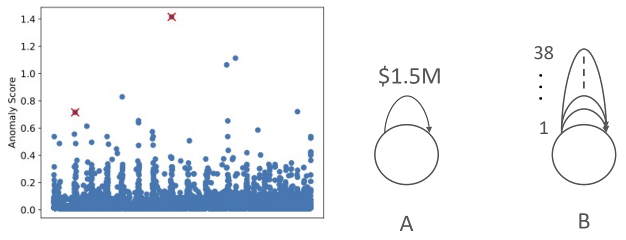

Fig. 4. Analyzing detected accounting anomalies. (Left) Anomaly scores (vs journal ID) of all $39,011$ entries in the SH dataset. (Right) Two example graphs, A and B, that are identified as anomalous by ADAMM in SH.

V. RELATED WORK

Anomaly detection (AD) has an extensive literature mainly considering outliers in tabular or vector data [29], [30], including the recently emerging deep neural network based approaches (see surveys [3], [31], [32]). However, these do not apply to AD for graphs with relational structure.

The majority of work on graph anomaly detection [6], including the recent graph NN (GNN) based techniques [33]–[36] focus on node, edge, or subgraph anomalies within a *single* graph, rather than graph-level anomalies in a *database*.

Different from these earlier work, we consider graph-level AD among a set of graphs within a database. There exist traditional encoding or compression-based techniques [9], [10], [37], [38] that aim to identify frequent structural motifs or graphlets that compress a graph database efficiently, and then flag those graphs with long encoding length as anomalous. Most recent work have shifted attention to employing deep learning and GNNs toward graph-based AD (for a recent survey, see [7]). The idea is to flatten each graph by leveraging their representation or embedding learning capability, and train the GNN parameters end-to-end through various AD objectives such as one-class [11], [12], mutual information-based [14], distributional distance [13], contrastive [39] as well as distillation losses [40]. While these have made progress in graph-level anomaly detection, they do not handle *multi*-graphs, nor are they designed to admit multi-modal input such as graphs with meta-features as in our case.

Examples of prior work on multi-graphs address summarization [41], partitioning [42]–[44], as well as anomaly detection [10], [45], however without considering additional meta-features. An earlier work on node-level (fake reviewers) AD in a single (reviewer-to-product) graph has attempted to bridge node-level meta-features with graph data—by first using the meta-features to estimate node outlierness scores and then propagating those over the graph to capture guilt-by-association [46]. Their method, however, does not generalize to graph database anomalies with graph-level meta-features.

In summary our proposed ADAMM, to our knowledge, is the first method for graph-level anomaly detection for directed node/edge-attributed multi-graphs with meta-features. It leverages (i) end-to-end *multi*-graph embedding, (ii) *joint* multi-modal representation learning and (iii) a *multi-centroid* AD loss to effectively capture complexities in the input data.

VI. Conclusion

In this work we addressed an anomaly detection problem that relates to one of the key challenges of big data mining, that is, data complexity. In particular, we considered a graph database consisting of node- and edge-attributed directed multi-graphs with associated metadata, and proposed a new multi-modal anomaly detection approach called ADAMM. In a unified neural network framework, ADAMM first captures a set representation of the multi-edges, learns a graph-level embedding, fuses the graph and metadata in a joint embedding space, on which it finally employs an unsupervised anomaly loss based on a multi-centered data distribution. To our knowledge, ADAMM is the first unified method that can tackle anomaly detection on complex data of this nature in an end-to-end fashion. Through extensive experiments on datasets from two real-world domains, namely accounting and urban mobility, we showed that ADAMM significantly outperforms all two-stage baselines that handle graphs and metadata separately. We open-source ADAMM's code for future research as well as practical use on possibly other real-world domains.

References

[1] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, pp. 45–66, 2004.

[2] C. C. Aggarwal and C. C. Aggarwal, *An introduction to outlier analysis*. Springer, 2017.

[3] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[4] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2013.

[5] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021.

[6] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data mining and knowledge discovery*, vol. 29, pp. 626–688, 2015.

[7] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Trans. on Knowledge and Data Engineering*, 2021.

[8] L. Akoglu, "Anomaly mining: Past, present and future," in *International Conference on Information & Knowledge Management*, 2021, pp. 1–2.

[9] M.-C. Lee, H. T. Nguyen, D. Berberidis, V. S. Tseng, and L. Akoglu, "Gawd: graph anomaly detection in weighted directed graph databases," in *IEEE/ACM ASONAM*, 2021, pp. 143–150.

[10] H. T. Nguyen, P. J. Liang, and L. Akoglu, "Detecting anomalous graphs in labeled multi-graph databases," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, no. 2, pp. 1–25, 2023.

