Anomaly Detection for Finance

Leman Akoglu
lakoglu@andrew.cmu.edu

June 25, 2024

Carnegie Mellon University
Heinz College

CARNEGIE MELLON UNIVERSITY
PITTSBURGH
1900

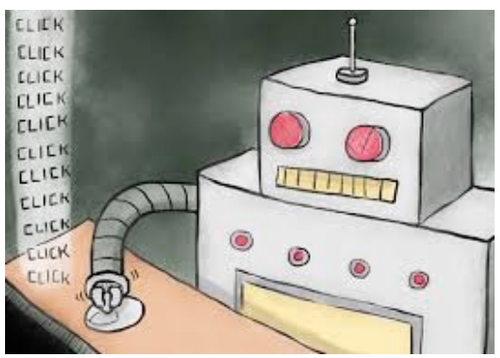
Anomaly Detection at Work

Monitoring

Security

amazon yelp tripadvisor Expedia

Opinion fraud



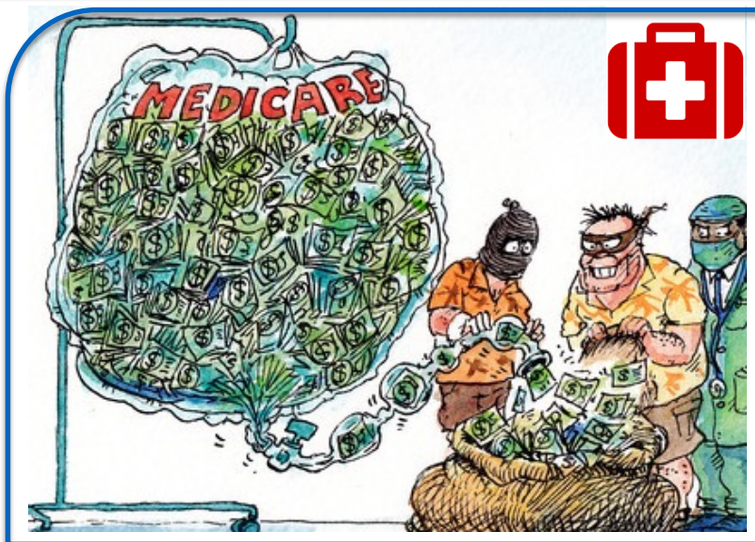
Ad fraud

Fake posts

Banking

Accounting

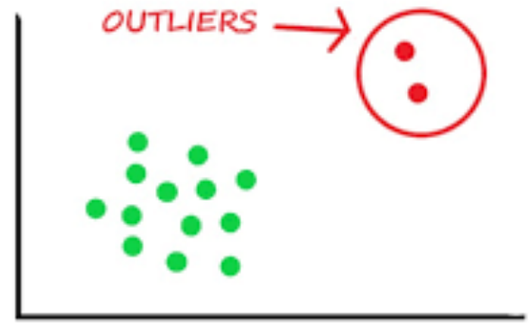
Tax



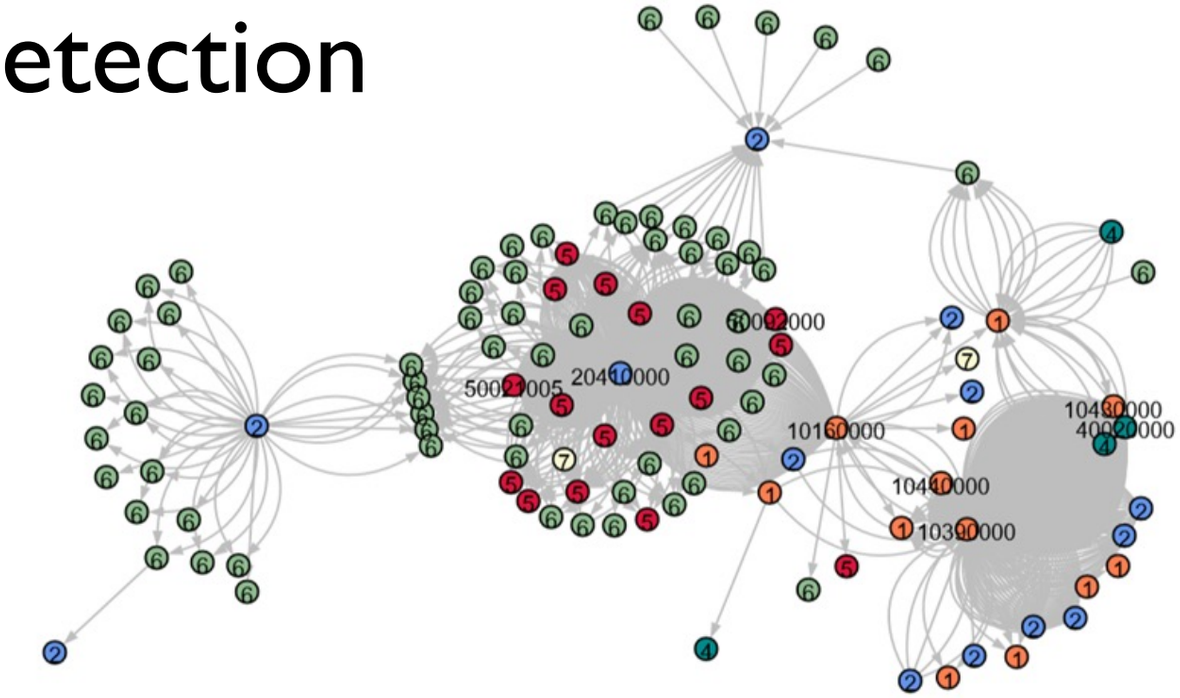
Claims

Anomaly Detection for Finance

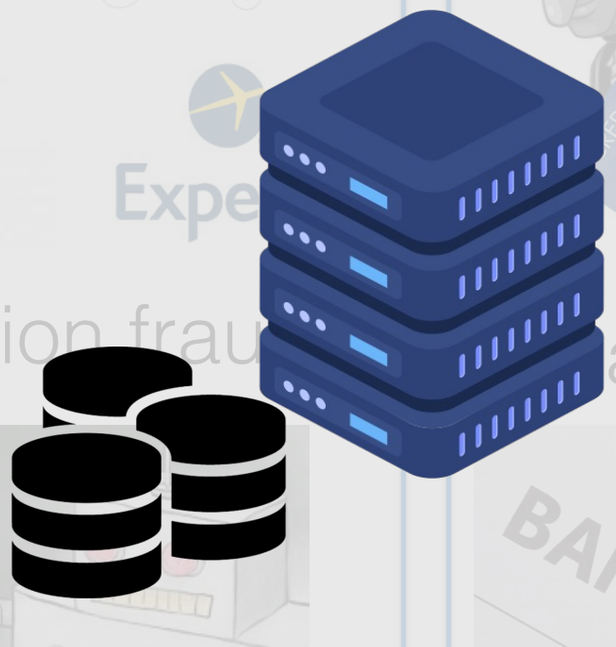
Distributed outlier detection [Tabular data]



- Relational anomaly detection [Graph data]



Financial data is <u>often</u> cloud-resident

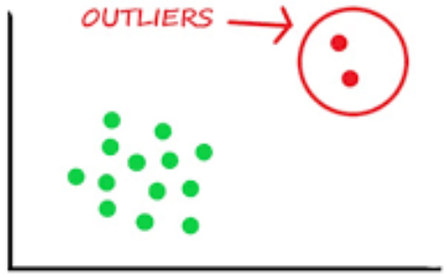


How can we design an outlier detection algorithm that is easy to distribute?

without resorting to subsampling, local model avg.'ing, etc.

Motivation: Distributed OD

- Outlier detection has extensive literature
  BUT ***distributed* OD under-studied**



	small d	**Large d**
small n	Many algorithms	SPIF[2]
Large n	DBSCOUT[3]	Sparx

- Open-source algorithms:
  [1] DDLOF (uses Hadoop), [2] SPIF (distributed Isolation Forest),
  [3] DBSCOUT (distributed DBSCAN)

[1] Yizhou Yan, Lei Cao, Caitlin Kulhman, Elke Rundensteiner. KDD 2017. Distributed local outlier detection in big data.
[2] https://github.com/titicaca/spark-iforest
[3] Matteo Corain, Paolo Garza, and Abolfazl Asudeh. ICDE 2021. DBSCOUT: A Density based Method for Scalable Outlier Detection in Very Large Datasets.

Key Points

Sparx *is a scalable open-source tool for outlier detection*

✓ Distributed, data-parallel detection

✓ Scalable to massive data – linear time/space complexity

✓ Handles not only Large n but also Large d

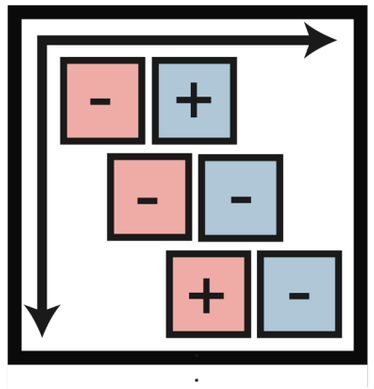
✓ Handles mixed-type attributes

✓ Robust to hyperparameter settings

✓ Open-source Apache Spark-based framework

Challenges

- Large number of points
- Large (**possibly evolving**) feature-space
- Fast streaming pts, limited memory

