YIKUN BAN

EECS, PEKING UNIVERSITY

RESEARCH SUMMARY

RESUME

Yikun BAN

Birthplace: Henan Provence, China
Birthday: 6th, Sep, 1994

Education:
M.S. Peking University, Beijing, China, 2016~
B.S. Wuhan University, Wuhan, China, 2012-2016

Selected Awards and Honors:

- First Prized (Top 0.1%, 2859 teams involved). The 4th “China Software Cup” Software Design Competition for College Students (National Competition). - Aug. 2015
- Second Prized (Top 0.2%, 5000+ teams involved) in National Finals, First Prized in Provincial Finals. “Challenge Cup” China College Student’s Entrepreneurship Competition in 2014. - Nov. 2014
- The Best Financial Creative Award, “CITI CUP” Citi Financial Innovation Application Competition in 2014. - Sep. 2014
- Outstanding Students of Peking University, 2017-2018.
- Outstanding Graduate Student of Wuhan University, 2012-2016.
- Excellent Students Scholarship of Wuhan University, 2013-2015.
- Outstanding Students of Wuhan University, 2012-2016.

RESEARCH WORKS & OUTLINE

- **Yikun Ban**, Xin Liu, Ling Huang, Yitao Duan, Xue Liu, and Wei Xu. "No Place to Hide: Catching Fraudulent Entities in Tensors". Submitted to **WWW 2019**. arXiv preprint arXiv:1810.06230, 2018.
  Slides: 4-24

- **Yikun Ban**, Jiao Sun, Xin Liu, Ling Huang, Yitao Duan, and Wei Xu. "FraudTrap: Catching Loosely Synchronized Behavior in Face of Camouflage", Submitted to **WWW 2019**. arXiv preprint arXiv:1810.08885, 2018.
  Slides: 25-44

- **Yikun Ban,** Yitao Duan, and Ling Huang. "On Finding Dense Subgraphs in Bipartite Graphs: Linear Algorithms with Applications to Fraud Detection". Submitted to **ICDE 2019.** arXiv preprint arXiv:1810.06809, 2018.
  Slides: 45-66

No Place to Hide: Catching Fraudulent Entities in Tensors

Yikun Ban

Advisors:

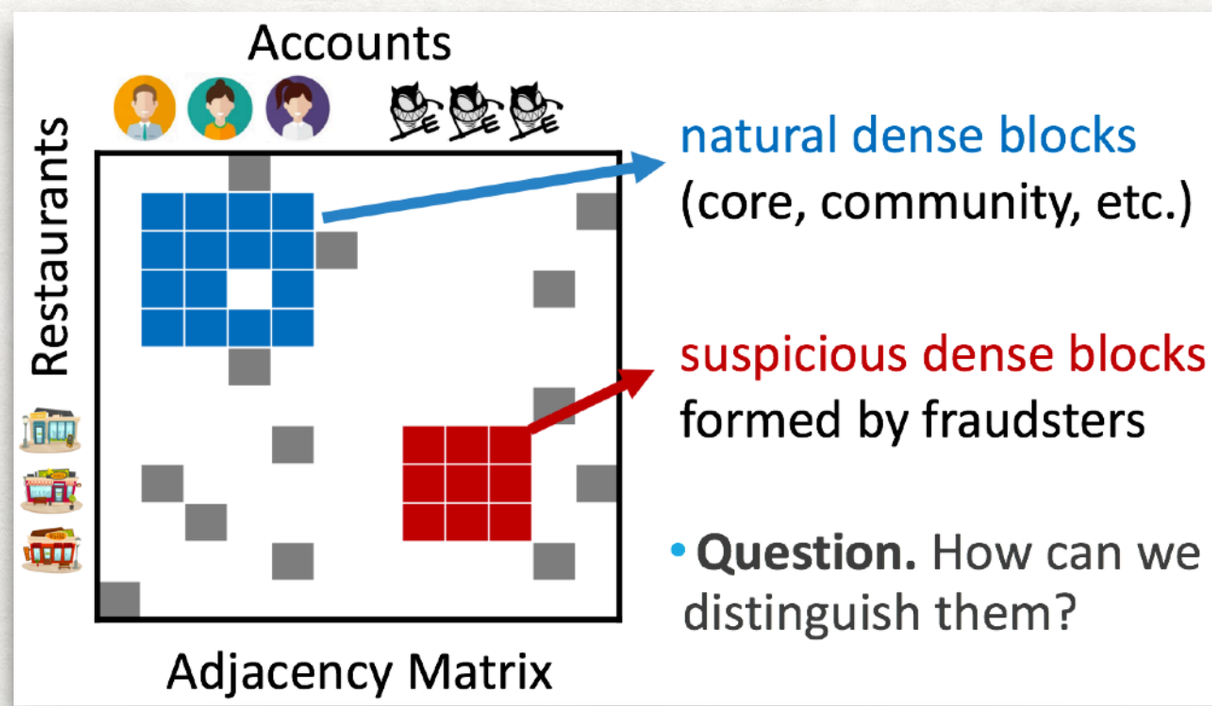
- Wei Xu(Assistant professor of Tsinghua (THU), Ph.D. at UC Berkeley),
- Yitao Duan(chief scientist of Netease Youdao Inc., Ph.D. at UC Berkeley),
- Ling Huang(senior research scientist of Intel Labs Berkeley, Ph.D. at UC Berkeley),
- Xue Liu (Professor, McGill University)

Teammates: Xin Ling (THU)

- ✦ Motivation
- ✦ Problem Definition
- ✦ Metheds
- ✦ Experiments and Evaluation

MOTIVATION

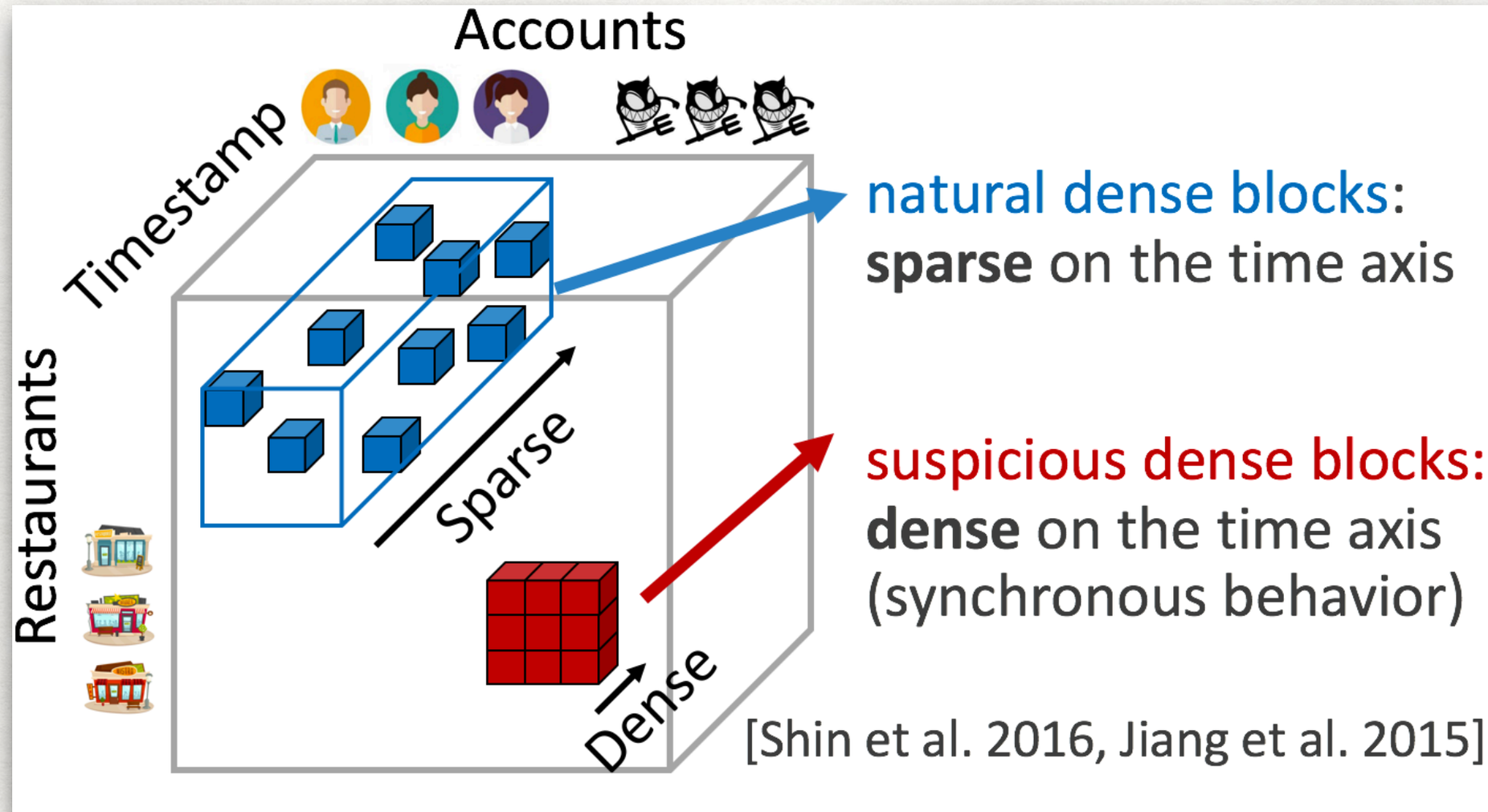
Problem: natural dense blocks



- **Question.** How can we distinguish them?

MOTIVATION

Solution: tensor modeling, to leveraging more features

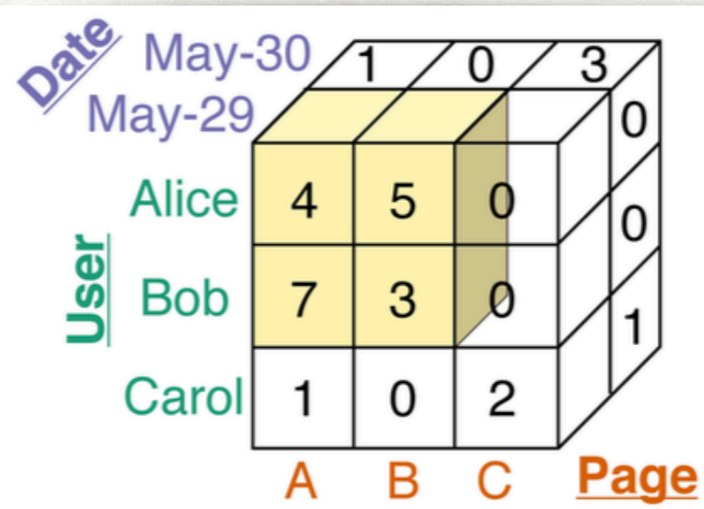


PROBLEM DEFINITIONS

The representation of a tensor

User	Page	Date	Count
Alice	A	May-29	4
Alice	B	May-29	5
Bob	A	May-29	7
Bob	B	May-29	3
Carol	C	May-30	1
⋮	⋮	⋮	⋮

(a) Relation $\mathcal{R}$



(b) Tensor Representation of $\mathcal{R}$

Related work:

Existing methods[15,16,18,6] directly search on a tensor by greedily delete or add distinct values from the tensor until finding a block maximizing a density metric.

PROBLEM DEFINITIONS

Limitations of existing approaches[22,15,16,18,6]

[22,15,16,18,6] focus on detecting dense blocks on N-dimensional tensor. However, what about a block B does not have a high-density signal on N dimensions of a tensor, but evidently has a high density on a subset of N dimensions.

To find B, how to solve solve exploded feature combinations?

How to find the dense block in a100-dimensional tensor?

Existing approaches[22,15,16,18,6] cannot solve the problem.

PROBLEM DEFINITIONS

Key Idea:

What is the reason of forming dense block?

Value sharing !

40 users login in 2 IPs in 120 minutes **12375 times**!

