Detecting Cash-out Users via Dense Subgraphs

Yingsheng Ji*†
China Etek Service & Technology
Shanghai, China
jiyingsheng@gmail.com

Zheng Zhang*
China Etek Service & Technology
Shanghai, China
zhang.zh0707@gmail.com

Xinlei Tang
China Etek Service & Technology
Shanghai, China
tangxinlei0117@gmail.com

Jiachen Shen
China Etek Service & Technology
Shanghai, China
js2111jiachenshen@gmail.com

Xi Zhang†‡
Beijing University of Posts and Telecommunications
Beijing, China
zhangx@bupt.edu.cn

Guangwen Yang
Tsinghua University
Beijing, China
ygw@tsinghua.edu.cn

ABSTRACT

Cash-out fraud refers to the withdrawal of cash from a credit card by illegitimate payments with merchants. Conventional data-driven approaches for cash-out detection commonly construct a classifier with domain specific feature engineering. To further spot cash-out behaviors in complex scenarios, recent efforts adopt graph models to exploit the interaction relations rich in financial transactions. However, most existing graph-based methods are proposed for online payment activities in internet financial institutions. Moreover, these methods commonly rely on a large amount of online user data, which are not well suitable for the traditional credit card services in commercial banks. In this paper, we focus on discerning fraudulent cash-out users by taking advantage of only the personal credit card data from banks. To alleviate the scarcity of available labeled data, we formulate the cash-out detection problem as identifying dense blocks. First, we define a bipartite multigraph to hold transactions between users and merchants, where cash-out activities generate cyclically intensive and high-volume flows. Second, we give a formal definition of cash-out behaviors from four perspectives: time, capital, cyclicity, and topotaxy. Then, we develop ANTICO, with a class of metrics to capture suspicious signals of the activities and a greedy algorithm to spot suspicious blocks by optimizing the proposed metric. Theoretical analysis shows a provable upper bound of ANTICO on the effectiveness of detecting cash-out users. Experimental results show that ANTICO outperforms state-of-the-art methods in accurately detecting cash-out users on both synthetic and real-world banking data.

CCS CONCEPTS

• **Social and professional topics** → **Financial crime**; • **Information systems** → **Network data models**; • **Computing methodologies** → **Anomaly detection**.

KEYWORDS

finance, cash-out fraud, anomaly detection, dense subgraph

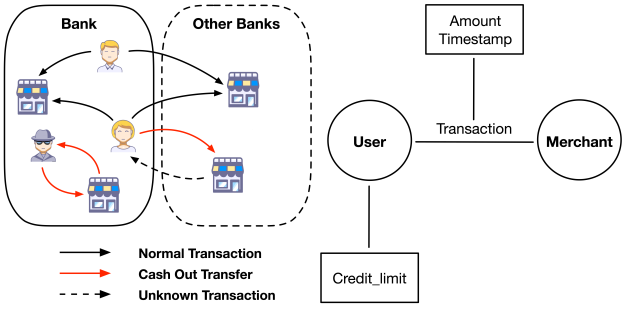
*Equal contributions.
†Corresponding authors.
‡Key Laboratory of Trustworthy Distributed Computing and Service (MoE).

1 INTRODUCTION

As a major fraud in credit card services, cash-out is to withdraw funds within the personal credit limit by illegitimate means. In a typical case, the user extracts cash from a credit card through fake transactions at point-of-sales (POS) machines. Fraudulent merchants or third-part agencies provide fictitious POS purchase and cash transfer for commission gains. Comparing with the bank loan, cash-out is charged extremely low. This is because no interests are generated within a billing cycle. Hence, cardholders with capital demands have strong incentives to engage in such unlawful activities. Massive cash-out frauds may increase joint debt risks and ulteriorly incur substantial default losses for banks, which raises the significance of fraud-fighting.

Anti-fraud advances and fraudsters hold a shifting contest over a long time. Intuitive attempts build a repository of expert knowledge [10, 30]. Mandatory rules (such as blacklist, compliance) are practical in detection, while considerable rules are empirical and inefficient for ever-evolving frauds. To address such limitations, traditional efforts perform subtle feature engineering and train machine learning [4, 35, 38] or further deep learning classifiers [20, 26] on large historical data. It requires massive manually labeled data to learn a robust model, whereas verified frauds are always scarce in real-world banks. Recently, a few works identify financial frauds by using graph-based methods. The key idea is to exploit rich multi-party interactions involved in financial data. However, most of these studies are proposed for online payments in internet financial companies [15, 16, 25, 37, 40], while few works have concerned on credit card services in commercial banks. Note that they are quite different scenarios, especially in the amount and types of available data, where online services can provide more user data, such as login, review, location. How to spot the cash-out behaviors with limited available banking data is less explored in existing studies.

Given a graph of credit card transactions, effectively detecting a group of users that are engaged in cash-out activities are challenging due to the following reasons. First, cash-out usually involves a chain transfer from sources to untraceable destinations. This is because most fraudsters manipulate inter-bank transactions to evade the direct detection, as shown in Figure 1a. Second, fraudsters explore diverse sophisticated patterns. For example, the agency services, called perfect-bill, can offer highly imitation daily expenses for cash-out purpose, which makes it indistinct from ordinary bills. Third, it is assumed that only credit card data (transactions and



(a) Banking scenario (b) Graph schema

Figure 1: A toy example of credit card cash-out frauds.

user information) are available for modeling. Besides, financial institutions are inclined to adopt explainable approaches owing to the strong enforcement by regulators.

To remedy these challenges, we regard identifying whether cardholders are engaged in cash-out activities as a problem of finding dense subgraphs. Detecting densely linked groups of nodes in a graph is a fundamental graph mining primitive [13]. The problem takes advantage of the internal structure of a subgraph by optimizing a formally-defined measure in the entire dataset. Subgraph discovery has been extensively studied for its theoretical approaches [23, 36] and practical applications: detecting biological motifs [12], spotting communities in queries [34], identifying stories in real-time streams [3], discriminating brain networks [22], and various anomaly detection tasks in social networks [14, 18, 27] and financial analysis [24, 39]. However, few attempts employ subgraph discovery for credit card frauds.

In this paper, we propose a novel approach for discerning cash-out users from credit card transaction data. In particular, we consider single-step transaction relations and adopt bipartite graph for modeling. This is because banks have complete card data but few external merchant data. Then, we capture cash-out users in graph by exactly exploiting density signals, instead of using a supervised learning method. Our major idea is generalized from the real-world observations. That is, cash-out activities periodically generate intensive and high-volume flows from user accounts, which is distinguishable from common transactions. We then propose a class of metrics to measure cash-out suspiciousness from four perspectives: time, capital, cyclicity and topotaxy. To fully utilize these suspicious signals, we develop ANTICO (**ANTI**-**C**ash-**O**ut), a near-greedy search algorithm with theoretical guarantees, to capture the cash-out frauds.

Our main contributions are as follows:

- **Graph**: to our knowledge, this work is among the first to model cash-out user detection as a problem of finding dense bipartite subgraphs.
- **Metric**: we define a class of cash-out suspiciousness metrics by leveraging time, capital, cyclicity and topotaxy.
- **Theoretical guarantee**: we provide an upper bound on the number of transaction relations between fraudsters without being caught, given limited cash-out merchants.

Table 1: Comparison between ANTICO and other fraud detection algorithms.

