Identifying Fraud Rings Using Domain Aware Weighted Community Detection

Shaik Masihullah, Meghana Negi(✉), Jose Matthew, and Jairaj Sathyanarayana

Swiggy, Bangalore, Karnataka, India
{shaik.masihullah1,meghana.negi,jose.matthew,jairaj.s}@swiggy.in

Abstract. With the increase in online platforms, the surface area of malicious activities has increased manifold. Bad actors abuse policies and services like claims, coupons, payouts, etc., to gain material benefits. These fraudsters often work collusively (rings), and it is difficult to identify underlying relationships between them when analyzing individual actors. Fraud rings identification can be modeled as a community detection problem on graphs where nodes are the actors, and the edges represent common attributes between them. However, the challenge lies in incorporating the attributes' domain-informed importance and hierarchy in coming up with edge weights. Treating all edge types as equal (and binary) can be fairly naive; we show that using domain knowledge considerably outperforms other methods. For community detection itself, while the weight information is expected to be learned automatically in deep learning-based methods like Graph Neural Networks (GNN), it is explicitly provided in traditional methods. In this paper, we propose a scalable and extensible end-to-end framework based on domain-aware weighted community detection to detect fraud rings. We first convert a multi-edge weighted graph into a homogeneous weighted graph and perform domain-aware edge-weight optimization to maximize modularity using the Leiden community detection algorithm. We then use features of communities and nodes to classify both community and a node as fraud or not. We show that our methods achieve up to 9.92% lift in F1-score on internal data, which is significant at our scale, and up to 4.81% F1-score lift on two open datasets (Amazon, Yelp) vs. an XGBoost based baseline.

Keywords: Fraud detection · Graph learning · Community detection · Fraud rings identification · Graph neural network

1 Introduction

Similar to any e-commerce marketplace, every day, new loopholes are being exploited by fraudsters in hyper-local, online food delivery platforms. Fraudsters exploit business policies and services around food-issue claims, payments, coupons, etc., which have a material impact on the platforms' bottom line. Most often, these fraudsters operate in collusive groups. In many cases, when looked at in isolation, it is easy to mistake the behavior of these fraudsters as not that dissimilar to normal, non-fraudulent patterns; it is only when looked at as a group that fraudulent patterns become apparent. Such fraud groups can cause more damage than individual fraudsters due to the collective knowledge they employ and can typically cover more surface area than an individual actor. As a result, identifying fraud rings is a crucial piece in a platform's overall risk mitigation plan.

For identifying frauds, graph-based approaches have been widely studied. Entities (like customers, sellers) in a marketplace and connections (like payment instruments, reviews) between them can be variously represented by homogeneous-, heterogeneous-, relational-, multi-, weighted graphs, and more. Nilforoshan [23] proposes a multi-view graph representation for mining fraud where multiple edges exist between nodes. Belle [2] proposes variants of graph representation learning such as traditional, inductive, and transductive for fraud detection in credit card transactions. While weighted and multi-view graphs solve for multiple edge types and weights, to the best of our knowledge, providing domain information to identify optimal weights has not been explored. Domain information is usually available in the form of, say, payment-instrument linkages being 'stronger' indicators of connectedness vs. linkages based on shared wi-fi addresses. Such fraud groups can cause more damage than individual fraudsters due to the collective knowledge they employ and can typically cover more surface area than an individual actor. As a result, identifying fraud rings is a crucial piece in a platform's overall risk mitigation plan.

GNNs play an important role in building graph-based machine learning (ML) approaches because of their ability to learn from graph structure. Graph Convolution Networks (GCN) have been applied in domains like opinion fraud [9,17] and insurance fraud [16]. For applications where graphs scale to millions of nodes with frequent updates and additions, scalability and run-time have been bottlenecks for GNN based approaches. On the other hand, community detection methods [26,36], which attempt to capture group structure by partitioning the graphs into communities, are typically more scalable. The authors of [4,13,30] have extensively studied GNN-based community formation and proposed methods to combine both approaches. In such methods, the number of communities to be formed has to be provided beforehand as a hyper-parameter, and this limits the attainability of optimal separation of communities.

Most fraud detection literature typically targets a single fraud detection task, like fake reviews or spam detection. However, in most real-world marketplaces, new fraud modus operandi (M.O) emerge all the time, and it is typically impractical, or even infeasible, to build M.O-specific detection models. It is our observation that a large swath of fraud is perpetrated by rings operating in collusion, constantly cooking up new M.Os. There is not much literature on methods or frameworks that can be extended across multiple M.Os being committed by similar sets of fraudsters (i.e., rings).

In this paper, we propose a novel generalizable framework to detect fraud rings using community detection on a graph whose edge weights are learned in a domain-aware manner. Our contributions are:

- We explicitly provide domain knowledge to community detection where optimal weights are algorithmically determined using weight bounds and edge priorities for fraud identification. This explicit feeding of information cannot be done in GNNs, as we expect them to learn this automatically, adding uncertainty.
- Our framework is modular – graph construction, community detection, downstream discriminator. We first convert a multi-edge graph into a weighted homogeneous graph which is then used by Leiden-based community detection. Community information is then used by downstream tasks to perform fraud ring detection. The node-to-community mapping can be used to develop rules and models for multiple M.Os using simple community feature aggregations.
- We perform extensive experiments on two public benchmark datasets (Yelp and Amazon Reviews) and an internal dataset demonstrating the effectiveness of the proposed framework, both in terms of run-time and F1-score performance. Our framework shows a 1.7% relative improvement in F1-score on Amazon, 4.8% on Yelp, and 9.92% on internal datasets.