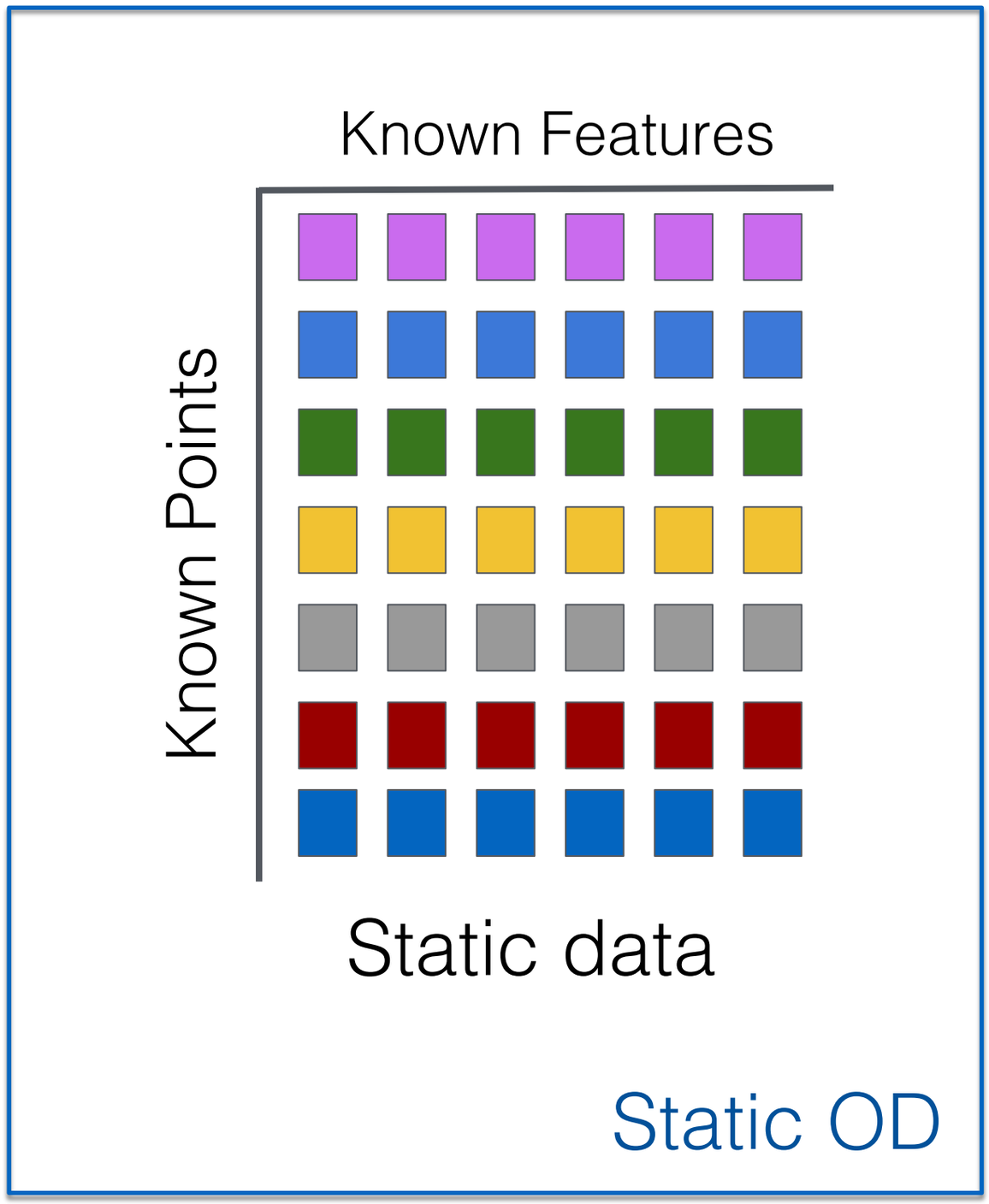


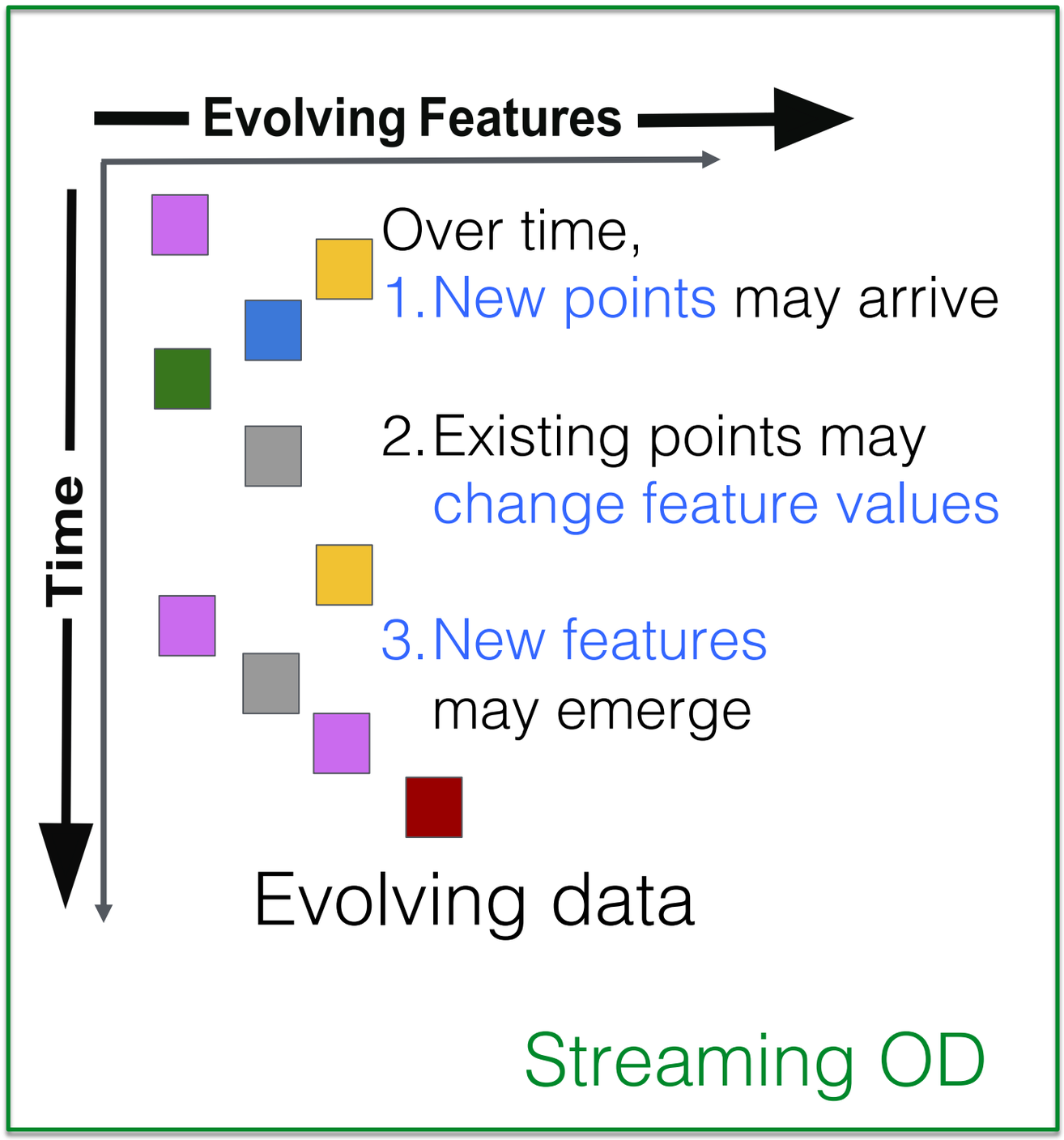
PROPERTIES	STORM	HST	LODA	RS-HASH	RS-FOREST	xSTREAM
MULTI-SCALE						✓
SUBSPACES		✓	✓	✓	✓	✓
PROJECTIONS			✓			✓
EVOLVING POINTS						✓
EVOLVING FEATURES						✓



xStream: Outlier Dete‘x’ion in Feature-Evolving Data Streams.
Emaad A. Manzoor, Hemank Lamba, Leman Akoglu.
ACM SIGKDD 2018

Problem Definition

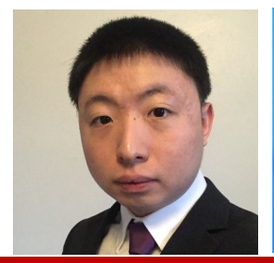




Sparx: Distributed xStream

Three stages:

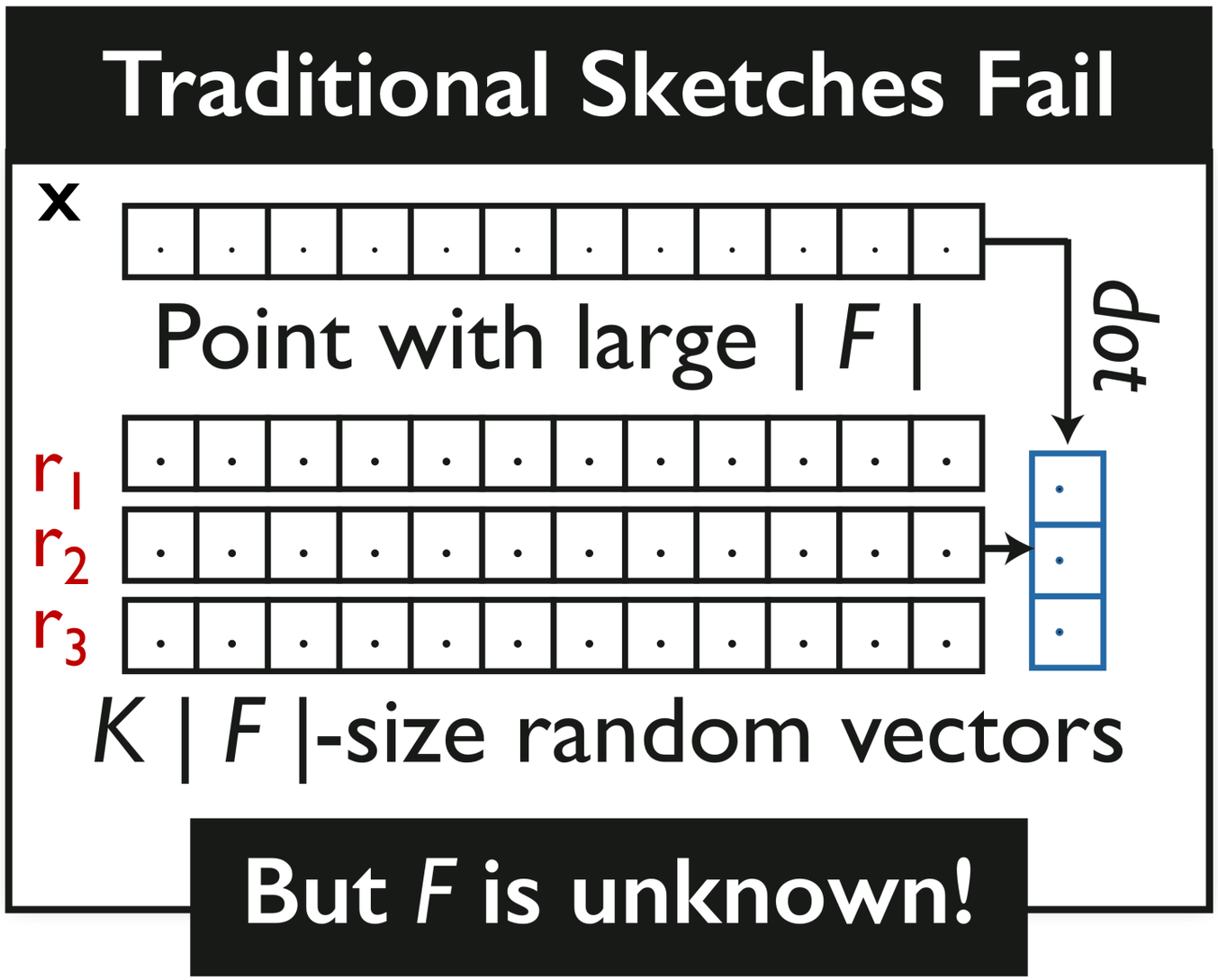
1. Projection/sketching
2. Bin-counting
3. Anomaly scoring



Sparx: Distributed Outlier Detection at Scale
Sean Zhang, Varun Ursekar, Leman Akoglu
ACM SIGKDD 2022

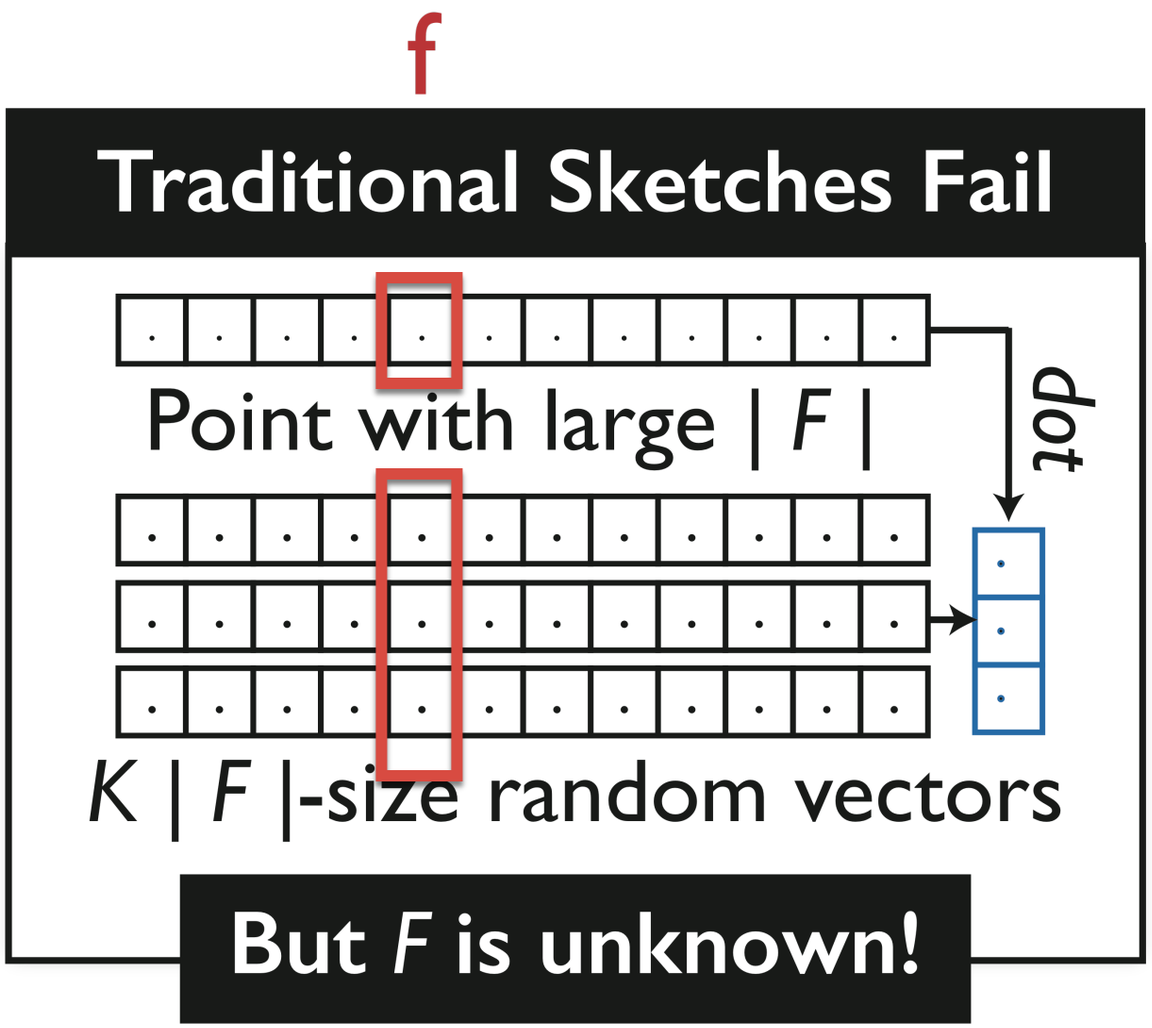
Stage 1: Sparse Evolving-data Sketches

- Typical approach to high-dimensional data: sketching



Stage 1: Sparse Evolving-data Sketches

- Typical approach to high-dimensional data: sketching



Idea: don't cache, hash!

