Using Graph Embeddings and Hardware Acceleration to Fight Financial Fraud and Improve Healthcare Recommendations

Dr. Victor Lee
Dr. Bill Shi
Parker Erickson

Speaker Bio

Victor Lee Ph.D.
Vice President of ML / AI
TigerGraph

Product leader and educator, with a passion for algorithms, languages, user experience, and ethics. 7 years at TigerGraph, 3 years as university professor, 20+ years in tech industry.

Bill Shi, Ph.D.
Sr ML Solution Architect
TigerGraph

Accomplished in both academia (UNC Chapel Hill, U Chicago) and industry (Amazon). Areas of speciality: network structure, dynamics, and analysis, machine learning.

Parker Erickson
Data Science/ML Intern
TigerGraph

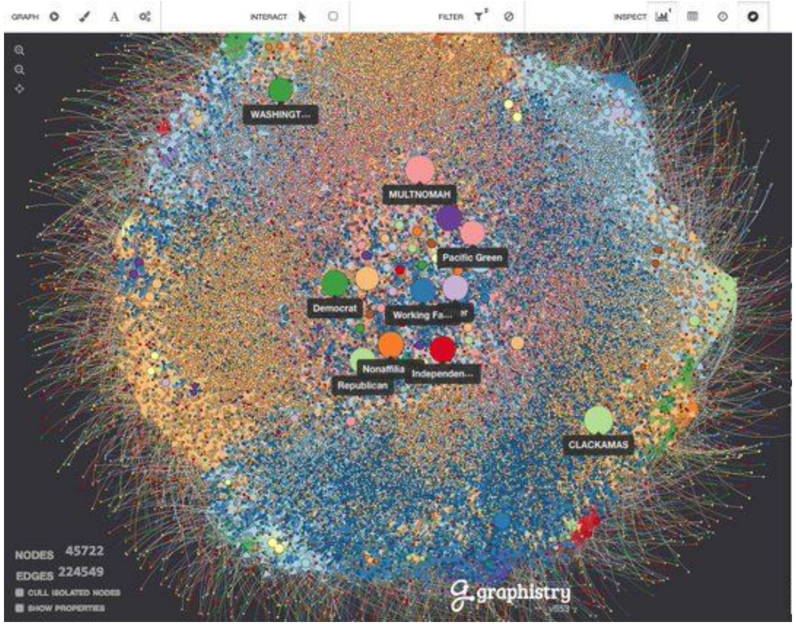
University of Minnesota M.S. Computer Science student. Founding developer of pyTigerGraph. Formerly interned at Optum, working on ML solutions with TigerGraph.

Outline

1. What is a Graph Embedding?
2. Advantages and Use cases
3. Optimizing with Hardware Acceleration
4. Demo - Detecting cryptocurrency fraud
5. Demo - Finding similar healthcare providers

Challenges of Graph ML

Richness of graph data is a double-edge sword:



courtesy: graphistry

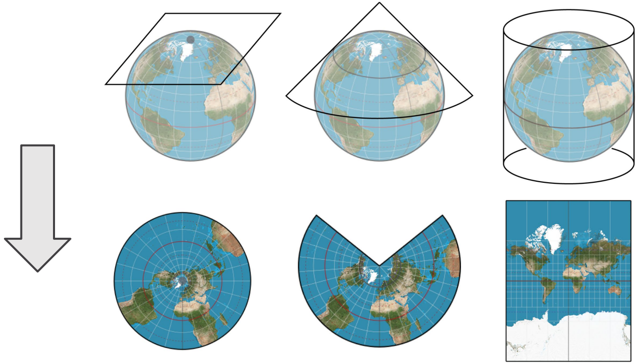
- Expresses a wealth of information
- Full-graph analytics can be expensive
- Conventional ML techniques need matrices, not graphs

$$
\begin{array}{c}
 \\ 1 \\ 2 \\ 3 \\ \vdots \\ m
\end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & \ldots & n \end{array} \\
\begin{bmatrix}
a_{11} & a_{12} & \ldots & a_{1n} \\
a_{21} & a_{22} & \ldots & a_{2n} \\
a_{31} & a_{32} & \ldots & a_{3n} \\
\vdots & \vdots & \vdots & \vdots \\
a_{m1} & a_{m2} & \ldots & a_{mn}
\end{bmatrix}
\end{array}
$$