The remainder of the paper is organized as follows. Section 2 outlines the existing literature in related works. Section 3 introduces the problem statement of fraud rings and challenges involved. Section 4 details the proposed framework. Section 5 demonstrates the experimentations conducted and the ablation study. We end in Sect. 6 by concluding the paper.

2 Related Work

2.1 Community Detection

While recent research has been backed by GNN based approaches, there has been significant research on non-GNN based approaches for community detection [31]. While GNN approaches focus on representation learning, traditional methods use various graph theory strategies. Traditional methods distribute graphs into communities and do not require the number of communities to be supplied beforehand. This is a major advantage as estimating the number of communities upfront is typically impossible. Modularity and Constant Potts Model (CPM) are well-known metrics to evaluate the quality of generated communities. Ghosh et al. [10] proposed a strategy to run the Louvain algorithm in a distributed computing environment. Even though Louvain, by nature, is not scalable, using these types of distributed implementations can process larger graphs, albeit at the cost of extra resources. You et al. [37] proposed a three-stage algorithm that includes central-node identification, label propagation, and community merging. Community detection can also be performed using optimization algorithms like Particle Swarm Optimization (PSO) [26] or semi-supervised approaches like label propagation [36]. The Leiden algorithm [33] introduced in 2019 by Traag et al. can run faster and find better partitions compared to other methods.

A majority of the recent literature on GNNs is based on the assumption of incorporating ring information within the embeddings. Luo et al. [20] attempted to identify communities in a heterogeneous graph by applying their Context Path based GNN model. While most community detection approaches focus on segregating graphs into non-overlapping communities, Shchur et al. [30] focus on generating overlapping communities using a GNN-based Neural Overlapping Community Detection (NOCD) model. Moreover, the NOCD model is also compared with non-deep models such as multi-layer perceptrons and free-variable models, studying the effectiveness of using deep network models. Bandyopadhyay et al. [1] attempted to solve community detection using GNN in an unsupervised approach. The authors integrated a self-expressive layer in GNN and designed a loss function to classify nodes into communities directly. Wang et al. [35] leverage the bipartite representation to perform community detection in heterogeneous graphs using GNNs. The attention mechanism is adopted for increased focus on prime nodes while segregating the graph. Jia et al. [14] presented a generative adversarial network based community detection framework solving overlapping community detection and graph representation learning.

Identifying edge priority has been an active area of research but focuses on learning weights rather than explicit inputs. Shang et al. [29] propose SACN (Structure-Aware Convolution Network) that uses an encoder-decoder architecture. The encoder consists of a weighted graph convolution network, which learns the network weights by utilizing the graph node structure, node attributes, and edge relation types. This generates accurate graph node embeddings capturing most of the important information from the graph. The decoder is a convolution network used for link prediction. In [12], the authors designed a fast algorithm known as MGFS (Multi-label Graph-based Feature Selection algorithm) that utilizes the Page Rank algorithm to estimate feature importance based on edge weights.

2.2 Graph-based Fraud Detection

Graph-based approaches for identifying frauds have been extensively studied in financial fraud [16,19,34] and opinion-fraud detection [9,15,24]. Wang et al. [34] proposed a SemiGNN variant solving the label uncertainty problem in fraud detection where only a small percentage is tagged as fraud with certainty. Liu et al. [19] present a GEM model utilizing graph embeddings and attention mechanisms to identify irregular malicious accounts on a financial platform. Dou et al. [9] proposed a CARE-GNN (CAmouflage-REsistant GNN) architecture emphasizing the identification of camouflaged fraudsters who had built ways to hide themselves among normal users. Liu et al. [18] present a DGFraud (Deep Graph Fraud) model that identifies frauds in social network graphs despite inconsistencies in context, features, or relations among the users. A few researchers [25,28,32] have also attempted to identify fraud community groups in medical, banking, and telecom sectors using traditional community detection such as label propagation and group mining. Sun et al. [32] proposed a person's similarity adjacency graph and Maximal Clique Enumeration-based approach to identify fraud in medical insurance.

Although using deeper networks or complex strategies might outperform during experimentations, these are often constrained by high inference time, complex retraining processes, and non-trivial deployment challenges, when applied in real-world applications. To the best of our knowledge, ours is the first novel and scalable end-to-end framework using traditional graph theory-based community detection, which outperforms GNNs in terms of both run-time and F1-score performance.

3 Fraud Detection Problem

3.1 Identifying Fraud Rings

A graphical representation of the user base of an e-commerce platform can have customers sharing stationary and non-stationary attributes. These customers can be grouped into communities, combined behavior of which can be used to identify fraud rings. Figure 1 illustrates how an entity's fraud status changes when looked at from the lens of its connections. Using only an entity's attributes might classify the entity as not-fraud. However, if the entity is connected to a number of high-risk entities, then the whole community and the entity at hand could be classified as fraudulent. It should be noted that it is not possible to constrain rings identified to be composed solely of fraudulent actors. It is typically a business decision to either add some post-processing to reduce such false positives or choose to live with it.