$h_i(f): f \rightarrow \{+1, 0, -1\}$

$h_1 \ldots h_K$ take **constant space!**

Constant-time Point Updates

Stream update: (id, f, δ)

$$\begin{bmatrix} 0.2 \\ -0.4 \\ -1.0 \end{bmatrix} \text{ +/- } \begin{matrix} h_1(f) \\ h_2(f) \\ h_3(f) \end{matrix} \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} \times \delta$$

Projection of point *id*

Hash updates of feature *f*

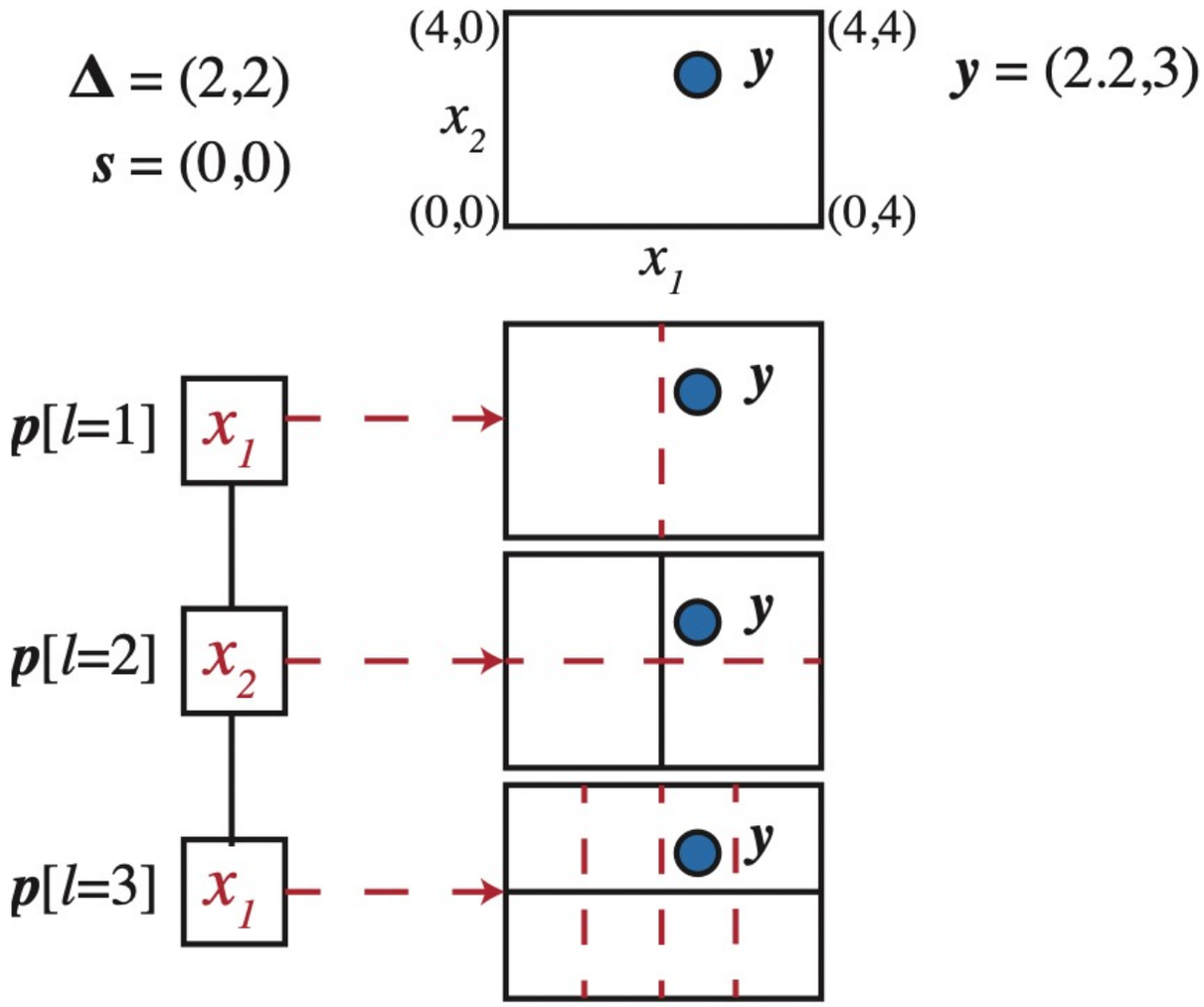
Sparx: Distributed xStream

Three stages:

1. **Projection/sketching**
   - ➢ K hash functions passed to workers
   - ➢ **map**: pass each point through $h_1 \ldots h_K$
   - ➢ Fully-local (no **reduce**/network communication)

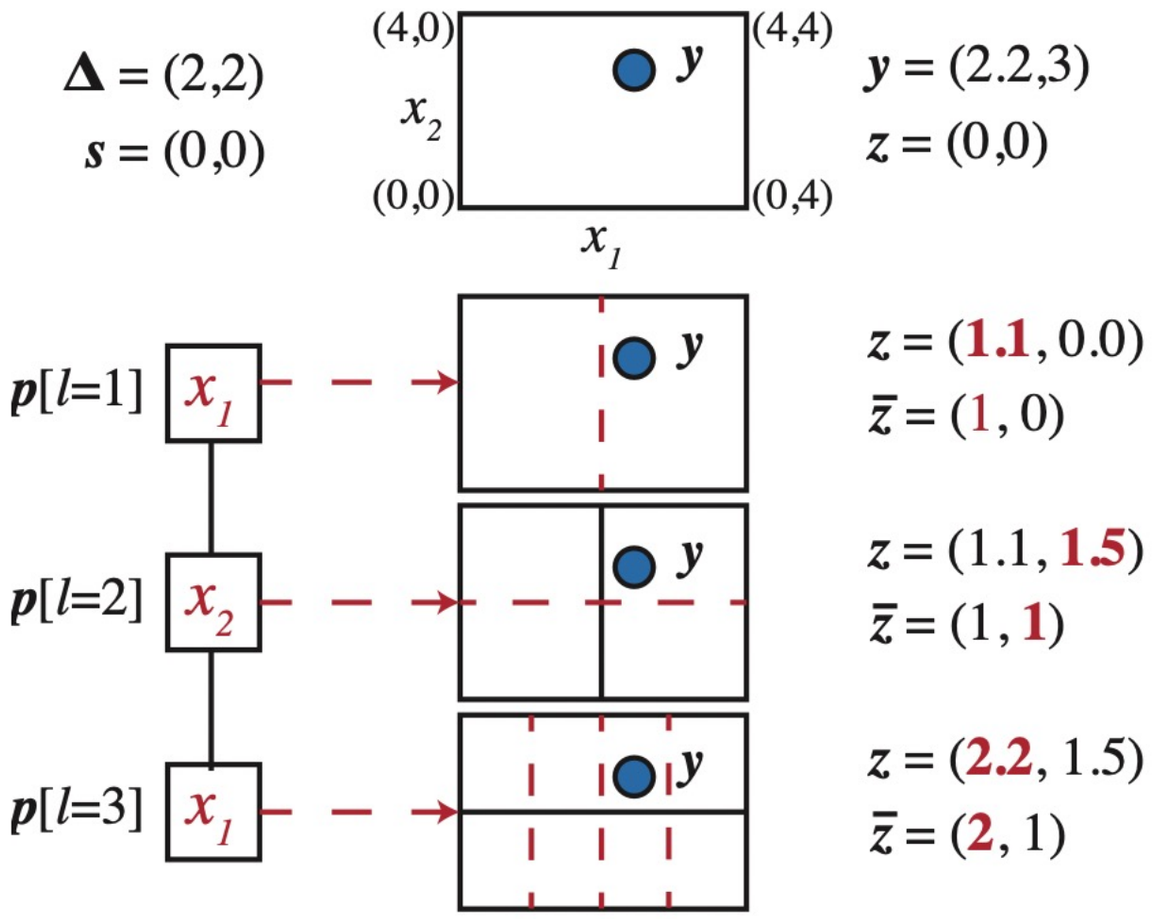
Stage 2: Multi-scale Density Estimation

- Anomaly detection by density estimation at multiple scales
- Half-space Chains w/ depth D
  - Randomly pick **split-dimension** $p \in \mathcal{P} = \{1, \ldots, K\}$ at each level $\ell = 1, \ldots, D$
  - Recursively split each bin along split-dimension in half



Stage 2: Multi-scale Density Estimation

- Anomaly detection by density estimation at multiple scales
- Half-space Chains w/ depth D
  - Randomly pick **split-dimension** $p \in \mathcal{P} = \{1, \ldots, K\}$ at each level $\ell = 1, \ldots, D$
  - Recursively split each bin along split-dimension in half
  - Let $\bar{\boldsymbol{z}} \in \mathbb{Z}^K$ denote **bin-vector**
  - Index $\bar{\boldsymbol{z}}$ into a **counting data structure** $\mathbf{H}_\ell$, e.g. (mxL) count-min-hash, at $\ell = 1, \ldots, D$



$$\boldsymbol{z}[p] = (\boldsymbol{y}[p] + \boldsymbol{s}[p])/\boldsymbol{\Delta}[p] \quad \text{if } o(p,l) = 1$$
$$\boldsymbol{z}[p] = 2\boldsymbol{z}[p] - \boldsymbol{s}[p]/\boldsymbol{\Delta}[p] \quad \text{if } o(p,l) > 1$$
$$\bar{\boldsymbol{z}} = \lfloor \boldsymbol{z} \rfloor$$

Sparx: Distributed xStream

Three stages:

1. Projection/sketching
2. **Bin-counting**
   - ➢ Having been passed $C = \{\boldsymbol{p}, \boldsymbol{\Delta}, \boldsymbol{s}, \mathcal{H}\}$ for each chain C,
     map: output key-value pair (bin-vector $\bar{\boldsymbol{z}} \in \mathbb{Z}^K$, 1)
     per point at each level
   - ➢ **reduceByKey** with sum func. (total count per bin)

Stage 3: Anomaly Scoring

- Train an **ensemble** of Half-Space Chains

$$\mathcal{C} = \{C_1 \ldots, C_M\}$$

where each $C = \{\boldsymbol{p}, \boldsymbol{\Delta}, \boldsymbol{s}, \mathcal{H}\}$

- Outlier scoring:

$$S(\boldsymbol{y}) = \frac{1}{M} \sum_{C \in \mathcal{C}} S_C(\boldsymbol{y}) = \frac{1}{M} \sum_{C \in \mathcal{C}} \min_l 2^l H_l[\bar{\boldsymbol{z}}]$$

Sparx: Distributed xStream

Three stages:

1. Projection/sketching
2. Bin-counting
3. **Anomaly scoring**

➢ Having been passed dictionary per level per chain, single-pass **map** to score each sample

Experiments

- 3 large public datasets, 3 different size settings

Name	*n* pts.	*d* dim.	size (GB)	type	outl.
Gisette	40,000	4,971	4.69	small-*n*/large-*d*	10%
OSM	2,772,233,904	2	51.50	large-*n*/small-*d*	0.036%
SpamURL	2,396,130	3,231,962		large-*n*/large-*d*	33%

- Baselines: two open-source distributed OD
  1. SPIF : *model-parallel Spark version of Isolation Forest*
  2. DBSCOUT: *data-parallel Spark version of DBSCAN*

	#partitions	driver memory	exec memory	#execs	#exec cores	#threads
`config-mod`	64	25GB	4GB	4	4	4
`config-gen`	128	45GB	8GB	64	8	128

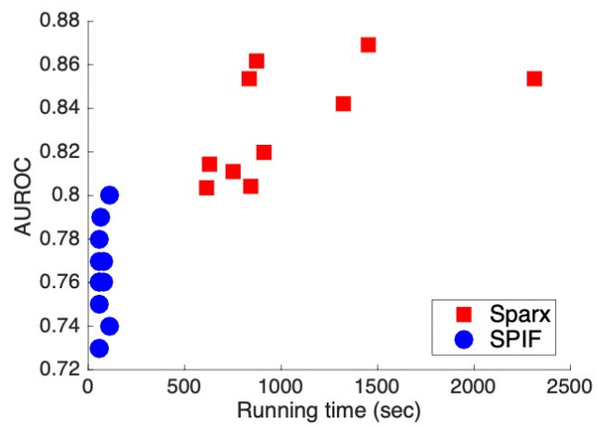
Gisette: small n / Large d

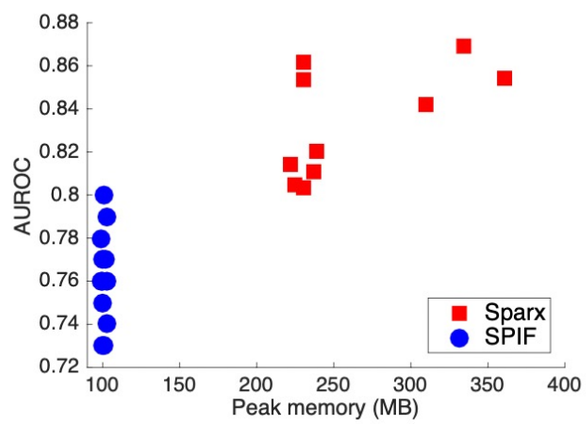
- n = 40,000; d = 4971
  Outlier% = 10%
- (!) **DBSCOUT** scales poorly with dimension d

d dim.	Runtime (sec)	Peak memory (MB)
2	11.3	1,650
4	13.0	1,630
6	31.1	133,000
8	429.8	254,000
10	3,420.0	350,000
11	TIMEOUT	N/A

- **Sparx** outperforms **SPIF** in AUROC for various HPs
- Data-parallel **Sparx** has runtime/memory vs. performance tradeoff

				AUROC		Time(s)		Mem (MB)	
conf.	#comp.	sampl.	depth	Sx	DIF	Sx	DIF	Sx	DIF
1	50	0.01	10	0.80	0.77	610.5	58.0	230.2	99.6
2	100	0.01	10	0.85	0.76	836.0	58.0	230.7	99.0
3	100	0.1	10	0.86	0.79	874.2	61.1	229.9	102.8
4	100	0.1	20	0.87	0.78	1455.6	60.1	334.5	99.0
5	100	1	20	0.85	0.80	2312.8	111.0	360.8	101.1





