2018 IEEE International Conference on Data Mining Workshops (ICDMW)

Anomaly Detection in Car-Booking Graphs

Oleksandr Shchur[1], Aleksandar Bojchevski[1], Mohamed Farghal[1,2], Stephan Günnemann[1], Yusuf Saber[2]
[1]*Technical University of Munich, Germany* [2]*Careem, UAE*
{shchur,bojchevs,farghl,guennemann}@in.tum.de yusuf.saber@careem.com

***Abstract*—The use of car-booking services has gained massive popularity in the recent years – which led to an increasing number of fraudsters that try to game these systems. In this paper we describe a framework for fraud detection in car-booking systems. Our core idea lies in casting this problem as an instance of anomaly detection in temporal graphs. Specifically, we use unsupervised techniques, such as dense subblock discovery, to detect suspicious activity. The proposed framework is able to adapt to the variations in the data inherent to the car-booking setting, and detects fraud with high precision. This work is performed in collaboration with Careem, where the described framework is currently being deployed in production.**

I. Introduction

Car-booking services, provided by companies such as Uber, Lyft, and Careem, have become an important mode of transportation in urban areas. In car-booking systems, users can book car trips and get automatically matched to a nearby driver (a.k.a. captain) by the booking system. The system itself is completely responsible for making the assignment, thus, neither the user nor the captain are involved in the assignment. Along with the increasing popularity of these services, the problem of fraudsters that try to fool these systems (e.g. creating fake trips, where the payment is declined after the trip ends) gets increasingly important as well. Therefore, effective techniques to detect fraud in car-booking systems are essential.

One of the main challenges with fraud prevention is that fraud grows and changes rapidly. Thus, techniques are needed to quickly detect new fraud patterns, before they spread and cause significant damage. Although rule-based systems based on hand-crafted rules can capture many types of fraud, they are not agile enough to adapt to the evolving behavior of the adversaries. Moreover, in car-booking systems, monitoring and finding fraud patterns based on hand-crafted rules is hard because of the regional variability, seasonal trends, heterogeneity of the data and the potentially complex fraud patterns. As a consequence, automatic data-driven techniques for fraud detection are required handling the underlying complex data.

In this paper, we present a framework designed in collaboration with and currently deployed at Careem. Our core idea is to treat the described problem as an instance of *anomaly detection in car-booking graphs*. As we will discuss later, we can conceptually think of the raw data in a car-booking system as a dynamic heterogeneous graph of captains, customers and other nodes with different types of relations between them. For example the *booked_with* relation denotes that a customer booked a trip with a captain. Given this graph structure, our goal is to find anomalous patterns (potentially) indicating ride sharing fraud.

We designed our anomaly detection framework for car-booking systems with the following requirements in mind: **(1) Adaptability**. The framework should be able to adapt to new data and new anomalies. Fraud is a reactive process, where fraudsters adapt to the attempts to prevent it. Thus, the framework must be able to adapt to such changes, and should not only consider fixed periods of time or one predefined anomaly pattern. **(2) Interpretability**. The anomaly detection results need to be explainable. Since after detecting anomalies some action follows (e.g., blocking the corresponding captains), an explanation for every action is essential. **(3) True Alarms**. Since actions often relate to blocking accounts (i.e. the users/captains can no longer use the system), a high certainty in marking patterns as anomalies is required. The system does not need to recall every fraud instance, but it is crucial to be sure of the ones reported (high precision).

Given these requirements, we introduce a framework that makes use of graph-based anomaly detection ideas. Specifically, to ensure interpretability we base on the idea of suspiciously dense subgraphs. In the following, we describe this framework and some related work in more detail.

II. Related Work

So far the problem of fraud detection in car-booking graphs has not been discussed in the literature. However, closely related is the field of unsupervised anomaly detection in relational data. For a comprehensive survey see [1].