	Fraudar	Spoken	CopyCatch	BP-based methods	Holoscope	Crossspot	M-Zoom/D-cube	ANTICO
Scalability	✓	✓	✓	✓	✓	✓	✓	✓
Camouflage	✓		?		✓	?	?	✓
Accuracy guarantees	✓				✓		✓	✓
Temporal attention					?			✓

- **Effectiveness**: experiments on real-world data show that ANTICO outperforms other competitors by precisely catching cash-out users.

Moreover, the time complexity of the method is near-linear with the number of edges. Our code is open-sourced for reproducibility[1].

2 RELATED WORK

We summarize the related studies on fraud detection from two perspectives: method and application.

2.1 Graph-based Anomaly Detection

Graph-based methods detect crowds of fraudsters, often by discerning structural anomalies with density signals. It is hard to escape such detection since frauds unavoidably incur interrelated associations, i.e. edges in the graph. Recent studies have demonstrated its effect in a wide range of anomaly detection tasks [2].

Many works study the problem of mining dense blocks within adjacency matrices or multi-mode tensors. Spectral approaches detect dense regions by singular value decomposition (SVD) of the adjacency matrix. Spoken [29] exploited the "eigenspokes" patterns in large graphs, and was extended for anomaly detection [19]. SpecGreedy [11] proposed a unified framework with graph spectral properties. Tensor decomposition [28] is another field of detecting dense blocks. CrossSpot [17] estimated dense subtensors by using a Poisson model. M-zoom [32] and D-cube [33] provided fast dense-block detection across large-scale tensor data.

In addition, CopyCatch [6] employed one-class clustering and sub-space clustering to identify deceitful page likes. Belief propagation (BP) has been developed for suspicious behavior detection [1]. CoreScope [31] performed k-core analysis to spot dense blocks. GENPEEL [36] unified the problems of capturing the general dense subgraphs and the maximum k-core patterns.

For more accurate detection, existing works focus on the specific suspiciousness of nodes or edges on a real graph. Fraudar [14] proposed a camouflage-resistant weighting scheme based on the inverse document frequency like metric. HoloScope [27] presented contrast suspiciousness for graph topology mining and temporal spikes for the sudden break of attacking patterns. HiDDen [39] extracted dense subgraphs at different granularities from the financial

[1] https://github.com/transcope/antico

identity information network. Flowscope [24] adopted a tripartite graph to organize massive capitals for money laundering.

Our work differs from these in its customization of bipartite graph to detect cash-out users in the banking scenario. We compare our work with other competitors in Table 1.

2.2 Cash-out Detection

Most credit card frauds are single-side schemes for benefits, whereas cash-out frauds involve the collusion between users and merchants. Internet financial platforms tend to detect and regulate suspicious merchants [25], e.g. charges are adjusted to raise cash-out costs. In addition, recent graph-based studies consider a general concept of credit risks which includes cash-out frauds owing to its high correlation with financial default [15, 16, 40].

From the perspective of economics, the fundamental basis is that executing a scam can bring benefits. For excessive profits, most fraudsters share and reuse resources (e.g. accounts, devices, addresses) [5]. Under the economic law, financial fraudulent activities usually give rise to the temporal-spatial aggregation. DeepTrax [21] empirically validated the shared-card-based merchant network, where nodes represent merchants and edges are the presence of short-time transactions made by the same card at two merchants. Flowscope [24] discerned money laundering by the suspicious inflows and outflows of middle accounts. The idea is triable for cash-out merchant detection. Most closely related to our hypothesis is Cheng's work [9] on the unauthorized use of cards through extracting the temporal-spatial aggregation in transactions.

3 PROBLEM FORMULATION

From the bank's perspective, cash-out frauds refer to a procedure involving suspicious flows of funds from user accounts to merchant accounts. It turns out that capital transfer chains are seldom found in the bank database (see Figure 1a). Thus, the problem is focused on single-step transactions to identify cash-out users.

3.1 Analysis

We assume that the cash-out behaviors may involve the following measurable traits.

Trait 1 (Intensiveness). *Fraudsters tend to draw funds from credit cards within a short time.*

The main intention is to maximize the capital occupation time. Besides, cash-out users distrust their collusive merchants.

Trait 2 (Mass). *Large-scale credit limits are engaged for cash-out purpose.*

We assume that cash-out may exhaust massive credit limits. Meanwhile, users may have general payments.

Trait 3 (Cycle). *Cash-out activities occur periodically.*

Subject to the credit card billing cycle, most cash-out frauds behave monthly.

Trait 4 (Shared merchants). *Cash-out transactions are concentrated on limited merchants.*

To maximize profits, fraudsters tend to share resources, i.e. merchants or POS machines.

Table 2: Description of symbols.

Symbol	Interpretation
U	Users or credit cards
M	Merchants
G	Bipartite multi-graph, $G = (V, E, X)$
V	Nodes of graph G, $V = U \cup M$
E	Transaction edges of graph G, $E = \{e_{ij}(t)\}$
$e_{ij}(t)$	Specific transaction from user i to merchant j at time t, its value is the transaction amount
X	Personal credit limits of l time cycles, $X \in \mathbb{R}^{\|U\|\times l}$
A	Subset of users
B	Subset of merchants
G'	Bipartite graph $G' = (V, E')$
E'	Binary transaction relations, $E' = \{e'_{ij}\}$
e'_{ij}	Binary relation between user i and merchant j
$g(A \cup B)$	Density metric
T	Transaction time cycles, $T = \{T^{(1)}, \cdots, T^{(l)}\}$
$T^{(k)}$	Transaction time slices, $T^{(k)} = \{T_1^{(k)}, \cdots, T_d^{(k)}\}$
$\mathcal{H}$	Group of suspiciousness metrics, $\mathcal{H} = \{\mathcal{T}, \mathcal{R}\}$
$\mathcal{F}(\cdot)$	Transition function
$\phi(\cdot)$	Joint probability function

In addition, Trait 1 and Trait 3 are with respect to temporal behaviors. Trait 2 is to measure the capital occupation. Trait 4 shows graph topology, where numerous cash-out funds are flowed in a few merchants. To sum up, cash-out activities periodically generate intensive and high-volume flows from user accounts, which distinguishes from common transactions.

3.2 Definitions and Notations

Considering that transactions refer to a binary relation between users and merchants, we choose bipartite graph for modeling.

Specifically, we define $G = (V, E, X)$ to be a bipartite multi-graph, with vertices V forming accounts and edges E forming transactions, and X forming feature vectors. Here, $V = U \cup M$ is composed of m users (i.e. U) and n merchants (i.e. M). $E(U, M) = \{e_{ij}(t)\}$ is a set of attributed edges, where $e_{ij}(t)$ details a specific transaction from the user $i \in U$ to the merchant $j \in M$ at the time t and its value denotes dealing amount. Let $X \in \mathbb{R}^{|U|\times l}$ be a matrix involving credit limits of l time cycles. Figure 1b shows the graph schema of G. The definition of our problem is as follows.

Problem 1 (cash-out detection). ***Given*** *a user-merchant transaction graph G,*

- ***find*** *dense subgraphs in G, consisting of cash-out users and merchants,*
- ***to optimize*** *a class of suspiciousness metrics under the traits in terms of temporal, capital and topological behaviors.*

Besides, we also demonstrate that our proposed metric can be optimized with approximation guarantees, which provides theoretical bounds on the efficiency of cash-out detection. Table 2 shows the mainly used symbols throughout the paper.

4 PROPOSED APPROACH

Given this problem definition, we propose how to measure the cash-out suspiciousness of subgraphs in G.