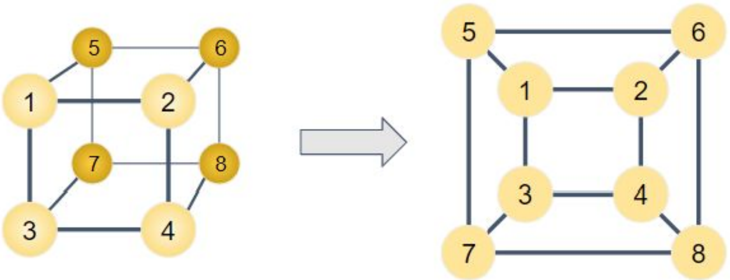
Enter: Embedding

Embedding transforms high-dimensional data into a lower-dimension.

- May not preserve 100% of details, but captures what is most important
- Tradeoff between accuracy, format, and efficiency



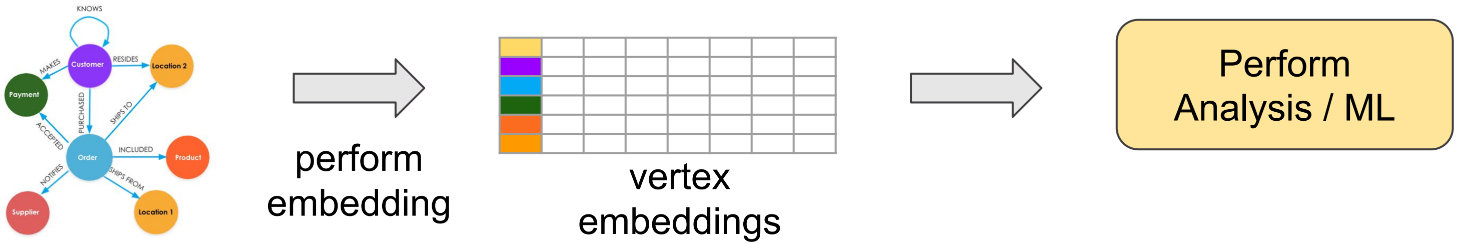
Map Projections. DOI: 10.22224/gistbok/2017.2.7



Two examples of embedding a 3-D object into 2-D space.

Using Graph Embeddings

- ***Graph Embedding*** transforms graph structure into a compact set of vertex vectors.
  - Captures the essence of a vertex's "nature" as a set of latent features
  - Enables graph data to run efficiently on non-graph neural networks



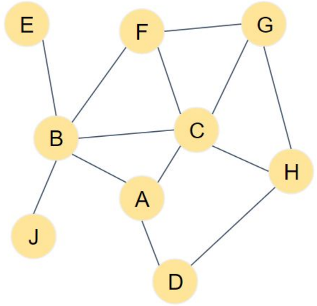
Works for numerous cases

- Recommendation (similarity)
- Fraud detection (classification)