[11] L. Zhao and L. Akoglu, "On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights," *Big Data*, vol. 11, no. 3, pp. 151–180, 2023.

[12] C. Qiu, M. Kloft, S. Mandt, and M. Rudolph, "Raising the bar in graph-level anomaly detection," *arXiv preprint arXiv:2205.13845*, 2022.

[13] L. Zhao, S. Sawlani, A. Srinivasan, and L. Akoglu, "Graph anomaly detection with unsupervised GNNs," *Preprint arXiv:2210.09535*, 2022.

[14] G. Zhang, Z. Yang, J. Wu, J. Yang, S. Xue, H. Peng, J. Su, C. Zhou, Q. Z. Sheng, L. Akoglu *et al.*, "Dual-discriminative graph neural network for imbalanced graph-level anomaly detection," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 144–24 157, 2022.

[15] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *NeurIPS*, 2017.

[16] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019.

[17] P. J. Liang, "Bookkeeping graphs: Computational theory and applications," *Foundations and Trends® in Accounting*, vol. 17, no. 2, pp. 77–172, 2023.

[18] C. M. Schneider, V. Belik, T. Couronné, Z. Smoreda, and M. C. González, "Unravelling daily human mobility motifs," *Journal of The Royal Society Interface*, vol. 10, no. 84, p. 20130246, 2013.

[19] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.

[20] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *ICML*, 2018, pp. 4393–4402.

[21] A. Das, A. Dasgupta, and R. Kumar, "Selecting diverse features via spectral regularization," *NeurIPS*, vol. 25, 2012.

[22] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[23] J. S. Taylor and N. Cristianini, "Support vector machines and other kernel-based learning methods," *Cambridge University*, 2000.

[24] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *arXiv preprint arXiv:1707.05005*, 2017.

[25] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019, pp. 594–602.

[26] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*. IEEE, 2008, pp. 413–422.

[27] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong, "A meta-analysis of the anomaly detection problem," *arXiv preprint arXiv:1503.01158*, 2015.

[28] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *ACM SIGKDD*, 2005, pp. 157–166.

[29] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[30] C. C. Aggarwal, "Outlier analysis," in *Data mining*. Springer, 2015, pp. 237–263.

[31] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[32] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection." *arXiv:2009.11732*, 2020.

[33] R. Yu, X. He, and Y. Liu, "Glad: group anomaly detection in social media analysis," *TKDD*, vol. 10, no. 2, pp. 1–22, 2015.

[34] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *KDD*. ACM, 2018, pp. 2672–2681.

[35] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *SDM*. SIAM, 2019, pp. 594–602.

[36] X. Wang, Y. Du, P. Cui, and Y. Yang, "OCGNN: one-class classification with graph neural networks," *CoRR*, vol. abs/2002.09594, 2020.

[37] C. C. Noble and D. J. Cook, "Graph-based anomaly detection." in *KDD*. ACM, 2003, pp. 631–636.

[38] W. Eberle, L. Holder, and D. Cook, "Identifying threats using graph-based anomaly detection," in *Mach. Learn. in Cyber Trust*, 2009.

[39] X. Luo, J. Wu, J. Yang, S. Xue, H. Peng, C. Zhou, H. Chen, Z. Li, and Q. Z. Sheng, "Deep graph level anomaly detection with contrastive learning," *Scientific Reports*, vol. 12, no. 1, p. 19867, 2022.

[40] R. Ma, G. Pang, L. Chen, and A. van den Hengel, "Deep graph-level anomaly detection by glocal knowledge distillation," in *WSDM*. ACM, 2022, pp. 704–714.

[41] D. Berberidis, P. J. Liang, and L. Akoglu, "Summarizing labeled multi-graphs," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2022, pp. 53–68.

[42] W. Tang, Z. Lu, and I. S. Dhillon, "Clustering with multiple graphs," in *ICDM*. IEEE, 2009, pp. 1016–1021.

[43] E. Papalexakis, L. Akoglu, and D. Ience, "Do more views of a graph help? community detection and clustering in multi-graphs," in *International Conference on Information Fusion*. IEEE, 2013, pp. 899–905.

[44] Z. Kang, G. Shi, S. Huang, W. Chen, X. Pu, J. T. Zhou, and Z. Xu, "Multi-graph fusion for multi-view spectral clustering," *Knowledge-Based Systems*, vol. 189, p. 105102, 2020.

[45] K. Maruhashi, F. Guo, and C. Faloutsos, "Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis," in *ASONAM*. IEEE/ACM, 2011, pp. 203–210.

[46] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *SIGKDD*, 2015, pp. 985–994.