4.1 Metric

A key idea of our approach is that we aggregate and mark the transaction edges between node pairs. To give a formal definition of our approach, we first define $G' = (V, E')$ as the simple transaction graph of $G = (V, E, X)$. For $i \in U$ and $j \in M$, the edge satisfies

$$e'_{ij} = \begin{cases} 1 & e_{ij}(t) \in E(U, M) \\ 0 & \text{else} \end{cases} \tag{1}$$

which depicts a binary relation between node pairs.

Now, we can build the framework of our metric. Afterwards, we show how to evaluate the cash-out frauds from three perspectives (Trait 1-3), and then integrate them for joint use.

4.1.1 Overall framework. Let $g(A \cup B)$ denote a class of metrics for optimization, where the subset of user nodes $A \subseteq U$ and the subset of merchant nodes $B \subseteq M$ constitute the target dense subgraph that the algorithm is identifying.

We propose using density metrics g formed as follows:

$$g(A \cup B) = \frac{1}{|A| + |B|} \sum_{i \in A} \sum_{j \in B} a_{ij} \cdot e'_{ij} \cdot c_{ij} \tag{2}$$

where e'_{ij} is a binary defined in formula and $a_{ij} \geq 0$ is a prior constant from the extra knowledge. We use a_{ij} to eliminate super nodes validated in the white list. The notation c_{ij} is a measure on the transaction relation between user i and merchant j and also known as an edge suspiciousness of G'. It is approximated by

$$c_{ij} = \sum_{e_{ij}(t) \in E} \sigma_{ij}(t) \cdot b_{ij}(t) \tag{3}$$

which is aggregated with the suspiciousness of each transaction between i and j. The factor $\sigma_{ij}(t)$ is defined as $\sigma_{ij}(t) = \frac{e_{ij}(t)}{\sum_\tau e_{ij}(\tau)}$, i.e. namely weighting by scaling transaction amounts. The transaction suspiciousness $b_{ij}(t)$ is defined as follows:

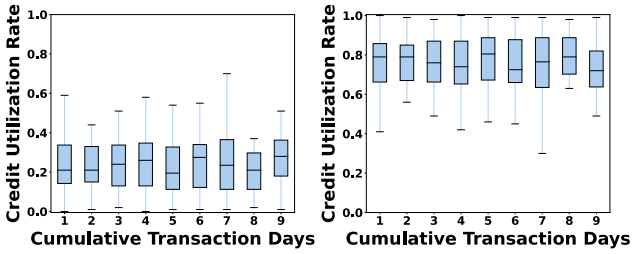
$$b_{ij}(t) := P(i \in A, j \in B \mid t) = P(i \in A, j \in B \mid i \in A, t) \cdot P(i \in A) \tag{4}$$

where $P(i \in A)$ denotes a priori estimation on the likelihood of a user node i that belongs to A (suspicious set), and $P(i \in A, j \in B \mid i \in A, t)$ is defined as the conditional likelihood of a specific transaction for cash-out purpose, given user i and time t. We approximate the suspiciousness of user i and then each transaction from i. That is, to detect dense graph structure, we calculate edge suspiciousness from a single side.

To sum up, the main benefits are that the metrics of this form can obey some basic properties listed in [17] and also the optimization is scalable and with theoretical guarantees.

4.1.2 Prior estimate. We next introduce how to approximate $P(i \in A)$ satisfying the above traits (see Section 3.1).

Intensiveness. Fraudsters tend to reach their credit limits within a short time (see Trait 1). In Figure 2, we can observe from the average statistics that suspicious users exhaust far more credit limits than general users within the given time. That is, cash-out activities



(a) General users (b) Suspicious users

Figure 2: Boxplots of temporal-capital analysis on transactions made by general users and suspicious users.

constantly spring up, which create high-volume flows of funds with adjacent timestamps.

Many methods can explore temporal uneven patterns [38]. We adopt Gini Index (Gini), an economic indicator on the wealth inequality within a nation. For our problem, we consider the distribution of transaction amounts (viewed as wealth) over the time slices (viewed as people). Besides, Gini has an applicable data range $[0, 1]$. We also prove that Gini can lead to strong theoretical bounds.

Assume that the k-th time period $T^{(k)}$ is split into d time slices, formed as $\left\{T_1^{(k)}, T_2^{(k)}, \cdots, T_d^{(k)}\right\}$. Let $\left\{x_1^{(i)}, x_2^{(i)}, \cdots, x_d^{(i)}\right\}$ be indexed in ascending sort, where $x^{(i)}$ is the aggregation of transactions within a time slice given user i. It is formulated as follows:

$$\mathcal{T}_i\left(T^{(k)}\right) = \frac{2\sum_{j=1}^{d} j \cdot x_j^{(i)}}{d\sum_{j=1}^{d} x_j^{(i)}} - \frac{d+1}{d} \tag{5}$$

Then, we have $P(i \in A) \propto \mathcal{F}(\mathcal{T}_i)$, where $\mathcal{F}(\cdot)$ denotes a transition from metrics to likelihood estimation. Naturally, we divide $T^{(k)}$ into calendar days as the choice of time slice widths.

Mass and cycle. We first assume that cash-out activities engage massive credit limits (see Trait 2). A natural choice is the credit utilization rate, formed as

$$P(i \in A) \propto \mathcal{F}(\mathcal{R}_i), \text{ where } \mathcal{R}_i\left(T^{(k)}\right) = \frac{\sum_{\tau \in T^{(k)}} e_{i*}(\tau)}{X_{ik}} \tag{6}$$

where X_{ik} is the credit limits of user i within $T^{(k)}$.

A sudden surge that creates one or more large volumes is far from abnormality. Hence, we consider the cyclical time of the activities for long-term capital occupation (see Trait 3). Given the data of l cycles, we have $\mathcal{T}_i = \frac{1}{l}\sum_{k=1}^{l} \mathcal{T}_i\left(T^{(k)}\right)$ and $\mathcal{R}_i = \frac{1}{l}\sum_{k=1}^{l} \mathcal{R}_i\left(T^{(k)}\right)$. Natural month should be a good choice for the settings of T.

Fusion. The following problem is how to integrate them for $P(i \in A)$. We define the requirements for a fusion of metrics: (a) saturated regime for high suspiciousness; (b) probability form and nonzero output; (c) scalability to hold more metrics.

An intuitive way is to multiply metrics directly. It seems ineffective since small variables may incur a suspiciousness vanishing problem. Inspired by [27], we employ a sigmoid function for better probability estimate. Let $\mathcal{H}$ be a set of metrics, where $\mathcal{H} = \{\mathcal{T}, \mathcal{R}\}$ in this paper. Given user i, $\mathcal{F}(\cdot)$ is defined as follows:

$$P(i \in A) \propto \mathcal{F}(\mathcal{H}_i), \text{ where } \mathcal{F}(\mathcal{H}_i) = \frac{1}{1 + b^{(1-2\phi(\mathcal{H}_i))}} \tag{7}$$

where the joint probability ϕ is

$$\phi\left(\mathcal{H}_i\right) = \sum_{h_{ij} \in \mathcal{H}_i} \gamma_j \cdot h_{ij} \tag{8}$$

for constants $\gamma_j > 0$ satisfy $\sum \gamma_j = 1$. The constants can serve as a prior for manual tuning (otherwise, the default is $\frac{1}{|\mathcal{H}|}$). $\mathcal{F}(\cdot)$ involves a sigmoidal curve constrained by probabilistic bounds. Furthermore, in a real-world environment, cash-out users may behave suspicious only in some perspective. We can still get a high suspiciousness probability by using the saturation function. The saturation interval can be adjusted by setting $b > 1$.

