EnsemFDet: An Ensemble Approach to Fraud Detection based on Bipartite Graph

Yuxiang Ren*, Hao Zhu†, Jiawei Zhang* , Peng Dai‡ and Liefeng Bo‡
*IFM Lab, Department of Computer Science, Florida State University, FL, USA
†College of Engineering & Computer Science, Australian National University, Canberra, Australia
‡AI Lab, JD Finance America Corporation, CA, USA
Email: yuxiang@ifmlab.org, hao.zhu@anu.edu.au, jiawei@ifmlab.org, peng.dai@jd.com, liefeng.bo@jd.com

***Abstract*—Fraud detection is extremely critical for e-commerce business platforms. It is the intent of the companies to detect and prevent fraud as early as possible. Utilizing graph structure data and identifying unexpected dense subgraphs as suspicious is a category of commonly used fraud detection methods. Among them, spectral methods solve the problem efficiently but hurt the performance due to the relaxed constraints. Besides, existing heuristic methods cannot be accelerated with parallel computation and fail to control the scope of returned suspicious nodes because they provide a set of subgraphs with diverse node sizes. These drawbacks affect the real-world applications of existing methods. In this paper, we propose an *Ensemble based Fraud DETection* (ENSEMFDET) method to scale up fraud detection in bipartite graphs by decomposing the original problem into subproblems on small-sized subgraphs. By oversampling the graph and solving the subproblems, the ensemble approach further votes suspicious nodes without sacrificing the prediction accuracy. Extensive experiments have been done on real transaction data from JD.com, which is one of the world's largest e-commerce platforms. Experimental results demonstrate the effectiveness, practicability, and scalability of ENSEMFDET. More specifically, ENSEMFDET is up to 100x faster than the state-of-the-art methods due to its parallelism with all aspects of data.**

***Index Terms*—Bipartite Graph, Ensembles, Fraud Detection, Graph Mining**

I. INTRODUCTION

Fraud detection in large scale graph structures is very important. In many scenes, fraudsters try to manipulate networks for personal gain [1]. As online services are becoming increasingly popular, it attracts fraudsters to look for measures to abuse their virtual currency system. For example, a large number of accounts are created and controlled by a group of fraudsters, and these malicious fraudsters may cash out and obtain ill-gotten wealth, thereby greatly damaging the entire virtual currency system.

Among many fraudulent activities, arbitrage frauds that use online promotional campaigns are common but not easy to deal with. E-commerce platforms usually launch many promotional campaigns to increase platform clickstream and attract user consumption. For example, each transaction can enjoy a 5-dollar discount if the transaction cost is greater than $5.01, or the discount can be a random value within 5 dollars no matter what the total transaction cost is. Fraudsters usually register many malicious accounts in batches and control malicious accounts to make bulk purchases during the promotion period to illegally profit. This kind of fraud not only damages the company's interests, where the costs invested in promotional activities are not converted into clickstreams but also deprives many users of the discounts they should have received. However, fraud behaviors in promotional campaigns are not easy to detect and defend. Most promotional campaigns will not last long and change a lot next time. According to our observation of the business on JD.com, most fraudulent accounts used by fraudsters will not be reused after a period of time, and the features of fraud behaviors will also change with the different promotional campaigns. Therefore, it is difficult to label fraud behaviors and utilize historical data. In other words, feature-based supervised learning [34], [27], [11], [29], a classical way to detect fraud behaviors, is very hard to be applied in such a scenario. In order to reduce the growing abuse of promotional campaigns, expert-rule systems are developed to identify suspicious accounts. However, fraudsters would update means of fraudulence using techniques such as device and IP obfuscation to evade the rules, and there are increasing difficulties in detecting these more sophisticated attacks. In addition, millions of fake user accounts are being created every day, and they further reduce the density of attacking and thus evade related rules. Besides, expert-rule systems heavily depend on the expert experiences finding in manual monitoring.

Compared with some camouflage measures to bypass expert-rule systems, the traces (e.g., links and nodes) left by fraudsters on graphs are difficult to eliminate. Graph-based approaches [30], [31], [17], [13] can analyze all events and users at the same time. And thus they can detect fraud behaviors and promotion abuse systematically. By finding the hidden relationships and behavior consistency among all accounts, graph-based methods can detect different groups of fraudsters without any label for training which is extremely expensive and time-consuming to get in the real world. As a data-driven approach, graph-based methods can reduce the costs generated by the lagged effect of making rules as well. Graph-based approaches detect groups of fraudsters by identifying unexpectedly dense regions of the graph. Intuitively, graph-based fraud detection methods share the same idea with density-based clustering approaches: both looking for subgraphs with a higher density than the remainder of the graphs/data. The task of graph-based fraud detection is to find all subgraphs that have anomalous patterns (usually of high

density) from the provided objective graph. There are two main types of approaches for this task: spectral methods [30], [31], [17] and heuristic algorithms [13], [33]. Compared with spectral methods, the heuristic algorithms commonly have better performance. However, heuristic methods commonly detect suspicious subgraphs with diverse sizes, and all nodes in one suspicious subgraph will be labeled as suspicious nodes. This property hurts the practicability of heuristic methods in real-world applications because they will lead to a zigzag ROC curve and precision-recall curve. Therefore, we cannot select suspicious nodes by configuring the recall or false positive rate. Besides, heuristic methods commonly tend to return the densest block because there is no efficient strategy to control how many blocks the algorithms should find. The most important thing is that the heuristic process in algorithms is sequential and thus it is difficult to be parallelized. These drawbacks greatly hinder their applications in the real world.

To address the aforementioned drawbacks, we propose a model, namely ***Ensem****ble based* ***F****raud* ***DET****ection* (ENSEMFDET), in this paper. In ENSEMFDET, we first define the optimization problem of graph-based fraud detection corresponding to our business scenarios. Second, we show how to implement graph-based fraud detection practical by decomposing the original graph into small-sized subgraphs so as to lower the search costs. An ensemble framework is utilized in ENSEMFDET, which employs three structural sampling methods for the bipartite graph with theoretical guarantees. The ensemble framework aggregating the outputs of several subproblems can reduce the overall risk of achieving a particularly poor suboptimal solution, and thus maintaining high prediction accuracy. Third, a method FDET is deployed in ENSEMFDET to detect fraudsters, which enables a more efficient search for the top-k fraud subgraphs. The selected parameter k can be determined by a strategy in FDET. In addition, ENSEMFDET can compute fraud detection in sampled graphs in parallel, thus accelerating the detection process. Last, using real-life datasets of promotion campaigns, we conduct extensive experiments and show that ENSEMFDET for fraud detection is effective, practical, scalable and stable.

The remaining paper is organized as follows. We review the related works in Section II. Then we introduce several important concepts and formulate the problem in Section III. The proposed model ENSEMFDET is introduced in Section IV, whose effectiveness is evaluated in Section V. Finally, we conclude this paper in Section VI.

II. Related Work

Approaches in the field of fraud detection can be classified into two categories: feature-based and graph-based.