Approaches for anomaly detection in static graphs usually either consider the graph structure [2], [3], or combine it with node features [4]. While these approaches provide good interpretability, they are not suited for car-booking graphs, since temporal evolution is an essential aspect in this setting. When dealing with temporal (a.k.a. dynamic) graphs we can distinguish several categories of approaches: (i) feature based event detection methods model certain properties of the graph and detect anomalous timestamps w.r.t. the features in the evolution of the graph [5]; (ii) decomposition based approaches detect anomalies by interpreting the spectra of certain matrices/tensors representing the graph [6]; (iii) community based approaches monitor changes in the structure of communities over time [7]; and (iv) window based event methods model earlier snapshots of the graph as 'normal' and compare them to the current state [8].

The class of approaches we are most interested in are methods that aim to find suspicious subgraphs [9]–[12]. Specifically, we consider approaches where the graph is modeled as a tensor (where the modes represent e.g. customers, time).

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00093

Such methods are usually based on discovering suspiciously dense subblocks of the tensor. This view is justified since real-world networks are usually sparse. Indeed, manual monitoring for anomalous behavior in the car-booking domain has also identified unusually dense subgraphs as suspicious. This, coupled with the interpretability, as well as the scalability of these approaches makes them a good choice for our use-case.

III. Framework

Fraudsters are able to obtain in-app credit using various methods (e.g., using promotional vouchers or stolen credit cards). However, getting money out of the system can only be done by performing trips. This means that we need to focus on the interactions between different agents in the system (e.g. customers and captains) in order to detect fraudulent behavior.

We cast the problem of detecting fraudulent behavior in car-booking data as that of finding dense blocks in car-booking graphs. Detecting dense subblocks is a common way of spotting anomalies in multi-relational data. A number of algorithms for tackling this problem have been proposed, that provide theoretical guarantees and are suitable for distributed computation. Still, we faced important multiple challenges when developing our system:

1) Given the highly heterogeneous and unstructured nature of the car-booking data, how do we construct graphs, where dense blocks correspond to suspicious behavior?
2) How do we incorporate available side information (such as user attributes) into our fraud detection framework?
3) How do we make sure that the detected dense blocks actually correspond to fraudulent behavior?

A. Flexible Graph Construction

The car-booking history between the drivers D and the passengers P over the sequence of time intervals T is stored in the system as tuples of the form (d, p, t). With a simple SQL query we can count how many rides driver $d \in D$ had with passenger $p \in P$ over a time period $t \in T$, and denote this quantity as $r(d, p, t)$. Additionally, each passenger p has attributes associated with him, such as $user\_id$ or $device\_id$. For a given time window t, we can represent this data as a bipartite graph with edge weights given by $r(d, p, t)$, as can be seen in pane 1 of Figure 1. A traditional approach for detecting anomalies in such setting would involve performing dense subblock discovery on a dynamic bipartite driver-passenger graph, represented as a $|D| \times |P| \times |T|$ tensor $\mathcal{R}$ with $\mathcal{R}_{d,p,t} = r(d, p, t)$.

We found that the problem can be simplified by performing one-mode projection of the bipartite graph $\mathcal{R}$. The flexibility of our approach partly comes from the choice of the projection. For example we can create a driver-driver graph (further referred to as $\mathcal{G}$), or a passenger-passenger graph. As the number of drivers is significantly lower than the number of users ($|D| \ll |P|$), working with $\mathcal{G}$ makes our approach a lot more computationally economical. Moreover, since fraud typically involves drivers (as they are the ones able to withdraw money from the system), we are able to capture most malicious behavior by only considering $\mathcal{G}$. To further improve the accuracy of our system, we create a separate graph for each geographic region since: (i) different regions have different baseline legitimate behavior and thus different anomalous behavior, and (ii) the computational complexity is reduced. Since information is lost when performing the projection, we need to carefully choose the edge weights in the resulting one-mode graph $\mathcal{G}$ to ensure that the relevant information is preserved [13].