4.1.3 Edge suspiciousness. So far, we have got the approximation of $P(i \in A)$. Next, we define to evaluate each transaction. The main idea is that we assign higher suspiciousness to any transaction within the time slice when the volume is large, satisfying Trait 1. That is, the suspiciousness of each transaction edge is determined by the temporal distribution when $P(i \in A)$ is given.

Recall that Gini is employed to measure the transaction inequality within a specific group of time slices. Conversely, the suspiciousness can be distributed to each time slice by the proportion of transaction amounts within a cycle. Then, we treat every transaction equally and assign the same suspiciousness with the corresponding time slice. For $t \in T_s^{(k)}$, the approximation of $P(i \in A, j \in B \mid i \in A, t)$ is formed as follows:

$$P(i \in A, j \in B \mid i \in A, t) \propto \frac{\sum_{\tau \in T_s^{(k)}} e_{i*}(\tau)}{\sum_{\tau \in T^{(k)}} e_{i*}(\tau)} \tag{9}$$

A natural follow-up question is whether the suspiciousness can be further allocated to transactions within a time slice by amount. It turns out that this weighting is not camouflage resistant. This is because fraudsters can lower suspiciousness by faking a batch of petty trades. In contrast, regardless of how many payments and how much of each payment, our equal treatment would not decrease the suspiciousness of transactions within a time slice.

In addition, our weighting can tolerate normal transactions occurring in nearby time to be assigned a high weight. We can handle the mistaken impact by exploring topology in graph, even if there is a single transaction between i and j. Thus, our ANTICO can guarantee anti-noise in most cases.

4.2 Algorithm

We next state how to optimize g. Developed on the Charikar's greedy peeling approach [8], ANTICO is to iteratively identify and then remove a single node with the minimum score. We first define the score assigned to each node i. It is required to be homologous with the elements of g. Hence, we consider node information, i.e. the degree of nodes in G'. The score is defined as

$$w_i(S) = \begin{cases} \sum_{j \in B} a_{ij} \cdot e'_{ij} \cdot c_{ij} & i \in A \\ \sum_{j \in A} a_{ji} \cdot e'_{ji} \cdot c_{ji} & i \in B \end{cases} \tag{10}$$

where $S = A \cup B$. Note that it does matter that the constants a_{ij} are used to add or reduce prior suspiciousness. For example, considering that we adopt complete credit card data including online and offline payments, the online purchases may create some super nodes which make serious impacts on the node selection, such as Amazon, Meituan, Alipay, WeChat Pay.

Algorithm 1 ANTICO

Given graph $G(U \cup M, E, X)$

```
S* ← ∅
S^(0) ← U ∪ M
{w_i} ← compute all node scores by formula (10)
P^(0) ← build priority tree from U ∪ M
while k = 1, ··· , |U| + |M| do
    i ← find the minimum node from P^(k−1)
    S^(k) ← S^(k−1) \ {i}
    {w_i} ← ∀j if e'_ij or e'_ji then recompute node score
    P^(k) ← assign ∞ to leaf i then recompute tree
    if g(S^k) > g(S*) then
        S* ← S^k
    end if
end while
return S* that maximizes g in formula (2)
```

Let $S^{(k)}$ be the nodes of subgraph at the k-th iteration, and then $S^{(0)}$ be the initial set of nodes $U \cup M$. The algorithm involves a greedy procedure of peeling nodes from the graph. Concretely, the k-th iteration is to find a locally optimum g evaluated on the rest nodes $S^{(k)} = S^{(k-1)} \backslash \{i\}$ while removing the node i with the minimum score. The iterations generate a diminishing sequence of subsets of nodes $S^{(0)} \supset S^{(1)} \supset \cdots \supset S^{(|U|+|M|)}$ until the rest is empty. Then, the algorithm outputs the densest one with the maximum g and the subset of nodes S^* as the final result. The whole procedure is described in Algorithm 1. Besides, to find the next subgraph, the intuitive way is to remove edges of the current one, and then repeat the above procedure.

4.3 Theoretical Analysis

From a theoretical perspective, we propose the approximation guarantees made by ANTICO for cash-out user detection. Recall that the optimization is inspired by [8], we have

Theorem 1 (Theoretical Bounds[2]). *Given a graph instance and the metric g, let (A_{opt}, B_{opt}) be the subset with the maximum suspiciousness. Then, ANTICO can achieve*

$$g(A \cup B) \geq \frac{1}{2} g\left(A_{opt} \cup B_{opt}\right) \tag{11}$$

Therefore, our detection result can promise half of the optimal density at the worst case. Further, we present what cash-out fraudsters that our proposed algorithm can capture.

Theorem 2 (Bounding Cash-out). *Let (A^*, B^*) be a block with m_0 users and n_0 merchants returned by ANTICO. Such subset can evade the detection if it satisfies*

$$\sum_{e'_{ij} \in E'} e'_{ij} \leq \frac{2(|A^*| + |B^*|) g(A^* \cup B^*)}{\eta} \left(1 + b^{2\epsilon + \frac{\Delta t}{d} - 1}\right) \tag{12}$$

[2]The proof details of the theorem refer to [8]

Table 3: The statistic information of real-world datasets sampled from different regions and months, where the density represents the sum of credit utilization rates over the total number of users and merchants.

Datasets	#Nodes	#Edges	Density	Time Span
nx21co	388K (233K, 155K)	4.29M	0.41	May.01,2021 - Jun.30,2021
jl21co	689K (446K, 243K)	8.89M	0.34	Jun.01,2021 - Jul.31,2021
gz21co	625K (360K, 265K)	7.34M	0.30	Jul.01,2021 - Aug.31,2021
sx21co	1156K (854K, 302K)	19.62M	0.25	Jul.01,2021 - Aug.31,2021
dl21co	285K (151K, 134K)	5.69M	0.26	Jun.01,2021 - Aug.31,2021

where d denotes the number of time slices. Let η be the minimum cash-out scale within a time slice. Let ϵ be the maximum scale of credit limits that are engaged for general purchase purpose and Δt be the maximum time slices of the cash-out activities, both of which satisfy $1 - \epsilon - \frac{\Delta t}{d} > 0$.

Proof. Consider another suspicious block $\hat{S}$ with size (m_0, n_0). By formula (11), we claim that $g\left(\hat{S}\right) \leq 2g(A^* \cup B^*)$. For $m_0 + n_0$ nodes, we define f be the total suspiciousness as

$$f\left(\hat{S}\right) = \sum_{i,j \in \hat{S}} c_{ij} \leq 2\left(m_0 + n_0\right) g(A^* \cup B^*) \tag{13}$$

For any suspicious user, assume that cash-out transactions exhaust at least $1 - \epsilon$ proportion of credit limits within Δt time slices at most of a time period, each time slice of which exhausts at least η proportion of credit limits. Then, we have that the Gini based metric $\mathcal{T}$ satisfies $\mathcal{T} \geq 1 - \epsilon - \frac{\Delta t}{d}$ (see Section A.1 of appendix) and the capital occupation metric $\mathcal{R}$ satisfies $\mathcal{R} \geq 1 - \epsilon$. Let $N_e = \sum_{e'_{ij} \in E'} e'_{ij}$ be the number of cash-out transaction relation edges. Then

$$f\left(\hat{S}\right) \geq N_e \cdot c_m = N_e \cdot \frac{\eta}{1 + b^{2\epsilon + \frac{\Delta t}{d} - 1}} \tag{14}$$

where c_m is the minimum edge suspiciousness.