OSM: Huge n / small d

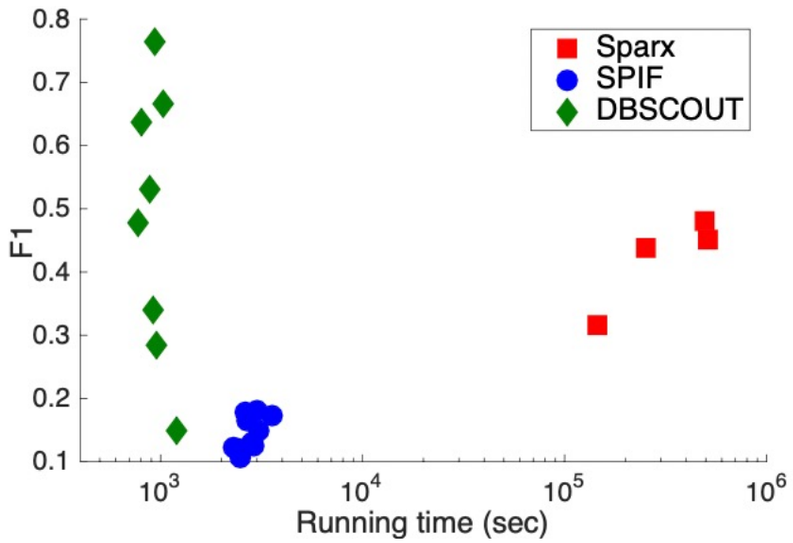
- n ~ 2.7 **billion**; d = 2
  Outlier% = 0.036%

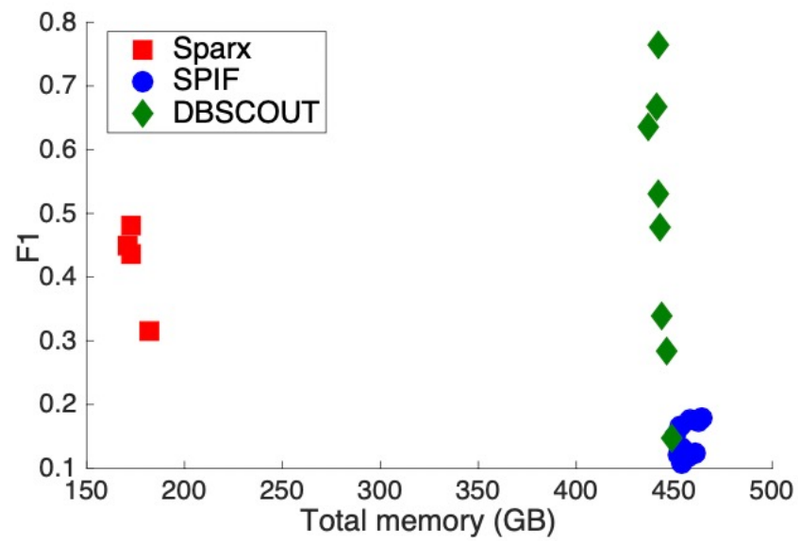
Frac.	#pts/tree	Time (s)	Mem (GB)	AUPRC	AUROC
0.00128	35,471	1396	454	0.19	0.987
0.00256	70,943	1402	455	0.27	0.989
0.00512	141,887	1531	461	0.38	0.991
0.01024	283,774	1834	463	0.42	0.993
0.02048	567,548	MEM ERR	-	-	-
0.04096	1,135,097	MEM ERR	-	-	-
0.08192	2,270,194	TIMEOUT	-	-	-
0.16384	4,540,389	TIMEOUT	-	-	-

- Model-parallel **SPIF** scales poorly with input size n
  - ➢ AUPRC increases with sample size per tree - but leads to memory/timeout errors

- **DBSCOUT** performs well for small d - but is sensitive to HPs

- **Sparx** takes longer to run - but has a lower memory footprint





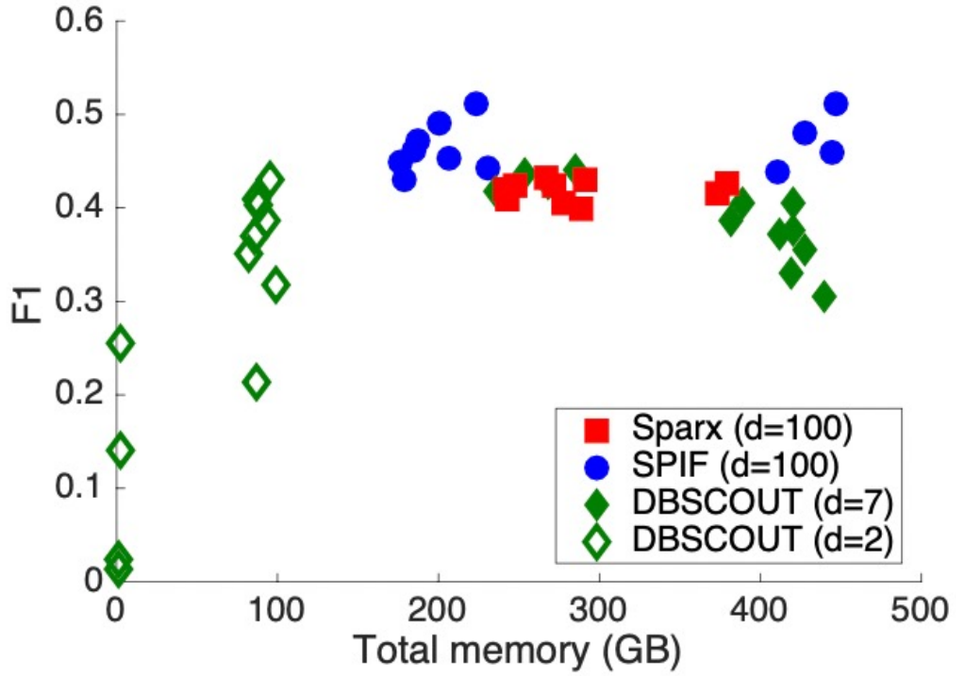
SpamURL: Large n / Large d

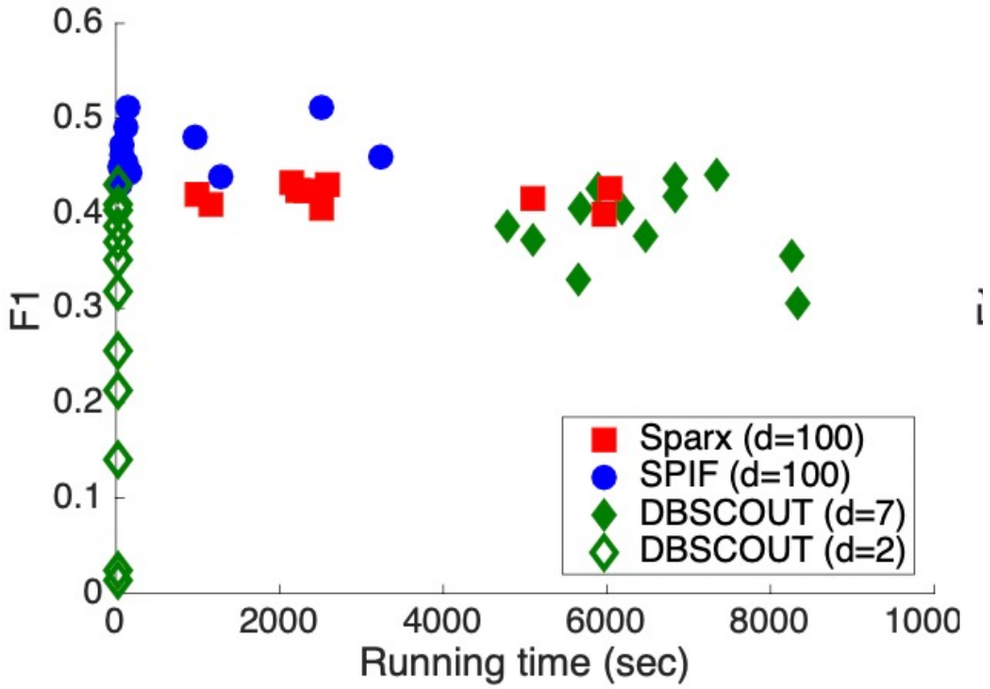
- d ~ 3.2 million,
  n ~ 2.3 million,
  Outlier% = 33%

- **DBSCOUT** cannot handle large d.
  We used random projections to first reduce dim. to d=2, 7

- **DBSCOUT** is resource-frugal but not robust to the choice of HPs

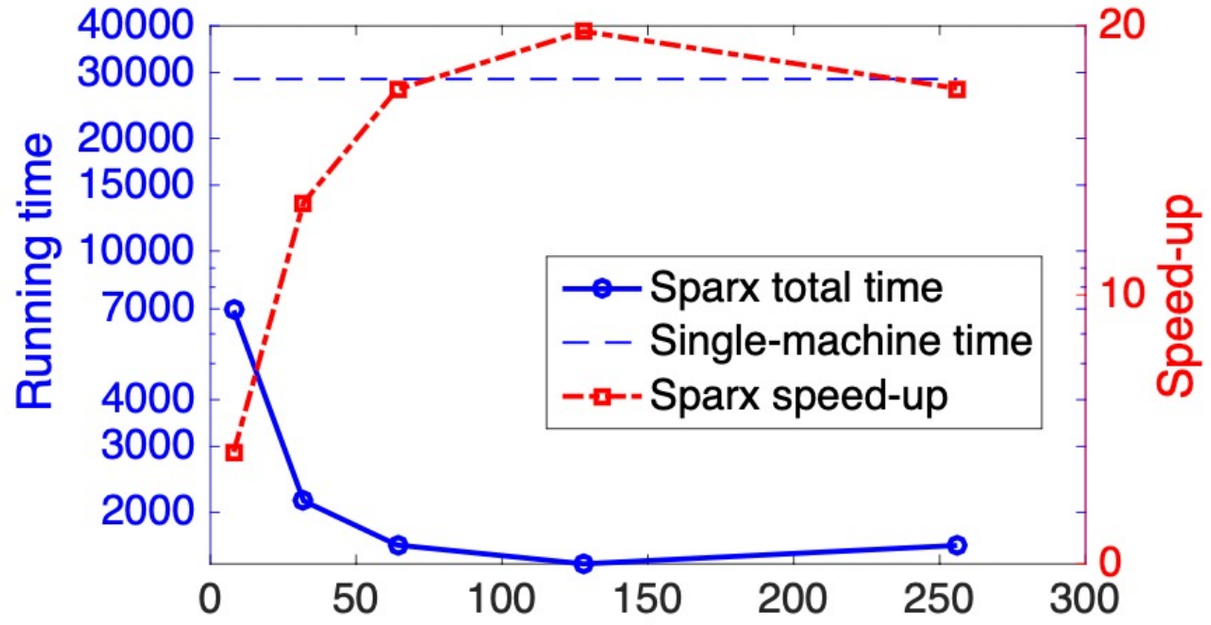
- **Sparx** performs on-par with SPIF.



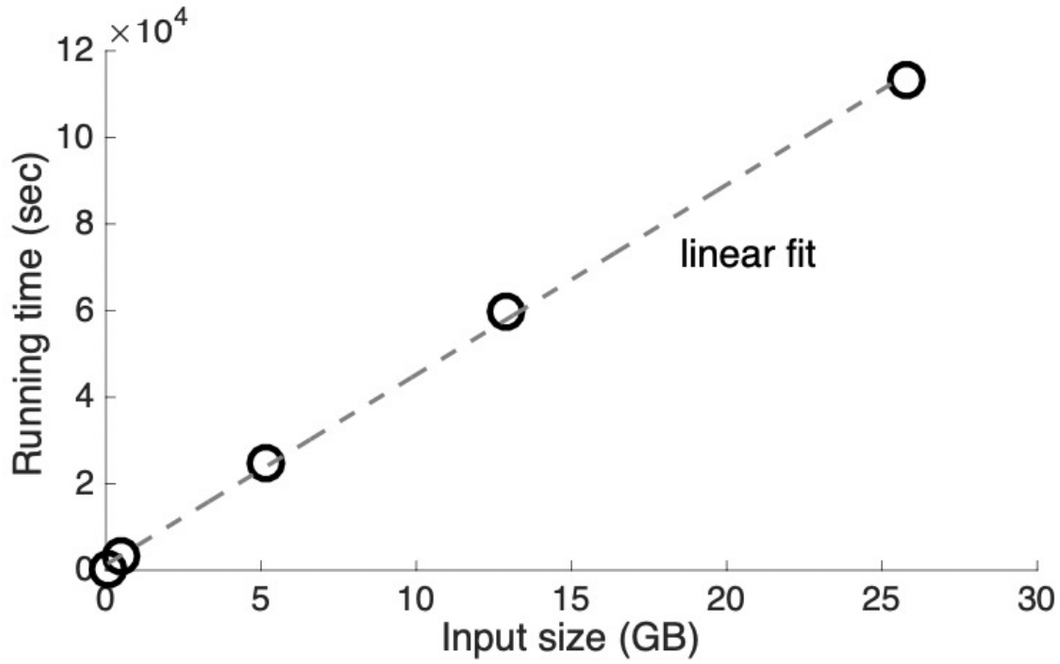


Runtime Scaling

- **Sparx** offers 4-20x speed up vs single-machine as #Spark partitions increase



- **Sparx** runtimes scale linearly in the input data size



Summary

Sparx – *a scalable open-source tool for distributed outlier detection*

✓ **Distributed**, data-parallel detection

✓ Scalable to massive data – linear time/space complexity

✓ Handles not only **Large n** but also **Large d**

✓ Handles mixed-type attributes

✓ Robust to hyperparameter settings

✓ Open-source Apache Spark-based framework

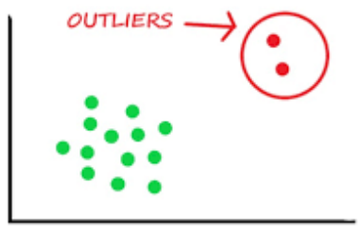
✓ Competitive with existing methods in small n/Large d and Large n/small d settings