Feature-based approaches model the users and activities by representing them through a set of attributes, such that each entity is represented as a vector in a multi-dimensional space. In order to distinguish normal users from fraudsters, the appropriate set of attributes should be designed hand-crafted, so that entities will lie in significantly different regions in this feature space. With the rise of deep learning, neural networks have also been used to extract features from fraudsters and relating events, and accomplish fraud detection through a classifier [34], [27], [25], [11], [29]. This class of algorithms often require pre-labeled training data, which are used to find the distinguishing attributes. These algorithms have been successfully used to predict trolls [7], vandals [22], hoaxes [23], and sockpuppets [21], among several other anomaly entities. However, obtaining reliable labels is extremely expensive and time-consuming in real business scenarios, which may limit us from detecting the fraud in the very early stage. Some of these feature-based algorithms are also efficient and effective to find anomaly users from unlabeled training data like linear models [28], proximity-based models [4], probabilistic models [20] and ensemble models [26].

Graph-based approaches are designed based on the intuition that anomalous entities act synchronously, i.e., they often take similar actions in near-similar times. When representing entities using their actions in a multi-dimensional space, the synchronous behavior of anomalous entities will form blocks with a higher density. Graph-based algorithms aim at identifying these dense-blocks in large-scale behavior logs. These algorithms have been successfully used to predict ill-gotten page Likes [3], zombie followers [16], and spam [15]. Graph-based approaches are more robust when facing adversarial attacking [13], because any strange behavior would unavoidably generate edges in the graph like 'who-buy-where' and 'who-buy-what'. There are two main types of approaches in graph-based algorithms: spectral-based methods and heuristic methods. Usually, spectral-based methods also identify unexpectedly dense regions of the graph and treat related nodes as suspicious. They try to transform the problem of identifying dense regions as the graph partitioning problem. Therefore, there are two different ways to solve the problem. First, by spectral relaxation, graph partitioning can be efficiently solved by SVD or eigenvalue decomposition. SPOKEN [30] observes the 'eigenspokes' pattern produced by pairs of eigenvectors of graphs and is later generalized for fraud detection in [17]. FBOX [31] focuses on the residual of SVD to detect attacks and [15] extends SVD to multimodal data. Several methods have used HITS-like [19] ideas to detect fraud in graphs [5], [10], [12], [16], [8]. Heuristic methods also solve the problem efficiently and effectively. Charikar et al. propose a heuristic method to find subgraphs with a large average degree [6], which shows that subgraph average degree can be optimized with approximation guarantees. Tsourakakis et al.[33] extend [6] to the K-clique densest subgraph problem. Hooi et al. [13] propose Fraudar to deal with camouflages and provide upper bounds on the effectiveness of fraudsters. However, Fraudar detects fraud subgraphs of different sizes, and all nodes in a suspicious subgraph will be marked as fraud nodes. Moreover, Fraudar cannot automatically determine the number of suspicious subgraphs. ENSEMFDET proposed in this paper can handle these drawbacks. At the same time, the parallelism of ENSEMFDET enables it to be more applicable to larger data.

III. Problem Formulation

In this section, we will provide several critical definitions at first, and then formulate the problem we need to handle.

A. Terminology Definition

As an important part of marketing campaigns, e-payment usually provides many promotional campaigns with interesting services like a random discount. For example, each transaction can enjoy a $5 discount if the transaction cost is greater than $5.01, or the discount can be a random value within 5 dollars no matter what the total transaction cost is. We observe that fraudsters register many accounts to make the rules-based anti-fraud system fail. We can usually represent the information of marketing campaigns as a bipartite graph, which can be defined as the 'who buy-from where' graph:

Definition 1 ('who buy-from where' graph): Consider user nodes $\mathcal{U} = \{u_1, ..., u_m\}$, merchant nodes $\mathcal{V} = \{v_1, ..., v_n\}$, and $\mathcal{E}$ which denotes the set of connections between $\mathcal{U}$ and $\mathcal{V}$ representing a purchase, we define that the bipartite graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ is a 'who buy-from where' graph.

Based on the 'who buy-from where' graph, there exist two clues about the fraudsters according to our observations:

- *synchronized behavior*: Fraudsters are limited by the time of promotional campaigns. Basically, Fraudsters need to achieve a specific goal in a short term. Therefore, suspicious nodes have extremely synchronized behavior patterns within a short time, because they are often required to perform similar tasks together such as payments in specific stores.
- *rare behavior*: The connectivity patterns among high suspicious nodes are very different from the majority. Usually, the density of subgraphs composed of highly suspicious nodes is significantly higher than other parts of the full graph.

Through our analysis of historical transaction data on JD.com, it is found that there are usually multiple groups of fraudsters in the same period of promotional campaigns. The fraudsters of different groups are reflected in the 'who buy-from where' graph, that is, there are high-density disjoint subgraphs. We need to find all suspicious groups in a graph.

B. Problem Definition

Based on the 'who buy-from where' graph, instead of identifying the densest subgraph, the target of the fraud detection problem is not only finding the densest subgraph but also extracting all other unexpected dense subgraphs. In this paper, we formulate the fraud detection as finding k-disjoint subgraphs that maximize the sum of densities and propose a simple strategy to select the parameter k automatically and stably. Here, we formally define our problem as follows:

Problem Definition: Given a 'who buy-from where' graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, the fraud detection problem is to find a group of vertex subsets $S = \{S_1, ..., S_{\hat{k}}\}$ and $S_i \subseteq \mathcal{U} \cup \mathcal{V}$. $\hat{k}$ is the selected value of the parameter k, which is determined by the property of the graph G.

Here, we make use of the *density score* to measure the density instead of the average density degree. According to [13], given a graph G, its density can be effectively measured with the following *density score*:

Definition 2 (Density Score): Let the density score $\phi(G)$ to be

$$\frac{1}{|\mathcal{U}| + |\mathcal{V}|} \left(\sum_{j \in \mathcal{V}} \frac{1}{\log(d_j + c)} \right)$$

where d_j denotes the node degree of the j_{th} merchant, and c is a small constant to prevent the denominator from becoming zero.

The reason for penalizing high-degree merchant nodes in *density score* is to keep the detection effective, even in the face of camouflage. The more detailed explanation can reference to [13] as well.

Based on the graph *density score* measure, the objective function of the proposed fraud detection problem can be represented as :

$$\max \sum_{i=1}^{\hat{k}} \phi(G(S_i)) \;\; s.t. S_l \cap S_m = \emptyset \;\; l \neq m \;\; l, m \in \{1, 2, \cdots, \hat{k}\}. \tag{1}$$

Here, $G(S_i)$ represents a subgraph which is composed of the vertex subset S_i. It is not hard to prove that Equ. 1 is a NP-hard problem [2].

IV. Proposed Method

In this section, we introduce the proposed model EnsemFDet. At the very beginning, we introduce three structural sampling methods with theoretical guarantees. After that, the fraud detection algorithm FDet will be proposed which can be conducted for all sampled graphs in parallel. Finally, we will describe the ensemble approach EnsemFDet from a holistic perspective.

A. Sampling methods for bipartite graph

In e-commerce business platforms, the amount of transaction data is normally huge, and the response time for fraud detection is extremely demanding. In order to make our approach applicable to large scale data in the real world, we make use of sampling methods to decompose the fraud detection problem in a large scale graph into multiple smaller scale graphs which can be solved simultaneously through the parallel implementation. In this section, we will introduce multiple sampling choices for bipartite graphs in detail.