By formula (13) and (14), we conclude that N_e has an upper bound, i.e. $\frac{2(|A^*| + |B^*|)g(A^* \cup B^*)}{\eta}\left(1 + b^{2\epsilon + \frac{\Delta t}{d} - 1}\right)$. □

In brief, Theorem 2 provides an upper bound estimate of cash-out transaction relation edges within the block. In other words, our algorithm can effectively detect the block once it contains more suspicious edges.

5 EVALUATION

The experiments are organized as follows: effectiveness study, real-world effects, posteriori analysis, and scalability.

To evaluate our ANTICO under various cash-out behaviors, we employ extensive datasets including synthetic data and real-world data. The synthetic data are detailed in effectiveness study. Table reftab:datasets illustrates the statistics of our used real-world data derived from a top commercial bank in China. We compared ANTICO with the following baseline methods: SpokEn [29], Crossspot [17], Dcube [33], Fraundar [14], Holoscope [27]. The experiments are running on the Linux machine with four Intel Xeon processors at 2.50 GHz and 8 GB physical memory.

5.1 Effectiveness study

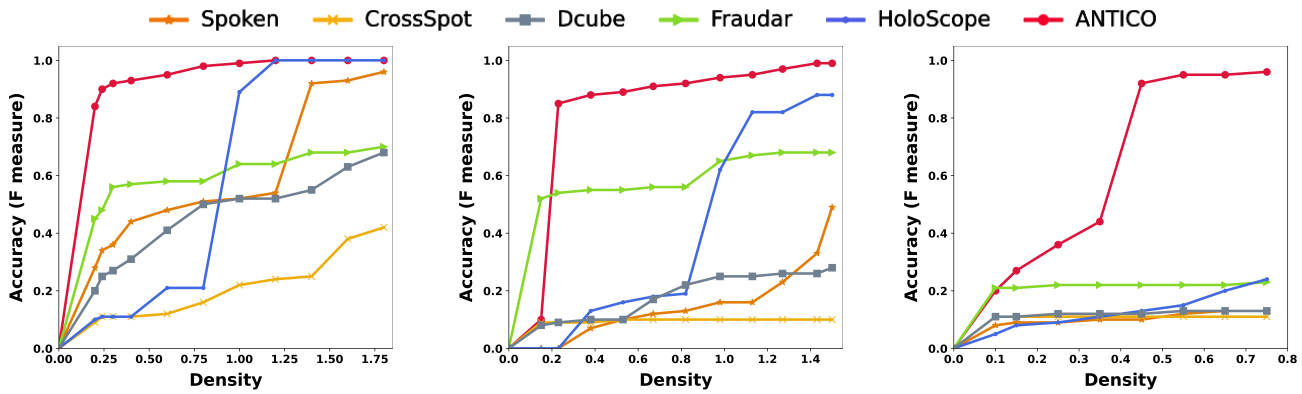
To demonstrate the effects of our method, we conduct three experiments with respect to validity and robustness: 1) cash-out connections, 2) credit utilization, 3) camouflage attacks.

We synthetically generate data with 20000 users and 20000 merchants. The banking users have separate credit limits, which lead to a wide gap in monthly expenses. To eliminate such influence, we assume users with the same credit limit in synthetic settings. From the perspective of topology, real transactions may generate arbitrary connections. We randomly add general transaction edges between users and merchants. To inject cash-out attacks, we first randomly pick 1000 suspicious users and mimic cash-out behaviors of diversiform scenarios, as detailed in the following cases. To evaluate performance, we compare ANTICO to other competitors by using F-measure in detecting suspicious users. As shown in Figure 3, the experimental results demonstrate that our method works effectively against cash-out frauds in all three cases.

We first assume the graph with no camouflage. That is, suspicious nodes have no link to honest nodes. In Figure 3a, we investigate the influence made by the number of fraudulent transactions. In real cases, fraudsters may disperse cash-out transactions to evade the detection. We inject cash-out attacks by incrementally adjusting the number of suspicious merchants (from 100 to 9000) while assuming that the credit limits are used up each month. The results show that ANTICO can achieve the best F-measure under variant edge densities, and even reach the ideal value in some cases. In contrast, other competitors are overly affected by the dispersive transactions due to its modeling of suspiciousness on edges.

In Figure 3b, we consider another major factor on densities, i.e. credit utilization. To investigate the behavior detection, we randomly select 1000 suspicious merchants and gradually decrease the credit utilization (on average) while keeping other conditions unchanged. The results empirically demonstrate the effects of ANTICO on the cases where credit limits are not exhausted for cash-out purpose. Note that our algorithm cannot detect any suspicious users under extreme conditions, where the entire credit occupations are lower than 20%. In a real-world environment, banks are more concerned with the risk of suspicious users with large occupation.

Next, our focus is to discern cash-out users in the presence of camouflage attacks. In the above experiments, we assume no capital transfer connections between suspicious nodes and honest nodes. However, suspicious users may have general purchase transactions. Even honest users may add edges to suspicious merchants in some offline scenarios, which is referred as "reverse camouflage". Thus, we inject camouflage attacks from both sides in Case 3. In detail, we incorporate camouflage edges pointed to random honest nodes and biased to high degree nodes. We assume that suspicious users have 20% credit utilization (on average) for general purchase purpose. Meanwhile, we add arbitrary reverse camouflage edges between



(a) ascending cash-out merchants (b) ascending credit utilization (c) adding camouflage transactions

Figure 3: ANTICO outperforms other competitors under variant density settings by: (a) adjusting the number of suspicious merchants; (b) ascending credit utilization for cash-out purpose; (c) adding camouflage edges between suspicious nodes and honest nodes. The horizontal axis represents the fraudulent density, i.e. the total proportion of credit utilization for cash-out purpose over the number of suspicious nodes.

Table 4: Comparison of performance on ANTICO with other competitors on real-world banking datasets.

Methods	nx21co		jl21co		gz21co		sx21co		dl21co	
	AUC	K-S	AUC	K-S	AUC	K-S	AUC	K-S	AUC	K-S
Spoken	0.6426	0.2861	0.5778	0.1563	0.6313	0.2684	0.6715	0.3404	0.6359	0.2755
CrossSpot	0.5392	0.2303	0.5173	0.2464	0.6073	0.3033	0.5125	0.1964	0.5458	0.2532
Dcube	0.5542	0.2312	0.5655	0.2881	0.6684	0.4002	0.4389	0.2692	0.5556	0.3377
Fraudar	0.7324	0.4648	0.6606	0.3236	0.6899	0.3602	0.6399	0.3323	0.7185	0.3643
HoloScope	0.5876	0.1735	0.5558	0.1658	0.5955	0.1899	0.5711	0.1918	0.5643	0.2011
ANTICO	**0.7472**	**0.4944**	**0.7051**	**0.4099**	**0.7325**	**0.4652**	**0.7534**	**0.5068**	**0.7563**	**0.5127**

half of honest users and suspicious merchants, but with sparser density than the fraud blocks. Compared to Case 2, all other conditions are unchanged. The experimental results are shown in Figure 3c. In spite of some loss when the entire credit occupations are lower than 40%, ANTICO works robustly and efficiently against camouflage attacks. In contrast, some competitors, such as Holoscope, performing well in Case 2 suffer serious performance degradation under camouflage.