✓ Readily adaptable to **streaming** settings

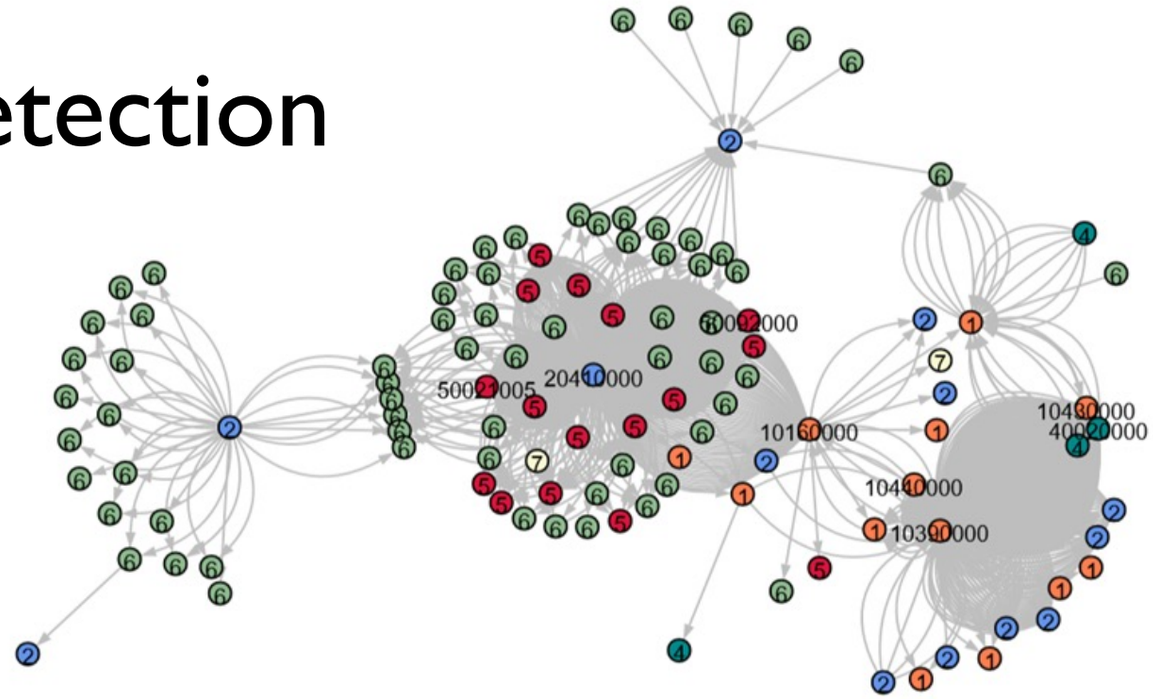
Anomaly Detection for Finance

- **https://tinyurl.com/sparx2022**

distributed outlier detection at scale



Relational anomaly detection
[Graph data]



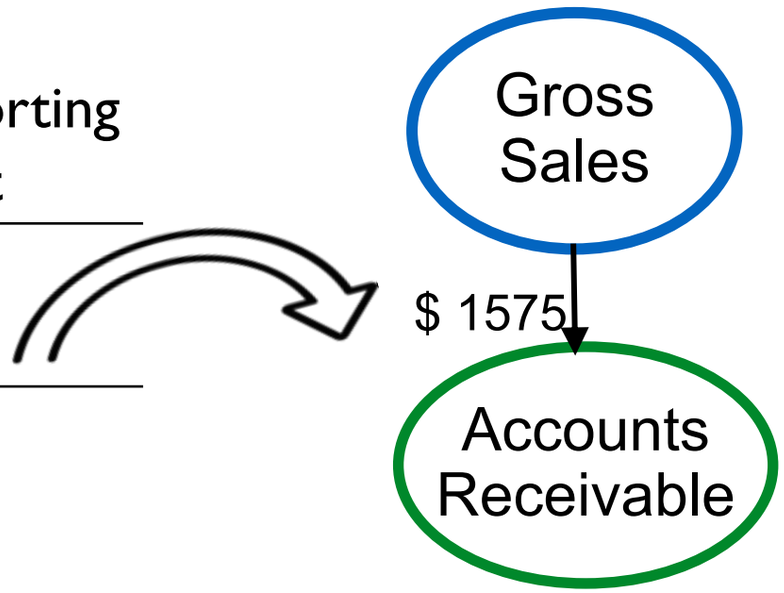
Problem

- Informal: **Given** millions of journal entries
  - **Find** anomalies (entry errors, misconduct, etc.)

Account book-keeping graphs

- Double-entry book-keeping journal entry to graph

GL_Account_ Number	CA_FS_Caption	Cr/Db	GL_Reporting _Amount
40060000 (Revenue)	Gross Sales (GSL)	C	-1575.00
10415000 (Assets)	Accounts Receivable (ARV)	D	1575.00

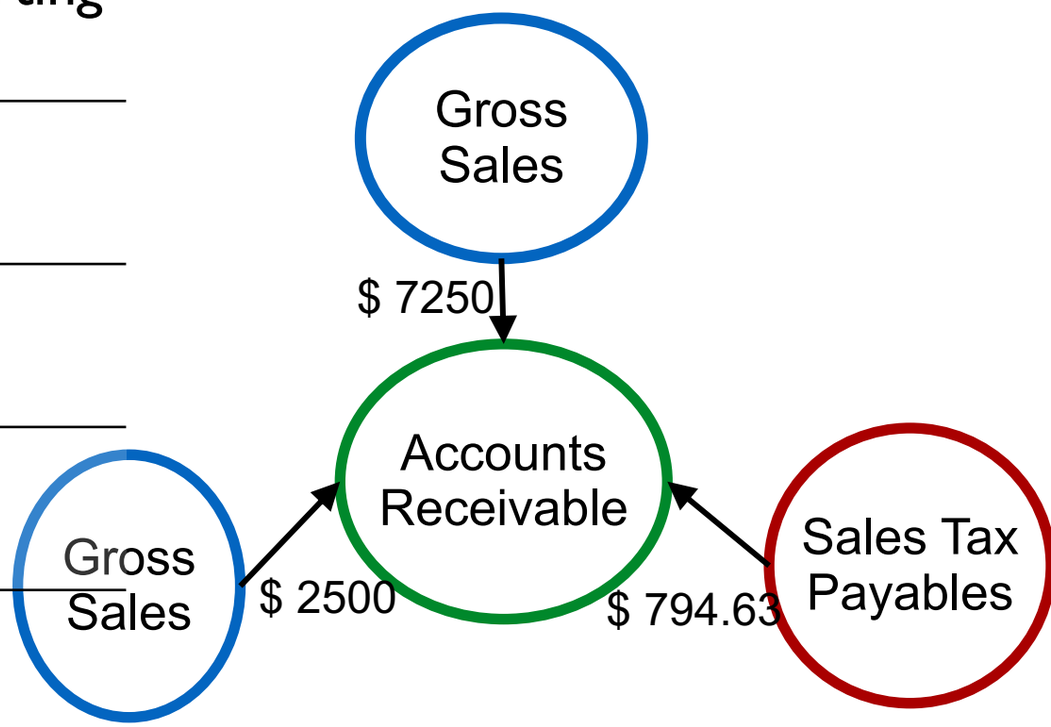


Account book-keeping graphs

- Double-entry book-keeping journal entry to graph

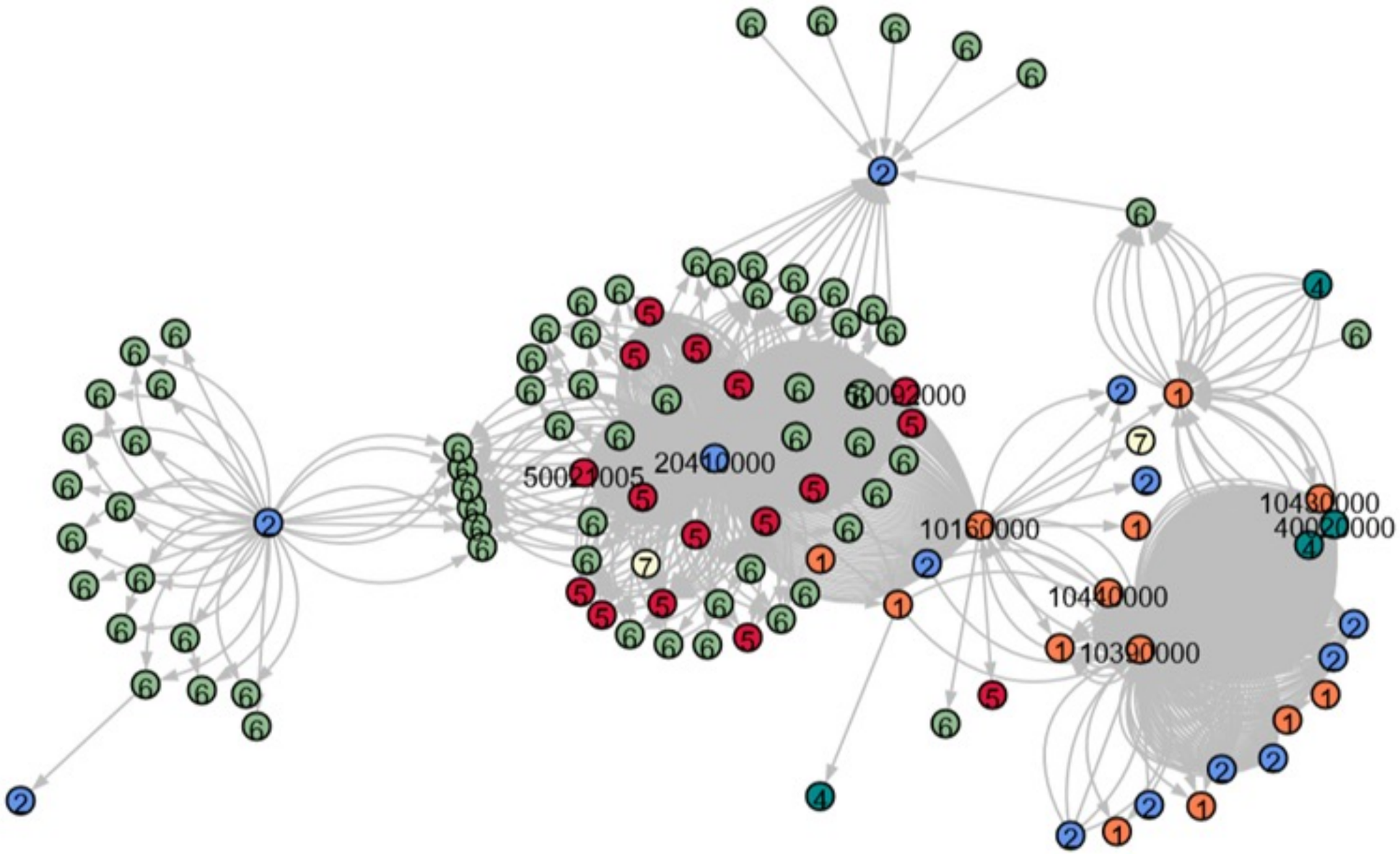
GL_Account_Number	CA_FS_Caption	Cr/Db	GL_Reporting_Amount
40060000 (Revenue)	Gross Sales (GSL)	C	-1575.00
10415000 (Assets)	Accounts Receivable (ARV)	D	1575.00

GL_Account_Number	CA_FS_Caption	Cr/Db	GL_Reporting_Amount
40020000 (Revenue)	Gross Sales (GSL)	C	-7250
40020001 (Revenue)	Gross Sales (GSL)	C	-2500
20830000 (Liabilities)	Sales Tax Payables (STP)	C	-794.63
10390000 (Assets)	Accounts Receivable (ARV)	D	10544.63



Account book-keeping graphs

- Transaction graph of journals over 10-day window:



Problem & Proposed ADAMM

- Formal: **Given** a database of node- & edge- attributed multi-graphs with auxiliary information (metadata),
  - **Find** instances (graphs, metadata) that are unusual
  - *ADAMM; a DL based anomaly detection model jointly detects anomalies on multi-graph/metadata level*

ADAMM: Anomaly Detection of Attributed Multi-graphs w/ Metadata: A Unified Neural Network Approach

Konstantinos Sotiropoulos, Lingxiao Zhao, Pierre J. Liang, Leman Akoglu

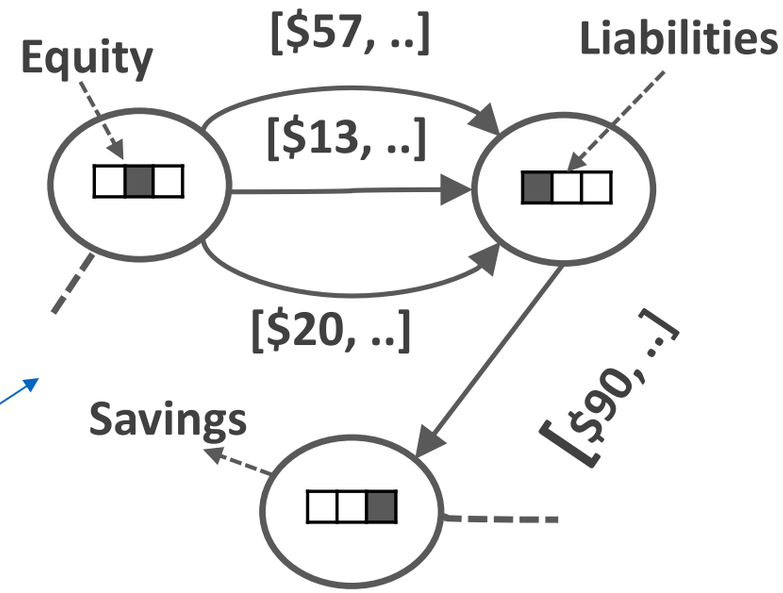
IEEE BigData 2023

Multi-modal Complex data

- How can we spot errors/frauds in general-ledger journal entries?