1) Why sampling can work: Most of the real-world graphs are still too large to efficiently acquire, store and process. In order to facilitate the development and testing of systems in network domains, it is often necessary to sample smaller subgraphs from the larger network structure [24]. There are two different sampling strategies: node sampling and edge sampling. In classic node sampling, nodes are chosen independently and uniformly at random from the original graph for inclusion in sampled graphs. All edges among the sampled nodes are added to form the sampled graph. But for a bipartite graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, $\mathcal{U}$ is not as the same type as $\mathcal{V}$, thus we consider two different sampling strategies further:

one-side node sampling and *two-side node sampling*. Edge sampling focuses on the selection of edges rather than nodes to build sampled subgraphs. Thus, the node selection step in edge sampling algorithm proceeds by just sampling edges, and including both nodes when a particular edge is sampled. The subgraph is created just out of the sampled edges, which means no extra edges are added in addition to those chosen during the random selection process. We provide Theorem 1 to prove we can get a $\epsilon - approximation$ solution of Equ. 2 if we use the sampled subgraphs.

Theorem 1 Given an bipartite undirected graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, let n be the number of vertices in the graph and $c = \Omega(\ln n)$ be the minimum node-degree. We sample edges of G with probability $p = \frac{3(d+2)\ln n}{\epsilon^2 c}$ independently to construct a subgraph $G(S_i)$ and set their weights in $G(S_i)$ as the original multiplied by $\frac{1}{p}$. Then $G(S_i)$ is an $\epsilon - approximation$ of G under the density metric [9]:

$$(1-\epsilon)\phi(G(S_i)) < \phi(G(S_i)) < (1+\epsilon)\phi(G(S_i)) \quad (2)$$

Let $f_D(q)$ be the number of nodes of degree q in the original graph. Let $|\mathcal{V}_s|$ be the target number of nodes in the sample graph. Let $p_v = \frac{|\mathcal{V}_s|}{|\mathcal{V}|}$ be the probability of sampling a node in Node Sampling (NS). Let $|\mathcal{E}_s|$ be the number of sampled edges in Edge Sampling (ES), then $p_e = \frac{|\mathcal{E}_s|}{|\mathcal{E}|}$ is the probability of sampling a particular edge in ES. Let $E_*[d_q]$ refers to the expected number of sampled nodes that have degree $d = q$ in the original graph, where $*$ refers to different sampling methods like following:

$$\begin{aligned} E_{NS}[d_q] &= f_D(q) \cdot p_v \\ E_{ES}[d_q] &= f_D(q) \cdot [1 - (1-p_e)^q] \end{aligned} \quad (3)$$

The above equations show each node has a uniform probability of being sampled in NS. For ES, the probability of selection is proportional to a node's degree. More specifically, the likelihood of selection corresponds to the complement of the probability that none of the node's q edges is sampled. Thus ES selects high degree nodes with greater probability than NS. Formally, we have:

Lemma 1: For degree $q > \frac{\log(1-p_v)}{\log(1-p_e)}$, ES will sample degree q nodes at a higher rate than NS (i.e., $E_{ES}[d_q] > E_{NS}[d_q]$).

2) Random Edge Sampling (RES): *RES* is the most intuitive and simplest choice of sampling methods in a bipartite graph. The random edge sampling can be conducted with the following steps:

1. Select a random set of edges $\mathcal{E}_s$ from $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ with the sample ratio $S = |\mathcal{E}_s^i|/|\mathcal{E}|$. Collecting nodes connected by edges in $\mathcal{E}_s^i$ into sets $\mathcal{U}_s^i$ and $\mathcal{V}_s^i$ separately.
2. Based on $\mathcal{E}_s^i$, $\mathcal{U}_s^i$ and $\mathcal{V}_s^i$, we can construct the random sampled subgraph $G_s^i = (\mathcal{U}_s^i \cup \mathcal{V}_s^i, \mathcal{E}_s^i)$.

It's obvious that RES can sample subgraphs evenly and control the time consumption of each subgraph is similar. Based on Lemma 1, RES will sample nodes with larger degrees at a higher rate. What's more, from the view of the spectral, RES is likely to select dense components from the original bipartite graph with the rise of sample ratio S. Based on the problem we face, we know that such dense components have a greater likelihood of containing fraudulent nodes. At the same time, the sparse part will not be considered in the process, thus the method prunes low-risk nodes and reduces the hypothetical space during the procedure of ensemble construction.

3) One-side Node sampling (ONS): A notable difference between unipartite graphs and bipartite graphs is the node type. Nodes constituting a unipartite graph only belong to one type, but a bipartite graph has two types of nodes which can be seen as two sides of the graph. Obviously, *ONS* is a straightforward way to process the sampling of a bipartite graph. *ONS* can be operated as the following steps:

1. Construct the adjacency matrix $W \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{V}|}$ for the original bipartite graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$.
2. Determine the node type to sample. Here, assuming $\mathcal{U}$.
3. According to the sample ratio $S = |\mathcal{U}_s^i|/|\mathcal{U}|$, sampling $|\mathcal{U}_s^i|$ rows from W to form the subgraph adjacency matrix W_s^i,
4. Construct the random sampled subgraph G_s^i based on $\mathcal{W}_s^i$.

From the steps above, we can find that determining which side nodes to sample is a non-trivial option for *ONS*. Here, we have summed up some paradigms through practice.

- *Task-oriented principle*: First of all, we insist that the decision should depend on the problem to solve. For instance, if to deal with the link prediction or closeness measurement problems for one-side nodes, we should sample the target side nodes in order to measure the closeness with the support of global and complete information from the other side. Meanwhile, with the rise of sampled subgraphs, no node pair will be missed. However, when facing problems relating to dense subgraph detection, it's another kind of situation where we should take other properties into consideration like the specific topology. We analyze the situation in the next bullet point.
- *Retain topology*: We still take the problem relating to dense subgraph detection as an example. Here, we have a bipartite graph with $D_{avg}(\mathcal{V}) \gg D_{avg}(\mathcal{U})$ where $D_{avg}(\mathcal{U})$ is the average degree of $\mathcal{U}$. In response to the task requirement, we care about retaining the topology of dense components from the original graph after sampling. Obviously, if we sample $\mathcal{U}$, the sampled subgraph will become a subgraph of uniform distribution, especially when $D_{avg}(\mathcal{U}) \sim 1$. In contrast, sampling $\mathcal{V}$ can retain dense topology from the original graph effectively. Because once $v \in \mathcal{V}$ with a high degree is sampled, the dense components can be extracted.

ONS can conquer disparate structures more effectively. Considering dense subgraphs problem, it's more effective to select denser subgraphs from relatively dense ones after sampling, and subgraphs that are not very significant to be selected with the global setting become distinct on sampled graphs.

4) Two-sides Node Sampling (TNS): It's hard to avoid thinking about what if we conduct sampling to both sides of the bipartite graph. In fact, it's a fairly intuitive idea. From the view of the adjacency matrix, it is equivalent to sampling both rows and columns of W and using the cross-section of

these rows and columns to construct the adjacency matrix W_s^i. Hence the detailed sampling steps are similar to *ONS*.

Here we need to remind a point to better understand the number of sampled subgraphs N and the sample ratio S. Because we sample both sides of the bipartite graph, the sampled graph is smaller than RES or ONS with the same sample ratio S. In fact. when $S = 0.1$, the size of a subgraph from random edge sampling is 10% of the original one, but the subgraph sampled from two sides is only near S^2 of the original graph. So in practice, we usually need to enlarge the sampling ratio S or increase the number of samples N to ensure the effectiveness of the two-sides sample. However, considering the parallel implementation of ENSEMFDET, with the rise of N, more computing resources need to be occupied at the same time which may be unacceptable in real-world scenes. Besides, it's easy to understand that *TNS* will preserve the structure of the graph more finely and make it difficult to preserve specific topology as needed like *ONS*, which is an aspect need to be considered.