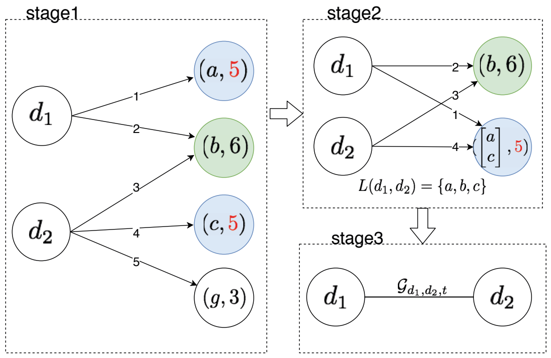

Fig. 1. Graph construction

An important issue we had to resolve first is the fact that not all passengers in the set P correspond to unique persons. Fraudsters often create fake passenger accounts that are then used for cheating purposes. We address this problem by considering the attributes of the passengers (like $user\_id$ or $device\_id$). Passengers sharing the same $device\_id$, even though having different $user\_ids$, likely correspond to the same individual. Therefore, we first construct for each driver d_i a set of passengers he traveled with

$$P_i = \{p \in P : r(d_i, p, t) > 0\}.$$

Then, for every attribute j and every pair of drivers (d_1, d_2) we construct a set $L_j(d_1, d_2)$ consisting of the passengers that traveled with both drivers, potentially using different identifies (i.e. the passenger data does not need to be completely identical but only the value in attribute j needs to be the same):

$$L_j(d_1, d_2) = \{p_1 \in P_1 : \exists p_2 \in P_2 \text{ s.t. } attr_j(p_1) = attr_j(p_2)\} \\ \cup \{p_2 \in P_2 : \exists p_1 \in P_1 \text{ s.t. } attr_j(p_1) = attr_j(p_2)\}$$

In Figure 1 we see that passengers a and c are added to $L(d_1, d_2)$ because they have the same $device\_id$ of 5, and each of them booked a trip with one of the drivers. Likewise passenger b is added since both drivers had a trip with him.

Edge weights in the driver-driver graph at time window t are then determined as

$$\mathcal{G}_{d_1,d_2,t} = \max_j \min \left\{ \sum_{p \in L_j} r(d_1, p, t)\eta(p), \right. \\ \left. \sum_{p \in L_j} r(d_2, p, t)\eta(p) \right\}$$

where $\eta : P \to \mathbb{R}$ is a passenger scoring function. In the simplest case we can set $\eta(p) = 1$, thus simply counting the number of common users between drivers weighted by the number of trips. In a more sophisticated scenario we can use the scoring function to incorporate additional information about the users available to us (more on this in Section III-B). The edge weight is set to the minimum over d_1 and d_2 since we want conservative estimates of the graph density to avoid false positives. We construct a graph with the above edge weights for every time interval $t \in T$ (typically, 2 hours long), and represent the graph as a matrix $\mathcal{G}_t$. Stacking these matrices together for every $t \in T$ produces a three mode tensor $\mathcal{G}$.

B. Dense Subblock Discovery

We are interested in finding anomalies in the graph that we constructed. Our approach is based on the M-Zoom algorithm [9]. This choice is motivated by the efficiency of M-Zoom (linear scaling), and its theoretical guarantees on the density of the detected subblocks.

Consider an N-way tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots \times n_N}$. Let $\mathcal{T}_{I_1,\ldots,I_N}$ be a subblock of the tensor $\mathcal{T}$ (where is $I_j \subseteq \{1, ..., n_j\}$ for $j = 1, ..., N$). Arithmetic density of the subblock $\mathcal{T}_{I_1,\ldots,I_N}$ is then defined as

$$\rho(\mathcal{T}_{I_1,\ldots,I_N}) = \frac{\sum_{i_1 \in I_1} \cdots \sum_{i_N \in I_N} \mathcal{T}_{i_1,\ldots i_N}}{\prod_{j=1}^{N} |I_j|/N}$$

M-Zoom uses a greedy strategy to find K blocks in the original tensor $\mathcal{T}$ that have the highest arithmetic density. It has been shown that dense subblocks in tensors correspond to anomalous activity in applications ranging from social networks to rating platforms and internet traffic [9], [10].

Recall, that in Section III-A we constructed the temporal driver-driver graph $\mathcal{G}$ (represented as a 3-way tensor in $\mathbb{R}^{|D| \times |D| \times |T|}$), where edge weights depend on the passengers shared by the drivers, and the scoring function $\eta(p)$. For the simple choice of the scoring function $\eta(p) = 1$, this corresponds to the number of customers that drivers have in common. Given the large number of users in the system and the fact that assignment is done automatically, it is extremely unlikely that multiple drivers have a large number of customers in common by chance. This matches our intuition that dense subblocks correspond to suspicious activity.