3.2 Incorporating Domain Knowledge

In e-commerce marketplaces, it is pretty likely that the customers can be connected to other customers. One can create connections based on identifiers like payments, wi-fi, etc., to behavioral aspects like similar buyers, areas, category preferences, etc. The importance of such connections varies based on the problem one is trying to solve. For example, behavioral relations can boost the quality of recommendations, and identifiers can help identify swindling tendencies. Further, in fraud rings identification, not all rings are equal. For example, for a hyper-local marketplace, a ring of entities connected by a common device(s) is a 'stronger' signal compared to rings of wi-fi addresses or broad geo-locations. It could be the opposite for a social network. It is crucial to encode this additional domain-specific knowledge.

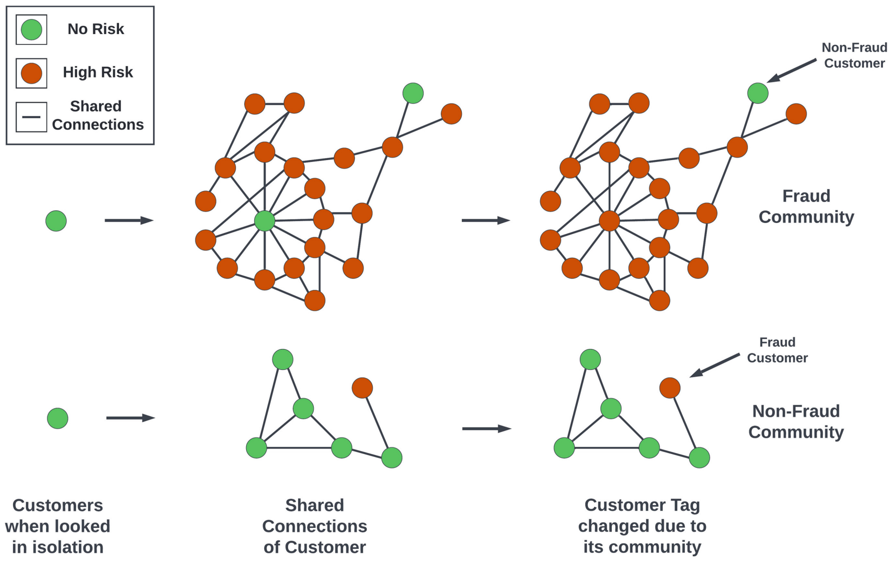
No Risk
High Risk
Shared Connections
Non-Fraud Customer
Fraud Community
Fraud Customer
Non-Fraud Community
Customers when looked in isolation
Shared Connections of Customer
Customer Tag changed due to its community

Fig. 1. An illustration of fraud rings identification problem.

Adding weights and domain knowledge to the graph helps enforce graph substructure by focusing on relationships that identify frauds with higher confidence and distilling the accidental or less relevant connections. The basic assumption is that "Not all the available information is useful or is of equal importance". Without weights, all the edges will have equal importance, which might increase false positives. Weights benefit the graph algorithms by providing domain knowledge information required to learn better representation from the neighborhood. These weights are learned automatically in GCNs but are taken as inputs in community detection. Explicitly setting weights to the edge relations provides us control over the amount of information to be used from the neighborhood.

3.3 Other Challenges in Fraud Detection

- **Concept Drift:** Fraudsters are constantly inventing new M.Os leading to the breaking of models trained on older data distributions. This phenomenon is known as concept drift. We tackle concept drift in two ways. Firstly, several M.O-agnostic features are employed which do not change with time, i.e., stationary attributes. Secondly, by abstracting community detection from fraud identification discriminator.
- **Scalability:** With millions of customers, it is challenging to represent them in a single graph or as multiple subgraphs. Further, it is computationally costly and time-consuming to train graphical ML models on large graphs. We rely on Leiden methods' fast local move approach to detect quality communities in much less time.

- **Cold Start:** A majority of the fraud detectors depend on entities' history or interactions on a platform to predict fraud. However, fraudsters continually create new accounts to commit frauds for which there will be no historical data. This is similar to cold-start problems in recommendation systems. To combat this, we match these new accounts with previously identified stationary attributes, linking them to a community. Since our framework tags an entire community as fraud or not, the same label is inherited by new accounts 'matched' to that community. We were thus able to uncover 20% more frauds on a daily basis.

4 Proposed Framework

In this section, we present our community-based fraud detection framework. Firstly, we provide an overview of the framework and its workflow. Then, we detail the different modules involved.

4.1 Overview

The proposed framework has three modules: Graph Construction (GC), Community Detection (CD), and Downstream Task (DT). The GC module molds the raw data into a graph representation. We then propose a way to convert a multigraph into a homogeneous one. The CD module processes this graph and segregates the nodes into possible rings based on their connectivity. This module also takes in the domain knowledge in the form of edge-weights optimized over the modularity metric. Finally, the DT module predicts communities as fraud rings and nodes as fraudulent customers. An illustration of the proposed framework is presented in Fig. 2.