Compact → **scalability**

Set of vectors → **compatibility**, **reduced complexity**

DeepWalk Embedding

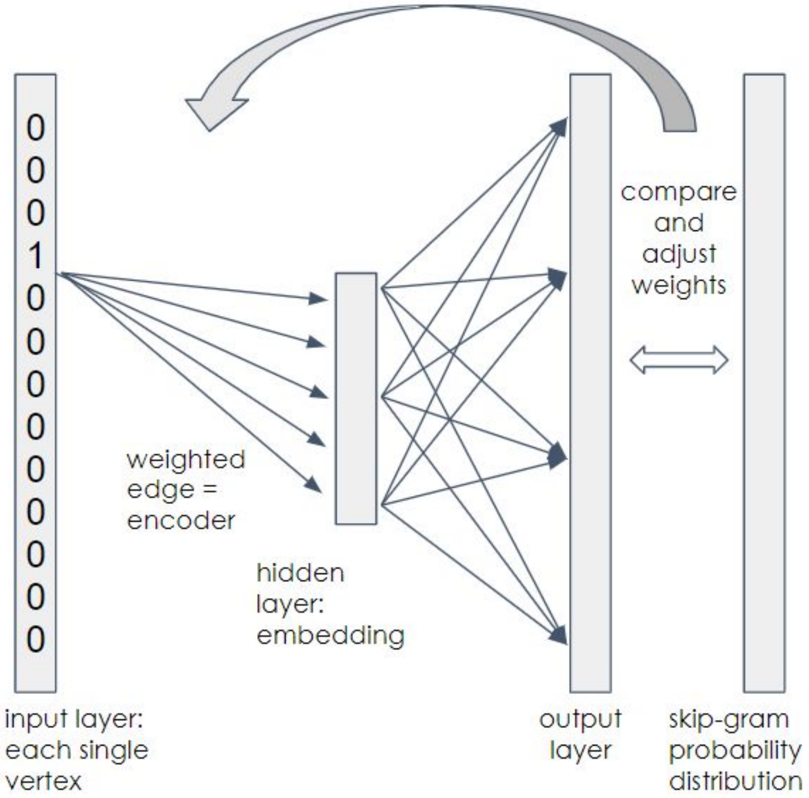


C	A	B	J	B	F	G	C	H	G	F	B	A	C	F	C

(a) random walk, λ = 16, w = 2

A	B	C	D	E	F	G	H	J
2	2	4	0	0	2	2	1	0

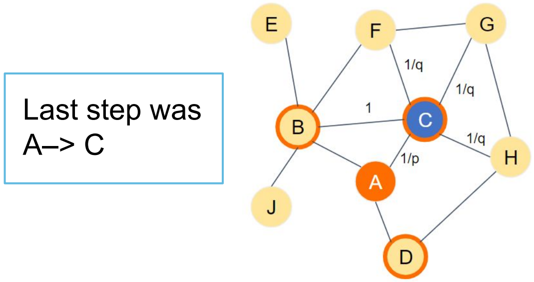
(b) corresponding skip-gram



Improvements: Node2Vec and FastRP

Node2Vec

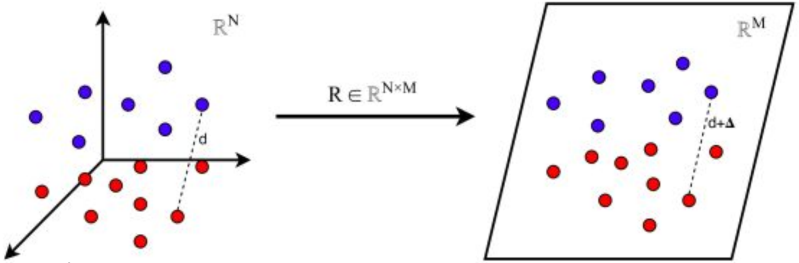
- **Goal**: Improved accuracy, semantics
- **Intuition**: Neighbor exploration isn't random
- **Answer**: biased random walk
  3 types of steps, with different probabilities:
  1. backwards (retrace your step)
  2. breadth (a neighbor of where you were)
  3. depth (a new neighbor)



- Con: Slower than DeepWalk

FastRP

- **Goal**: Scalability, size and speed
- **Intuition**: skipgrams compute a vertex similarity matrix that is n^2 (non scalable)
- **Answer**: use sparse random projection to go directly to a n × d matrix, where d << n



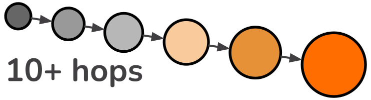
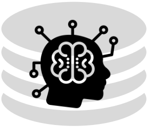
courtesy:
https://www.groundai.com/project/random-projections-of-mel-spectrograms-as-low-level-features-for-automatic-music-genre-classification/1

