EuroCC@Turkey

https://eurocc.truba.gov.tr/

This document is prepared by EuroCC@Turkey for EuroCC under GA NO 951732

CASE STUDY REPORT

Fraud Detection with Graph Processing

NCC Partner	*Sabanci University*
Company*	*Yapı Kredi Teknoloji* – http://www.ykteknoloji.com.tr/
Expert	*Öznur Taştan* -- *otastan@sabanciuniv.edu* *Kamer Kaya* -- *kamer.kaya@sabanciuniv.edu* *Ahmet Demirelli* -- *ahmet.demirelli@sabanciuniv.edu*
Start & End Date	*March 1, 2021 – March 1, 2022*

***Company** accepts that the Case Study Report is shared with the EuroCC Project and the community through the EuroCC@Turkey awareness creation activities and platforms.

1. Problem Identification

YKY aims to build a machine learning model to detect fraudulent bank transactions. The team has worked previously on detecting credit card fraud detection, and now they are aiming at the problem of general bank transaction detection. The goal is to enable the business unit with better predictions to reduce their human effort and increase the detection rate of accounts that participate in fraudulent transactions. The fraud detection problem is a challenging ML problem due to following observations

i) there is a severe class imbalance, there are very few frauds cases compared to non-fraud transactions
ii) there is also concept drift; fraudsters change their strategies over time
iii) the dataset is large, processing and computing requires high-performance computation.

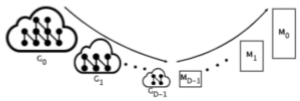
The problem involves building effective ML models. High-performance computing is critical as the experimentation to arrive at a good model requires repeatedly building the models with different algorithms, features, and hyperparameters. They are currently using jupyter-notebooks and Python. To effectively experiment with other models, HPC is critical. We will provide support on general machine learning, graph machine learning, and mostly high-performance computing.

2. First Suggestion

1) We suggest a solution that involves representing the data as a graph; here, Dr. Taştan's expertise on ML on graphs, and Dr. Kaya's expertise on large-scale graph processing will be used. We will use our in-house tool GOSH for graph embedding.

GOSH: Embedding Big Graphs on Small Hardware

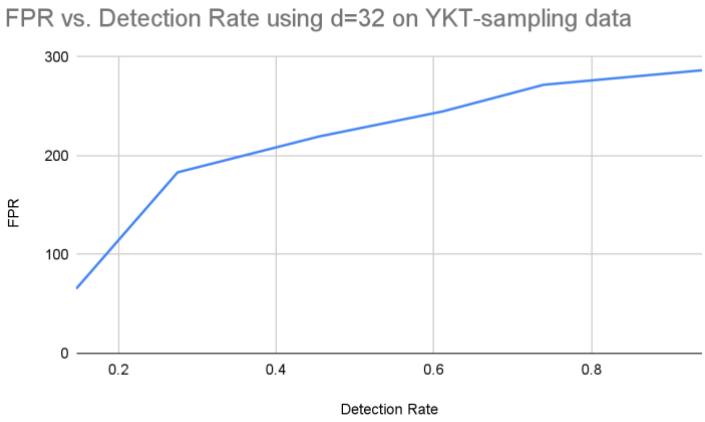
- **An ultra-fast** embedding on a ***single GPU*** for any **arbitrarily large graph**.
  - Provides link prediction AUC ROCs that **match the state-of-the-art** methods.
  - A parallel coarsening **with small overhead** that increases embedding performance
- A **flexible and dynamic task scheduling** for memory restricted devices.



Graph	GOSH	Speedup
Hyperlink (40m vertices, 600m edges)	**0.2 hours** (97% AUC) on a single TITAN X SotA with comparable AUC: **5.4 hours** on 4 Tesla P100	26.7x
Wiki-topcats (1.7m vertices, 28m edges)	**11 seconds** (98% AUC) on a single TITAN X SotA with comparable AUC: : **310 sec.** on a single TITAN X	27.4x

2) We suggest supervised and unsupervised models, providing advice on general machine learning, graph machine learning, and high-performance computing. Due to NDA restrictions, we will not be able to get into the details of these models.