4.2 Graph Construction

Graph Definition. We construct a graph where customers are the nodes, and stationary attributes are the edges between them. All edge types in our graph are undirected. Each edge type has a weight that signifies the importance of that attribute in identifying fraud. Since we intend to use off-the-shelf community detection algorithms which require input graphs to be homogeneous, we make a simplifying assumption of treating our edge types as homogeneous (our nodes are already homogeneous).

Graph Representation. A multigraph is defined as $G(V, E(e,t), W(t,w))$, where

V is a set of vertices representing customers with $\|V\| = n$,

E is the set of edges with $\|E\| = m$, each edge e having a relation type attribute t ∈ T' = {1, 2, ..., T} where T' is the set of possible edge relation types, and

W = $(w_1, w_2 ..., w_T)$ is a weight vector that maps each edge relation type t to a weight w_t. By default, all edge relation types have unit weight.

The multigraph (G) is converted into a homogeneous graph (G'), by merging multiple edges between the same two nodes as one edge and summing the edge weights.

G' is defined as $G'(V, E'(e, m), W')$,

where

V is the same set of vertices representing users with $\|V\| = n$,

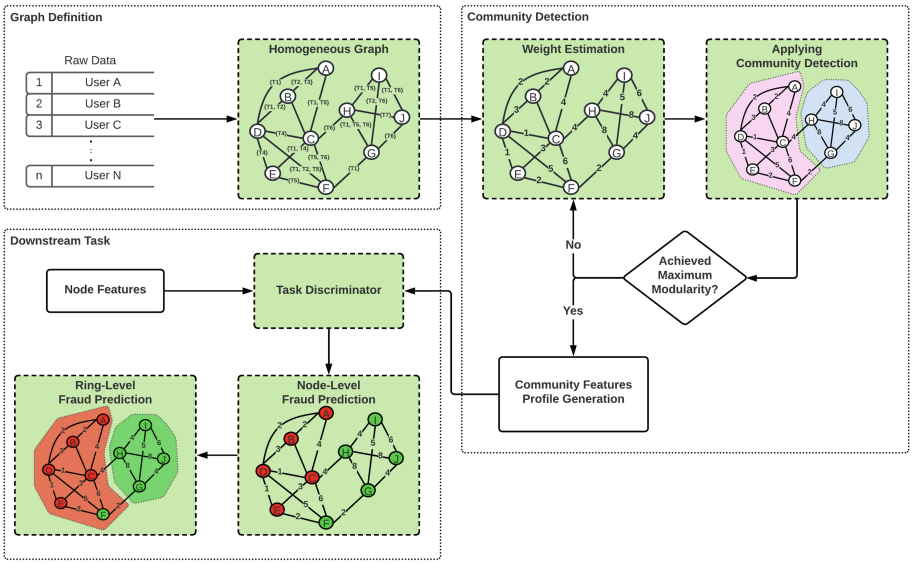

Fig. 2. Proposed community-based fraud detection framework.

E' is the set of edges with $\|E'\| = m'$, each edge e having a merge set type attribute u ∈ P(T'), power set of T'

W' = $(w'_1, \ldots, w'_{2n})$ is a new weight vector, where w'_u is the sum of weights of all edge types in the merge set u.

We experimented with summation, averaging, and multiplication as weight aggregation techniques, out of which, summation worked best (as indicated by goodness-of-fit in downstream tasks) in our experiments. Figure 3 illustrates the conversion from a multigraph to a homogeneous graph.

4.3 Community Detection

Weighted Community Detection. In a graph, a community is defined as a set of nodes that can be grouped together such that each set of nodes is densely connected internally, and loosely connected with the rest of the nodes. Several graph algorithms exist for community detection, which evolved over time from

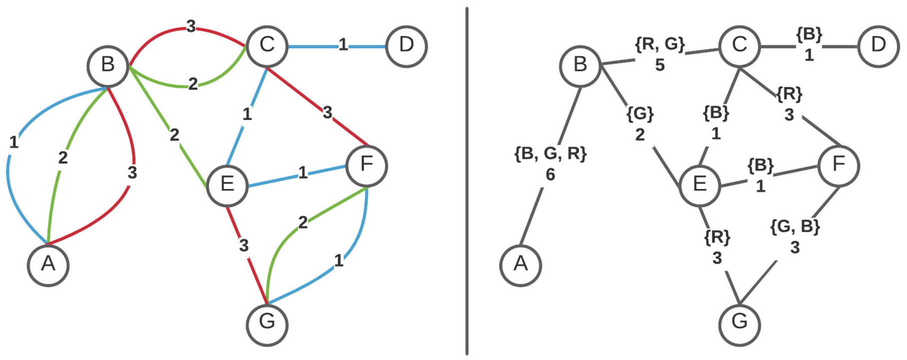

Fig. 3. Left: A sample multigraph representation. Edge colors and numbers represent different edge types and their edge weights respectively. Right: The homogeneous graph derived where edge attributes store the merged information such as edge types involved and their weights aggregation. A simple summation of edge weights for aggregation is illustrated here.

Newman [22], Louvain [3] to Leiden algorithm [33]. Our framework adopts the Leiden algorithm (LDN), which builds on the Louvain algorithm. LDN employs a three-step process for segregating communities:

1. local moving of nodes,
2. refinement of the partition, and
3. aggregation of the network based on the refined partition, using the non-refined partition to create an initial partition for the aggregate network.