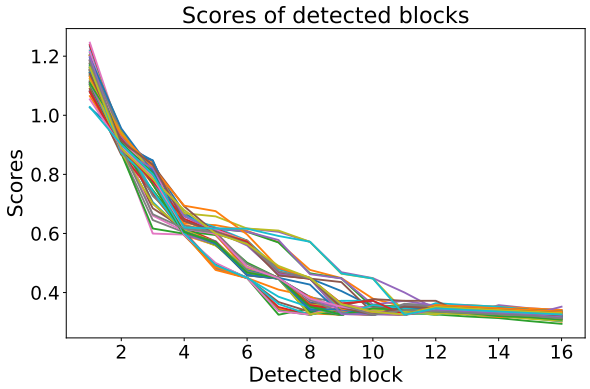


Fig. 1. Scores for each detected block.

B. *Fraud DETection(*FDET*) algorithm based on Heuristic*

The sampling methods are able to decompose the large scale graph into multiple smaller scale graphs. Now, we propose FDET to complete fraud detections for each sampled subgraph. As we mentioned in Section III-B, the fraud detection task in our realistic e-commerce scene is a disjoint case. We start from considering one natural heuristic for the disjoint case: at each step, we compute the densest subgraph $G(S_i) = (S_i, \mathcal{E}_i)$ in the current graph G. Formally, to achieve $G(S_i)$, nodes in G which result in the highest value of *density score* ϕ defined in Definition 2 will be repeatedly obtained. Then we remove edges in previously detected subgraphs from the current graph G. Iteratively searching the dense subgraph until we find the $\hat{k}_{th}$ subgraph or the current graph contains no edges.

How to determine $\hat{k}$ is a very important issue. We insist that $\hat{k}$ should be a parameter that exists objectively depending on how many dense subgraphs exist. Therefore, when detecting dense subgraphs, we have to truncate the detection process effectively instead of setting a number or the more the better. About truncating effective dense components, the basic idea

Algorithm 1: FDET for Equ. 1

Data: Bipartite Graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, density metric ϕ
Result: $S_d = (\mathcal{U}_d \cup \mathcal{V}_d)$

$\mathcal{U}_d \leftarrow \emptyset$, $\mathcal{V}_d \leftarrow \emptyset$;
repeat
  $n \leftarrow |\mathcal{U}| + |\mathcal{V}|$;
  $H_n \leftarrow G$;
  for $i = n : 2$ **do**
    find minimal degree node m in H_i;
    $H_{i-1} \leftarrow H_i - \{m\}$;
  $G(S_i) = (\mathcal{U}_i \cup \mathcal{V}_i, \mathcal{E}_i) \leftarrow$ $\arg\max_{H_i \in \{H_2, \ldots, H_n\}} \phi(H_i)$;
  $\mathcal{U}_d \leftarrow \mathcal{U}_d \cup \mathcal{U}_i$;
  $\mathcal{V}_d \leftarrow \mathcal{V}_d \cup \mathcal{V}_i$;
  remove $\mathcal{E}_i$ from G;
until $\arg\min_i \Delta^2\phi(G(S_i))$ *and* $G \neq \emptyset$;

behind partitioning methods, such as k-means clustering [14], is to define clusters such that the total intra-cluster variation or total WSS (within-cluster sum of square) is minimized. The total WSS measures the compactness of the clustering, and we want it to be as small as possible. The Elbow method [18], [32] treats the total WSS as a function of the number of the clusters: one should choose a number of clusters so that the total WSS doesn't improve significantly while adding another cluster. In this paper, we employ a similar idea to select the $\hat{k}$. We see the total density measure $\sum_i^{\hat{k}} \phi(G(S_i))$ as a function of $\hat{k}$: one should choose a number so that $\sum_i^{\hat{k}+1} \phi(G(S_i))$ doesn't improve much better. In other words, $\Delta^2\phi(G(S_i))$ which is the second-order finite difference of $\phi(G(S_i))$ can be utilized to measure the improvement of $\sum_i^{\hat{k}} \phi(G(S_i))$, and $min(\Delta^2\phi(G(S_i)))$ represents the density score ϕ suddenly decreases. Therefore, we can define *Truncating Point* as:

Definition 3 (Truncating Point):

$$\hat{k} = \arg\min_i \Delta^2\phi(G(S_i))$$

We plot the curve of $\phi(G(S_i))$ for multiple sampled graphs as shown in Figure 1, and one line represents one sampled graph. It can demonstrate this method is able to find a reasonable hyperparameter $\hat{k}$ because all curves are monotonically decreasing and achieve a similar low score finally which means detected subgraphs are meaningless after the truncating point $\hat{k}$. In the experiments, we will compare the performances between a fixed k and the optimal $\hat{k}$.

By using minimal heap [13], each update can be performed in $O(\log(|\mathcal{U}| + |\mathcal{V}|))$ time, totaling $O(\hat{k}|\mathcal{E}|\log(|\mathcal{U}| + |\mathcal{V}|))$ time because we need $|\mathcal{E}|$ updates to node degrees and repeat $\hat{k}$ times. We show FDET in Algorithm 1. Typically, the number $\hat{k}$ of groups varies from few to few tens. The required number of components $\hat{k}$ is often expected to increase with the network size. However, based on sampling methods, the large scale graph can be decomposed into multiple smaller scale graphs. Besides, the algorithm FDET is able to run on

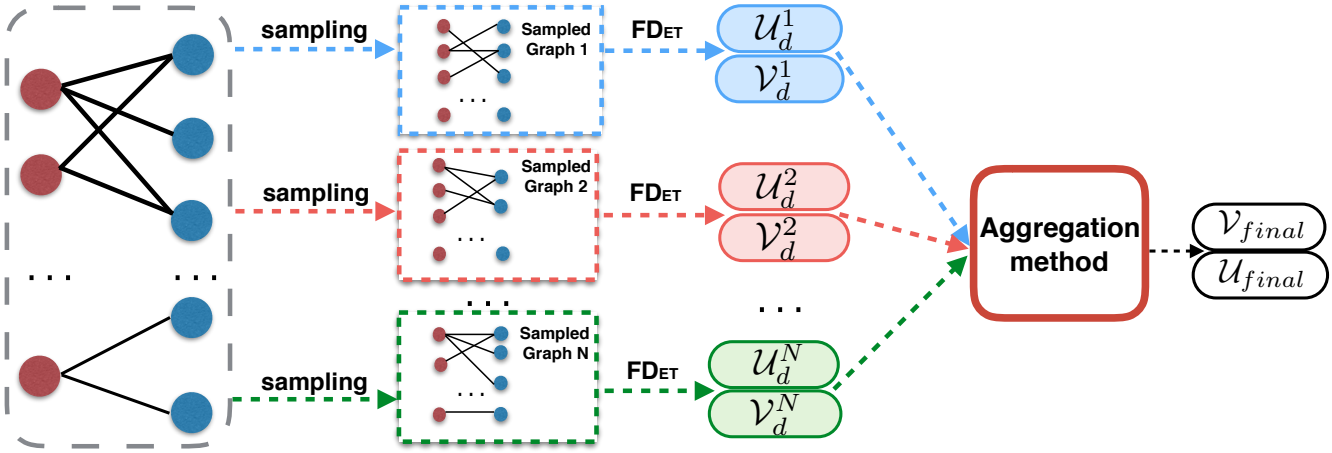


Fig. 2. The Structure of EnsemFDet

each sampled subgraphs in parallel. After describing graph sampling methods and the fraud detection algorithm in each subgraph, we will introduce our proposed model EnsemFDet from a holistic view.