- **Con**: May have reduced accuracy, too much sparsity

Optimizing Graph Analytics

1. Graph Database & Analytics Platform
2. Accelerated Graph Analytics
3. Compute Server

1. TigerGraph: Scalable Graph Platform

Feature	Design Difference	Benefit
Real-Time Deep-Link Querying  5 to 10+ hops	• Native Parallel Graph design • C++ engine for high performance • Storage Architecture	• Uncovers hard-to-find patterns • Operational, real-time • HTAP: Transactions+Analytics
Massive Scale 	• Distributed DB architecture • Massively parallel processing • Compressed storage reduces footprint and messaging	• Integrates all your data • Automatic partitioning • Elastic scaling of resource usage
In-Database Analytics & Machine Learning 	• GSQL: High-level yet Turing-complete language • User-extensible graph algorithm library, runs in-DB • ACID (OLTP) & Accumulators (OLAP)	• Avoids transferring data • Richer graph context • Graph-based feature extraction for supervised machine learning • In-DB machine learning training

TigerGraph In-Database
Graph Data Science Library

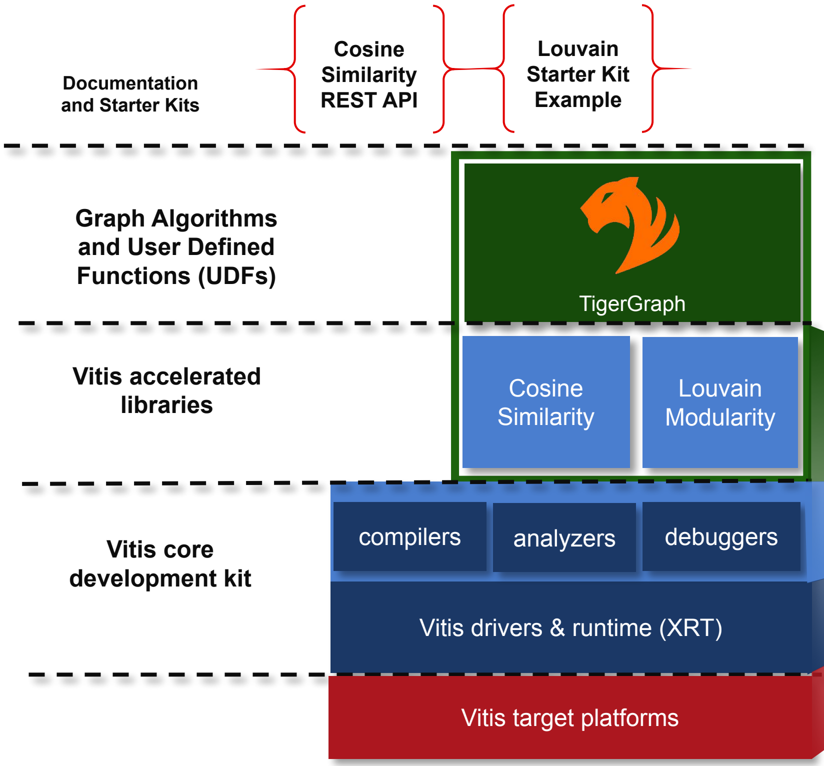
Signals our commitment to serving the needs of data scientists

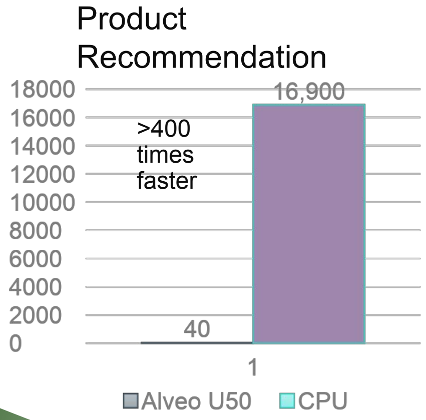
- More algorithms (15 released this week)
  - **Graph Embedding - node2vec, fastRP**
  - Topological link prediction
  - Similarity
  - Centrality
- More than just algorithms