LDN supports two objective functions known as Modularity and the Constant Potts Model (CPM). Modularity is a measure of how well a graph is partitioned into communities. It tries to maximize the difference between the actual number of edges in a community and the expected number of such edges and is defined as follows:

$$H = \frac{1}{2m} \sum_{c} \left(e_c - \gamma \frac{K_c^2}{2m} \right)$$

Here, e_c denotes the actual number of edges in community c. $\frac{K_c^2}{2m}$ denotes the expected number of edges, where K_c is the sum of the degrees of the nodes in community c and m is the total number of edges in the network. γ is the resolution parameter that ranges in [0, 1]. Higher resolution leads to more communities, while lower resolution leads to fewer communities.

Our framework's novelty is the inclusion of domain knowledge in community detection. The various edge types between the customers can differ in the information they convey with respect to a task. For example, to suggest friends in a social graph, subscribing to a common page is likely a more important edge than clicking on the same ad. A similar analogy applies to fraud detection in e-commerce graphs where the edge weight of stationary attributes could be different. Prior domain knowledge is required to define these weights. Weighted community detection can then exploit this domain knowledge to better segregate a graph, compared to an unweighted or domain-agnostic method. The brute-force way would be to take the best guesstimate and pick weights as constant values such that order is maintained between the edges type. The problem with this approach is that weights might not be optimal. The proposed way picks inspiration from constrained optimization. We first define upper and lower bounds on weights for each edge type. Then, we use a relative priority of edge types as constraints over these weights. The bounds can overlap among edge types, but estimated weight combinations should follow priority constraints. For example, consider two edge types with bounds as [0,10] and [5,10] respectively and edge_type_1 with a lower priority than edge_type_2. The set of weight combinations considered during optimization have edge_type_1_weight < edge_type_2_weight with [5<6,4<8,1<10] as valid and [8<5, 9<5] as invalid example combinations.

Community Profile. Each node is represented by a set of features (F) derived from the node's domain behavior. Key group indicators (KGI) are a subset of F that can be directly influenced by a group context. Community profile (CP) is defined by the combined representation of the member node. A CP vector is an aggregate of KGI feature values corresponding to each node in the community. Figure 4 explains CP in a fraud reviews detection problem.

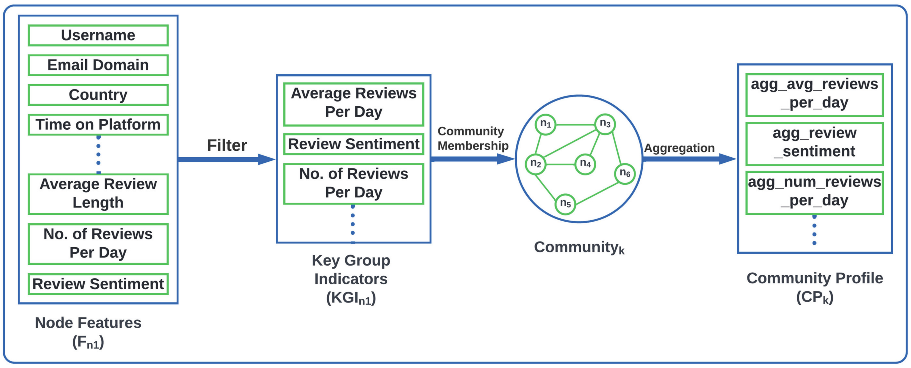

Fig. 4. Community profile

Proposed Algorithm. The CD module comprises of the below step by step process:

1. Define weight bounds and edge type priorities for the input homogeneous graph
2. Iteratively select combinations of weights using optimization methods like grid search.
3. Apply weighted community detection with weights from step 2 and modularity as the optimization metric

4. Output the community separation with maximum modularity as the best separation
5. For each community in the best separation, aggregate node features to create a community profile

The community detection process is presented as Algorithm 1.

Algorithm 1. Domain Aware Community Detection

```
Input: G' = (V, E'(e, m), W): A homogeneous graph with equal uni-weighted edge types
Output: A mapping of each node in V to a community and a community features vector representation for each community formed
  procedure CommunityDetection(Graph G')
      $max\_modularity \leftarrow 0$
      $best\_communities\_seperation \leftarrow None$
      $ub = (u_1, u_2, ..., u_t)$
      $lb = (l_1, l_2, ..., l_t)$
      $constraint = \{n\}$ #example $\rightarrow n_r > n_g$ & $n_g \geq n_r$ & $n_b = n_r$
      $weight\_combinations \leftarrow weight\_estimation(ub, lb, constraint)$
      for each w in $weight\_combinations$ do
          for each e in E' do
              $E" += e' = sum(m, w)$
          end for

          $G" \leftarrow graph(E")$
          $partitions \leftarrow leiden\_community\_detection(G")$
          $communities \leftarrow group\_partitions\_by\_community\_id(partitions)$
          $modularity \leftarrow evaluate(communities)$
          if $modularity > max\_modularity$ then
              $max\_modularity \leftarrow modularity$
              $best\_communities\_separation \leftarrow communities$
          end if
      end for

      $community\_features \leftarrow dict\{community\_id : aggregated\_features\}()$
      for $community$ in $best\_communities\_separation$ do
          $aggregated\_features \leftarrow vector()$
          for each $node$ in $community$ do
              Aggregate individual node features within a community as required and add the to the aggregated_features vector
              $aggregated\_features \leftarrow addup\_node\_features()$
              $community\_features[community\_id] \leftarrow aggregated\_features$
          end for
      end for
  return $partitions, community\_features$
  end procedure
```