3. Initial Results



After a successful NDA process, we have received the transaction data from YKT with the instructions to recreate their machine learning pipeline, i.e., how to collect the positive and negative samples for the machine learning algorithm. We extracted a graph out of the data and used GOSH (graph embedding tool) that we have implemented, to get vertex-based features for each user in the graph. By using these features, we tested a supervised and an unsupervised machine learning algorithm for anomaly detection; XGBoost and isolation forest, respectively. We found that the unsupervised algorithm performs better. The isolation forest algorithm reaches an 80 percent detection rate with a false positive rate of approximately 250. Our findings only use the topological information extracted from the data. We believe merging graph-based features with the user-specific features extracted by YKT will result in a state-of-the-art anomaly detection algorithm for transactional data.

4. Results and Achievements

Anomaly detection using graph embeddings

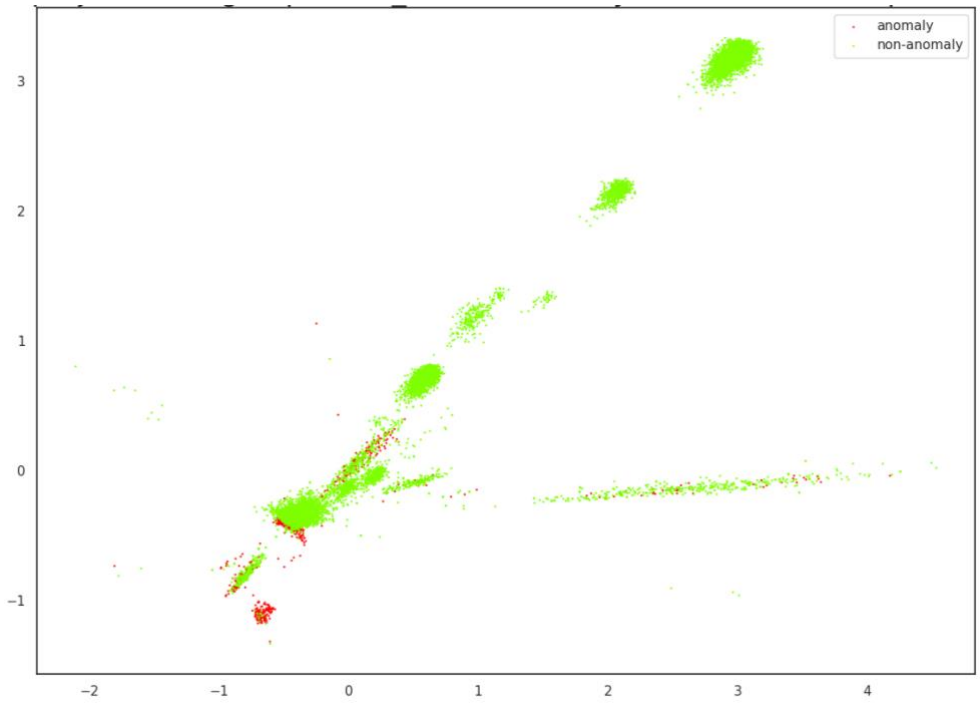
We were able to construct a graph from transaction data spanning seven months, totaling 247 million transactions and 42 million unique users. Using GOSH, our in-house embedding tool, we generated graph embeddings for the graph and used these embeddings to carry out anomaly detection. The process was as follows: Firstly, we generated embedding vectors for each user's node on the transaction graph. Then, we created an embedding vector for each transaction by elementwise multiplying the embedding vectors of the users taking part in the transaction. Finally, we used these transaction embedding vectors as features for the anomaly detection model. We used two different anomaly detection algorithms, a supervised algorithm, XGBoost, and an unsupervised algorithm, Isolation Forest. However, the latter produced better results.

Anomaly detection results using GOSH embeddings with 32 embedding dimensions. Model used is the Isolation Forest model.