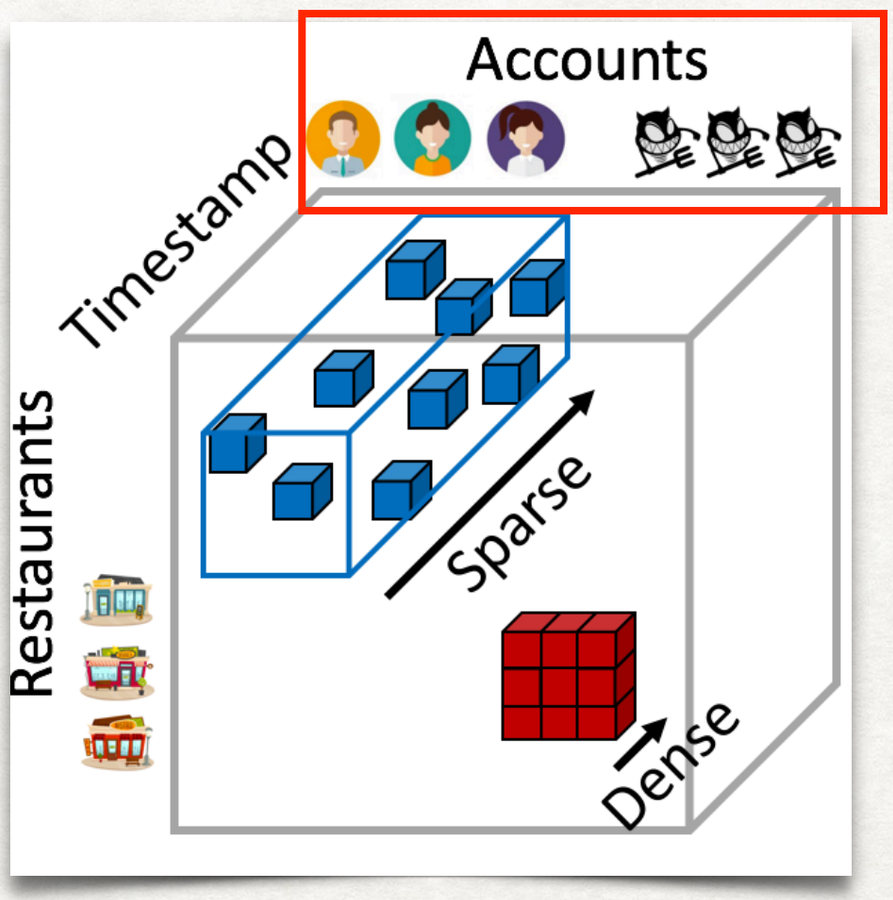
The values of user, ip, or time are shared by actions (entries) multiple times!

Thus, catching value sharing is identified with catching dense blocks!

PROBLEM DEFINITIONS

Target Dimension:

Catching fraudulent entities is equivalent to detecting a subset of distinct values in a certain dimension. Let U denote the target dimension in which a subset of distinct values form the fraudulent entities we wish to detect.



Target dimension, because we about to detect fraudulent accounts

PROBLEM DEFINITIONS

Formulate value sharing:

We define two types of value sharing:

Pairwise Value Sharing (PVS):

PVS

User	Page	Date	Count
Alice	A	May-29	4
Alice	B	May-29	5
Bob	A	May-29	7
Bob	B	May-29	3
Carol	C	May-30	1
⋮	⋮	⋮	⋮

(a) Relation $\mathcal{R}$

E.g. Alice and Bob share A

Self Value Sharing (SVS):

SVS

User	Page	Date	Count
Alice	A	May-29	4
Alice	B	May-29	5
Bob	A	May-29	7
Bob	B	May-29	3
Carol	C	May-30	1
⋮	⋮	⋮	⋮

(a) Relation $\mathcal{R}$

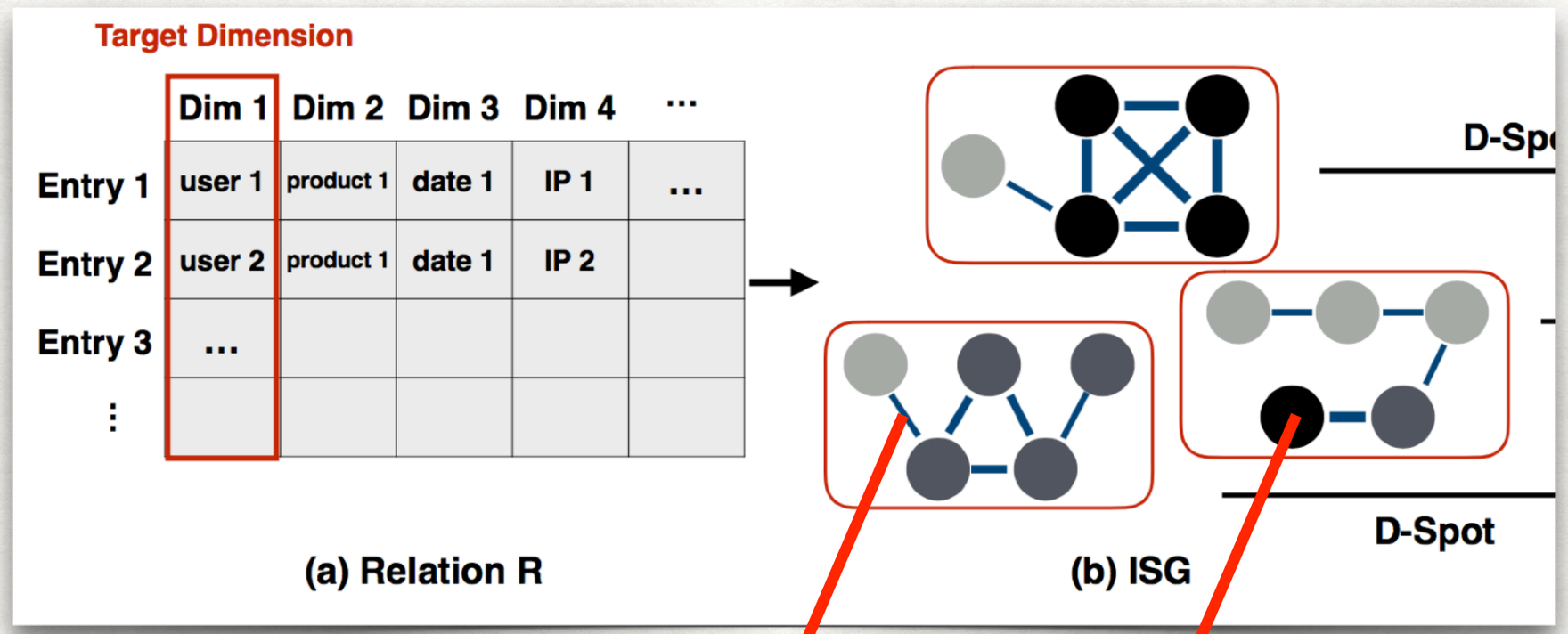
E.g. Alice share May-29 Twice

METHODS: ISG

How to represent PVS and SVS?

Weighted Graph!

Information Sharing Graph (ISG)



Weighted edge: PVS

Weighted Node: SVS

METHODS: ISG

How to calculate PVS and SVS?

Probability ! based on Information Theory[23].

Edge construction in ISG

Given a dimension A_k, the target dimension U, any $u_i \in \mathbf{V}$, and an entry $t_1 \in \mathbf{R}$ for which $t_1[U] = u_i$, then for each $a \in A_k$, we assume that the probability of $t_1[A_k] = a$ is $p^k(a)$.

Edge Construction. Based on information theory [30], the self-information of the event that u_i and u_j share a is:

$$\mathcal{I}_{i,j}^k(a) = \log(\frac{1}{p^k(a)})^2. \quad (1)$$

To compute the pairwise value sharing between u_i and u_j across all K dimensions, we propose the metric $\mathcal{S}$-score as the edge weight of ISG:

$$\mathcal{S}_{i,j} = \sum_{k=1}^{K} \sum_{a \in H_k(u_i, u_j)} \mathcal{I}_{i,j}^k(a), \quad (2)$$

where $H_k(u_i, u_j)$ is the set of all values shared by u_i and u_j on A_k. Note that $\mathcal{S}_{i,j} = 0.0$ if $\bigcup_{k=1}^{K} H_k(u_i, u_j) = \emptyset$.

METHODS: ISG

Node Setting in ISG

Node Setting. For a node $u_i \in \mathbf{V}$, let $\mathcal{B}(a, A_k, u_i, U)$ be the set $\{t \in \mathbf{R} : (t[A_k] = a) \wedge (t[U] = u_i)\}$. When $|\mathcal{B}(a, A_k, u_i, U)| \geq 2$. The information of forming $\mathcal{B}(a, A_k, u_i, U)$ is:

$$\mathcal{I}_i^k(a) = \log(\frac{1}{p^k(a)})^{|\mathcal{B}(a, A_k, u_i, U)|}. \tag{3}$$

We now define $\mathcal{S}_i$ to compute the self-value sharing for u_i across all K dimensions:

$$\mathcal{S}_i = \sum_{k=1}^{K} \sum_{\mathcal{B} \in H_k(u_i)} \mathcal{I}_i^k(a), \tag{4}$$

where $H_k(u_i)$ is the set $\{\mathcal{B}(a, A_k, u_i, U), \forall a \in \mathbf{R}_k\}$ and $\mathbf{R}_k$ is the set of distinct values of A_k. Note that $\mathcal{S}_i = 0.0$ if $\bigcup_{k=1}^{K} H_k(u_i) = \emptyset$.

Optimization of ISG Construction:

ISG: $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ it takes $O(K|\mathbf{R}| + |\mathbf{E}|)$

K is the number of dimensions of the tensor

|R| is the number of entries of the tensor

METHODS: ISG

Determining [$p^k(a)$].

(1) if the dimension A_k is fraud-related resource features such as IP subnets and device IDs, we assume a uniform distribution.

$$p^k(a) = 1/|\mathbf{R}_k|.$$

(2) if the dimension A_k is the object colluded with fraudulent user such as products on Amazon and followee on Twitter, we use empirical distribution:

.

$$p^k(a) = |\mathcal{B}(a, A_k)|/|\mathbf{R}|,$$

METHODS: D-SPOT

Duo to ISG, we novelly identify dense-block detection with dense-subgraph mining.

Problem:

Input: ISG

Output: multiple dense subgraphs of ISG

D-Spot is a dense-subgraph detection algorithm, inspired by [24]

Density Metric: $$\mathcal{F}_{\mathcal{G}} = \frac{\sum_{(u_i,u_j)\in\mathcal{E}} \mathcal{S}_{i,j} + \sum_{u_i\in\mathcal{V}} \mathcal{S}_i}{|\mathcal{V}|}.$$ (Average degree)

D-Spot try to find a set of nodes that maximizing the density metric.

The limitations of existing competitor[24,2]:

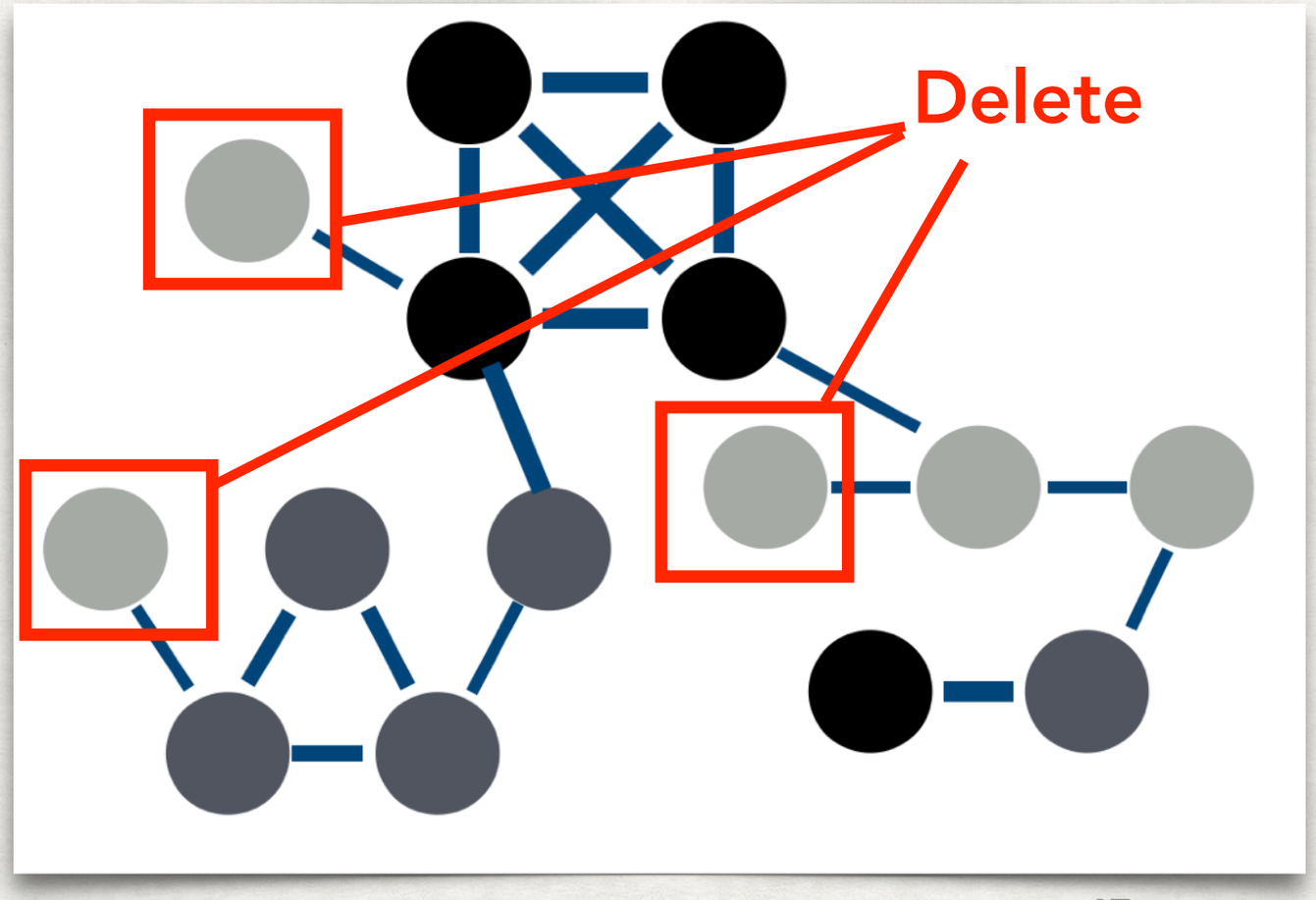
(1) only detect a single dense graph at one run.