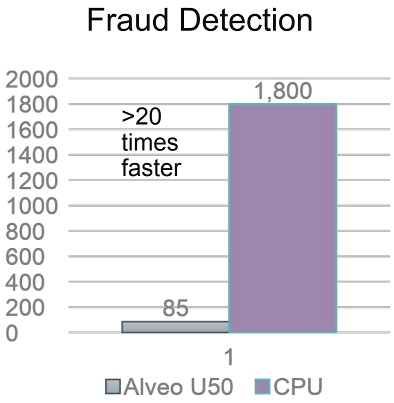
In-Database
- No export needed
- Live, updatable data
- Scaleable, Ultra-fast engine
- GSQL query language

For Data Scientists
- easier and faster to run
- include ML, such as graph embeddings
- will integrate with feature & model management
- will integrate with Graph+ML Workbench

2. XILINX FPGA-based Graph Analytics Acceleration







Cosine Similarity TCO 12X Cost Reduction

- Target Performance
  - <100ms latency
  - 100 queries/second
  - 15M patients

- TCO
  - w/o Alveo: $1,092,972
  - With Alveo: $82,266

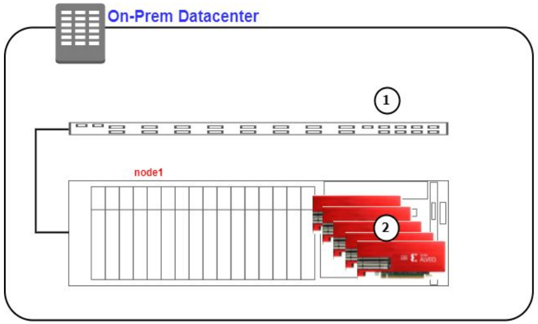
- Savings: $1M

Feature	Tigergraph	Tigergraph + U50
Server Configuration	Xeon-Platinum 8153 2x16cores/socket	Xeon-Platinum 8153 2x16cores/socket
Measured latency per query (ms)	1584	33
# of queries/sec can be achieved per system	0.63	30
# of systems required to meet the target perf	159	4
# of accelerator cards required per server	-	5
Total # of servers required	159	4
Server power without PCIe cards (W)	700	700
Power per solution (W)	700	1,000
Total Cost of Acquisition (TCA)	$636,000	$56,000
Maintenance Cost (10%) per yr for 3yr	$190,800	$16,800
Total Power (KW)	111	4
3yr Power Cost ($0.07KWH)	$204,747	$7,358
Datacenter PUE	1.30	1.30
3yr Cooling Cost	$61,424	$2,208
Total Cost of Ownership (TCO)	$1,092,972	**$82,366**
TCO Savings		$1,010,606

3. HPE REFERENCE ARCHITECTURE FOR SINGLE NODE

Proliant DL385 Gen10 Plus server v2

ARCHITECTURE



(1) **10g Switch**

This can be dedicated switch or part of corp network with isolation used for server access.

(2) **U50 CARD for acceleration**

This is used as acceleration card for graphDB and for communication between the cards

SOLUTION COMPONENTS

Hardware:

- 1 x HPE ProLiant DL385 Gen10 Plus server v2 (2 x AMD EPYC 7713, 64 cores / processor, base frequency 2.0 GHz, 256 MB L3 cache,
  16 x 32 GB DIMMs)
- 5 x Xilinx Alveo U50 Cards (SFF, 100 Gbps networking I/O, PCIe Gen4, and HBM)
- Storage: 6 TB minimum up to 60 TB (depends on the size of the database); SATA or NVMe

Software:

- OS: Ubuntu 20.04.1 LTS
- TigerGraph 3.1 Enterprise Edition
- HPE system ROM: A42 v1.24 April 27, 2020 or later
- HPE iLO 5 version 2.15 pass6 or later
- HPE iLO Advanced Platform Management Link (APML) version 2.11.00.24 or later