C. Ensemble based Framework EnsemFDet

EnsemFDet is a general fraud detection method and can be applied to detect various fraud behaviors in real-world scenes. The structure of EnsemFDet is shown in Figure 2. Meanwhile, the pseudo-code of EnsemFDet is available in Algorithm 2.

In EnsemFDet, the process before applying the aggregation method can be implemented in parallel. After sampling, we will apply FDet to all sampled graphs simultaneously with the multicore environment. The parallel property of EnsemFDet is very meaningful and practical because there are high demands for running time in the real-world where graphs are very huge in size. Then EnsemFDet applies the aggregation method to the results from all sampled graphs to get the final fraud detection output. In experiments, we choose the majority voting aggregation method as defined:

Definition 4 (Majority Voting Aggregation Method (MVA)): The majority voting aggregation method can be described by this equation:

$$H(u) = \begin{cases} accept & \text{if } \left(\sum_{i=1}^{N} h_i(u)\right) \geq T \\ reject & \text{otherwise.} \end{cases}$$

where $h_i(u)$ is the number of the vote-catching of the node u in sampled subgraph G_s^i, and the parameter T is a threshold.

The parameter T normally is determined by task requirements and the preference for detection results. Although the heuristic algorithm FDet is able to solve the fraud detection based on bipartite graphs with near-linear scalability, it is still unable to control the size of dense subgraphs which may limit its practicality. The threshold T is able to handle the quantity of detected suspicious nodes, and we will analyze the impact of T in Section V-D. In fact, the aggregation methods are flexible and can be set as the one suitable for the specific requirement.

Algorithm 2: EnsemFDet

Input: Bipartite Graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$; Density score metric ϕ; Sample Method $\mathcal{M}$; Number of sampled graphs N; Sample ratio S; Voting threshold T; Majority Voting Aggregation Method MVA;
Output: Detected sets of fraud nodes $\mathcal{U}_{final}$ and $\mathcal{V}_{final}$

```
U_d ← ∅, V_d ← ∅;
begin run in parallel
    for i ∈ {1, 2, ..., N} do
        Apply Sample Method M to G with S and get G_s^i;
        Apply FDet to G_s^i and get S_d^i = (U_d^i ∪ V_d^i);
        U_d ← U_d ∪ U_d^i;
        V_d ← V_d ∪ V_d^i;
U_final ← ∅, V_final ← ∅;
Apply MVA to U_d, V_d with the voting threshold T;
for u ∈ U_d do
    if H(u) = accept then
        U_final ← U_final ∪ u
for v ∈ V_d do
    if H(v) = accept then
        V_final ← V_final ∪ v
```

V. Experiments

To demonstrate the effectiveness, stability, and scalability of EnsemFDet, extensive experiments are conducted in three datasets that come from real transaction data on **JD.com**. In this section, we will describe datasets in detail at first. Then the experimental settings, including experimental setup, evaluation metrics, and comparison methods, will be introduced. Finally,

we will display the experimental results together with the parameters impact analysis.

A. Datasets Description

TABLE I
STATISTICS OF DATASETS

Dataset#	Node:PIN	Fraud PIN	Node:Merchant	Edge
1	454,925	24,247	226,585	1,023,846
2	2,194,325	16,035	120,867	2,790,517
3	4,332,696	101,702	556,634	7,997,696

Datasets we used in experiments are based on real transaction data on **JD.com**, one of the world's largest e-commerce business platforms. A set of policies, rules, and models will determine levels of risk of all transactions on **JD.com**, and a subset of them with relatively high risk will be sent for additional manual checking. A team consisting of experienced experts in manual checking will review those transactions carefully and determine if they should be rejected. Once transactions are rejected, the *PIN* of users participating transactions will be marked as dangerous and be recorded in the Blacklist. The Blacklist is used as the ground-truth for evaluating fraud detection algorithms. After cleaning up the original mass transaction data, we get *PIN-Merchant* bipartite graph ('who buy-from where' graph) which describes the trading relationship between users and merchants. For these three datasets, we also have three corresponding blacklists that contain dangerous *PIN* of users as the ground-truth. These three datasets actually have to be independent of each other, because they are collected from different time periods, and the business scene we face is extremely time-sensitive. For example, one *PIN* appears in three datasets, but only in one dataset, it is marked as black which may be due to the theft of accounts. And later because of some operations like appeals. the *PIN* can be removed from the blacklist. The key statistical data describing datasets can be found in Table I. We hold the view that in response to this kind of practical problem, experiments conducted on the datasets from the real-world are the most valuable and convincing.

B. Experimental Settings

1) Experimental Setup and Metrics: In the experiments, ENSEMFDET can be divided into three steps and conduct them step by step just like the description in Section IV-C. The key parameters are summarized in Table II , and extensive experiments with different combinations of parameters are conducted in order to discover interesting impacts.

The main goal of our experiments is to compare the quality and quantity of the detected fraud nodes, thus we can choose conventional evaluation metrics to measure the performance. The methods we test in experiments can all output detected fraud nodes, thus we can use F1, Recall, Precision as evaluation metrics. It should be noted that normally Accuracy in fraud detection problems seems not very significant, because the proportion of fraud samples is quite low.

TABLE II
PARAMETERS USED IN EXPERIMENTS

Parameters	Descriptions
N	Number of sampled graph
S	Sample ratio
T	Voting threshold in aggregation method
R	The repetition rate $R = S \times N$

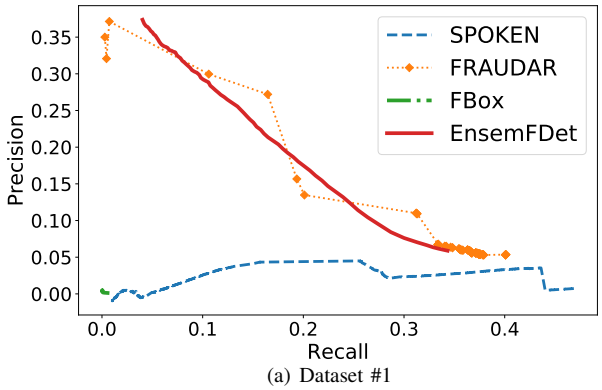
2) Comparison Methods: The methods used in experiments are listed as following:

- **ENSEMFDET**: ENSEMFDET is the model proposed in this paper. We aim at demonstrating the practicability and stability of ENSEMFDET and the advantage of the ensemble framework in time consumption without loss of detection performance in our experiments.
- **ENSEMFDET-FIX-K**: In ENSEMFDET-FIX-K, we fix the number of detected blocks K instead of truncating the detecting process automatically which is described in Section IV-B. The method is used to verify the effectiveness of the truncation.
- **SPOKEN**: By spectral relaxation, SPOKEN [30] is able to find the densest density regions with SVD method. SPOKEN is not parameter-free to make sure how many components should be used to estimate the suspicious nodes. So in our experiments, the number of components is set to 25 as same as the paper described.
- **FBOX**: FBOX[31] analyzes the reconstruction error and shows attacks of small enough scale cannot be efficiently detected in the top-k SVD components. FBOX needs to set the parameter K which is a determinant factor of the reconstruction error.
- **FRAUDAR**: FRAUDAR [13] tries to find a unexpected dense subgraph which commonly contains high suspicious users.