(2) in practice, the speed is low.

METHODS: D-SPOT

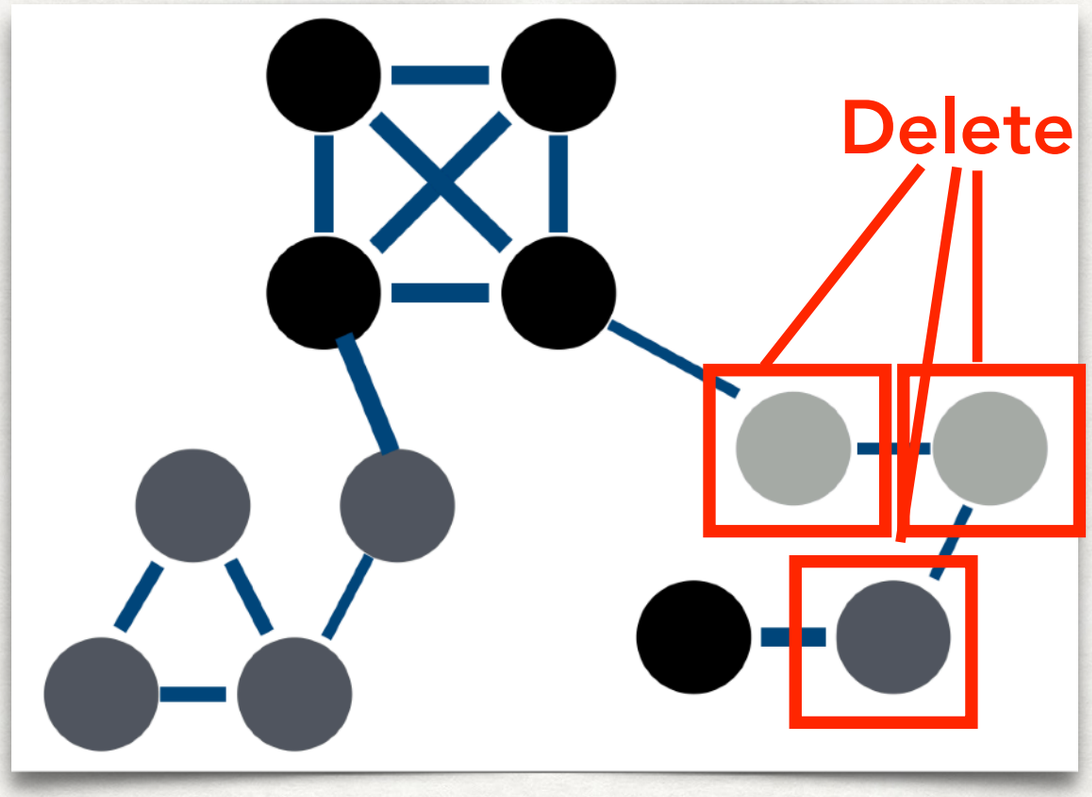
The improvement/difference of D-spot:

(1) D-Spot can detect multiple densest subgraphs simultaneously. D-Spot first partitions the graph, and then detects a single densest subgraph in each partition.
(2) Instead of removing nodes one by one, D-Spot removes a set of nodes at once, reducing the number of iterations.

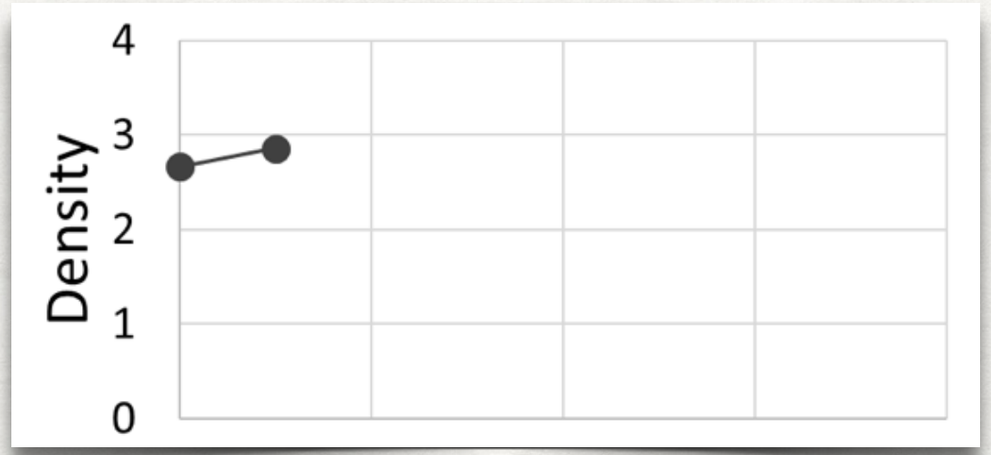


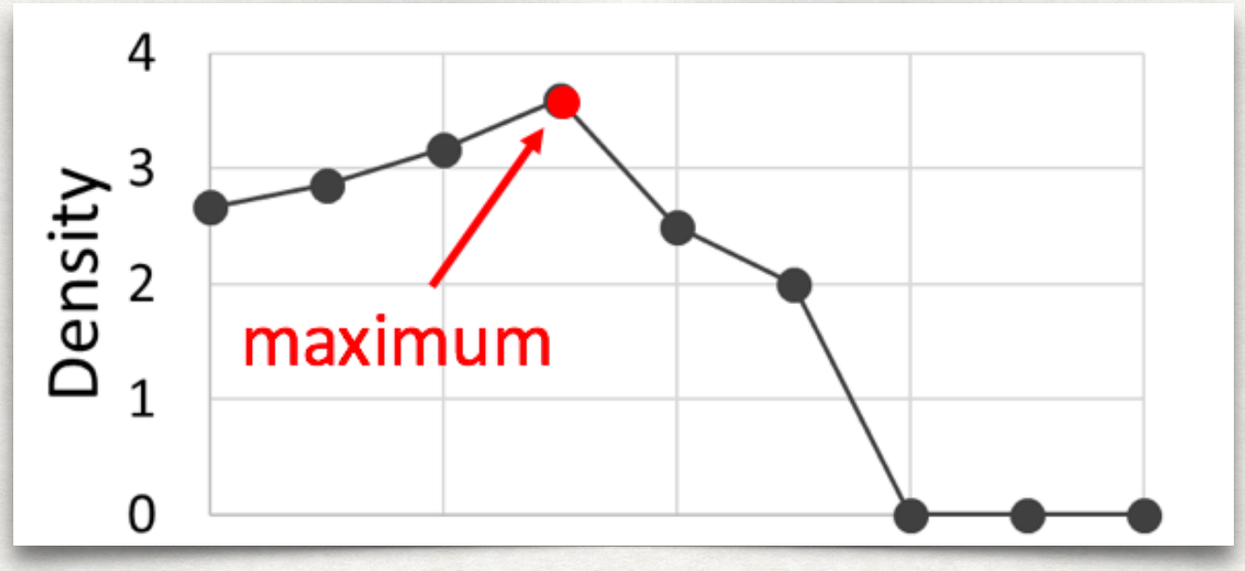
Iteration 1

METHODS: D-SPOT



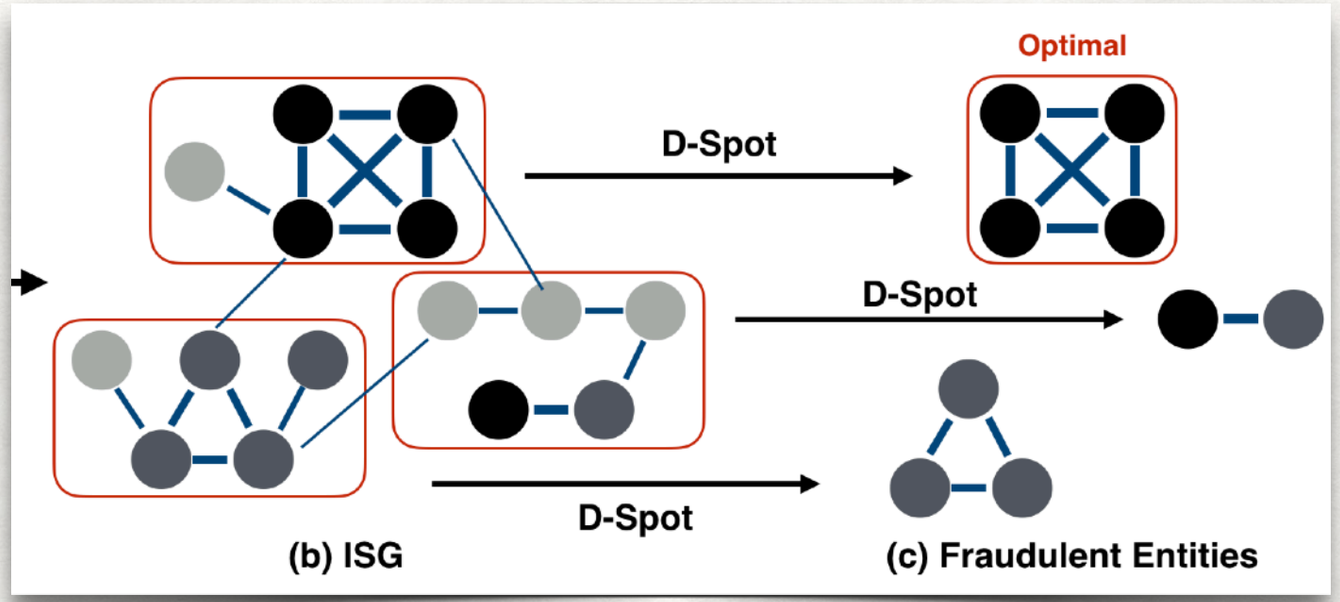
Iteration 2





Return the set of nodes maximizing the density metric

METHODS: D-SPOT



D-Spot Accuracy Guarantee: 1/2 Approximation Guarantee

<u>the multiple subgraphs found by D-Spot must contain some subgraphs that are at least 1/2 as dense as the optimum.</u>

(Proof in paper)

METHODS: ISG + D-SPOT

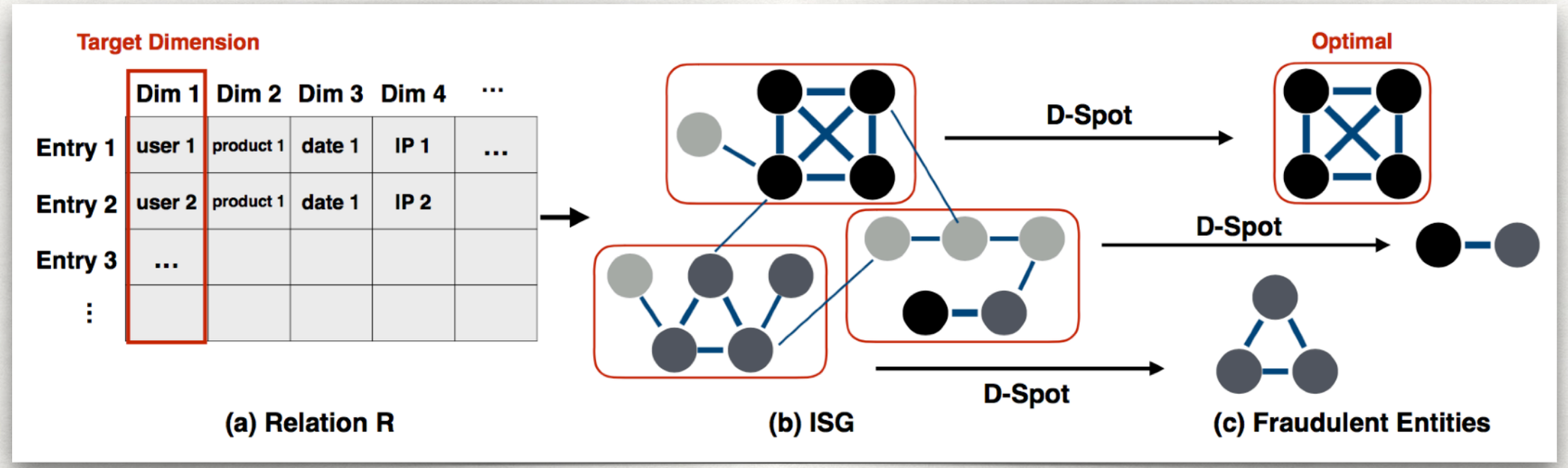


Table 1: ISG+D-Spot vs. existing dense-block detection methods.