SOLUTION RESULTS

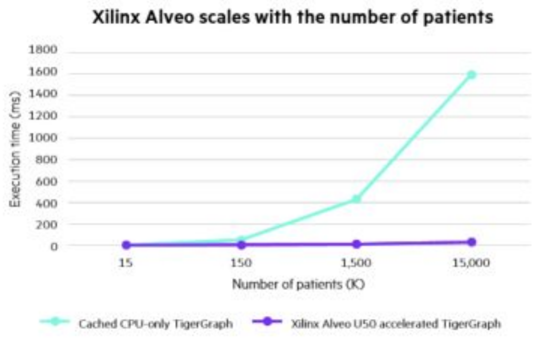
Use case: Patient360 / customer analytics

Data set: Synthetic patient data generated by Synthea™

Algorithm: Cosine similarity

Results:

- Optimized algorithm
- Reduced computational burden
- Improved prediction performance
  – 48x faster compared to CPU-only solution for 15 million patients[1]
  – Linearly flat versus exponential growth



[1] Based on HPE internal testing done on HPE hardware in May 2021.

Hardware Acceleration

- HPE REFERENCE ARCHITECTURE FOR ACCELERATED GRAPH ANALYTICS
  - HPE ProLiant DL385 Gen10 Plus v2 server
  - Xilinx Alveo U50 Data Center Accelerator (x7)
  - TigerGraph Analytics Platform

Hewlett Packard Enterprise

TigerGraph

XILINX

Graph Embedding Demos

1. Predict fraud, from a cryptocurrency transaction graph

2. Identify similar healthcare providers, from a Provider-Specialization graph

Detecting Cryptocurrency Fraud with Graph Embeddings

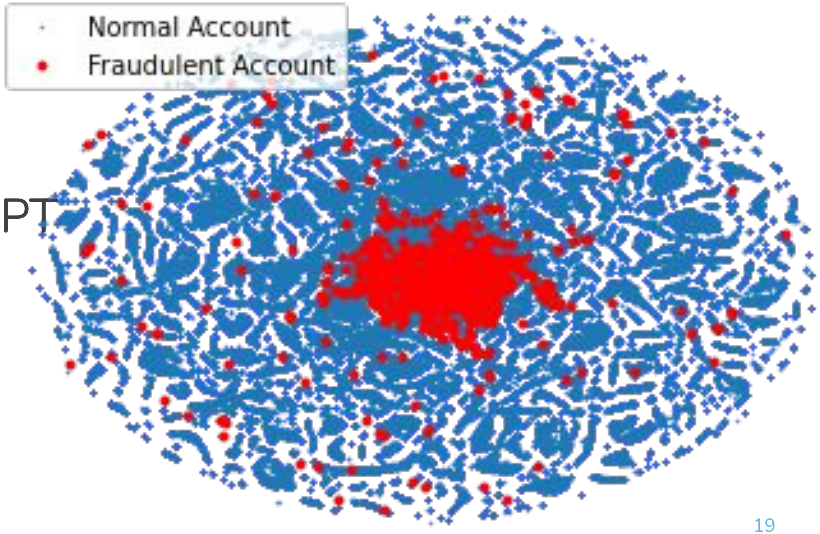
Overview

Ethereum Transactions → TigerGraph → Graph Embedding → Neural Network → Fraud Prediction

Graph Embedding Algorithm

Data

- Ethereum: platform of the second largest cryptocurrency, Ether (ETH).
- Transaction network of Ethereum
  - Vertices: wallets, i.e., accounts on the platform
  - Edges: transactions between the accounts
- Statistics
  - 2,973,489 vertices
  - 5,355,155 edges
  - 1,165 phishing (fraudulent) vertices
- Data source: http://xblock.pro/ethereum/#EPT