5.2 Evaluation on real-world data

Next, we evaluate the performance of ANTICO on detecting cash-out frauds in real-world graphs. We have collected five datasets of separate areas and time spans during 2021, involving 45.8 million transactions between 2 million users and 1.1 million merchants (see Table 3). We assign a binary label on each user, where cash-out users are labeled as 1 and honest users are labeled as 0. Totally, our used datasets contain 120 thousand suspicious cash-out users examined by domain experts. With respect to normal users, we pick users having the intersection of merchants with the chosen subset (which we refer to as "shared-merchant relation"). In addition, we report the performance mainly via AUC (Area Under the ROC Curve) and KS (Kolmogorov Smirnov), which are extensively applied in banking risk management.

In Table 4, the results demonstrate that ANTICO has achieved higher scores than other competitive baselines on detecting large blocks of cash-out accounts in all five bank graphs. Indeed, most of the detected cash-out users have the suspicious behavior on long-term capital occupation during the given time spans. We further investigate the missing samples by our algorithm and summarize three possible reasons as follows: 1) few transactions seldom contain topology information, especially external merchants; 2) some cash-out activities are not covered in the time spans; 3) some other cash-out patterns, e.g. installment loans for various specific purposes. Besides, it is worth noting that all the competitors achieve best performance by iteratively exploiting multiple densest subgraphs. We conjecture that it is mainly due to the presence of a hierarchical suspiciousness of transaction relations, as detailed below.

5.3 Posteriori analysis

To further investigate the cash-out fraud patterns, we adopt case studies on the dense blocks by ANTICO. Concretely, we pick a set of verified results by domain experts, and then casually extract

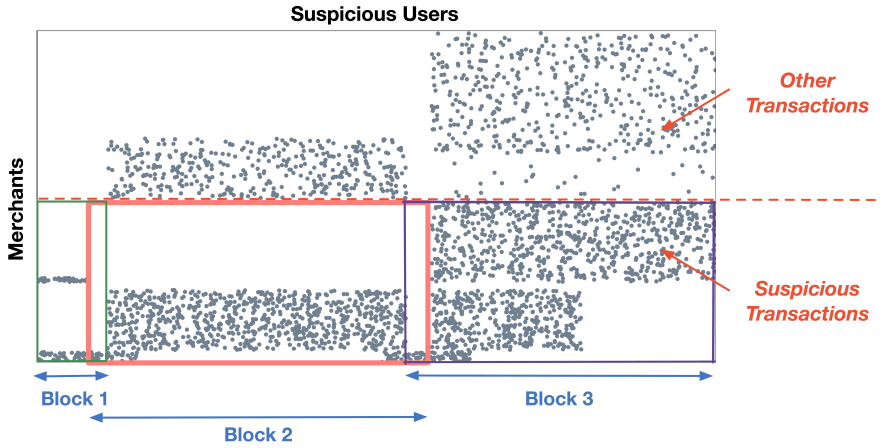


Figure 4: The distribution of transactions made by suspicious users of top three blocks detected by ANTICO. The users are sorted by the suspiciousness of blocks and then nodes within the block, along the horizontal axis. We add other transactions with merchants, which are not involved in blocks.

transactions within a time span. In Figure 4, we visualize the distribution of transactions between suspicious users and their paying merchants from top three blocks.

We summarize the following four major observations: 1) *Divergency*: a majority of suspicious users make cash-out transactions with several merchants. In contrast, a few cases with a single large-amount transaction in the dataset. 2) *Priority*: the merchants at the forefront of blocks involve more transactions with cash-out users on average, most of which are external-bank accounts (i.e. no other transactions). 3) *Multi-purpose*: most detected users have transactions with honest merchants as well as suspicious merchants. Our method can effectively exclude normal transactions and merchants. 4) *Hierarchy*: more suspicious dense blocks, more concentrated cash-out transactions by suspicious uses. Conversely, with the decline of block suspiciousness, users tend to disperse cash-out transactions and have more normal transactions.

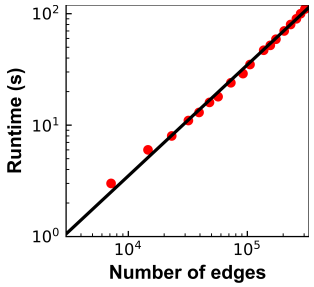


Figure 5: ANTICO is scalable with the transaction relation edges, where red plots along and around the main diagonal refer to linear growth.

5.4 Scalability

To analyze the runtime, we adopt incremental sampling from synthetic data to generate a series of datasets in a range of sizes. Figure 5 shows the running time of ANTICO with regard to the graph scales. We can observe that the taken time is closely linear with the number of transaction relation edges, which is consistent with the time complexity detailed in Section A.2 of appendix.

6 CONCLUSION

In this paper, we study the cash-out user detection by optimizing density signals on a defined bipartite graph. In particular, we reveal the traits of cash-out users, and propose a novel metric from these perspectives, namely temporal intensiveness, capital occupation, cyclicity and topotaxy. We develop ANTICO for searching the suspicious blocks by greedily optimizing our proposed metric. ANTICO provides theoretical bounds on the detection of cash-out frauds. Experiments demonstrate that ANTICO can effectively capture cash-out users in near-linear running time. Analysis results show that ANTICO can resist camouflage, where cash-out users may have real purchase payments. This is the first work in which dense subgraph mining has ever been used for spotting cash-out users in credit card services of conventional banks. As future work, our primary task is to upgrade the search process for high performance. Next, we will model the cash-out merchant detection as finding dense subgraphs, and then build a general framework for cash-out frauds. In addition, considering that ANTICO has exhibited the potential for labeling cash-out actions, another attempt is to cooperate with a semi-supervised graph learning model.

ACKNOWLEDGMENTS

This work was supported by the Key R&D Program of Ministry of Science and Technology of China (2020YFB0204800), and the Natural Science Foundation of China (No.61976026). We would like to thank experts from TigerGraph for assistance with the exploratory analysis on a large banking graph through their platform. Thank reviewers for their valuable comments and suggestions.

REFERENCES

[1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. In *ICWSM*, Vol. 7. 2–11.

[2] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data Min. Knowl. Discov.* 29, 3 (April 2015), 626–688.

[3] Albert Angel, Nick Koudas, Nikos Sarkas, Divesh Srivastava, Michael Svendsen, and Srikanta Tirthapura. 2014. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *The VLDB journal* 23, 2 (2014), 175–199.

[4] Alejandro Correa Bahnsen, Aleksandar Stojanovic, Djamila Aouada, and Björn Ottersten. 2014. Improving credit card fraud detection with calibrated probabilities. In *SDM*. 677–685.

[5] Yikun Ban, Xin Liu, Ling Huang, Yitao Duan, Xue Liu, and Wei Xu. 2019. No place to hide: Catching fraudulent entities in tensors. In *WWW*. 83–93.

[6] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. 2013. Copycatch: stopping group attacks by spotting lockstep behavior in social networks. In *WWW*. 119–130.

[7] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. 2020. Flowless: Extracting densest subgraphs without flow computations. In *WWW*. 573–583.

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

[9] Dawei Cheng, Sheng Xiang, Chencheng Shang, Yiyi Zhang, Fangzhou Yang, and Liqing Zhang. 2020. Spatio-temporal attention-based neural network for credit card fraud detection. In *AAAI*, Vol. 34. 362–369.