As a more sophisticated strategy, we can weigh users differently based on their features. These features are, for instance, their email address (fake accounts often have an email address where the characters are close to each other on a keyboard layout indicating they were randomly typed), IP addresses, phone number or log-in history. We use a decision tree classifier that takes the user basic info and outputs the likelihood of a user account being fake. When using this likelihood for $\eta(p)$, we expect that the sub-graphs containing less reputable users will be denser.

C. Filtering

The dense subblock search algorithm outputs the K densest blocks in the given tensor – even in the case if there are no anomalies in the underlying graph. This makes it clear, that we need a refinement step in order to avoid false positives.

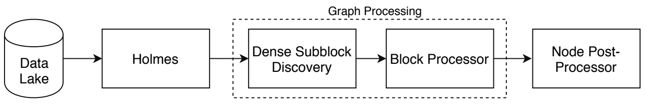

Fig. 2. Autonomous anomaly detection pipeline

Our system contains two levels of filtering: density filtering and rule-based filtering. In the density filtering step we discard the blocks whose density is within two standard deviations of the average block density in the graph. The candidate block density can be computed directly, as the blocks are much smaller than the entire graph. To efficiently estimate the average block density of $\mathcal{G}$ we do the following. We start by selecting M nodes uniformly at random and performing L-level BFS from each of them. Denoting by $\mathcal{B}_m$ the subgraph induced by the set of nodes L hops away from the the starting node d_m, we compute the vector of block densities

$$\gamma = [\rho(\mathcal{B}_1), \rho(\mathcal{B}_2), ..., \rho(\mathcal{B}_M)] .$$

Now, we discard a candidate block $\mathcal{B}$, if it's within two standard deviations of the average block density, that is

$$\frac{\rho(\mathcal{B}) - \mu(\gamma)}{\sigma(\gamma)} \leq 2$$

In the second layer of filtering we discard blocks and nodes that are known to be legitimate. In some cities there exist small communities, that correspond to denser than usual graphs. These cases are covered by a rule-based filter.

D. Interpretation and Action

After the filtering steps are complete, the suspicious blocks are forwarded to human annotators, whose job is to make an appropriate decision. While dense subblocks themselves are in most cases sufficient evidence of fraud and are more interpretable than say a spectral embedding, it is crucial to provide a clear justification when banning people from the service. Therefore, we perform a series of t-tests to compare the behavioral patterns of the suspects to (i) baseline behavior in the given city and (ii) behavior of the suspects in the past. These behavioral patterns include aspects like average rating or average trip duration, which complement the explanation of the fraud activity. The results are visualized in an interactive dashboard that allows annotators to analyze the relevant information and perform the necessary actions in a few clicks.

E. System Architecture

Figure 2 shows the main stages of our anomaly detection pipeline: (1) **Holmes** is a software used internally at Careem that connects to various data sources and creates materialized views of the data. Holmes uses an SQL source code repository to create materialzed views and stores the data using Hive metastore on top of Amazon S3. We use Holmes to construct the raw input tuples in realtime by extracting the data needed and then performing the joins and aggregations required to construct the different variations of the graphs. (2) **Dense subblock discovery** is performed using our own Python implementation of the M-Zoom algorithm. We use arithmetic density as the density measure. (3) **Block Processor**. A daily job runs to calculate an estimate of the average block densities γ for each of the graphs involved. The job also calculates other graph statistics that are used for generating explanations. This way the system stays up-to-date and adapts to the shifting trends in user activity. The Block Processor uses the values calculated to filter blocks and generate block-level explanations. (4) **Node post-processor** consists of a node-level augmenter and a filter. First, the nodes are augmented using features extracted from Holmes. These nodes are then filtered by the rule-based system and compared with other drivers in the city and the node's historical records (using t-tests) without considering the graph information.