➢ Relational Information as multi-graph:

- Accounts as **nodes**.
- Debit/credit as weighted directed **edges**.
- Account type is the **node label**.



Complex graph

➢ Metadata:

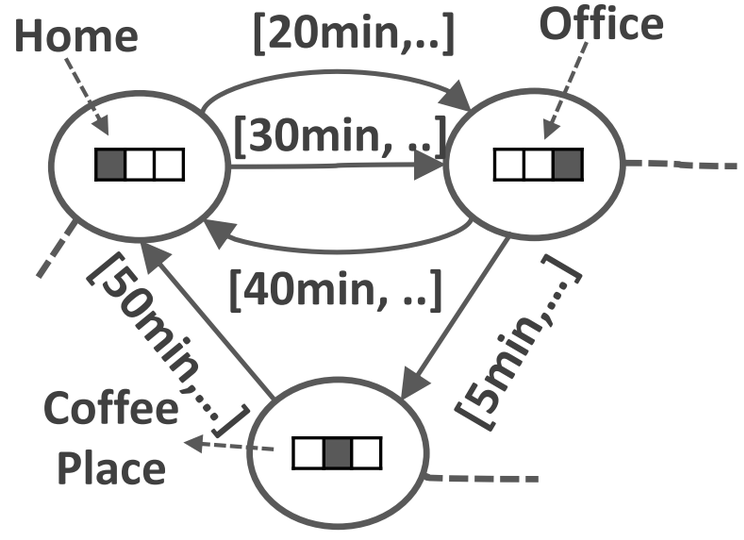
- Approver, entry date, effective date, etc.

Approver	Entry	Effective	…
John	02/01	01/01	

Different Modalities

Applications in other Domains

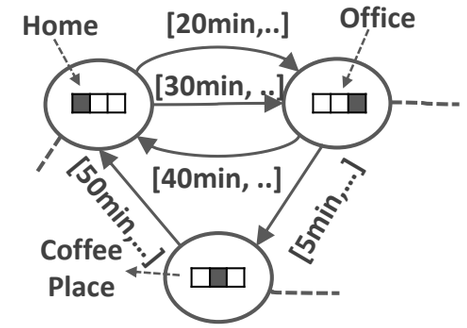
- Human Mobility.
  - Find unusual behavior in daily activity
  - Daily activity: Stay points & trips inbetween



Type	Day		...
Office Worker	Monday		

Applications in other Domains

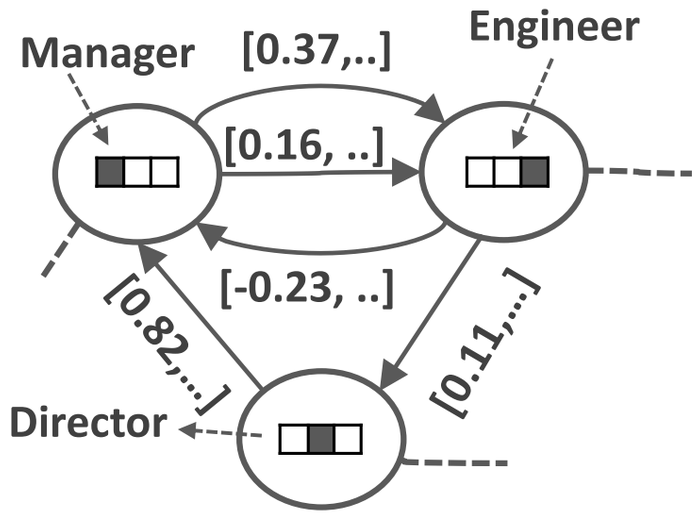
- Human Mobility.
  - Find unusual behavior in daily activity
  - Daily activity: Stay points & trips inbetween



Type	Day	
Office Worker	Monday	

...

- Communication Networks.
  - Detect significant events within a company.
  - E-mails exchanged between employees.



Division	Day	
Sales	Monday	

...

Existing Works

- Detect anomalies at **node/edge level only**.

  e.g., DOMINANT (Ding et al., 2009)

- Can **not** handle directed multi-edges

  e.g., GLAM (Zhao et al., 2022) & OCGTL (Qiu et al, 2022)

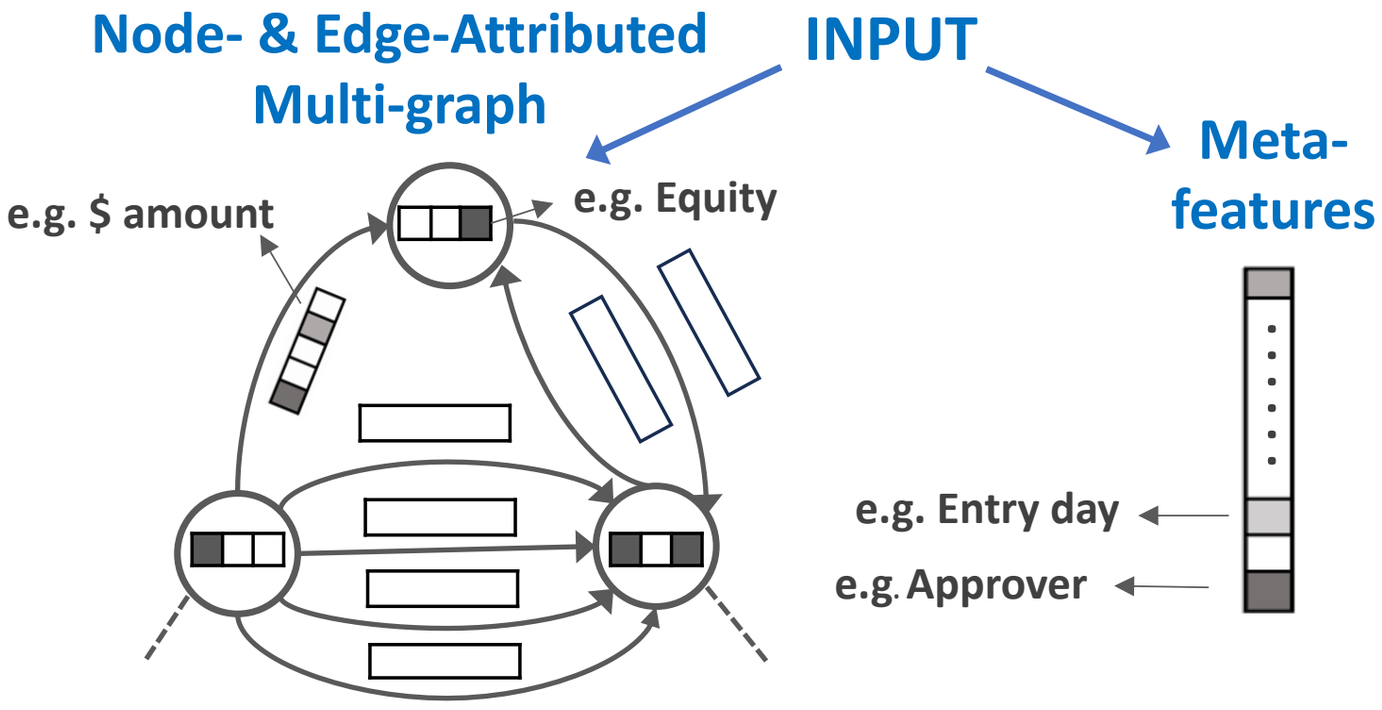
- **Non-DL**-based methods: no edge features, scalability issue

  e.g., GAWD (Lee et al., 2021) & CODEtect (Nguyen et al., 2023)

- Can **not jointly** handle graphs & metadata.

  Miss the opportunity for leveraging inter-dependencies

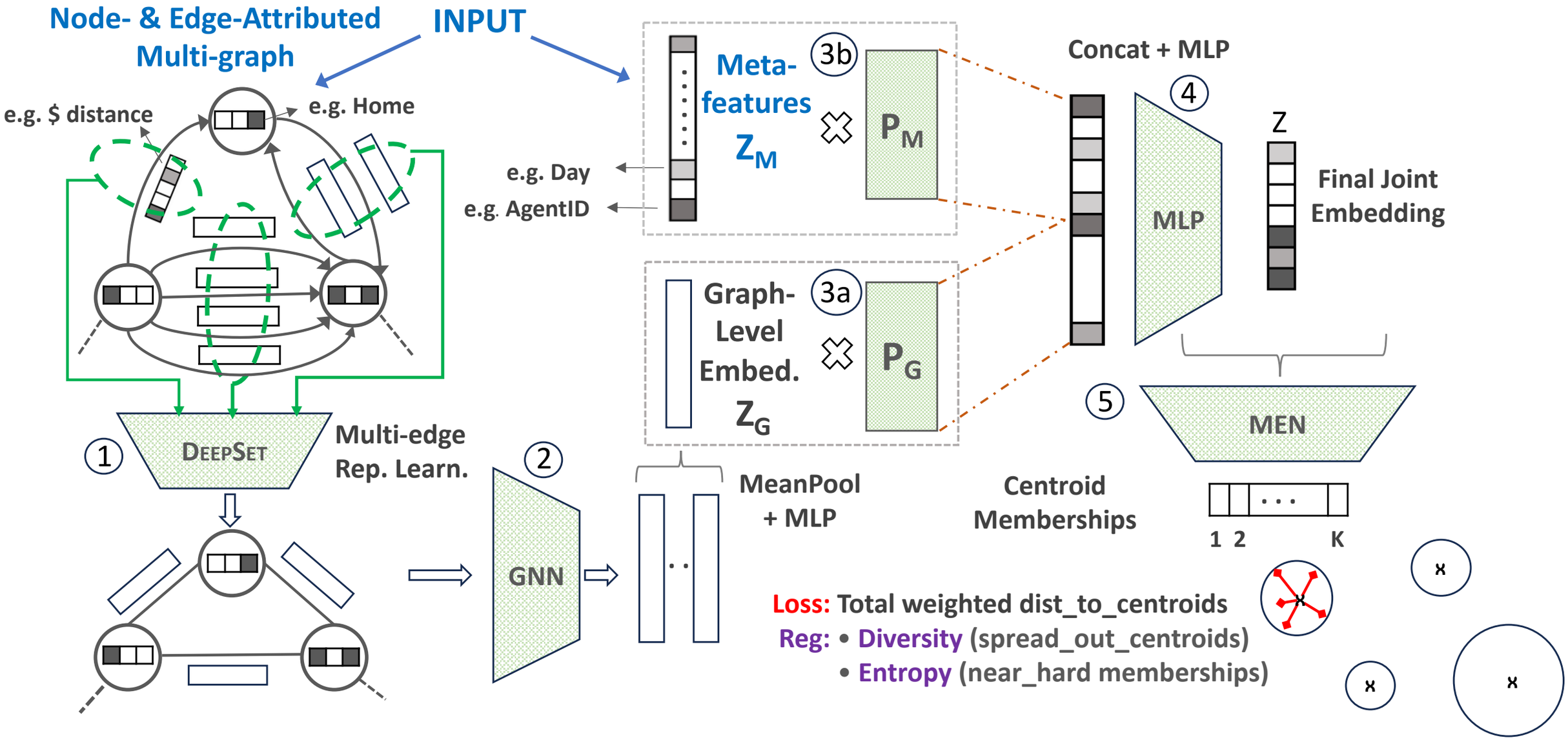
ADAMM



- Can handle complex graph data.
  - ✓ Directed, multi-edges & self-loops
    (e.g., multiple transactions between two accounts)
  - ✓ Node attributes & Edge Features.
- Jointly with graph-level metadata.

ADAMM Overview

- **ADAMM** learns embeddings tightly centered around one of K cluster centroids.
- Idea: Abnormal samples will **not** be embedded close to a cluster centroid.