False Positive Ratio	Detection Rate
65.36	14.65%
183.17	27.51%
219.53	45.49%
244.86	61.16%
271.79	73.94%
286.50	94.04%

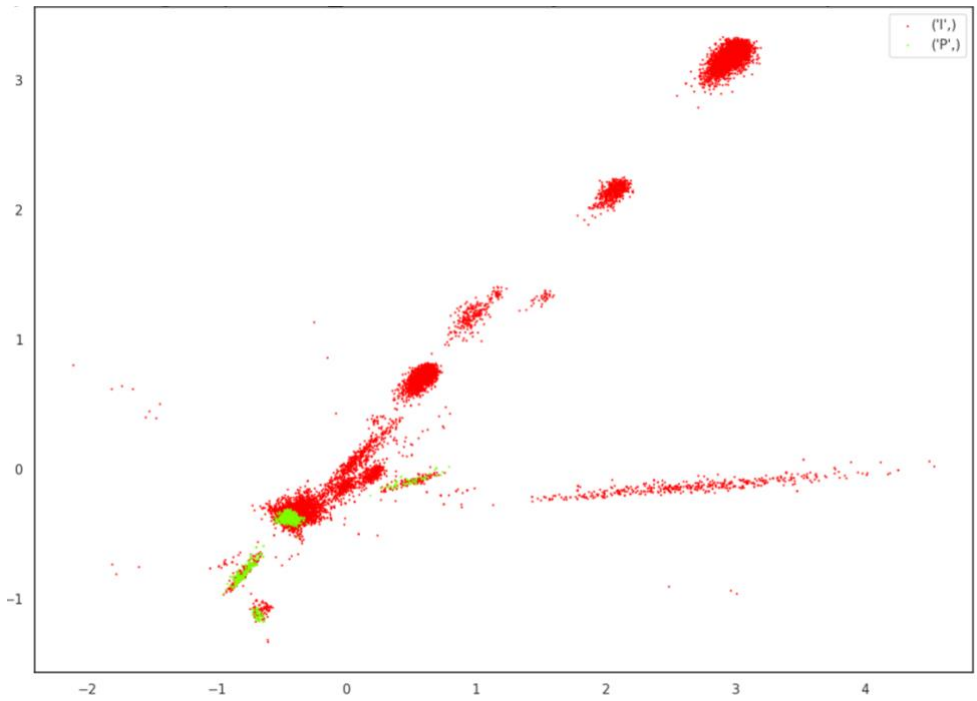
UMAP projections

We further analyzed the node embeddings that gosh produced by visualizing them using the UMAP projection. More specifically, we derived for each transaction an embedding vector by elementwise multiplying the embeddings of its users, and we use UMAP to reduce the dimensionality of that vector to two dimensions. The plot below shows such a projection, where points are separated as anomalies (red) and non-anomalies (green).



UMAP projections of transaction embeddings seperated by anomoulse (red) and non-naomolous (green) points.

We find that, even though the embeddings are not able to clearly separate anomalous from non-anomalous transactions, some form of clustering was occurring in the projections. To further analyze this effect, we try to color these embedding projections according to different transactions properties to see if the clustering aligns with these properties. However, none of the properties available to us seemed to align with the clustering created with UMAP projection. For instance, the following is the same UMAP projection in the previous figure, but the points are colored according to transaction type.



UMAP projections of transaction embeddings. Colors express the transaction type (value meanings are not available due to NDA).

Combining graph embeddings with user features

Even though the structural information by itself did not provide sufficiently high accuracies, we believe that combining it with user-specific features can improve the accuracy of the overall model significantly. However, we did not have access to the exact data features or to the model that the YKT team was using. Additionally, the transaction data we were using in our experiments was different from the data they had been using; ours was outdated.

We requested the latest data, futures, and model from the YKT team to integrate our model into it, but due to access issues, the team was not able to provide what we had requested. However, we provided the team with scripts to carry out the following:

a. Create graphs from transactional data.
b. Sample the graph into sub-graphs used for training and testing.
c. Generate graph embeddings for users.
d. Carry out anomaly detection using the isolation forest and XGBoost models.

Unfortunately, the YKT team was working on many different parallel directions and was not able to integrate graph embeddings into their model.

Data aggregation

The YKT team decided to experiment with certain aggregated features that they extract from the transaction data. These features included averages, minimums and maximums, and other simple aggregations. The team at YKT used a Redis in-memory database to aggregate

the data, and we provided a simple aggregation algorithm to accelerate the process. The algorithm used a simple However, the team at YKT found experimentally that these features were not improving the model's performance at anomaly detection, and so the aggregated features were not pursued further.