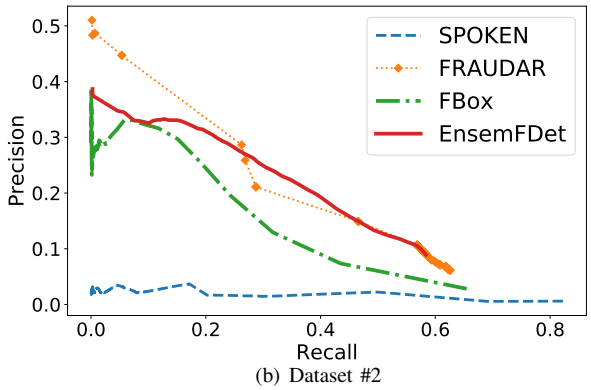
Based on fair considerations, as an unsupervised learning method, we will not compare with supervised learning methods, such as some feature-based models and emerging GNN-based models [34], [27], [29].

C. Experiment Result

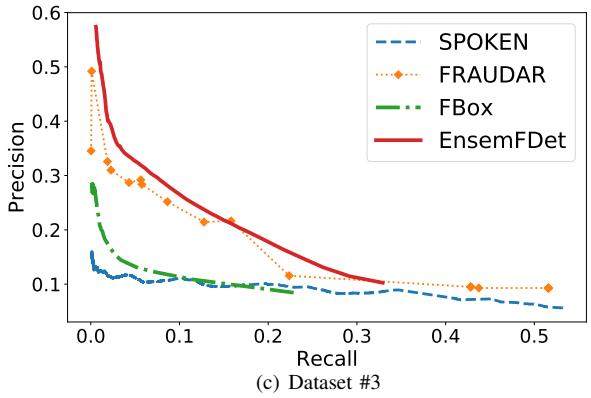
1) Evaluation on Comparison Methods: The experimental results of all comparison methods in three datasets are shown in Figure 3. We can find that methods based on SVD, including SPOKEN and FBOX, are not able to keep a stable performance in different datasets. Especially for FBOX, almost completely invalidated on No.1 Dataset: Precision and Recall are close to 0, but it works on the other two datasets. It's obvious that FRAUDAR and ENSEMFDET can get better performance in all datasets. Because the performance of FRAUDAR can not be represented with a continuous curve, we use diamond points to represent experimental results of FRAUDAR. Overall, ENSEMFDET has close performance compared with FRAUDAR. let's make more detailed comparison from Figure 4 where we set S as 0.1 and N as 80. The number of detected nodes is used in the x-axis of Figure 4, because in response to the algorithms' task, the performance has a



(a) Dataset #1



(b) Dataset #2



(c) Dataset #3

Fig. 3. Performance comparison of different methods

comparative value when EnsemFDet and Fraudar detect the equivalent fraud nodes. From Figure 4(a)- 4(c), we can find EnsemFDet and Fraudar have similar performance in F1. However, what we are trying to express from these figures is not only a similar performance but also the practicability of EnsemFDet. EnsemFDet shows a smooth curve on Figure 4, because the numbers of fraud nodes being detected in EnsemFDet are almost continuous with the control of T.

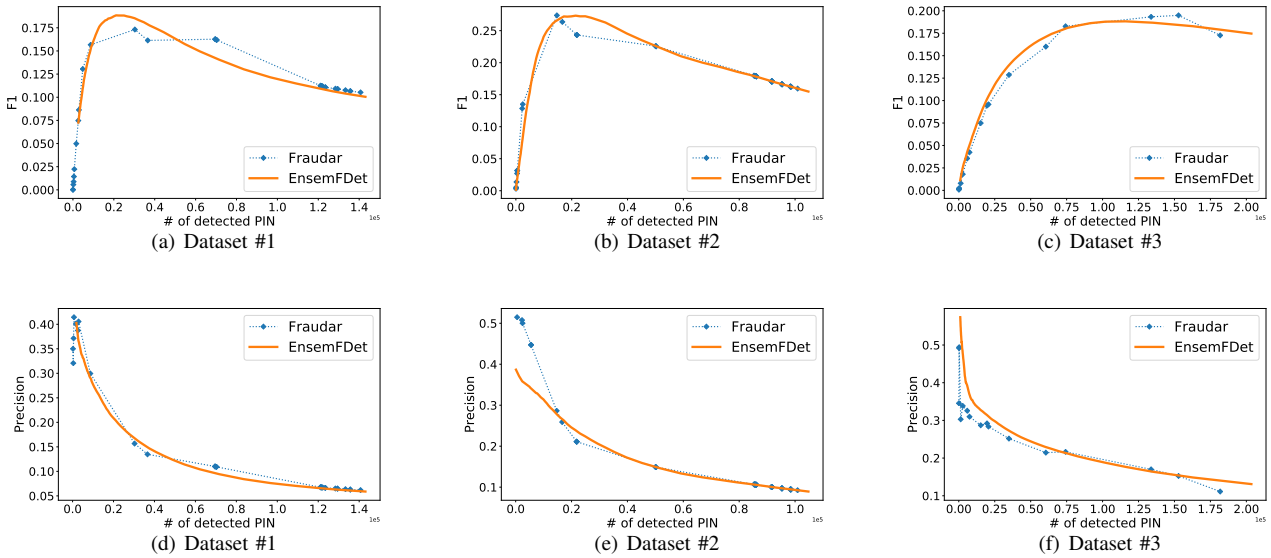
TABLE III
The comparison of time consumption between EnsemFDet and Fraudar

	Dataset #1	Dataset #2	Dataset #3
EnsemFDet	74.127 sec	162.102 sec	470.508 sec
Fraudar	805.533 sec	2365.659 sec	5681.591 sec

In comparison, the numbers of detected nodes in Fraudar are marked with diamond points, and it's obvious that their connection is a polyline. This means that the quantities of detected nodes Fraudar are very discrete and the number of nodes in the detected blocks is unstable so that we are unable to control the number of nodes being detected and select one point that best fits our requirements on the curve. Besides, on Figure 4(e) we can notice that there are several marks of Fraudar have apparent advantages compared with EnsemFDet, but from Figure 4(b) this kind of advantage disappears when taking F1 into consideration. It reflects Recall is very too low to be used in the real world. And at the next point, Fraudar is going to be weaker than EnsemFDet, which spans almost $20,000$ nodes. Normally, such a huge span is unacceptable in the business, but EnsemFDet can be used through the entire curve which makes it practical in the face of real-world scenes.

In terms of time consumption, we compared the running time of EnsemFDet and Fraudar which are both heuristic methods. Table III shows the running time of experiments where $S = 0.1$, $N = 80$ for EnsemFDet, and K is fixed as 30 for Fraudar. Fraudar and EnsemFDet both runs near-linearly in the input size which verifies and ensures the scalability. Because EnsemFDet can be parallel and make use of the truncation strategy to decrease the number of detected blocks. EnsemFDet is 10X faster than Fraudar. In theory, $Time(\text{EnsemFDet}) < S \times Time(\text{Fraudar})$ which is demonstrated in our experiments. For the smallest $S = 0.01$ we tested, EnsemFDet can be 100x faster due to its parallelism.