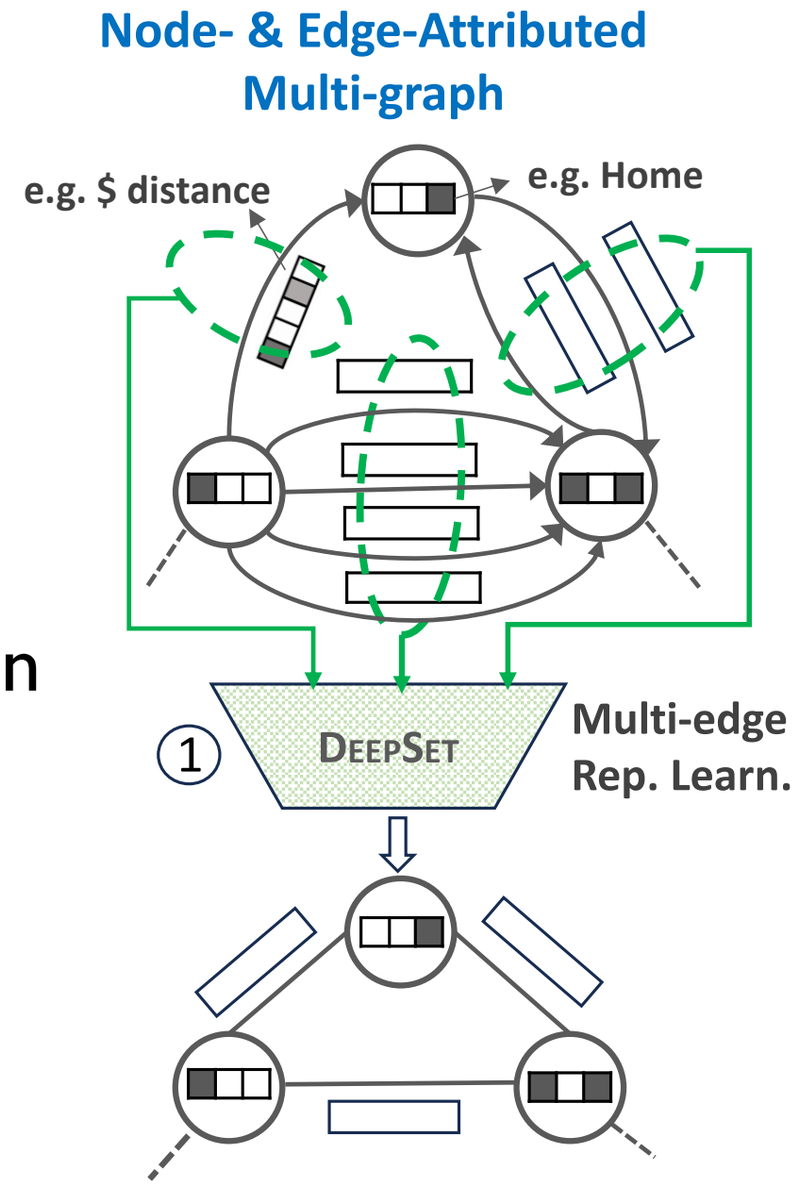


All ADAMM parameters ①-⑤ estimated end-to-end via unsupervised multi-centroid loss.

ADAMM Steps

① Multi-Edge Representation Learning

- Current GNNs can **not** handle multi-edges.
- We "flatten" all directed multi-edges between two nodes to a single undirected edge.
- By learning a permutation-invariant multi-set function using *DeepSets* (Zaheer et al., 2017).



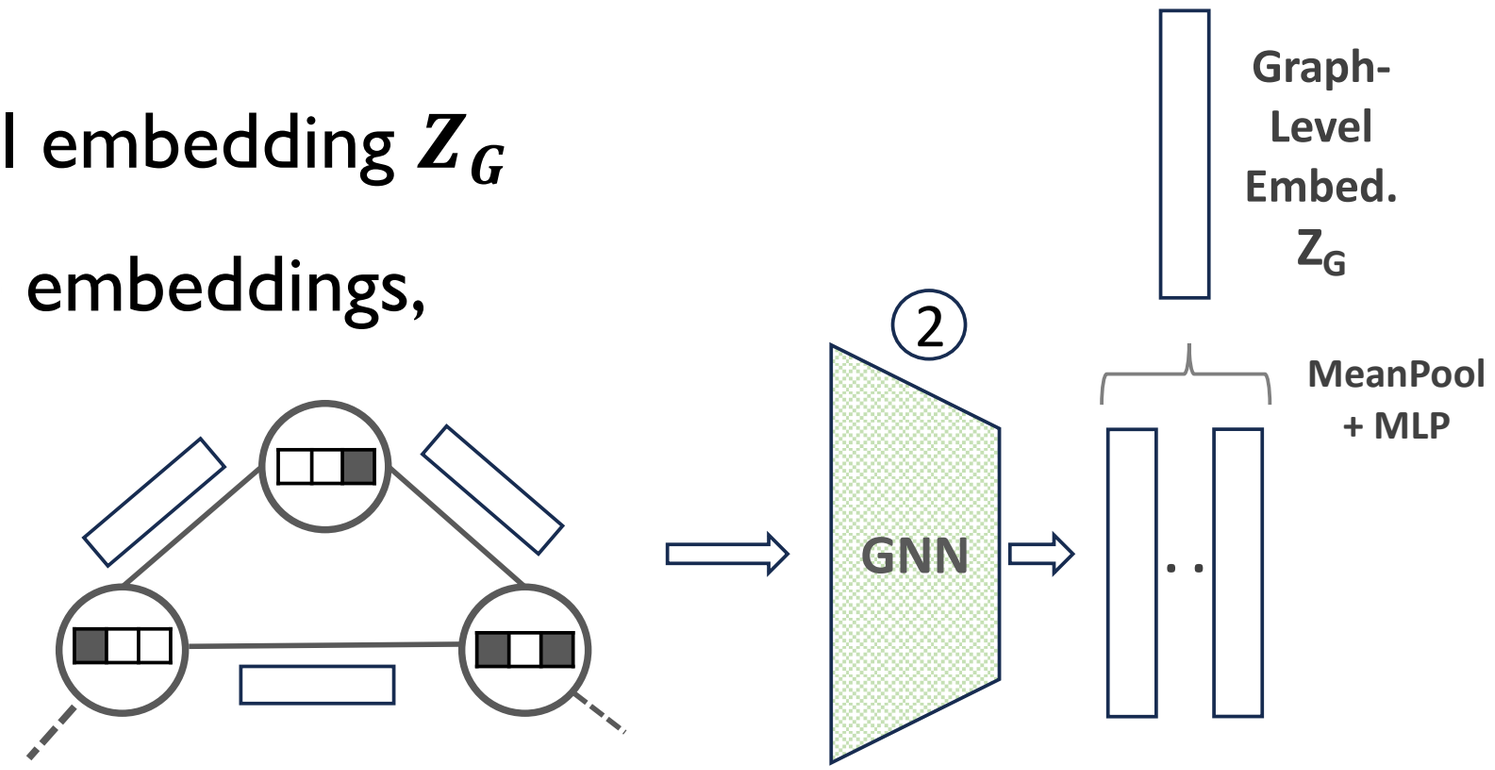
ADAMM Steps

2 Graph-Level Embedding

- We use **GIN** (Xu et al., 2018) a provably expressive GNN model to learn node embeddings.

- We learn a graph-level embedding $\boldsymbol{Z_G}$

  by mean-pooling node embeddings, followed by an MLP.



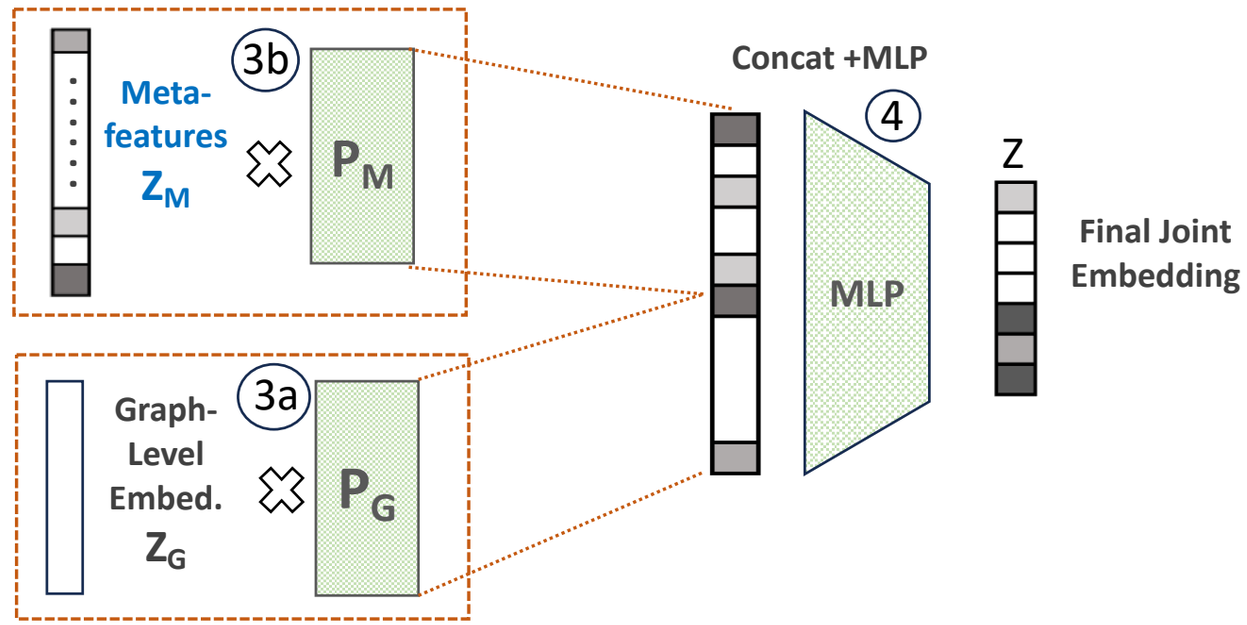
ADAMM Steps

3 - 4 Unifying Embedding Space for Graph and Metadata

- We learn projection matrices P_M and P_G to obtain two new vectors $Z_{M'}$ and $Z_{G'}$.

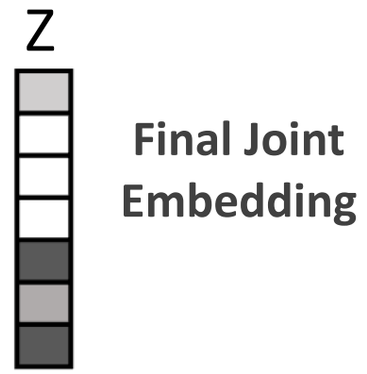
- We concatenate and use an MLP to obtain final embedding vector:

$$Z = MLP(CONCAT(Z'_M, Z'_G); \theta_J)$$



ADAMM Steps

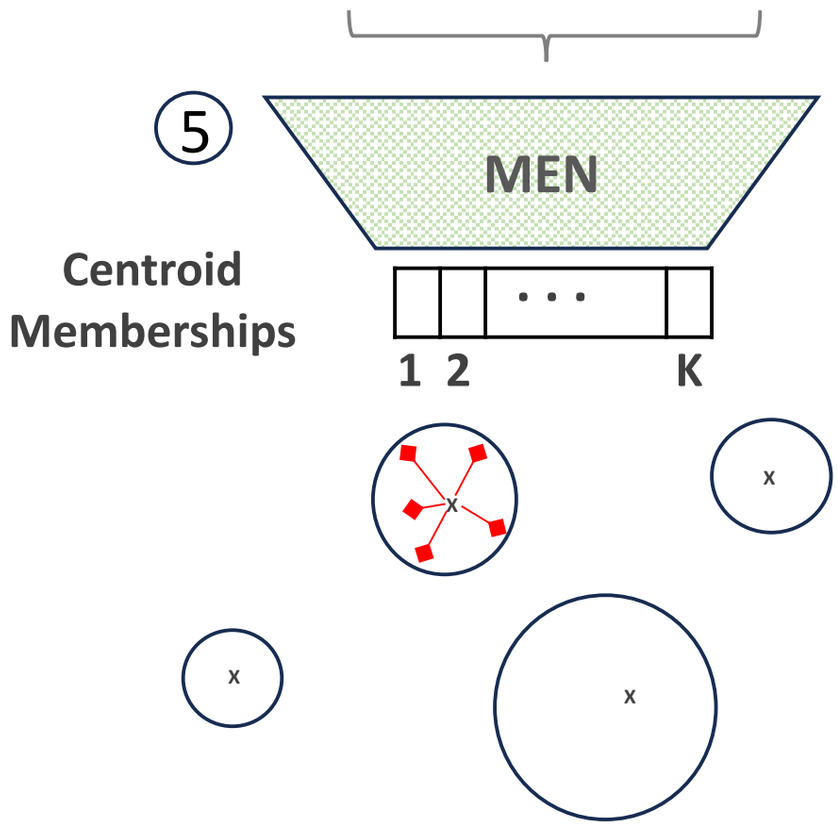
5 Membership Estimation Network (MEN)



- MEN estimates the membership probability of Z

for each of the K clusters:

$$\hat{\gamma} = \text{softmax}(MLP(Z; \theta_{MEN})$$



- Cluster centroids are calculated as a weighted avg. of embedding vectors Z_i , $i = 1, \ldots, N$:

$$\hat{c}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} \cdot Z_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

ADAMM Loss Function

- ADAMM parameters are optimized **end-to-end** to minimize:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \underbrace{\hat{\gamma}_{ik} \left|\left| Z_i - \hat{c}_k \right|\right|_2^2}_{1} + \underbrace{\lambda_1 H\left(\hat{\Gamma}\right)}_{2} + \underbrace{\lambda_2 D\left(\hat{C}\right)}_{3}$$

  - 1: Weighted distance of embeddings from cluster centroids
  - 2: Entropy Regularization; forces MEN for confident estimation:

$$H\left(\hat{\Gamma}\right) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} -\hat{\gamma}_{ik} \cdot \log(\hat{\gamma}_{ik})$$

  - 3: Diversity term; promotes separation between cluster centroids:

$$D\left(\hat{C}\right) = -\log(\det(\, Cov(\hat{C}))$$

$det(\, Cov(\hat{C}))$: determinant of the covariance matrix of cluster centroids

Anomaly Score & Model Selection

- For a sample $T_i = (G_i, M_i)$, its anomaly score is its total weighted distance to cluster centroids:

$$\text{score}(T_i) = \sum_{k=1}^{K} \hat{\gamma}_{ik} \left|\left| Z_i - \hat{c}_k \right|\right|_2^2$$

- Different hyperparameter config.s yield different models.
  - We select the model that in the training set minimizes:

$$\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \hat{\gamma}_{ik} \left|\left| Z_i - \hat{c}_k \right|\right|_2^2$$

Datasets

➢ Accounting Domain

Dataset	Graphs	Nodes	Multi-Edges	Node Attr.	Edge Attr.	Meta-feat.
SH	39,011	[1,15]	[1,338]	11	1	11
KD	152,105	[1,91]	[1,774]	10	1	9
HW	90,274	[1,25]	[1,59]	11	1	7

➢ Human Mobility Domain

Dataset	Graphs	Nodes	Multi-Edges	Node Attr.	Edge Attr.	Meta-feat.
MobiNet	140,000	[1,22]	[1,59]	41	4	9

Expert-Guided Anomaly Injections

1) Graph Anomalies

- **Label Change (GA1):** Change the label of a node to a randomly chosen new one.