	MAF [21]	CrossSpot [12]	M-zoom [31]	M-biz [32]	D-cube[33]	ISG+D-Spot
Applicable to N-dimensional data?	√	√	√	√	√	√
Catch densest blocks?	√	√	√	√	√	√
Catch hidden-densest blocks?	×	×	×	×	×	√
%-Approximation Guarantee?	×	×	1/N	×	1/N	1/2

EXPERIMENTS AND EVALUATIONS

Baselines:

- D-Cube[16], WSDM 2017
- M-Zoom[6], PKDD 2017
- M-Biz[17], TKDD 2018

Table 3: Multi-dimensional datasets used in our experiments

	Synthetic	TCP Dumps		Review Data					
		DARPA	AirForce	YelpChi	YelpNYU	YelpZip	AmaOffice	AmaBaby	AmaTools
Entries (Mass)	10k	4.6M	30k	67k	359k	1.14M	53k	160k	134k
Dimensions	7	3	3	3	3	3	3	3	3

λ, which denotes the number of dimensions on which B is the densest.

Table 4: Performance (AUC) on the Synthetic datasets

	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 4$	$\lambda = 5$
M-Zoom (ari)	0.5005	0.5005	0.5000	0.5489	0.6567
M-Zoom (geo)	0.5005	0.5005	0.5000	0.6789	0.7543
M-Zoom (sus)	0.7404	0.7715	0.8238	0.9685	0.9767
M-Biz (ari)	0.5005	0.5005	0.5005	0.5005	0.6834
M-Biz (geo)	0.5005	0.5005	0.5005	0.6235	0.7230
M-Biz (sus)	0.6916	0.7638	0.8067	0.9844	**0.9948**
D-Cube (ari)	0.5005	0.5005	0.5005	0.5670	0.6432
D-Cube (geo)	0.5005	0.5005	0.5340	0.6876	0.6542
D-Cube (sus)	**0.8279**	**0.8712**	**0.9148**	**0.9909**	0.9725
ISG+D-Spot	**0.9843**	**0.9957**	**0.9949**	**1.0000**	**1.0000**

ISG + D-Spot catch hidden-densest blocks.

EXPERIMENTS AND EVALUATIONS

Table 5: Performance (AUC) on the Amazon datasets

	AmaOffice	AmaBaby	AmaTools
M-Zoom (ari)	0.6795	0.5894	0.8689
M-Zoom (geo)	0.8049	0.8049	1.0000
M-Zoom (sus)	0.7553	0.6944	0.6503
M-Biz (ari)	0.6328	0.5461	0.8384
M-Biz (geo)	0.8049	**0.8339**	1.0000
M-Biz (sus)	0.7478	0.6944	0.6503
D-Cube (ari)	0.7127	0.5956	0.6250
D-Cube (geo)	**0.8115**	0.7561	1.0000
D-Cube (sus)	0.7412	0.6190	0.5907
ISG+D-Spot	**0.8358**	**0.9995**	**1.0000**

Table 6: Performance (AUC) on the Yelp datasets

	YelpChi	YelpNYU	YelpZip
M-Zoom (ari)	0.9174	0.6669	0.8859
M-Zoom (geo)	0.9752	0.8826	0.9274
M-Zoom (sus)	0.9831	**0.9451**	0.9426
M-Biz (ari)	0.9174	0.6669	0.8863
M-Biz (geo)	0.9757	0.8826	0.9271
M-Biz (sus)	**0.9831**	0.9345	**0.9403**
D-Cube (ari)	0.5000	0.5000	0.9033
D-Cube (geo)	0.9793	0.9223	0.9376
D-Cube (sus)	0.9810	0.9007	0.9365
ISG+D-Spot	**0.9875**	**0.9546**	**0.9529**

ISG + D-Spot all wins!

Table 7: Performance (AUC) on the DARPA dataset

Dataset	DARPA	
	U = source IP	U = target IP
M-Zoom (ari)	0.5649	0.5584
M-Zoom (geo)	0.7086	**0.5714**
M-Zoom (sus)	0.6989	0.3878
M-Biz (ari)	0.5649	0.5584
M-Biz (geo)	**0.7502**	0.5679
M-Biz (sus)	0.6989	0.3878
D-Cube (ari)	0.3728	0.5323
D-Cube (geo)	0.4083	0.3926
D-Cube (sus)	0.4002	0.3720
ISG+D-Spot	**0.7561**	**0.8181**

✦ EXPERIMENTS AND EVALUATIONS

Table 8: Performance (AUC) on the AirForce dataset

	Sample 1	Sample 2	Sample 3
M-Zoom (ari)	0.8696	0.8675	0.8644
M-Zoom (geo)	0.9693	0.9693	0.9738
M-Zoom (sus)	0.9684	0.9683	0.9726
M-Biz (ari)	0.9038	0.8848	0.8885
M-Biz (geo)	0.9694	0.9693	0.9741
M-Biz (sus)	0.9684	0.9683	0.9726
D-Cube (ari)	**0.9824**	**0.9823**	0.9851
D-Cube (geo)	0.9695	0.9691	**0.9862**
D-Cube (sus)	0.9697	0.9692	0.9737
ISG+D-Spot	**0.9835**	**0.9824**	**0.9877**

Table 9: Performance (AUC) on the Registration dataset. 'C' represents 'crucial feature' and 'N' represents 'noisy feature'

	1C	2C	2C+1N	2C+2N	2C+3N
M-Zoom (ari)	0.5000	0.5000	0.5031	0.5000	0.5430
M-Zoom (geo)	0.7676	0.8880	**0.8827**	**0.8744**	**0.8439**
M-Zoom (sus)	0.7466	0.8328	0.4009	0.4878	0.4874
M-Biz (ari)	0.5000	0.5000	0.5000	0.5000	0.5004
M-Biz (geo)	**0.7677**	0.8842	0.8827	0.8744	0.8439
M-Biz (sus)	0.7466	0.8328	0.4009	0.4878	0.4874
D-Cube (ari)	0.7073	0.8189	0.8213	0.8295	0.7987
D-Cube (geo)	0.7073	**0.9201**	0.8586	0.8312	0.7324
D-Cube (sus)	0.7522	0.8956	0.7877	0.7642	0.7080
ISG+D-Spot	**0.7699**	**0.9946**	**0.9935**	**0.9917**	**0.9859**

ISG + D-Spot all wins!

SUMMARY

1) [Graph Model]. We propose the novel ISG model, which converts every value sharing in a tensor to the representation of weighted edges or nodes (entities). Furthermore, our graph model considers diverse data types and their corresponding distributions based on information theory to automatically prioritize multiple features.
2) [Algorithm]. We propose the D-Spot algorithm, which is able to find multiple densest subgraphs in one run. And we theoretically prove that the multiple subgraphs found by D-Spot must contain some subgraphs that are at least 1/2 as dense as the optimum. In real- world graphs, D-Spot is up to 11× faster than the state-of-the-art competing algorithm.
3) [Effectiveness]. In addition to dense blocks, ISG+D-Spot also effectively differentiates hidden-densest blocks from normal ones. In an N-dimensional tensor, the entity group found by the ISG+D-Spot is at least 1/2 of the optimal group regarding density, compared with the 1/N guarantee ensured by competing methods. In experiments using eight public real-world datasets, ISG+D-Spot detected fraudulent entities more accurately than conventional methods.

FraudTrap: Catching Loosely Synchronized Behavior in Face of Camouflage

Yikun Ban

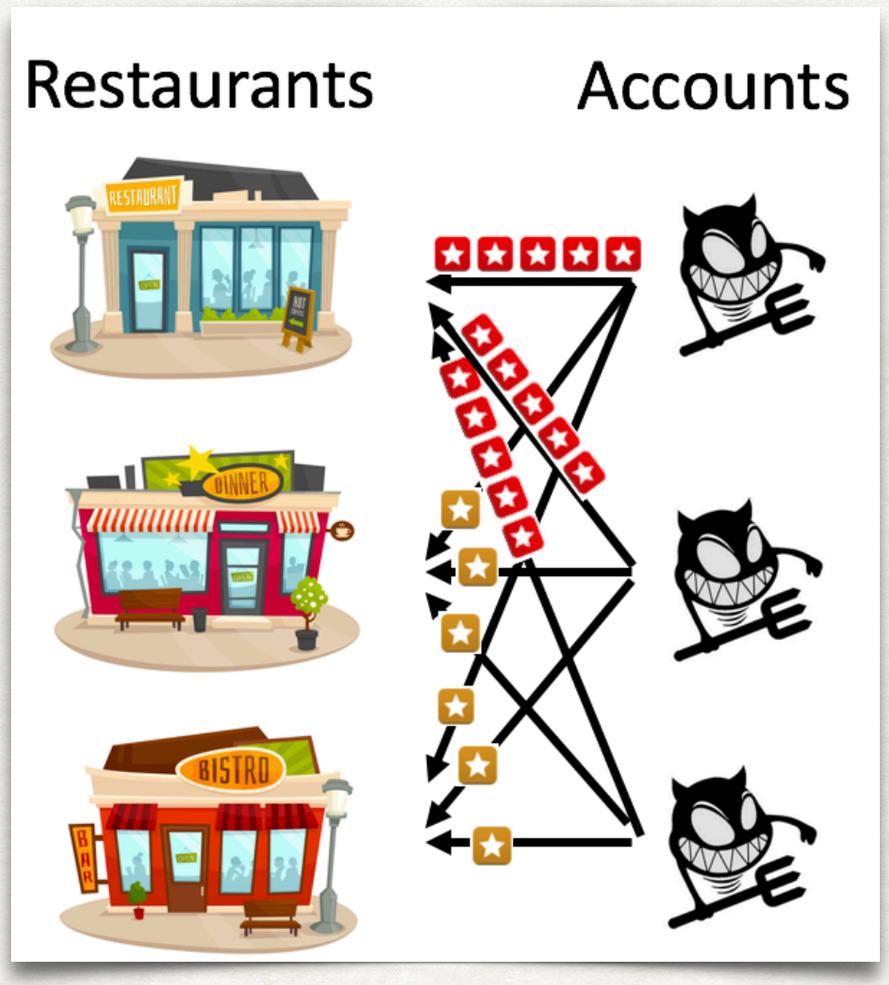
Advisors: Wei Xu, Yitao Duan, and Ling Huang

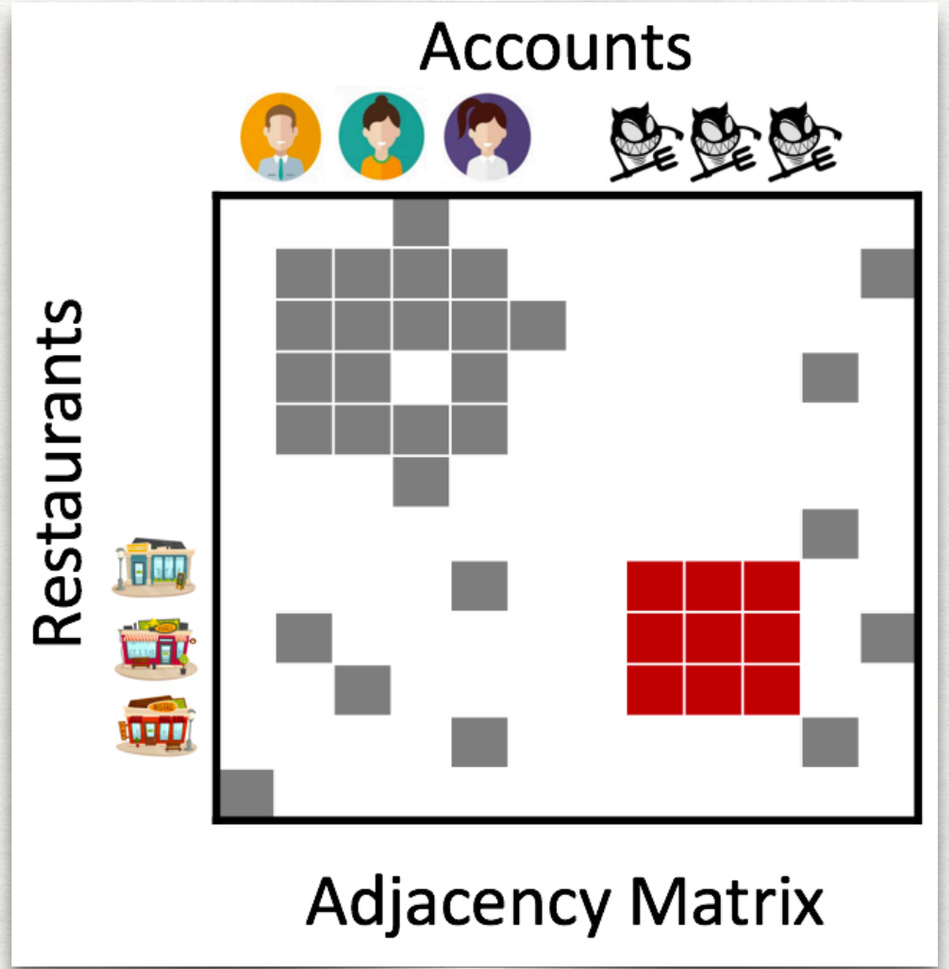
Teammates: Jiao Sun (THU), Xin Ling

- Motivation
- Problem Definition
- Metheds
- Experiments and Evaluations

MOTIVATION

Fraud is a severe threat to social networks such as Amazon, Yelp, and Twitter, etc.