4.4 Downstream Task

The DT module consists of a discriminator which takes node features and community profiles as inputs. In GCN-based methods, the task information is provided during the optimization process, and hence, the learned node-embeddings are tuned to the respective M.Os/tasks. In the proposed framework, the communities identified in the previous step(s) can be used across M.Os. Community 'profile' consists of aggregated features across all possible identified frauds. For example, for an M.O involving order cancellations, the incidence of cancellation in the community can be used. For M.Os around fraudulent claims, the incidence of claims at a community level can be used. Since the communities' profile vector already encapsulates the incidence of various fraudulent actions, it can help in creating a variety of lightweight discriminators with little to no future feature engineering. For example, a simple rule like flagging all communities with incidences of M.Os above a certain threshold can be employed to identify rings. This is especially useful for emerging M.Os or when it is challenging to get labeled data. For M.Os where sufficient labeled data is available, supervised methods with node and community features as inputs can be used.

5 Experiments

In this section, we demonstrate the effectiveness of our framework on real-world fraud detection problems, namely, opinion fraud and financial fraud. We also compare and contrast the framework in unweighted and weighted variants, using grid-search and PSO (particle swarm optimization) methods for weights estimation, in addition to GCN-based and Leiden community detection. We use the end-to-end run-time of the framework and the F1-score of the downstream task as the comparison metrics.

Table 1. Graph statistics of reviews datasets for opinion fraud detection

Dataset	#Nodes	#Edges	Degree	Fraud percentage
Amazon	11,944	4,398,392	739.71	14.5%
Yelp	45,954	3,846,979	167.47	9.5%
Our graph	˜3.9M	˜4.9M	2.48	< 5%*

* to preserve confidentiality, we do not reveal the exact number

5.1 Experimental Setup

Datasets. For opinion fraud detection, two open datasets are considered: restaurant-review spam data from Yelp [27] and product-review fraud data from Amazon [38]. Both have labels for each review/user being either fraud (spam) or benign (genuine). Both of these datasets can be represented as multi graphs, with one node type and multiple edge types.

Yelp dataset has reviews as nodes and 32 handcrafted node features with the following three relations:

1. R-U-R: links different reviews posted by the same user
2. R-S-R: links different reviews under the same product with the same star rating
3. R-T-R: links different reviews under the same product posted in the same month

Amazon dataset has users as nodes and 25 handcrafted node features with the following three relations:

1. U-P-U: links different users reviewing at least one same product
2. U-S-U: links different users having at least one same star rating within one week
3. U-V-U: links different users with top 5% mutual review text TF-IDF similarities

For benchmarking on our internal dataset, we tackle a M.O related to cash transactions where the graph is built on two months' worth of cash-transacting customers. Our graph has customers as nodes and 60 node features with our relations (for confidentiality reasons, we cannot reveal the exact relationship types).

As mentioned in Sect. 4.2, all datasets are converted to homogeneous graphs. Table 1 shows the statistics.

Methods Compared. We compare our proposed methods (domain-aware, weighted community-detection based) against GCN methods as shown in Table 2. The baseline is an XGBoost classifier which also serves as the discriminator for all non-baseline methods.

Table 2. Methods used for ablation

Name	Description	Community detection method	Input to discriminator
Baseline	XGBoost	-	Node features
GCN	Graph Convolutional Network	-	Node features & Graph embeddings
DMoN	Deep Modularity Network [17]	Modularity-based GCN	Node features & Graph embeddings
UWL	Unweighted community detection	Leiden	Node features & Graph embeddings
OWL*	Domain-aware weighted community detection using PSO optimization	Leiden	Node features & Graph embeddings
WL*	Domain-aware weighted community detection using grid search	Leiden	Node features & Graph embeddings

* our proposals

5.2 Experimental Settings

For GCN variants, we implemented Deep Graph Infomax [18] to learn node representations. A GCN with 2-layer and 128-dimension embeddings was used for comparison methods across all three datasets. For GCN-based community detection using DMoN, a 2-layer network each of 32 dimensions was trained for all datasets. Both GCN & DMoN were trained for 100 epochs with Adam optimizer with a learning rate of 0.001 and a dropout of 0.5. The choice of architectures for GCN and DMoN was primarily driven by the computing resources required. In the case of the XGBoost discriminator, hyper-parameter settings were kept the same for all the comparison methods but were specific to the dataset. To achieve a fair comparison, the classification threshold of the XGBoost discriminator was adjusted so that the fraud coverage (percentage of samples tagged as fraud by a model) was the same for all variants and equal to the dataset's fraud percentage. This is called the threshold-moving strategy in the literature [6].