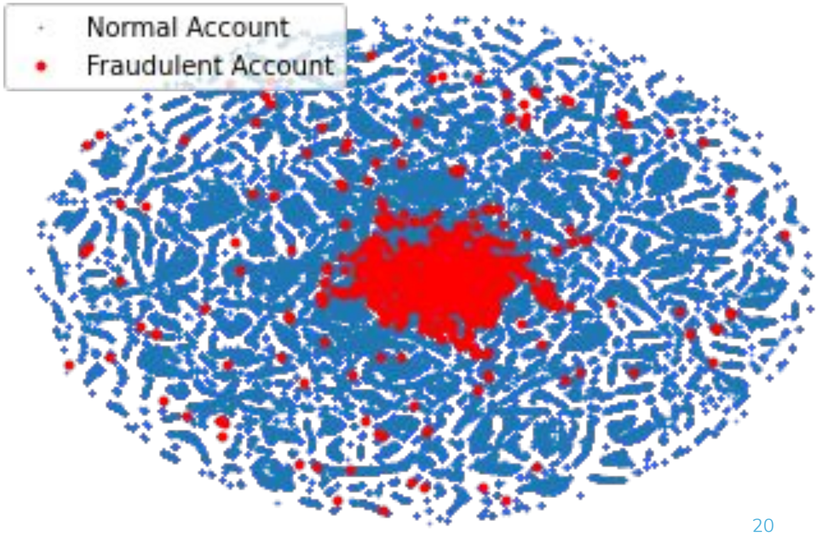


Method

- Goal: Predict phishing accounts in the transaction network

- Traditional approach
  - Rule based: if ... then ...
  - Manually created features + ML

- Our approach
  - Phishing accounts might share similar network structures
  - Graph embedding + ML



Method

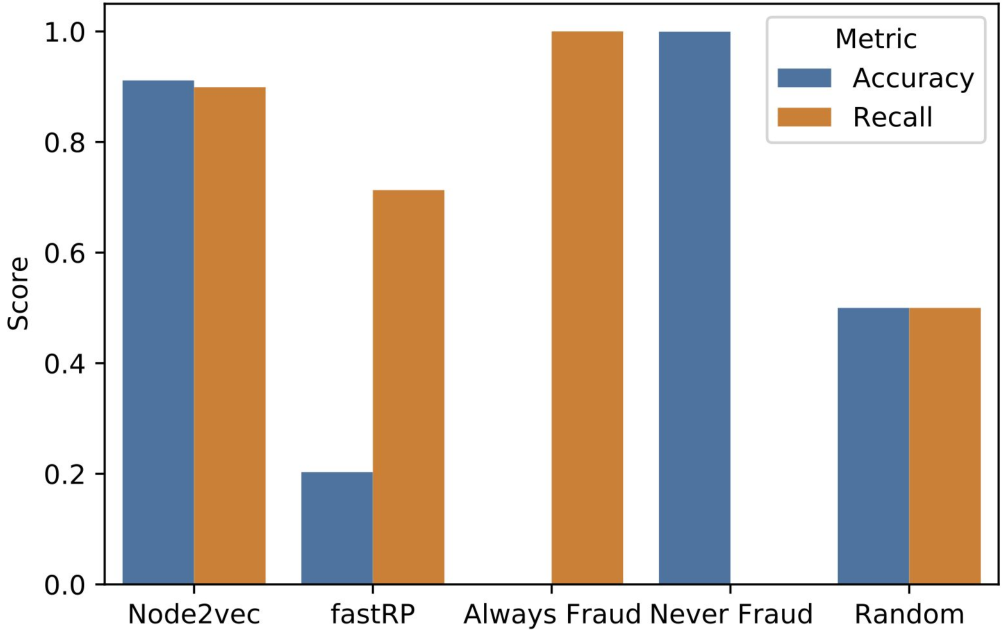
- Graph embedding algorithms
  - Node2vec
  - FastRP

- Neural network model
  - Input: node embedding
  - Output: whether a node is a phishing account
  - 3 fully connected layers
  - RELU activation
  - Cross entropy loss

Results

Predictive Performance

- Node2vec embedding
  - 91% accuracy
  - 90% recall
- fastRP embedding
  - Low accuracy
  - High recall



Demo

Github repo:
https://github.com/TigerGraph-DevLabs/detect-cryptocurrency-fraud

Healthcare Provider Recommendation with Graph Embeddings and Hardware Acceleration