MOTIVATION

Strategies of "smart" fraudster:

Loosely synchronized behavior and **Camouflage**

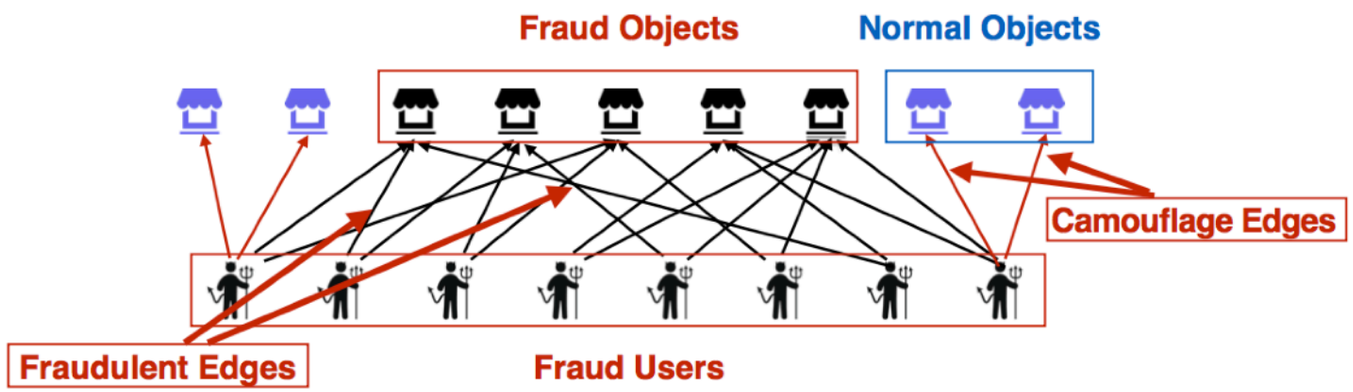


Figure 1: (1) Loosely synchronized behavior: fraudsters increase the number of fraud users and multiplex them. (2) Camouflage: fraud users create edges to normal objects.

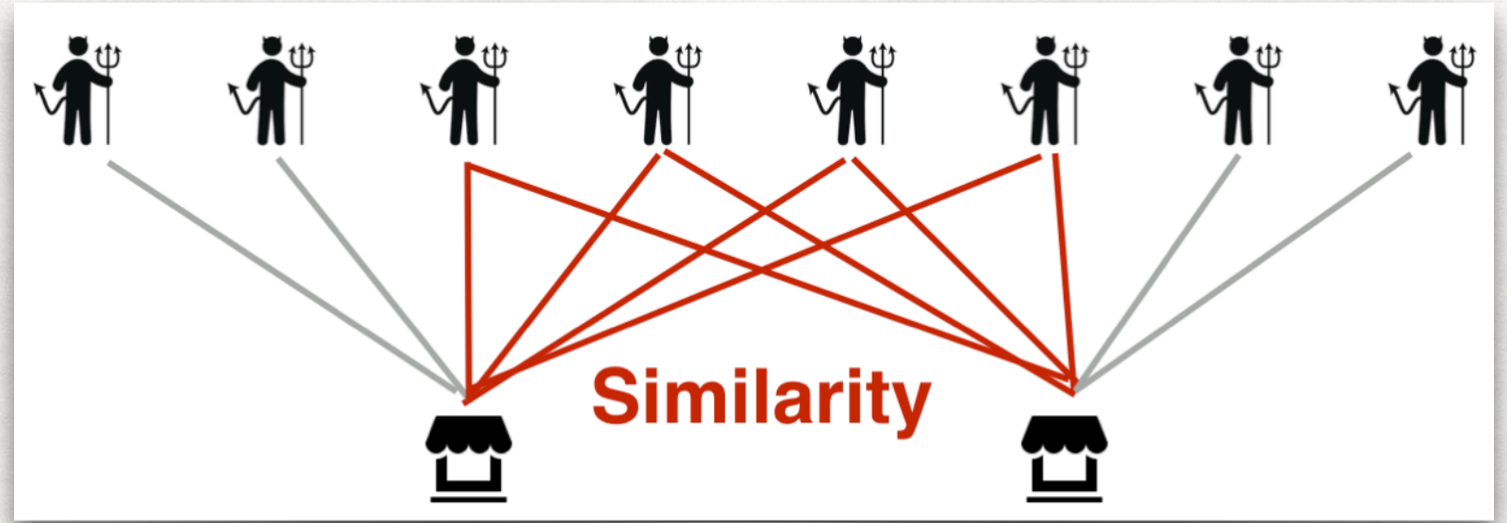
Challenge: by multiplexing a larger pool of fraud users, a fraudster can effectively reduce the density of the subgraph induced by a fraud group.

Dense-subgraph detection approaches **do not work any more !!!**

✦METHODS: OBJECT SIMILARITY GRAPH (OSG)

How to catch loosely synchronized behavior?

Similarity is key!



Similarity Score: following Jaccard similarity[19]

$$\mathcal{S}_{ij} = \frac{|\mathbf{I}_i \cap \mathbf{I}_j|}{|\mathbf{I}_i \cup \mathbf{I}_j|}.$$

where $\mathbf{I}_i = \{\varepsilon_{ii} : \forall n_i \in \mathbf{N}, (n_i, m_i) \in E\}$ be the set of edges pointing to m_i .

m_i is a object.

✦METHODS: OBJECT SIMILARITY GRAPH (OSG)

Unsupervised mode: sometimes, we can obtain a set of labeled fraud users.

The bipartite graph Gl formed by a set of labeled fraud users Nl and the same object set M.

In G^l, let $\mathbf{I}_i^l = \{\varepsilon_{ii} : \forall n_i \in \mathbf{N}^l, (n_i, m_i) \in E^l\}$ represent the set of edges pointing to m_i. Then the similarity score $\mathcal{S}_{ij}^l$ between m_i and m_j is given by:

$$\mathcal{S}_{ij}^l = \frac{\left|\mathbf{I}_i^l \cap \mathbf{I}_j^l\right|}{mean(\mathbf{I}^l)}, \tag{3}$$

where $mean(\mathbf{I}^l)$ is the mean of the set $\mathbf{I}^l = \{|\mathbf{I}_i^l \cap \mathbf{I}_j^l| : \forall m_i, m_j \in \mathbf{M}, |\mathbf{I}_i^l \cap \mathbf{I}_j^l| > 0\}$.

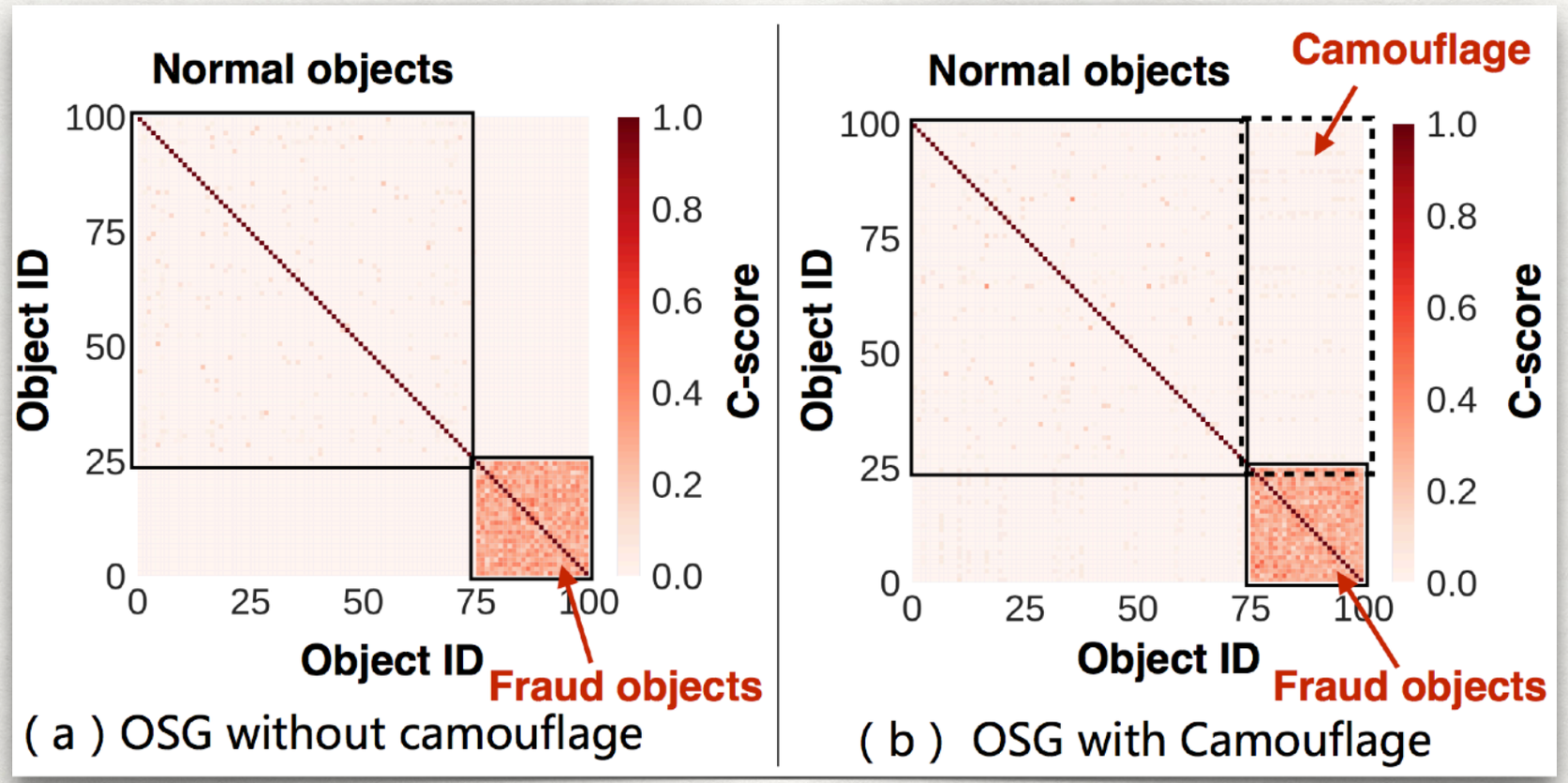
✦METHODS: OBJECT SIMILARITY GRAPH (OSG)

the similarity score between object m_i and object m_j as

$$C_{ij} = \mathcal{S}_{ij} + \mathcal{S}^l_{ij},$$

where $\mathcal{S}_{ij}$ is the similarity score calculated from the unlabeled G, while $\mathcal{S}^l_{ij}$ is obtained from the labeled G^l.

Edge representation: $$\varepsilon_{ij} = (n_i, Attr_1, Attr_2 \cdots),$$



(a) OSG without camouflage

(b) OSG with Camouflage

METHODS: ALGORITHM LPA-TK

LPA-TK is a similarity-based clustering algorithm, inspired by Label Propagation Algorithms[20,21].

Obviously, it is significant to design **the update criteria** f that determines the final results.

Algorithm 1 LPA-TK

Require: $\mathbf{G} = (\mathbf{M}, \mathbf{E})$
Ensure: $\hat{\mathcal{M}}$s

```
for each m_i ∈ M do L_i^0 = m_i # Initialize labels
{M_1, ..., M_δ} ← M # Color partition: no two adjacent vertices
share the same color.
t ← 0, M̂s ← ∅
while the stop criterion is not met do
    t ← t + 1  # The t-th iteration
    for M ∈ {M_1, ..., M_δ} do
        for each m_i ∈ M do
            L_i^t = f(m_i, t) # To Algorithm 2
for each l ∈ {L_i, ∀m_i ∈ M} do
    M̂s ← M̂s + {{m_i ∈ M : L_i = l}}
return M̂s
```

The workflow of LPA-TK

METHODS: ALGORITHM LPA-TK

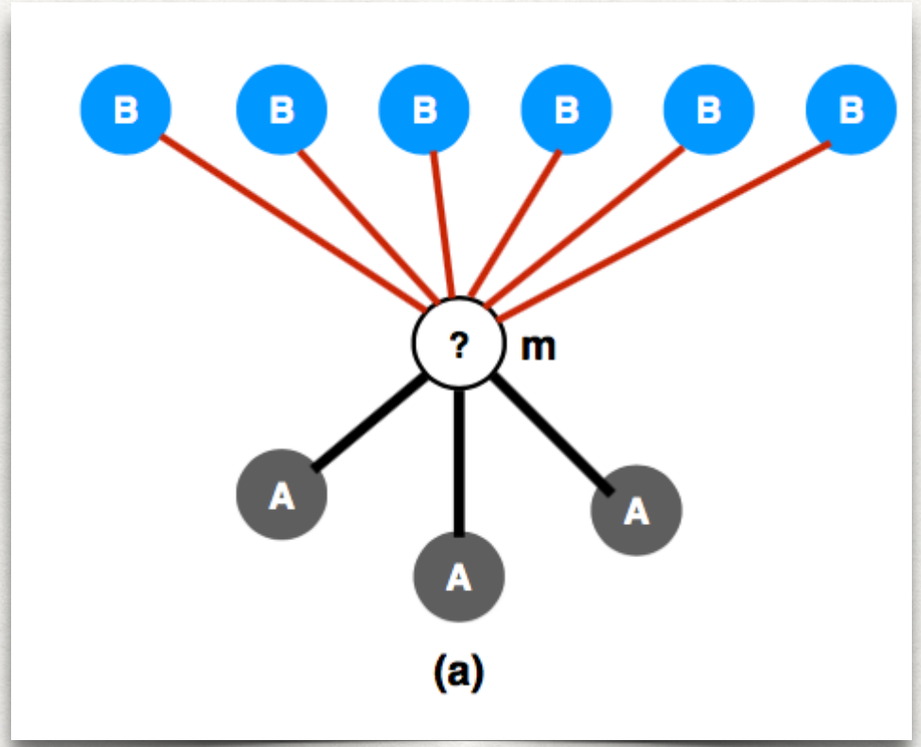
Update Criteria 1: Sum, inspired by [20,21]

$$f(m_i, t) = \underset{l \in \{L_j^t, \forall m_j \in \mathbf{H}(m_i)\}}{\arg\max} \sum_{m_j \in \mathbf{H}(m_i)} C_{ij} h(L_j^t, l), \qquad (4)$$

where $\mathbf{H}(m)$ is the set of neighbors of m_i and $h(L_j^t, l)$ is an indicator function:

$$h(L_j^t, l) = \begin{cases} 1 & \text{if } L_j^t == l. \\ 0 & \text{otherwise.} \end{cases}$$

the label of m_i is determined by the sum of edge weights of each distinct label among its neighbors.
Unfortunately, the results of clustering deteriorate as the camouflage edges increase.