5.3 Implementation

Graphs were constructed and maintained using networkx [11] and igraph [7] libraries. For PSO, the global optimization variant was adopted from pyswarms [21]. We used the Leiden implementation from igraph with default parameters. For GCN and XGBoost implementations, Stellargraph [8] and XGBoost [5] packages were used respectively. All models were trained on an AWS m4.4xlarge CPU instance.

Table 3. Performance comparison datasets

Comparison method	Amazon			Yelp			Our graph	
	F1 score			F1 score				
		Relative improvement	Time taken		Relative improvement	Time taken	Relative improvement in F1-score*	Time taken
Baseline	0.8383	-	2 s	0.7751		2 s	0%	2 s
GCN	0.8423	0.48%	3.5 min	0.7958	2.67%	2.65 min	3.58%	35 min
DMoN	0.832	−0.75%	10.2 min	0.7928	2.28%	8 min	−0.64%	48 min
UWL	0.8406	0.27%	8.6 s	0.8279	6.81%	2 s	2.47%	44.5 s
OWL**	0.839	0.08%	9.5 s	0.792	2.18%	2 s	2.94%	44.5 s
WL**	0.8532	1.78%	9.5 s	0.8124	4.81%	2 s	9.92%	44.5 s

* to preserve confidentiality, we only report relative improvement numbers for our data
** our proposals

5.4 Experimental Results

Table 3 shows the F1 scores and time taken on the CPU of each dataset. As previously mentioned, the baseline method uses only node features, while the other methods use both node and neighborhood information. On the Amazon dataset, only our WL method shows a pragmatically meaningful improvement (1.78%). On the Yelp dataset, all methods handily beat the baseline, demonstrating the usefulness of neighborhood information. Our WL method trails UWL on the Yelp dataset. We hypothesize that this is primarily due to our limited knowledge of Yelp's domain which has, in turn, had a direct bearing on and affects weight optimization and the quality of communities formed. On our dataset, DMoN under-performs the baseline signifying that, in larger graphs, limiting the number of communities can negatively affect downstream performance. Both OWL (+2.94%) and WL (+9.92%) outperform UWL (+2.47%), indicating the importance of domain-aware weights. Between OWL and WL, we hypothesize that PSO was not able to optimize better by using only modularity as the objective function and hence could not outperform the grid-search-based WL.

On the computation time front, while for smaller graphs like Amazon and Yelp, the difference is in seconds, for larger graphs like ours with 3 M nodes, LDN concludes in less than a minute, while GCN methods take at least half an hour. We only used two months of data for these experiments; expanding this horizon will have a direct bearing on graph size. Hence, we hypothesize that for even larger graphs, training GCN will take much longer.

We also investigated the temporal stability of the communities formed. We constructed graphs over a moving time window of two months at a weekly level. We then compared the movements of customers' assignments from one community to another in every iteration and found a 3–5% movement which is an acceptable threshold for us. As previously claimed, we were also able to create a rule-based classifier for an unseen-before M.O using the results from the community module within a few days (as opposed to weeks/months if we had to build models from scratch for this M.O). We were also able to change the threshold of these rules based on changing fraud behavior with no change in the underlying graph or community modules. This framework is currently deployed in production, inferencing millions of transactions per day, with the graph and community modules being updated weekly.

6 Conclusion and Future Work

In this paper, we proposed a novel end-to-end fraud detection framework to identify fraud rings. To the best of our knowledge, this is the first attempt at a scalable graph-based system utilizing domain knowledge as weighted edge priorities in Leiden community detection. Experiments were conducted on large-scale open and internal fraud datasets demonstrating the effectiveness of the proposed framework using F1 score and CPU run-times.

As an extension, we plan to experiment with different objective functions and strategies that can potentially outperform grid searching. On the community detection front, we want to explore how we can extend this work to handle overlapping communities. Our current implementation uses stationary attributes for edges. Given the dynamic nature of fraud, it is important to explore ways to incorporate non-stationary attributes, which can potentially help make detections resistant to changing fraud patterns. In this work, we also made the simplifying assumption of converting heterogeneous graphs into homogeneous ones. We would like to explore if there are additional benefits to be derived by researching ways to directly use heterogeneous graphs instead.

References

1. Bandyopadhyay, S., Peter, V.: Unsupervised constrained community detection via self-expressive graph neural network. In: Uncertainty in Artificial Intelligence, pp. 1078–1088. PMLR (2021)
2. Van Belle, R., Mitrović, S., De Weerdt, J.: Representation learning in graphs for credit card fraud detection. In: Bitetta, V., Bordino, I., Ferretti, A., Gullo, F., Pascolutti, S., Ponti, G. (eds.) MIDAS 2019. LNCS (LNAI), vol. 11985, pp. 32–46. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-37720-5_3
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Statist. Mech. Theory Exp. **2008**(10), P10008 (2008)
4. Cao, C., Li, S., Yu, S., Chen, Z.: Fake reviewer group detection in online review systems. In: 2021 International Conference on Data Mining Workshops (ICDMW), pp. 935–942. IEEE (2021)
5. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
6. Collell, G., Prelec, D., Patil, K.R.: A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. Neurocomputing **275**, 330–340 (2018)
7. Csardi, G., Nepusz, T., et al.: The igraph software package for complex network research. Int. J. Complex Syst. **1695**(5), 1–9 (2006)
8. Data61, C.: Stellargraph machine learning library (2018). https://github.com/stellargraph/stellargraph
9. Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., Yu, P.S.: Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 315–324 (2020)
10. Ghosh, S., et al.: Distributed Louvain algorithm for graph community detection. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 885–895. IEEE (2018)
11. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using network. Technical report, Los Alamos National Lab. (LANL), Los Alamos, NM (United States) (2008)
12. Hashemi, A., Dowlatshahi, M.B., Nezamabadi-Pour, H.: MGFS: a multi-label graph-based feature selection algorithm via PageRank centrality. Expert Syst. App. **142**, 113024 (2020)
13. He, D., Song, Y., Jin, D., Feng, Z., Zhang, B., Yu, Z., Zhang, W.: Community-centric graph convolutional network for unsupervised community detection. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pp. 3515–3521 (2021)