[10] María del Mar Roldán-García, José García-Nieto, and José F. Aldana-Montes. 2017. Enhancing semantic consistency in anti-fraud rule-based expert systems. *Expert Syst. Appl.* 90 (Dec. 2017), 332–343.

[11] Wenjie Feng, Shenghua Liu, Danai Koutra, Huawei Shen, and Xueqi Cheng. 2020. Specgreedy: unified dense subgraph detection. In *ECML-PKDD*. 181–197.

[12] Eugene Fratkin, Brian T Naughton, Douglas L Brutlag, and Serafim Batzoglou. 2006. MotifCut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics* 22, 14 (2006), e150–e157.

[13] Aristides Gionis and Charalampos E Tsourakakis. 2015. Dense subgraph discovery: Kdd 2015 tutorial. In *KDD*. 2313–2314.

[14] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *KDD*. 895–904.

[15] Binbin Hu, Zhiqiang Zhang, Chuan Shi, Jun Zhou, Xiaolong Li, and Yuan Qi. 2019. Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. In *AAAI*, Vol. 33. 946–953.

[16] Binbin Hu, Zhiqiang Zhang, Jun Zhou, Jingli Fang, Quanhui Jia, Yanming Fang, Quan Yu, and Yuan Qi. 2020. Loan default analysis with multiplex graph learning. In *CIKM*. 2525–2532.

[17] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2015. A general suspiciousness metric for dense blocks in multimodal data. In *ICDM*. 781–786.

[18] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2016. Spotting suspicious behaviors in multimodal data: A general metric and algorithms. *IEEE Trans. Knowl. Data Eng.* 28, 8 (2016), 2187–2200.

[19] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. Inferring strange behavior from connectivity pattern in social networks. In *PAKDD*. 126–138.

[20] Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre-Edouard Portier, Liyun He-Guelton, and Olivier Caelen. 2018. Sequence classification for credit-card fraud detection. *Expert Syst. Appl.* 100, 15 (June 2018), 234–245.

[21] Anish Khazane, Jonathan Rider, Max Serpe, Antonia Gogoglou, Keegan Hines, C. Bayan Bruss, and Richard Serpe. 2019. Deeptrax: Embedding graphs of financial transactions. In *ICMLA*. 126–133.

[22] Tommaso Lanciano, Francesco Bonchi, and Aristides Gionis. 2020. Explainable classification of brain networks via contrast subgraphs. In *KDD*. 3308–3318.

[23] Victor E. Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. 2010. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Chapter 10, 303–336.

[24] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. 2020. Flowscope: Spotting money laundering based on graphs. In *AAAI*, Vol. 34. 4731–4738.

[25] Yuan Li, Yiheng Sun, and Noshir Contractor. 2017. Graph mining assisted semi-supervised learning for fraudulent cash-out detection. In *ASONAM*. 546–553.

[26] Zhenchuan Li, Guanjun Liu, and Changjun Jiang. 2020. Deep representation learning with full center loss for credit card fraud detection. *IEEE Trans. Comput. Soc. Syst.* 7, 2 (April 2020), 569–579.

[27] Shenghua Liu, Bryan Hooi, and Christos Faloutsos. 2017. Holoscope: Topology-and-spike aware fraud detection. In *CIKM*. 1539–1548.

[28] Koji Maruhashi, Fan Guo, and Christos Faloutsos. 2011. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *ASONAM*. 203–210.

[29] B. Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. 2010. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *PAKDD*. 435–448.

[30] Quratulain Rajput, Nida Sadaf Khan1, Asma Larik, and Sajjad Haider. 2014. Ontology based expert-system for suspicious transactions detection. *Computer and Information Science* 7, 1 (Jan. 2014), 103–113.

[31] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. 2016. Corescope: Graph mining using k-core analysis—patterns, anomalies and algorithms. In *ICDM*. 469–478.

[32] Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2016. M-zoom: Fast dense-block detection in tensors with quality guarantees. In *ECML-PKDD*. 264–280.

[33] Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. 2017. D-cube: Dense-block detection in terabyte-scale tensors. In *WSDM*. 681–689.

[34] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *KDD*. 939–948.

[35] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. 2008. Credit card fraud detection using hidden Markov model. *IEEE Trans. Dependable Secure Comput.* 5, 1 (Jan. 2008), 37–48.

[36] Nate Veldt, Austin R Benson, and Jon Kleinberg. 2021. The generalized mean densest subgraph problem. In *KDD*. 1604—-1614.

[37] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. 2015. APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decis. Support Syst.* 75 (July 2015), 38–48.

[38] Yue Wu, Yunjie Xu, and Jiaoyang Li. 2019. Feature construction for fraudulent credit card cash-out detection. *Decis. Support Syst.* 127 (Dec. 2019), 113155.

[39] Si Zhang, Dawei Zhou, Mehmet Yigit Yildirim, Scott Alcorn, Jingrui He, Hasan Davulcu, and Hanghang Tong. 2017. Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In *SDM*. 570–578.

[40] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network. In *WWW*. 785–795.

A APPENDIX

A.1 Proof of the Gini Based Metric in Theorem 2

Given a time period with d time slices, let $\{\rho_1, \rho_2, \cdots, \rho_d\}$ be the credit utilization rate per time slice in ascending order, i.e. $0 \le \rho_1 \le \cdots \le \rho_d \le 1$ and $\rho_1 + \cdots + \rho_d \le 1$. We have assumed that cash-out activities take Δt time slices at most. Without loss of generality, we assume $\rho_1 + \cdots + \rho_{d-\Delta t} \le \epsilon$ and $\rho_{d-\Delta t+1} + \cdots + \rho_d \ge 1 - \epsilon$ to be credit utilizations respectively for general purchase and cash-out purpose.

For simplicity, we adopt $\mathcal{T}$ to represent the metric. Following our used Gini formula, $\mathcal{T}$ can be formed by

$$\mathcal{T} = \frac{2\sum_{j=1}^{d} j \cdot \rho_j}{d\sum_{j=1}^{d} \rho_j} - \frac{d+1}{d}$$

Let λ be the total credit utilization rate, where $\sum_{j=1}^{d} \rho_j = \lambda$ and $1 - \epsilon \le \lambda \le 1$. Then, we have the minimum problem

$$\mathcal{T} = \frac{2\sum_{j=1}^{d} j \cdot \rho_j}{d\lambda} - \frac{d+1}{d}$$

subject to equality constraints

$$\phi(\mathcal{P}) = \lambda - \sum_{j=1}^{d} \rho_j = 0$$

and inequality constraints

$$\psi_1(\mathcal{P}) = \sum_{j=1}^{d-\Delta t} \rho_j - \lambda + 1 - \epsilon \le 0$$

$$\psi_2(\mathcal{P}) = 1 - \epsilon - \sum_{j=d-\Delta t+1}^{d} \rho_j \le 0$$
$$\psi_3(\mathcal{P}) = -\rho_1 \le 0$$
$$\psi_4(\mathcal{P}) = \rho_1 - \rho_2 \le 0$$
$$\cdots$$
$$\psi_{d+2}(\mathcal{P}) = \rho_{d-1} - \rho_d \le 0$$