METHODS: ALGORITHM LPA-TK

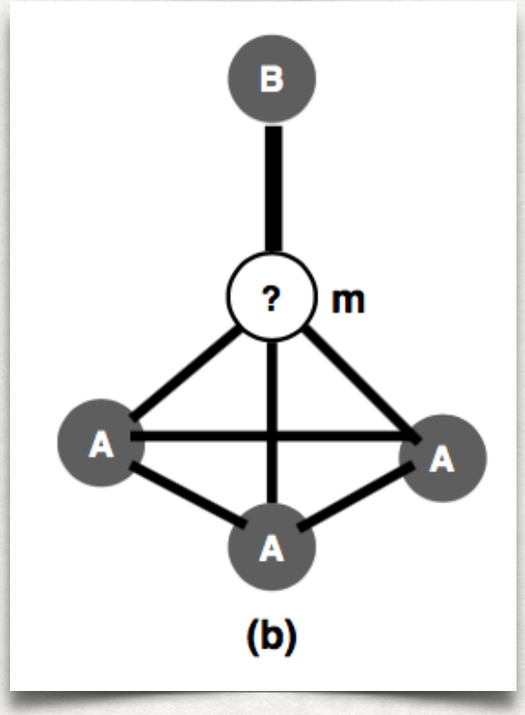
Update Criteria 2: Max

$$f(m_i, t) = \underset{l \in \{L_j^t, \forall m_j \in \mathrm{H}(m_i)\}}{\arg\max} \; h(l, m_i) \tag{5}$$

where $\mathrm{H}(m_i)$ is the set of neighbors of m_i and

$$h(l, m_i) = \max\{C_{ij} \in \mathbf{E} : (m_j \in \mathrm{H}(m_i)) \wedge (L_j^t = l)\}$$

the label of mi is determined by the maximal edge weight of each distinct label among its neighbors. Although Eq.5 can eliminate the influence of camouflage because the most similar neighbor of a fraud object should also be fraudulent, the result of clustering is not well and a group of objects often is divided into multiple parts.



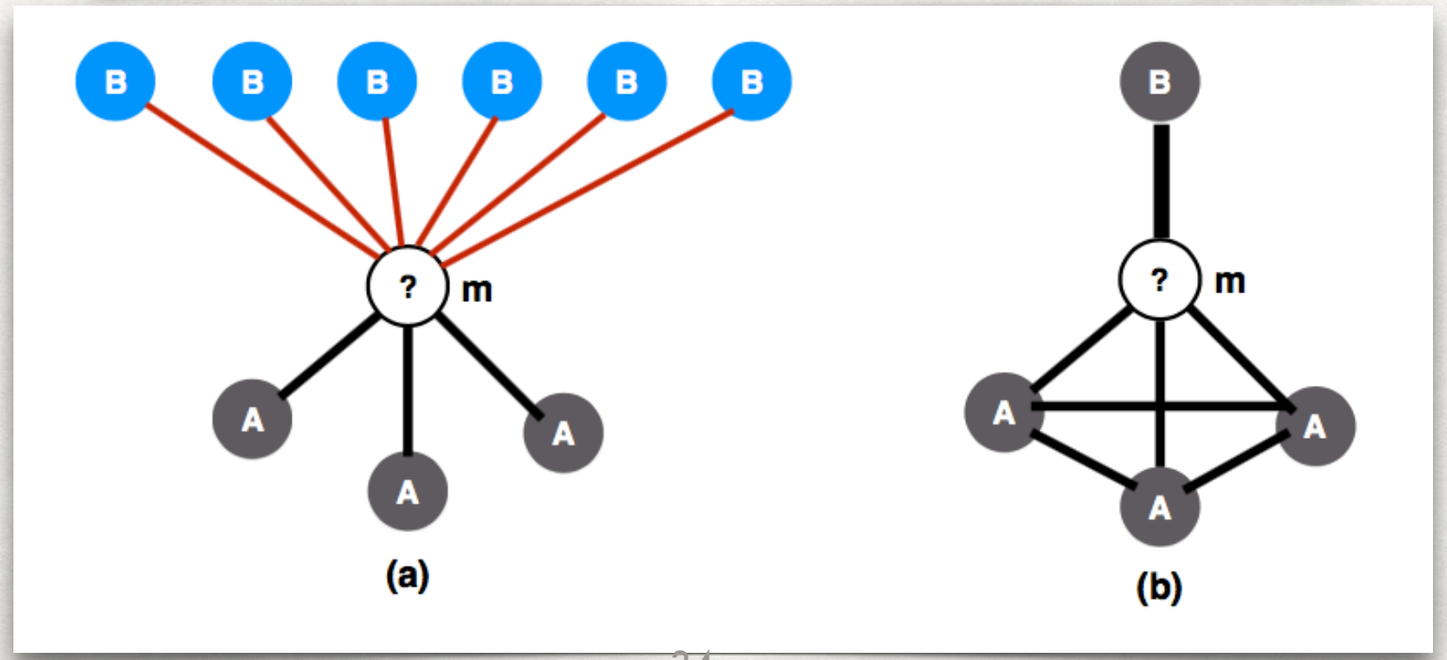
METHODS: ALGORITHM LPA-TK

Update criteria: Top-K most similar neighbors

Algorithm 2 $f(m_i, t)$

Require: K

for each $l \in \{L_j^t, \forall m_j \in \mathbf{H}(m_i)\}$ **do** # $\mathbf{H}(m_i)$ is the set of neighbors of m_i
  $h(l) \leftarrow 0$
  $\mathbf{C} \leftarrow \{C_{ij} \in \mathbf{E} : (m_j \in \mathbf{H}(m_i)) \wedge (L_j^t = l)\}$
  while $K > 0$ **do**
    if $|\mathbf{C}| \geq K$ **then**
      $h(l) \leftarrow h(l) + \max \mathbf{C}$
      remove $\max \mathbf{C}$ from $\mathbf{C}$
    $K \leftarrow K - 1$
return l if $h(l)$ is the maximum in h



✦ METHODS: FIND SUSPICIOUS GROUPS

After generating all candidate groups, how to spot fraud groups?

We propose an interpretable suspiciousness metric F to score each group and find top suspicious groups.

$$\mathcal{F}_{\mathcal{G}} = \frac{\sum_{(m_i, m_j) \in \mathcal{E}} (C_{ij} + |\mathbf{I}_i \cap \mathbf{I}_j|)}{|\mathcal{M}|(|\mathcal{M}| - 1)^2}, \tag{6}$$

where $\mathcal{F}_{\mathcal{G}} = \mathcal{F}_{\mathcal{G}}{}^1 \cdot \mathcal{F}_{\mathcal{G}}{}^2 \cdot |\mathcal{M}|$,

$$\mathcal{F}_{\mathcal{G}}{}^1 = \frac{\sum_{(m_i, m_j) \in \mathcal{E}} C_{ij}}{|\mathcal{M}|(|\mathcal{M}| - 1)}$$

and

$$\mathcal{F}_{\mathcal{G}}{}^2 = \frac{\sum_{(m_i, m_j) \in \mathcal{E}} |\mathbf{I}_i \cap \mathbf{I}_j|}{|\mathcal{M}|(|\mathcal{M}| - 1)}.$$

Intuitively, $\mathcal{F}_{\mathcal{G}}{}^1$ is the average value of C-score on all edges of $\mathcal{G}$, $\mathcal{F}_{\mathcal{G}}{}^2$ is the average number of edges pointed from same users on all object pairs of $\mathcal{G}$.

✦ METHODS: FIND SUSPICIOUS GROUPS

The advantage of F -score is that the score obeys the following good properties including axioms proposed in [15] that all good algorithms should have.

edge density denoted by $\rho_{edge} = \frac{|\mathcal{E}|}{|\mathcal{M}|(|\mathcal{M}|-1)}$. And '=' represents 'not change'.

(i) AXIOM 1. *[Object Size]*. Keeping ρ_{edge}, C_{ij}, and $|\mathbf{I}_i \cap \mathbf{I}_j|$ fixed, a larger $\mathcal{G}$ is more suspicious than one with a smaller size.

$$({\mathcal{F}_\mathcal{G}}^1 =) \wedge ({\mathcal{F}_\mathcal{G}}^2 =) \wedge (|\mathcal{M}| \uparrow) \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$$

(ii) AXIOM 2. *[Object Similarity]*. Keeping ρ_{edge}, $|\mathbf{I}_i \cap \mathbf{I}_j|$, and $|\mathcal{M}|$ fixed, a $\mathcal{G}$ with more similar object pairs is more suspicious.

$$C_{ij} \uparrow \Rightarrow ({\mathcal{F}_\mathcal{G}}^1 \uparrow) \wedge ({\mathcal{F}_\mathcal{G}}^2 =) \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$$

(iii) AXIOM 3. *[User Size]*. Keeping ρ_{edge}, C_{ij}, and $|\mathcal{M}|$ fixed, a fraud object group ($\mathcal{G}$) connected with more fraud users is more suspicious.

$$|\mathbf{I}_i \cap \mathbf{I}_j| \uparrow \Rightarrow ({\mathcal{F}_\mathcal{G}}^1 =) \wedge ({\mathcal{F}_\mathcal{G}}^2 \uparrow) \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$$

(iv) AXIOM 4. *[Edge Density]*. Keeping C_{ij}, $|\mathbf{I}_i \cap \mathbf{I}_j|$, and $|\mathcal{M}|$ fixed, a denser $\mathcal{G}$ is more suspicious.

$$\rho_{edge} \uparrow \Rightarrow ({\mathcal{F}_\mathcal{G}}^1 \uparrow) \wedge ({\mathcal{F}_\mathcal{G}}^2 \uparrow) \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$$

Let $\sum_{(m_i,m_j)\in\mathcal{E}}(C_{ij} + |\mathbf{I}_i \cap \mathbf{I}_j|)$ be the same total suspiciousness.

(v) AXIOM 5. *[Concentration.]* With the same total suspiciousness, a smaller $\mathcal{G}$ is more suspicious.

$$|\mathcal{M}| \downarrow \Rightarrow \mathcal{F}_\mathcal{G} \uparrow$$

METHODS: FRAUDTRAP

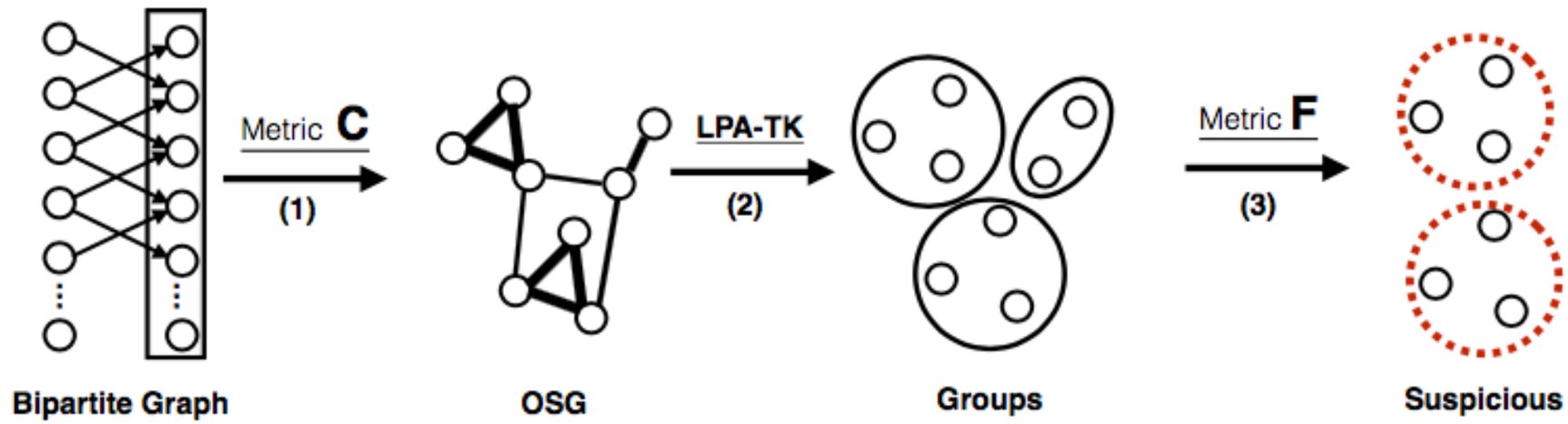


Figure 2: The workflow of FraudTrap ($\mathcal{C}$+LPA-TK+$\mathcal{F}$).