2) Comparison among Sampling Methods : In Section IV-A, we analyzed several sampling methods for a bipartite graph, and here experiments are conducted to demonstrate our strategy and analysis. The Precision-Recall curves in Figure 5 come from the experiments conducted on the No.3 dataset where $S = 0.1$ and $R = 8$. At first, from Figure 5 we can find the performance of *Node PIN Bagging* (apply *ONS* as the sampling method in EnsemFDet) is worst, but the curve of *Node Merchant Bagging* is much better. In fact, the performance verifies our analysis in Section IV-A3. In this dataset, the graph with $D_{avg}(Merchant) \gg D_{avg}(PIN)$ leads to the failure of retaining the dense topology from the original graph effectively by *Node PIN Bagging*. Conversely, *Node Merchant Bagging* can achieve better performance with the support of keeping critical topology. However, some sampled graphs



(a) Dataset #1 (b) Dataset #2 (c) Dataset #3

(d) Dataset #1 (e) Dataset #2 (f) Dataset #3

Fig. 4. Performance and properties Analysis between EnsemFDet and Fraudar

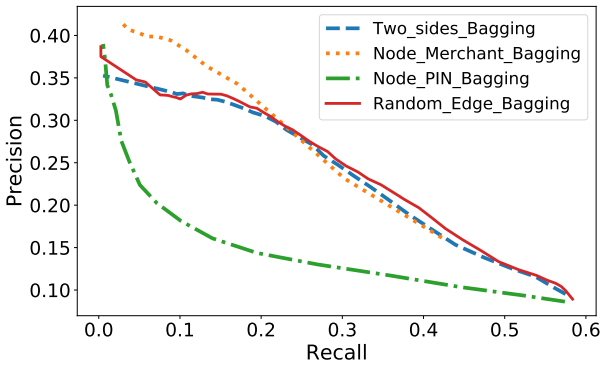


Fig. 5. Performance comparison among different sampling methods in EnsemFDet.

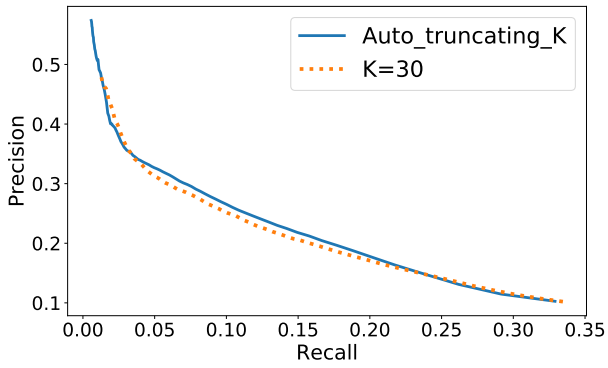
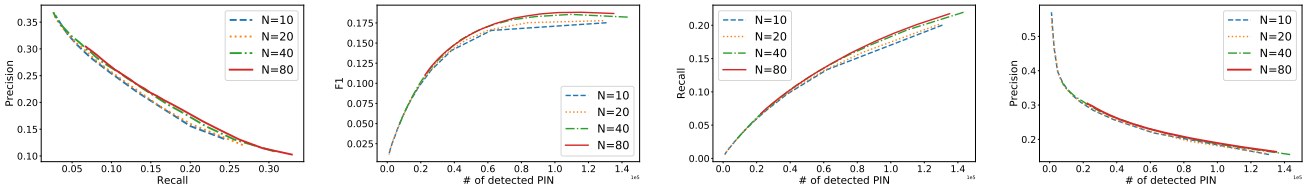


Fig. 6. Performance comparison between EnsemFDet and EnsemFDet-Fix-K.

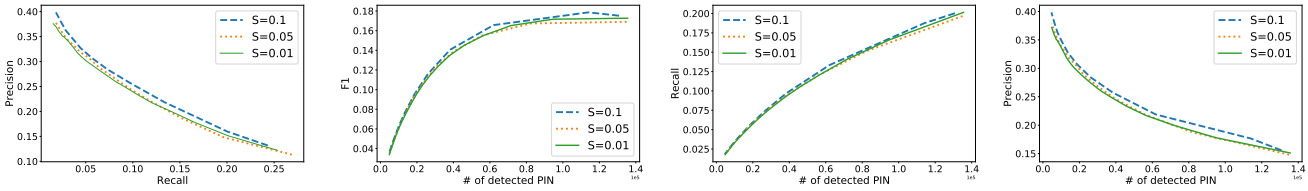
coming from *Node Merchant Bagging* can be very large in size due to some nodes with a very high degree. Although we set S as 0.1, the size of some sampled graphs can reach 30% of the original one which results in the higher time-consumption of parallel computation. Under this circumstance, we can discover *Node Merchant Bagging* has better performance. Besides, the similar and stable performance of *Node Merchant Bagging*, *Two-sides Node Bagging* and *Random Edge Bagging* reflects the stability of EnsemFDet to a certain extent.

3) Verification of the truncating point: Figure 6 displays the results from a comparative experiment between EnsemFDet and EnsemFDet-Fix-K where the number of detected blocks k is fixed instead of truncating the detecting process automatically. We set $k = 30$ for EnsemFDet-Fix-K, and EnsemFDet is based on the truncating point. In experiments, we also record the detected blocks number of EnsemFDet, and all of the records are smaller than 15. The performance expressed by the Precision-Recall curve shows EnsemFDet can achieve better outcomes than EnsemFDet-Fix-K. Although EnsemFDet-Fix-K can get higher Recall with the increase of k, actually Precision has been close to random selection. This kind of high Recall is meaningless and these blocks after truncation are not of value as we defined before. Therefore, the comparison verifies the effectiveness of the truncation strategy which can even level up the performance in Precision. What's more, the time-consumption has also been greatly reduced, because EnsemFDet only need to detect less than half of k sets for EnsemFDet-Fix-K in experiments.



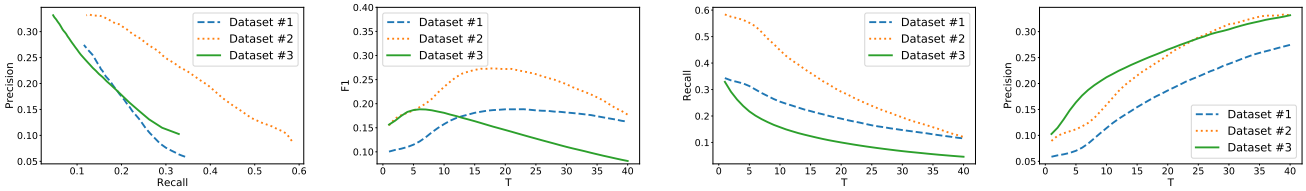
(a) Precision-Recall curve (b) F1 (c) Recall (d) Precision

Fig. 7. Performance Analysis under different N when $S = 0.1$



(a) Precision-Recall curve (b) F1 (c) Recall (d) Precision

Fig. 8. Performance Analysis under different S when fixing $S \times N = 1$.



(a) Precision-Recall curve (b) F1 (c) Recall (d) Precision

Fig. 9. Performance Analysis under different T when fixing $S = 0.1$ and $N = 80$.

D. Impacts of various parameters

In this section, we evaluate the impacts of parameters shown in Table II. We conducte experiments, where the sampling method is fixed as *RES*, in all three datasets, but only show the results on the No.3 dataset for the reason of the pages limitation when analyzing N and S. It should be emphasized that the impacts are consistent across the three datasets in our experiments.