IV. Results

Quantitative evaluation of a fraud detection system is challenging because labeled ground-truth datasets are hard to construct for the given setting. As we don't know about all the possible kinds of fraud that might be happening in the system, we are only able to measure the precision of our method. This is done by collecting the feedback from human experts. Currently, our approach has precision above 90% and allows the company to save significant amounts of money that would otherwise be lost to fraudsters. Additionally, we asked experts to perform manual inspection of the discovered anomalies.

When analyzing the results, we discovered a sophisticated class of fraud patterns that we call "captain gangs". Because of the existing rule-based filters it is almost impossible to successfully commit fraud as a single agent in the system. This led to the fraudulent captains uniting into groups, thus making it more difficult for the rule-based system to detect them. An example of such activity can be seen in Figure 3, that shows a two-day activity of a relatively small instance of this fraud pattern. Here, two captains claim to receive extra cash from passengers, which translates into in-app credit for these passenger accounts. This credit is then used to book fake trips with the other four captains. The catch is that the two captains never show up again to pay the amount to the company. Such type of fraud is hard to detect and maintain using rules that do not consider the graph structure.

In fact, such patterns were already manually discovered some time ago, and new threshold-based rules were added to the system to spot it. However, as we realized after analyzing the results of our density-based method, the fraudsters quickly adapted to the rules and continued their activity which led to the coverage of the rule-based system to go down from 70% to nearly 10% with respect to the density approach. It becomes more and more difficult to handcraft effective rules, the more agents are involved in a given pattern. On the other hand, our density-based approach is much better suited for spotting this kind of scam, because such behavior will always be an outlier with respect to the rest of the graph.

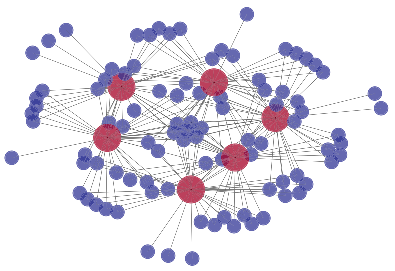
Fig. 3. Snapshot of a captain-passenger graph, illustrating an instance of the "captain gang" fraud pattern. Red nodes - captains, blue nodes - passengers.

V. Conclusion

In this work-in-progress paper we describe a system for detecting fraud in car-booking graphs, that is currently being tested and deployed at Careem. We formulate the problem of anomaly detection as that of discovering dense subblocks in dynamic graphs. One of the key advantages of the proposed approach is its ability to handle different kinds of data and incorporate side information. Our method is able to adjust to the varying conditions in different geographic locations, which is crucial for the business model of Careem. As we saw in the case study in Section IV, while fraudsters quickly adapt to evade the rule-based systems, our density-based framework is able to detect the abnormal behavior.

Acknowledgements

This research was supported by the Technical University of Munich - Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement no 291763, co-funded by the European Union.

References

[1] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, 2015.
[2] H. Tong and C.-Y. Lin, "Non-negative residual matrix factorization with application to graph anomaly detection," in *SDM*, 2011.
[3] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: graph mining using recursive structural features," in *KDD*, 2011.
[4] A. Bojchevski and S. Günnemann, "Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure," in *AAAI*, 2018.
[5] C. C. Aggarwal, "Outlier analysis," in *Data mining*. Springer, 2015.
[6] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Army science conference*, 2010.
[7] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "Graphscope: parameter-free mining of large time-evolving graphs," in *KDD*, 2007.
[8] C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park, "Scan statistics on enron graphs," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 229–247, 2005.
[9] K. Shin, B. Hooi, and C. Faloutsos, "M-Zoom: Fast dense-block detection in tensors with quality guarantees," in *ECML-PKDD*, 2016.
[10] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, "Spotting suspicious behaviors in multimodal data: A general metric and algortihms," *TKDE*, 2016.
[11] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, 2009.
[12] M. Araujo, S. Günnemann, S. Papadimitriou, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra, "Discovery of 'comet' communities in temporal and labeled graphs," *KAIS*, 2016.
[13] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang, "Bipartite network projection and personal recommendation," *Phys. Rev. E*, vol. 76, 2007.