Overview

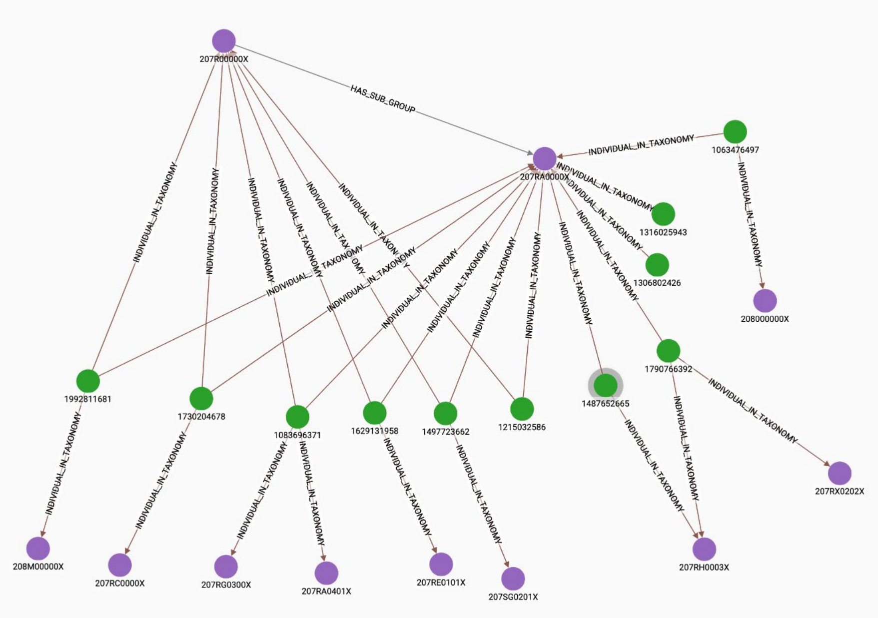
Provider Data → Hewlett Packard Enterprise / XILINX → Graph Embedding → Hewlett Packard Enterprise / XILINX → Cosine Similarity

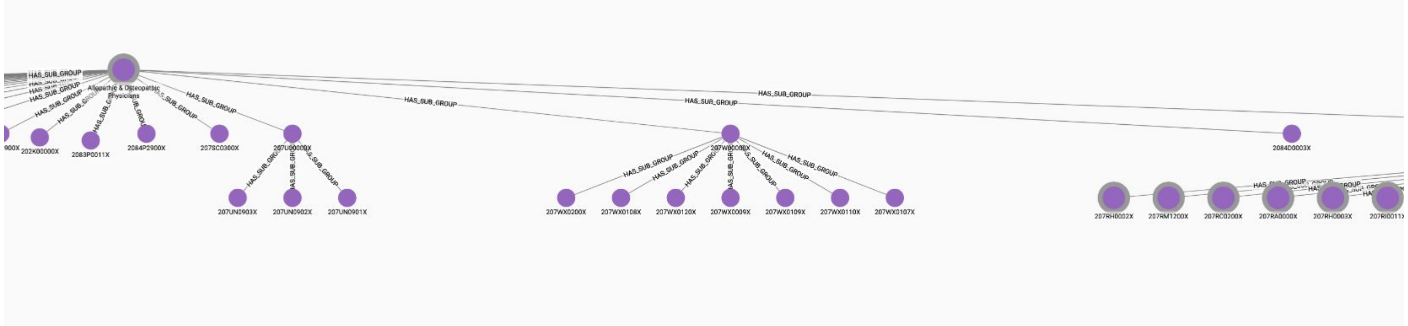
Graph Embedding Algorithm

Accelerated Cosine Similarity Algorithm

Data

- All Healthcare Providers have a National Provider Identifier (NPI)
- NPIs are associated with a provider's specialty
- Specialties are arranged in a taxonomic hierarchy





Method

- Create FastRP embeddings for each vertex in the graph
- Cache embeddings in Xilinx Alveo U50 Data Center Accelerator (x7)
- Given an input embedding, compare against 5.3 million others
  - Use Cosine Similarity
  - 200-dimensional embedding vector

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$