The optimal solution should satisfy the KKT conditions as follows:

$$\nabla\mathcal{T}\left(\rho^*\right) + \alpha\nabla\phi\left(\rho^*\right) + \sum_{j=1}^{d+2}\beta_j\nabla\psi_j\left(\rho^*\right) = 0, \quad \text{(I)}$$
$$\phi\left(\rho^*\right) = 0, \quad \text{(II)}$$
$$\psi_j\left(\rho^*\right) \le 0, \quad \text{(III)}$$
$$\beta_j \ge 0, \quad \text{(IV)}$$
$$\beta_j\psi_j\left(\rho^*\right) = 0, \quad \text{(V)}$$

where $j = 1, \cdots, d+2$. From (I), we have

$$\frac{2j}{d\lambda} - \alpha + \beta_1 - \beta_{j+2} + \beta_{j+3} = 0, \text{ where } j = 1, \cdots, d - \Delta t$$

$$\frac{2j}{d\lambda} - \alpha - \beta_2 - \beta_{j+2} + \beta_{j+3} = 0, \text{ where } j = d - \Delta t + 1, \cdots, d - 1$$

$$\frac{2}{\lambda} - \alpha - \beta_2 - \beta_{d+2} = 0$$

Then, we have

$$\beta_{j+2} = \frac{(d-j+1)(d+j)}{d\lambda} - (d-j+1)\alpha + (d-\Delta t-j+1)\beta_1 - \Delta t\beta_2,$$

where $j = 1, \cdots, d - \Delta t$

$$\beta_{j+2} = \frac{(d-j+1)(d+j)}{d\lambda} - (d-j+1)\alpha - (d-j+1)\beta_2,$$

where $j = d - \Delta t + 1, \cdots, d$.

By (V), we have

$$\beta_1\left(\sum_{j=1}^{d-\Delta t}\rho_j^* - \lambda + 1 - \epsilon\right) = 0$$
$$\beta_2\left(1 - \epsilon - \sum_{j=d-\Delta t+1}^{d}\rho_j^*\right) = 0$$
$$\beta_3\rho_1^* = 0$$
$$\beta_{j+2}\left(\rho_{j-1}^* - \rho_j^*\right) = 0, \quad j = 2, \cdots, d$$

Let $\beta_1 = 0$ and $\beta_2 = 0$. We deduce that $\Delta\beta_j := \beta_{j+2} - \beta_{j+1} = \frac{2-2j}{d\lambda} + \alpha$ is monotonically decreasing for $j = 2, \cdots, d$. Then, we consider the following three cases.

Case 1. *when* $\beta_3 = 0, \beta_4 > 0, \cdots, \beta_{d+2} > 0$, *where* $3 < l \le d+2$

It follows that $\rho_1^* = \cdots = \rho_d^* = \frac{\lambda}{d}$. To satisfy (III), we claim that $\lambda \ge \frac{d(1-\epsilon)}{\Delta t} > 1$, which contradicts with the assumption $\lambda \le 1$.

Case 2. *when* $\beta_3 > 0, \cdots, \beta_l = 0, \cdots, \beta_{d+2} > 0$, *where* $3 < l \le d+2$

It follows that $\beta_{l-1} = -\frac{d-l+4}{d\lambda} < 0$, which contradicts with (IV).

Case 3. *when* $\beta_3 > 0, \beta_4 > 0, \cdots, \beta_{d+2} > 0$

It follows that $\rho_1^* = \cdots = \rho_d^* = 0$, which contradicts with (II). Then, it is incompatible that $\beta_1 = 0$ and $\beta_2 = 0$. Now, we have

$$\sum_{j=1}^{d-\Delta t}\rho_j^* = \lambda - 1 + \epsilon, \quad \sum_{j=d-\Delta t+1}^{d}\rho_j^* = 1 - \epsilon \quad \text{(VI)}$$

For the rest parameters, we also have

$$\Delta\beta_j := \beta_{j+2} - \beta_{j+1} = \begin{cases} \frac{2-2j}{d\lambda} + \alpha - \beta_1 & j = 2, \cdots, d - \Delta t + 1 \\ \frac{2-2j}{d\lambda} + \alpha + \beta_2 & j = d - \Delta t + 2, \cdots, d \end{cases}$$

which is monotonically decreasing in each piece. In a similar way, we can assert that neither of the following two cases is possible.

Case 1. $\beta_3 > 0, \cdots, \beta_l = 0, \cdots, \beta_{d-\Delta t+2} > 0$, *where* $3 < l \le d - \Delta t + 2$

It contradicts the fact that $\Delta\beta_j$ is monotonically decreasing for $j = 2, \cdots, d - \Delta t + 1$.

Case 2. $\beta_{d-\Delta t+3} > 0, \cdots, \beta_l = 0, \cdots, \beta_{d+2} > 0$, *where* $d - \Delta t + 3 < l \le d + 2$

It follows that $\beta_{l-1} = -\frac{d-l+4}{d\lambda} < 0$, which contradicts with (IV). So we consider the following four cases.

Case 1. *when* $\beta_3 = 0, \beta_4 > 0, \cdots, \beta_{d-\Delta t+3} = 0, \cdots, \beta_{d+2} > 0$,

It follows that $\rho_1^* = \cdots = \rho_{d-\Delta t}^* = \frac{\lambda-1+\epsilon}{d-\Delta t}$, $\rho_{d-\Delta t+1}^* = \cdots = \rho_d^* = \frac{1-\epsilon}{\Delta t}$. Then, we can deduce that $\mathcal{T} = \frac{1-\epsilon}{\lambda} - \frac{\Delta t}{d}$.

Case 2. *when* $\beta_3 > 0, \beta_4 > 0, \cdots, \beta_{d-\Delta t+3} = 0, \cdots, \beta_{d+2} > 0$,

It follows that $\rho_1^* = \cdots = \rho_d^* = \frac{1-\epsilon}{\Delta t}$, which contradicts with (VI).

Case 3. *when* $\beta_3 = 0, \beta_4 > 0, \cdots, \beta_{d-\Delta t+3} > 0, \cdots, \beta_{d+2} > 0$,

It follows that $\rho_1^* = \cdots = \rho_d^* = \frac{\lambda-1+\epsilon}{d-\Delta t}$, which contradicts with (VI).

Case 4. *when* $\beta_3 > 0, \beta_4 > 0, \cdots, \beta_{d+2} > 0$,

It follows that $\rho_1^* = \cdots = \rho_d^* = 0$, which contradicts with (VI).

Overall, when $\rho_1^* = \cdots = \rho_{d-\Delta t}^* = \frac{\lambda-1+\epsilon}{d-\Delta t}$, $\rho_{d-\Delta t+1}^* = \cdots = \rho_d^* = \frac{1-\epsilon}{\Delta t}$, it also satisfies the regularity condition, and we have

$$\min\mathcal{T} = \frac{1-\epsilon}{\lambda} - \frac{\Delta t}{d}$$

Finally, we conclude that

$$\mathcal{T} \ge 1 - \epsilon - \frac{\Delta t}{d}$$

A.2 Complexity Analysis

In Algorithm 1, we can see that the weights c_{ij} are calculated at the initialization stage, which runs in $O(|E|)$ time. The hotspot is in the greedy loop. The procedure runs $|V|$ iterations, each of which is to search and remove the node with the minimum score, and then update scores of its neighbors. A general speedup method is to maintain a priority queue [7]. Thus, we build a priority tree for search efficiency. It is a binary tree data structure where leaf nodes store V with the score as its priority, and each middle node iteratively stores the smaller priority of its two children. The priority

tree searches in $O(\log |V|)$ time, which moves quickly from the root to the minimum leaf. The priority updates can run in $O(\log |V|)$ time by fixing the leaf nodes. The greedy procedure requires $O(|E'|)$ update operations, each of which takes $O(\log |V|)$ time. The time complexity of our algorithm is $O(|E'| \log |V|)$ time.