(1) OSG construction: The OSG captures object similarity, and we design a metric, C-score, to capture the similarity between two objects based on user interactions.

(2) Clustering on OSG: We propose a similarity-based clustering algorithm that clusters each object into a group based on its most similar neighbors on OSG.

(3) Spot suspicious groups: Given candidate groups, it is important to use an interpretable metric to capture how suspicious an object/user group is, relative to other groups. We design the F -score metric for the purpose.

EXPERIMENTS AND EVALUATIONS

Baselines:

- Fraudar[2], the best paper, KDD 2016;
- CatchSync[1], best paper finalist, KDD 2014;
- CrossSpot[15], TKDE 2016

Table 3: Datasets used in experiments.

datasets	edges	datasets	edges
AmazonOffice[18]	53K	YelpChi[32]	67K
AmazonBaby[18]	160K	YelpNYC[32]	359K
AmazonTools[18]	134K	YelpZip[32]	1.14M
AmazonFood[18]	1.3 M	DARPA[23]	4.55M
AmazonVideo[18]	583K	Registration	26k
AmazonPet[18]	157K		

✦ EXPERIMENTS AND EVALUATIONS

LPA-TK VS Baselines:

Injected a fraud group M into Amazon datasets

Table 4: Clustering performance on the [Amazon] datasets. 'Num' represents 'the number of groups $\mathcal{M}$ is divided into. θ denotes the number of camouflage edges of each fraud user.

	$\theta = 0$	$\theta = 5$	$\theta = 10$	$\theta = 20$
	Num|AUC	Num|AUC	Num|AUC	Num|AUC
LPA	1 |1.0	1 |0.787	1 | 0.727	1 |0.731
LPA-Sum	1 | 1.0	1 | 1.0	1 | 0.787	1 | 0.761
LPA-Max	14 | 0.998	14 | 0.996	13 | 0.998	10 |0.991
LPA-TK	**1 |1.0**	**1 |1.0**	**1 | 0.999**	**1| 0.998**

LPA-TK is the best on accuracy and clustering performance!

✦ EXPERIMENTS AND EVALUATIONS

The performance of FraudTrap

In a fraud object group M, for a fraud n, \rho is synchrony metric:

$$\rho = \frac{\overline{|\mathcal{W}|}}{|\mathcal{M}|},$$

W is the subset of M that n is connected with,
M is the set of all fraud objects

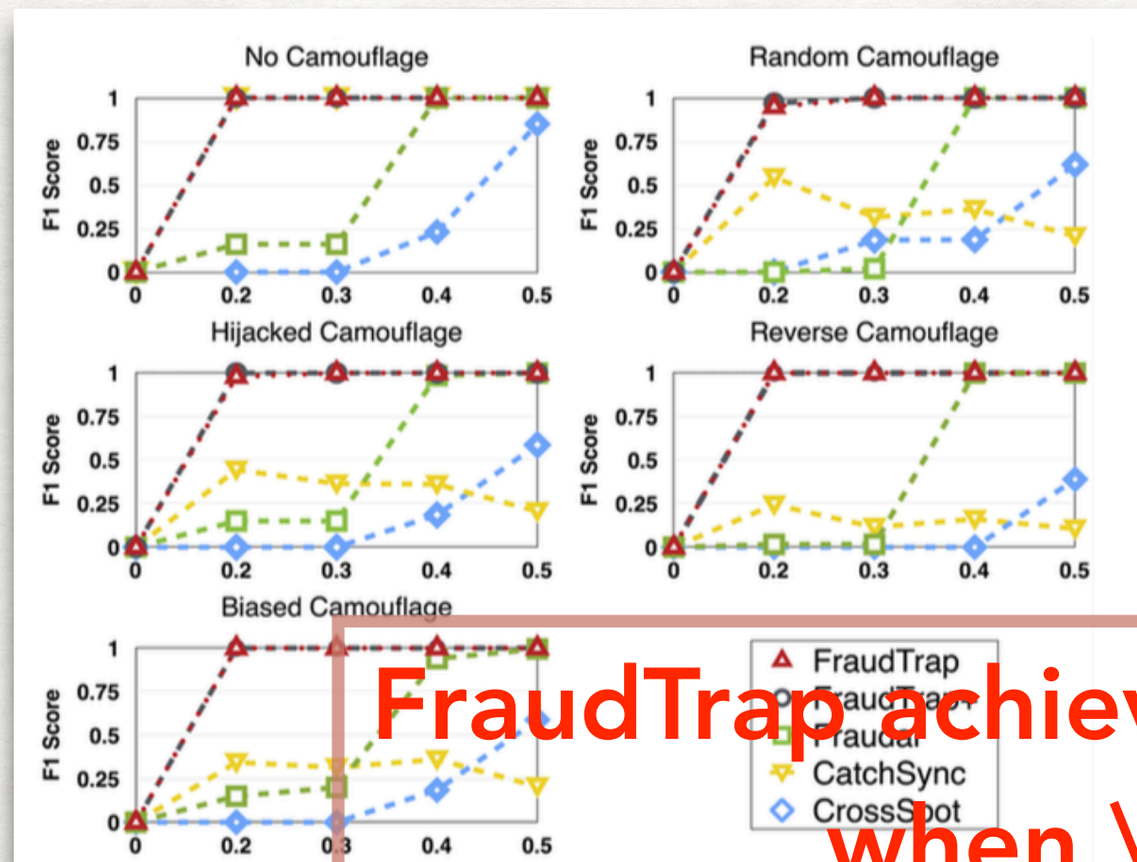


Figure 5: X-axis: ρ. Performance on detecting fraud objects

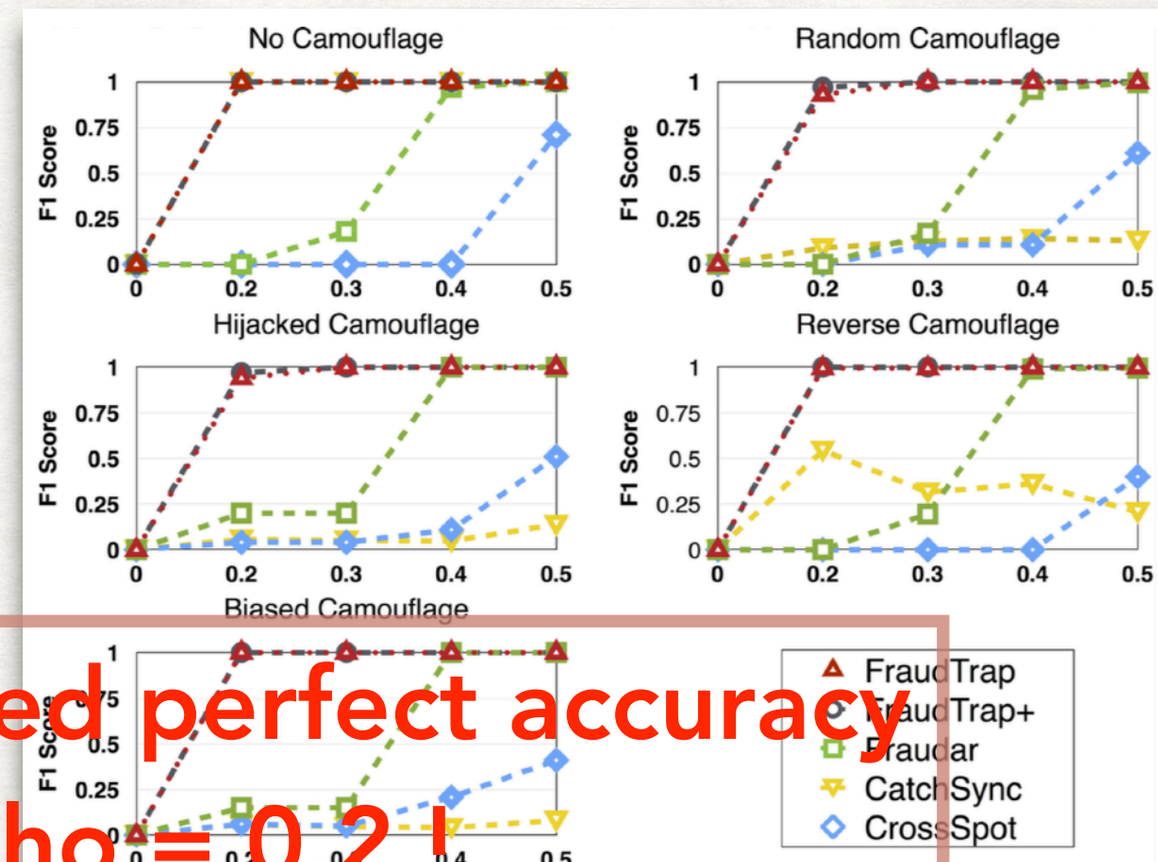


Figure 6: X-axis: ρ. Performance on detecting fraud users

FraudTrap achieved perfect accuracy when \rho = 0.2 !

EXPERIMENTS AND EVALUATIONS

The performance of FraudTrap

Table 5: Performance(AUC) of detecting fraud objects on the [Amazon] datasets

	AmazonOffice	AmazonBaby	AmazonTools
Fraudar	0.8915	0.8574	0.8764
CatchSync	0.8512	0.8290	0.8307
CrossSpot	0.8213	0.8342	0.7923
FraudTrap	**0.9987**	**0.9495**	**0.9689**
FraudTrap+	**0.9987**	**0.9545**	**0.9675**
	AmazonFood	AmazonVideo	AmazonBook
Fraudar	0.6915	0.7361	0.8923
CatchSync	0.7612	0.7990	0.7634
CrossSpot	0.7732	0.7854	0.8324
FraudTrap	**0.8458**	**0.8651**	**0.9534**
FraudTrap+	**0.8758**	**0.8951**	**0.9644**

Table 6: Performance(AUC) of detect fraud users on the [Amazon] datasets

	AmazonOffice	AmazonBaby	AmazonTools
Fraudar	0.9015	0.8673	0.8734
CatchSync	0.8732	0.8391	0.8304
CrossSpot	0.8113	0.8422	0.7823
FraudTrap	**1**	**0.9795**	**0.9796**
FraudTrap+	**1**	**0.9845**	**0.9855**
	AmazonFood	AmazonVideo	AmazonBook
Fraudar	0.7213	0.7451	0.8815
CatchSync	0.7234	0.8243	0.7763
CrossSpot	0.7653	0.7913	0.8532
FraudTrap	**0.8637**	**0.8843**	**0.9572**
FraudTrap+	**0.8818**	**0.9111**	**0.9579**

FraudTrap all wins!

Table 7: Performance(AUC) on the [YELP] datasets

	YelpChi	YelpNYC	YelpZip
Fraudar	0.9905	0.8531	0.7471
CatchSync	0.9889	0.8458	0.7779
CrossSpot	0.9744	0.7965	0.7521
FraudTrap	**0.9905**	**0.8613**	**0.7793**
FraudTrap+	**0.9905**	**0.8653**	**0.7953**

Table 8: Performance(AUC) on the [DARPA] dataset

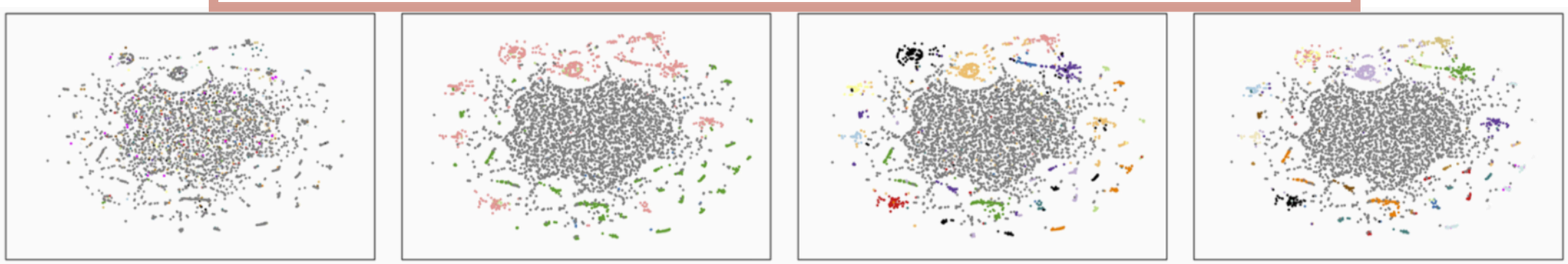
Detection of	source IP	destination IP
Fraudar	0.7420	0.7298
CatchSync	0.8069	0.8283
CrossSpot	0.7249	0.6784
FraudTrap	**0.9968**	**0.9920**
FraudTrap+	**0.9968**	**0.9920**

EXPERIMENTS AND EVALUATIONS

Table 9: Performance (AUC) on the [Registration] dataset

	Feature: IP	Feature: phone
Fraudar	0.7543	0.8742
CatchSync	0.7242	0.8435
CrossSpot	0.6976	0.8231
FraudTrap	**0.7658**	**0.8979**
FraudTrap+	**0.7826**	**0.9113**
FraudTrap*	**0.7724**	**0.9215**

FraudTrap all wins!
and has best clustering results!