1) Impact of N: In order to evaluate the impact of the parameter N, we fixed S as 0.1, and N changed within {10, 20, 40, 80}. The Precision-Recall curve, F1, Precision and Recall are displayed in Figure 7. Here we have to explain why the number of detected nodes is used in the x-axis of Figure 7(b)- 7(d). Obviously, we doesn't mention the control of the last parameter T consistent in the comparative experiments. In fact, when the parameter N is not the same, the consistent T in not reasonable instead, because in this case there is a huge difference in the total number of votes behind the same T. Thus for the sake of fairness, we compare performance when EnsemFDet detects the equivalent fraud nodes under different N.

The results show that EnsemFDet achieves better performance with a rise of N. In fact, this elevation of performance comes directly from the nature of the bagging method. However, we should note that the improvement in performance is not significant, especially with the increase of N. When comparing $N = 40$ and $N = 80$, we can find the improvement has become negligible. At the same time, the rise of the cost of equipments is enormous which is not a fair trade-off. In addition, we can also discover that EnsemFDet is very stable when $N \in \{10, 20, 40, 80\}$ which means the repetition rate R is between 1 and 8 times. The stability actually makes EnsemFDet has loose requirements for the computational environment: even if there are not enough parallel computing cores, EnsemFDet can still achieve relatively stable and acceptable performance.

2) Impact of S: To evaluate the impact of S, we fixed $S \times N = 1$, instead of setting N as a constant value. The reason why we choose such an experimental setting is that the same repetition rate R is fairer for points and edges in the bipartite graphs. Besides, R itself is determined by the S and N, so after the analysis of S and N, no additional

analysis is needed for the dependent variable R. The Figure 8 shows the Precision-Recall curve, F1, Precision and Recall in the experiments with $S \in \{0.01, 0.05, 0.1\}$. From Figure 8(a)-8(d), we have two major findings. On the one hand, we can see that the rise of S can bring a certain improvement in performance. We should remind that S will determine the size of sampled subgraphs, and the graphs in the real-world have a very large scale normally. A sampled subgraph with a relatively large scale still challenges the storage structure and the single computing core which is against the original intention of EnsemFDet. Thus we don't pay much attention to the performance improvement with the rise of S. On the other hand, stability is also shown in this set of experiments. When $S = 0.01$, the performance shown in Figure 8 is still close to the one of $S = 0.1$. It means that when facing a large-scale graph structure, the stability of EnsemFDet allows you to sample the graph to a much smaller size without losing a lot of performance. Of course, when sampling large-scale graphs to ones of smaller size, N will increase to keep R constant. But we think this trade-off can be done according to task requirements and equipment conditions. If you are more concerned about time consumption with enough parallel computing cores or the original graph is too large to deal with, you can set a smaller S. Otherwise, if the performance is more critical or the original graph itself is not too large to handle, a relatively large S is a good choice. In fact, parameters S and N let EnsemFDet very flexible enough to adapt to a variety of scenarios rather than make EnsemFDet complicated to be manipulated.

3) Impact of T: The results shown in Figure 9 come from the set of experiments with $S = 0.1$, $N = 80$ and $T \in \{1, 2, \dots 39, 40\}$. Obviously, the experiment results show that Precision would go up and Recall would drop with the rise of T. The phenomenon is easy to understand because the fraud nodes with more votes are equivalent to having a higher risk in multiple sampled graphs. Meanwhile, the number of detected nodes will decrease with the increase of the threshold T which necessarily leads to the fall of Recall. We can find the curves are smooth and monotonous in Figure 9(c)- 9(d) which is a nice property that can be exploited. Based on the curves, we can determine T in response to the task requirements: do we prefer to reduce the detecting error rate or to try to find the fraud nodes as many as possible. In this case, we have a definite direction when we need to tune the parameter T.

VI. Conclusion

In this paper, we propose an ensemble approach EnsemFDet to solve the fraud detection problem in large graphs. First, we formulate the optimization problem for fraud detection according to our business scenarios. EnsemFDet scales up fraud detection through the ensemble framework, and we analyze the sampling methods in bipartite graphs as well. A heuristic method FDet, which is one critical part of EnsemFDet, is proposed to detect dense subgraphs. The FDet can also speed up the search process through an efficient truncation. Extensive experiments conducted on three real-world datasets demonstrate that EnsemFDet is effective, scalable, and stable. After successful experiments on real data, EnsemFDet has been deployed in the risk control department of JD.com for further tests.

References

[1] Kayode Sakariyah Adewole, Nor Badrul Anuar, Amirrudin Kamsin, Kasturi Dewi Varathan, and Syed Abdul Razak. Malicious accounts: dark of the social networks. *Journal of Network and Computer Applications*, 79:41–67, 2017.

[2] Oana Denisa Balalau, Francesco Bonchi, TH Chan, Francesco Gullo, and Mauro Sozio. Finding subgraphs with maximum total density and limited overlap. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 379–388. ACM, 2015.

[3] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22nd international conference on World Wide Web*, pages 119–130. ACM, 2013.

[4] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[5] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 15–15. USENIX Association, 2012.

[6] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 84–95. Springer, 2000.

[7] Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. Antisocial behavior in online discussion communities. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

[8] Brian Davison. Propagating trust and distrust to demote web spam. 2006.

[9] Ruohan Gao, Huanle Xu, Pili Hu, and Wing Cheong Lau. Accelerating graph mining algorithms via uniform random edge sampling. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

[10] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st international conference on World Wide Web*, pages 61–70. ACM, 2012.

[11] Jon Ander Gómez, Juan Arévalo, Roberto Paredes, and Jordi Nin. End-to-end neural network architecture for fraud scoring in card payments. *Pattern Recognition Letters*, 105:175–181, 2018.

[12] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment, 2004.

[13] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–904. ACM, 2016.

[14] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8), 2010.

[15] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. A general suspiciousness metric for dense blocks in multimodal data. In *2015 IEEE International Conference on Data Mining*, pages 781–786. IEEE, 2015.

[16] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Catchsync: catching synchronized behavior in large directed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 941–950. ACM, 2014.

[17] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. Inferring strange behavior from connectivity pattern in social networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 126–138. Springer, 2014.

[18] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996.

[19] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[20] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452. ACM, 2008.

[21] Srijan Kumar, Justin Cheng, Jure Leskovec, and VS Subrahmanian. An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 857–866. International World Wide Web Conferences Steering Committee, 2017.

[22] Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. Vews: A wikipedia vandal early warning system. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 607–616. ACM, 2015.

[23] Srijan Kumar, Robert West, and Jure Leskovec. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web*, pages 591–602. International World Wide Web Conferences Steering Committee, 2016.

[24] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

[25] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2703–2711. ACM, 2019.

[26] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[27] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2077–2085. ACM, 2018.

[28] Junshui Ma and Simon Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1741–1745. IEEE, 2003.

[29] Jongchan Park, Min-Hyun Kim, Seibum Choi, In So Kweon, and Dong-Geol Choi. Fraud detection with multi-modal attention and correspondence learning. In *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–7. IEEE, 2019.

[30] B Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 435–448. Springer, 2010.

[31] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. In *2014 IEEE International Conference on Data Mining*, pages 959–964. IEEE, 2014.

[32] Robert L Thorndike. Who belongs in the family? *Psychometrika*, 18(4), 1953.

[33] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 104–112. ACM, 2013.

[34] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 310–316. ACM, 2019.