14. Jia, Y., Zhang, Q., Zhang, W., Wang, X.: Communitygan: community detection with generative adversarial nets. In: The World Wide Web Conference, pp. 784–794 (2019)
15. Li, A., Qin, Z., Liu, R., Yang, Y., Li, D.: Spam review detection with graph convolutional networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2703–2711 (2019)
16. Liang, C., Liu, Z., Liu, B., Zhou, J., Li, X., Yang, S., Qi, Y.: Uncovering insurance fraud conspiracy with network learning. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1181–1184 (2019)
17. Liu, Y., et al.: Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In: Proceedings of the Web Conference 2021, pp. 3168–3177 (2021)
18. Liu, Z., Dou, Y., Yu, P.S., Deng, Y., Peng, H.: Alleviating the inconsistency problem of applying graph neural network to fraud detection. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1569–1572 (2020)
19. Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., Song, L.: Heterogeneous graph neural networks for malicious account detection. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 2077–2085 (2018)
20. Luo, L., Fang, Y., Cao, X., Zhang, X., Zhang, W.: Detecting communities from heterogeneous graphs: A context path-based graph neural network model. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1170–1180 (2021)
21. Miranda, L.J.: PySwarms: a research toolkit for particle swarm optimization in python. J. Open Source Softw. **3**(21), 433 (2018)
22. Newman, M.E.: Fast algorithm for detecting community structure in networks. Phys. Rev. E **69**(6), 066133 (2004)
23. Nilforoshan, H., Shah, N.: Slicendice: mining suspicious multi-attribute entity groups with multi-view graphs. In: 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 351–363. IEEE (2019)
24. Noekhah, S., binti Salim, N., Zakaria, N.H.: Opinion spam detection: using multi-iterative graph-based model. Inf. Process. Manage. **57**(1), 102140 (2020)
25. Peng, L., Lin, R.: Fraud phone calls analysis based on label propagation community detection algorithm. In: 2018 IEEE World Congress on Services (SERVICES), pp. 23–24. IEEE (2018)
26. Rahimi, S., Abdollahpouri, A., Moradi, P.: A multi-objective particle swarm optimization algorithm for community detection in complex networks. Swarm Evol. Comput. **39**, 297–309 (2018)
27. Rayana, S., Akoglu, L.: Collective opinion spam detection: bridging review networks and metadata. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 985–994 (2015)
28. Sarma, D., Alam, W., Saha, I., Alam, M.N., Alam, M.J., Hossain, S.: Bank fraud detection using community detection algorithm. In: 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 642–646. IEEE (2020)
29. Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B.: End-to-end structure-aware convolutional networks for knowledge base completion. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 3060–3067 (2019)
30. Shchur, O., Günnemann, S.: Overlapping community detection with graph neural networks. arXiv preprint arXiv:1909.12201 (2019)

31. Souravlas, S., Anastasiadou, S., Katsavounis, S.: A survey on the recent advances of deep community detection. Appl. Sci. **11**(16), 7179 (2021)
32. Sun, C., Yan, Z., Li, Q., Zheng, Y., Lu, X., Cui, L.: Abnormal group-based joint medical fraud detection. IEEE Access **7**, 13589–13596 (2018)
33. Traag, V.A., Waltman, L., Van Eck, N.J.: From Louvain to Leiden: guaranteeing well-connected communities. Sci. Rep. **9**(1), 1–12 (2019)
34. Wang, D., et al.: A semi-supervised graph attentive network for financial fraud detection. In: 2019 IEEE International Conference on Data Mining (ICDM), pp. 598–607. IEEE (2019)
35. Wang, L., Li, P., Xiong, K., Zhao, J., Lin, R.: Modeling heterogeneous graph network on fraud detection: a community-based framework with attention mechanism. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1959–1968 (2021)
36. Yang, G., Zheng, W., Che, C., Wang, W.: Graph-based label propagation algorithm for community detection. Int. J. Mach. Learn. Cybern. **11**(6), 1319–1329 (2019). https://doi.org/10.1007/s13042-019-01042-0
37. You, X., Ma, Y., Liu, Z.: A three-stage algorithm on community detection in social networks. Knowl. Based Syst. **187**, 104822 (2020)
38. Zhang, S., Yin, H., Chen, T., Hung, Q.V.N., Huang, Z., Cui, L.: GCN-based user representation learning for unifying robust recommendation and fraudster detection. In: Proceedings of the 43rd international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 689–698 (2020)