  "Entry-error in accounting or visit to unusual POI in mobility"

- **Path Injection (GA2):** Delete an edge u-v and rewire through an intermediary, creating a path u-z-v.

  "Money-laundering in finance or unusual stop in mobility"

2) Metadata Anomalies

- **Unusual back-dating (MA1):** Change entry date to follow effective date.

- **Combination of unrelated transactions (MA2):** Merge two unrelated transactions by creating one with a unique journal ID.

Anomaly Injections (cont.) & Baselines

3) Potpourri anomalies

Pick a graph anomaly & a metadata anomaly and inject both

- We compare ADAMM against two-stage baselines:

  Stage 1-Graph: **Graph-level** Anomaly Detectors

  Stage 1-Metadata: **Tabular** Data Outlier Detectors

  Stage 2: Rankings from 1-g & 1-m are **aggregated** using different methods.

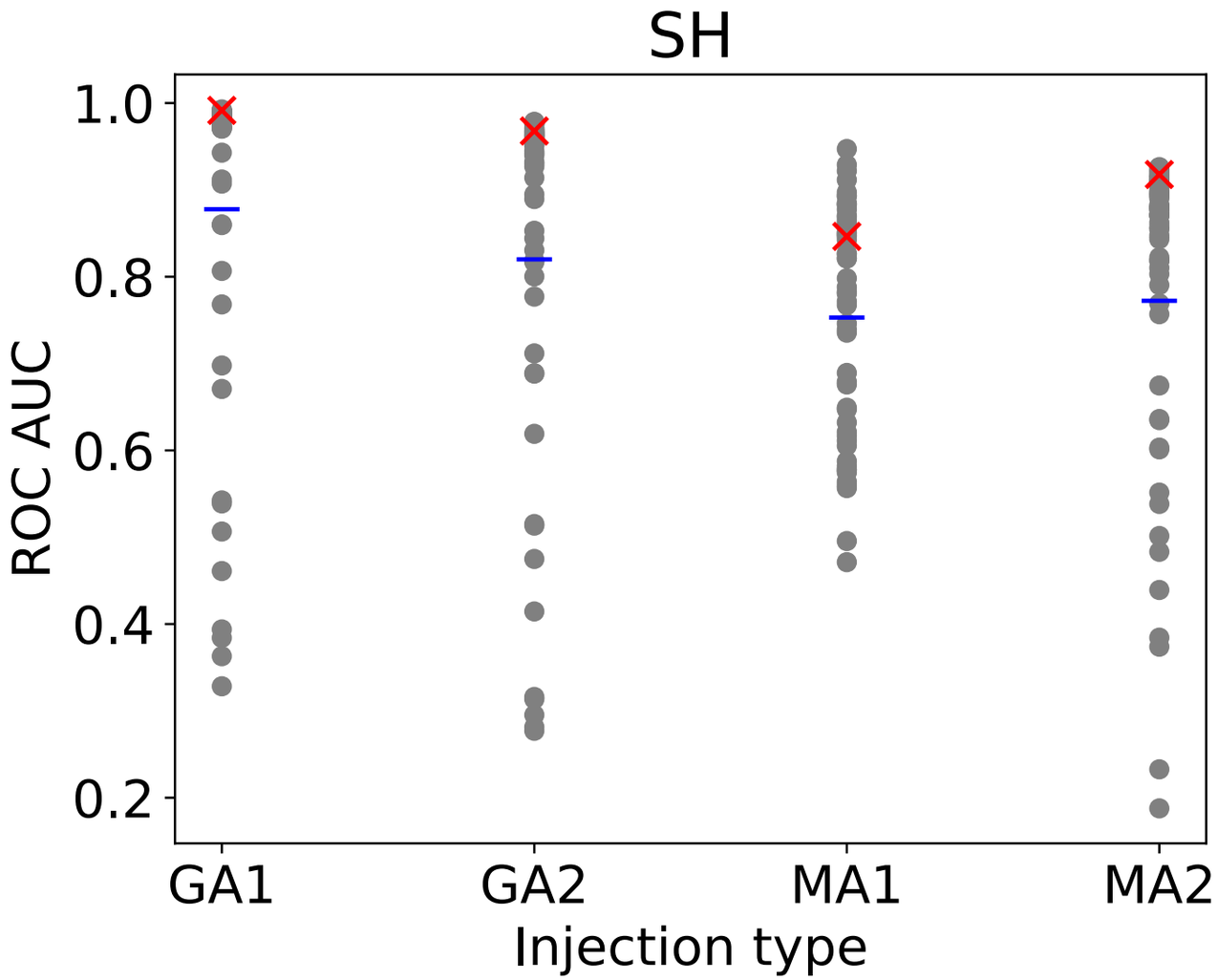
ADAMM is Effective

- ADAMM outperforms two-stage baselines on most datasets and injection types.
- By >= 7.8% on average in terms of AUROC.
- & 2x better detection in terms of AUPRC.

Dataset	Anomaly Type	ADAMM	WL+BFS	WL+IR	G2V+BFS	G2V+IR	DOM.+BFS	DOM.+IR
SH	GA1	**0.992**	0.925 ± 0.01	0.922 ± 0.01	0.839 ± 0.09	0.833 ± 0.09	0.824 ± 0.01	0.821 ± 0.01
	GA2	**0.968**	0.827 ± 0.02	0.829 ± 0.02	0.854 ± 0.02	0.854 ± 0.02	0.834 ± 0.01	0.837 ± 0.01
	MA1	**0.846**	0.591 ± 0.02	0.610 ± 0.03	0.586 ± 0.01	0.592 ± 0.01	0.602 ± 0.02	0.613 ± 0.02
	MA2	**0.918**	0.638 ± 0.02	0.642± 0.02	0.614 ± 0.01	0.618 ± 0.01	0.615 ± 0.01	0.618 ± 0.01
	GA1 + MA1	**0.955**	0.899 ± 0.01	0.897 ± 0.02	0.807 ± 0.01	0.800± 0.01	0.811 ± 0.01	0.807 ± 0.02
	GA2 + MA1	**0.977**	0.841 ± 0.03	0.840 ± 0.01	0.871 ± 0.01	0.869±0.02	0.836 ± 0.02	0.838 ± 0.01
KD	GA1	**0.928**	0.885 ± 0.01	0.880 ± 0.01	0.835 ± 0.04	0.828 ± 0.04	0.462 ± 0.01	0.450 ± 0.01
	GA2	**0.939**	0.825 ± 0.03	0.828 ± 0.04	0.820 ± 0.01	0.824 ± 0.01	0.543 ± 0.01	0.528 ± 0.01
	MA1	**0.841**	0.727 ± 0.01	0.716 ± 0.03	0.729 ± 0.01	0.718 ± 0.01	0.610 ± 0.01	0.588 ± 0.01
	MA2	**0.854**	0.738 ± 0.02	0.743 ± 0.02	0.736 ± 0.02	0.741 ± 0.02	0.518 ± 0.01	0.505 ± 0.01
	GA1 + MA1	**0.933**	0.901 ± 0.01	0.895 ± 0.01	0.814 ± 0.07	0.805 ± 0.01	0.458 ± 0.01	0.448 ± 0.01
	GA2 + MA1	**0.916**	0.849 ± 0.03	0.849 ±0.04	0.818 ± 0.01	0.805 ± 0.07	0.537 ± 0.01	0.522 ± 0.01
HW	GA1	**0.973**	0.922 ± 0.01	0.916 ± 0.01	0.922 ±0.03	0.920 ± 0.03	0.713 ± 0.21	0.710 ± 0.21
	GA2	**0.994**	0.895 ± 0.01	0.888 ± 0.01	0.666 ± 0.05	0.660 ± 0.05	0.400 ± 0.07	0.406 ± 0.07
	MA2	**0.967**	0.691 ± 0.02	0.661 ± 0.02	0.706 ± 0.02	0.694 ± 0.01	0.535 ± 0.01	0.527 ± 0.01
MobiNet	GA1	0.526	0.676 ± 0.01	**0.678** ± 0.02	0.466 ± 0.07	0.467 ± 0.05	0.320 ± 0.01	0.323 ± 0.01
	GA2	0.491	0.487 ± 0.02	0.482 ± 0.03	0.499 ± 0.05	**0.505** ± 0.04	0.341± 0.01	0.346 ± 0.01
	MA3	0.441	0.558 ± 0.01	0.563 ± 0.01	0.556 ± 0.02	**0.574** ± 0.03	0.411± 0.01	0.415 ± 0.01
	MA4	0.450	0.492 ± 0.01	0.490 ± 0.01	0.517 ± 0.02	**0.524** ± 0.01	0.349± 0.01	0.353 ± 0.0
	GA1 + MA3	0.563	0.750 ± 0.01	**0.754** ± 0.02	0.451 ± 0.01	0.454 ± 0.02	0.326 ± 0.01	0.329 ± 0.03
	GA1 + MA4	0.678	0.743 ± 0.02	**0.747** ± 0.01	0.457 ± 0.02	0.460 ± 0.01	0.350± 0.01	0.353 ± 0.02
	GA2 + MA3	0.470	0.483 ± 0.01	0.480 ± 0.02	0.505 ± 0.01	**0.510** ± 0.02	0.329 ± 0.04	0.332 ± 0.01
	GA2 + MA4	0.477	0.494 ± 0.02	0.489 ± 0.01	0.520 ± 0.01	**0.526** ± 0.03	0.332 ± 0.01	0.336 ± 0.01
Average AUROC		0.787	0.730	0.728	0.678	0.677	0.524	0.522
Average Rank		2.13	3.08 (**)	2.78 (**)	4.09 (***)	3.74 (***)	6.17 (***)	6 (***)

Model Selection

- We succeed in selecting a model that performs better than picking hyperparameters at random.



Ablation Study

- ADAMM achieves better detection results than its counterparts:
  - i. **w/o** using Metadata information.
  - ii. **w/o** DeepSet (only average multi-edge attributes).
  - iii. Uses the **OneClass** DeepSVDD Loss (Ruff et al., 2018).

Dataset	Anomaly Type	ADAMM		ADAMM **w/o Metadata**		ADAMM **w/o DeepSet**		ADAMM **with OCDL**	
		AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
SH	GA1	**0.992**	**0.938**	0.989	0.920	0.988	0.920	0.986	0.894
	GA2	**0.968**	**0.708**	0.961	0.622	0.965	0.758	0.930	0.489
	MA2	**0.918**	**0.598**	0.898	0.580	0.816	0.427	0.868	0.467

Summary

ADAMM – ***a unified open-source deepNN model for AD on attributed multi-graphs (w/ metadata)***

✓ ADAMM effectively detects anomalies on a database of complex graphs with metadata

✓ Can handle complex graph data of virtually any type

✓ Widely applicable to various domains

✓ A unified, end-to-end trainable model

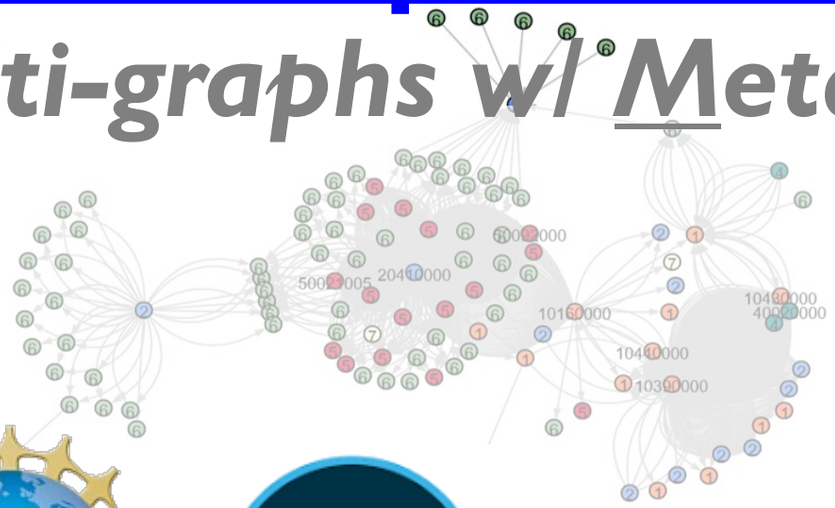
Anomaly Detection for Finance

- **https://tinyurl.com/sparx2022**

  Distributed outlier detection at scale

- **https://github.com/konsotirop/ADAMM**

  AD of Attributed Multi-graphs w/ Metada



Thanks!