(a) CrossSpot (b) Fraudar (c) FraudTrap (d) Golden Labels

normal users other colors: fraud groups

SUMMARY

- [Metric C]. We build Object Similarity Graph (OSG) by a novel similarity metric, C-score, which transforms the sparse subgraphs induced by fraud groups in the bipartite graph into the much denser subgraphs in OSG, by merging information from unlabeled and labeled(if available) data.
- [Algorithm LPA-TK]. We propose a similarity-based cluster- ing algorithm, LPA-TK, that perfectly fits in OSG and outperforms the baseline (LPA) in face of noise edges (camouflage).
- [Metric F]. Given candidate groups returned by C + LPA- TK, we propose an interpretable suspiciousness metric, F -score, meeting the all basic “axioms” proposed in [15].
- [Effectiveness]. Our method FraudTrap (C + LPA-TK + F) can operate in two modes: unsupervised and semi-supervised. The unsupervised mode outperforms other state-of-the-art methods for catching synchronized behavior in face of camou age. Semi- supervised mode naturally takes advantage of partially labeled data to further improve the performance.

On Finding Dense Subgraphs in Bipartite Graphs: Linear Algorithms with Applications to Fraud Detection

Yikun Ban

Advisors: Yitao Duan and Ling Huang

- Motivation
- Problem Definition
- Metheds
- Experiments and Evaluation

MOTIVATION

- Dense subgraph detection is a major topic in algorithmic graph theory and their applications can be found in a wide range such as social network analysis [4], link spam detection [1,2], and bioinformatics [3].
- In this paper, we focus on mining dense subgraphs in a bipartite graph that is useful for representing relationships across pairs of disparate data.
- The work is motivated by the task of **fraud detection**

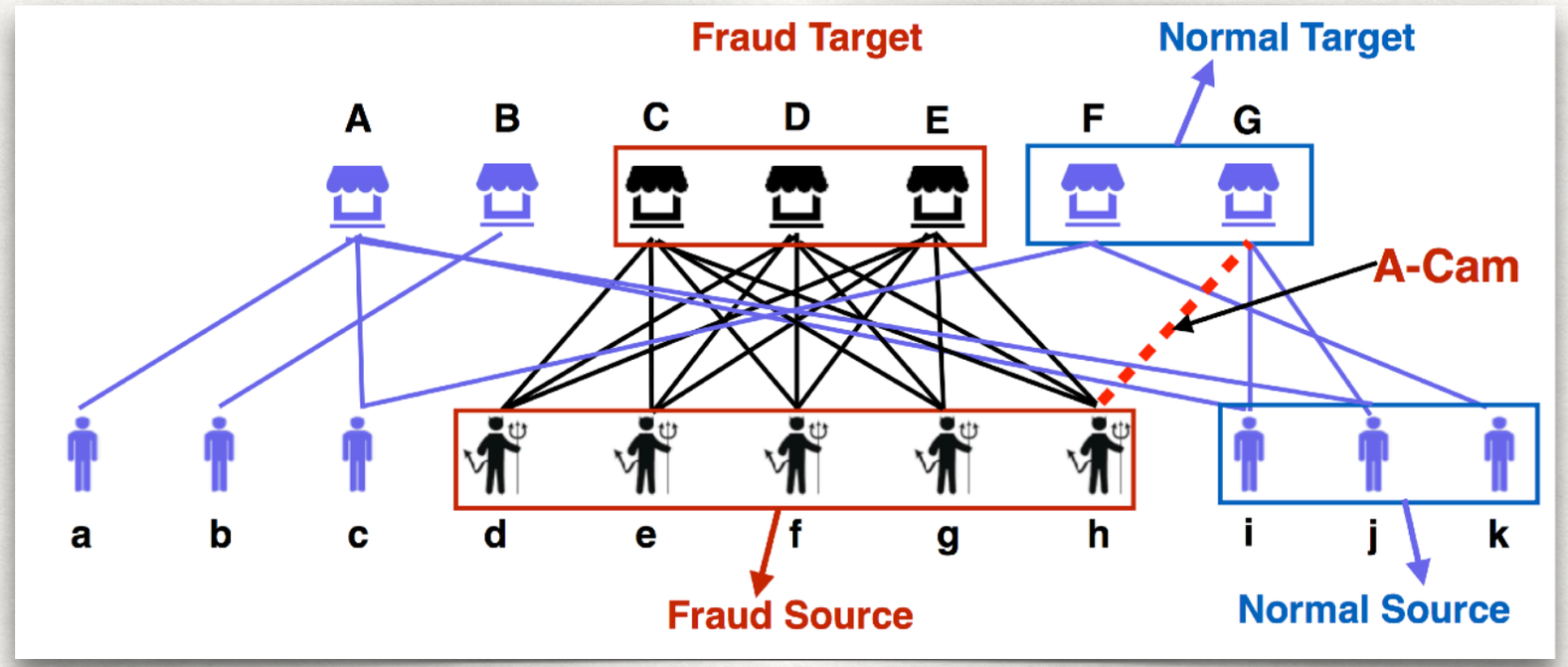
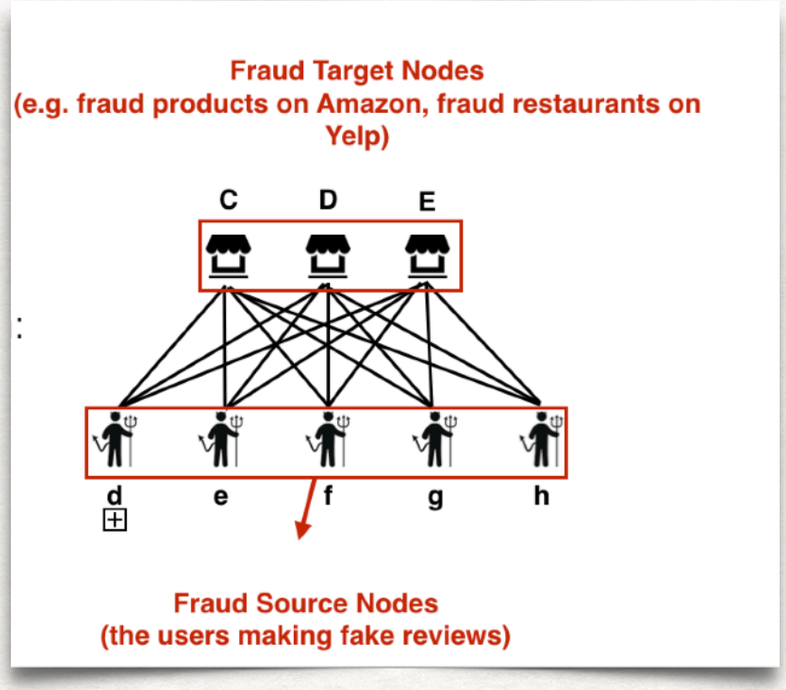


Figure 1: A bipartite graph formed by a set of source nodes and a set target nodes, where an edge represents an interaction between a source node and a target node.

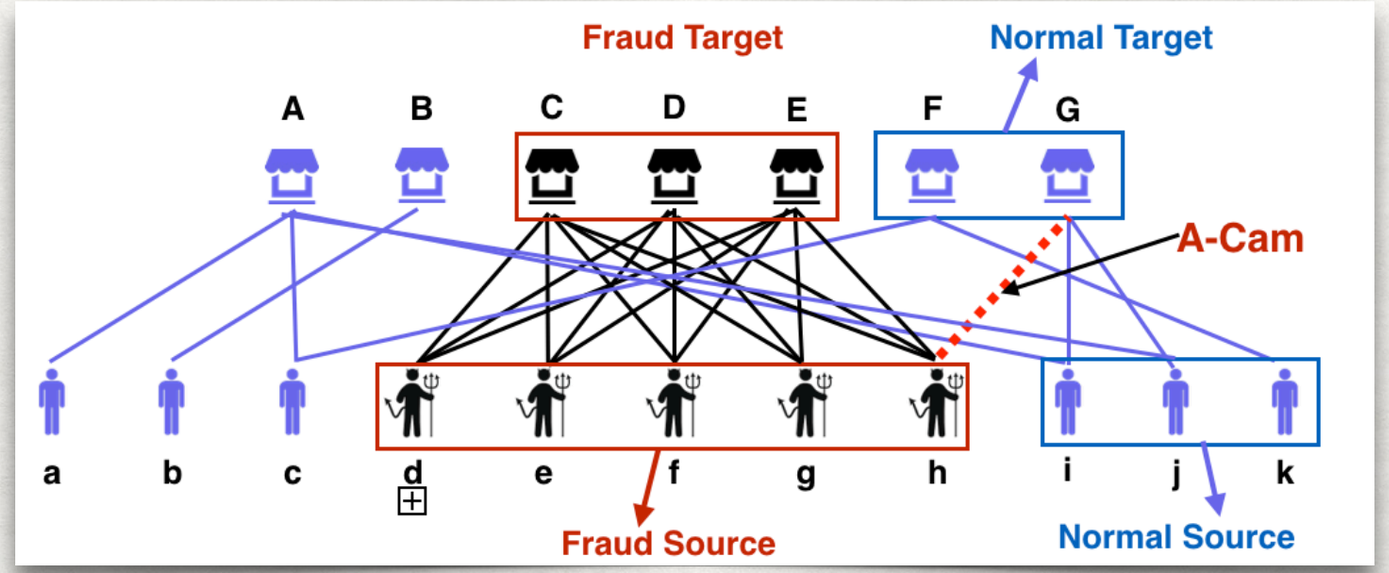
PROBLEM DEFINITION

Simple fraud attack pattern:
Isolated Biclique



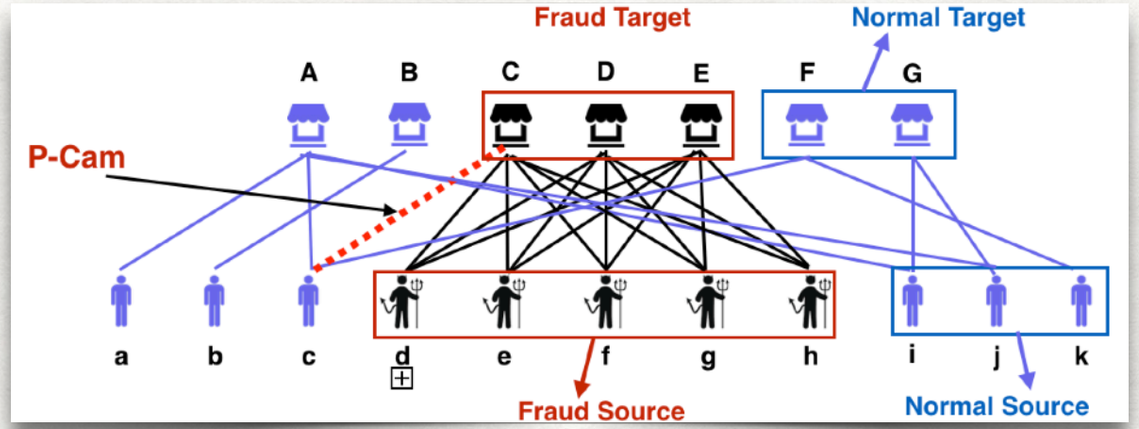
Camouflage: To mask their attack pattern, fraud source nodes introduce additional edges pointing to other non-fraud target nodes.
Camouflage can be classified into two types:
Active Camouflage (**A-Cam**) and Passive Camouflage (**P-Cam**)

Fraud attack pattern:
Isolated Biclique + A-Cam



PROBLEM DEFINITION

Fraud attack pattern:
Isolated Biclique + P-Cam



- Finding all maximal bicliques in a bipartite graphs cannot be solved in polynomial time[5].
- To formulate the attack pattern, we propose

  Half Isolated Biclique (HI Biclique) : the subgraph induced by fraud group connect to other nodes only through at most one side.

Definition 4 (Half Isolated Biclique (HI Biclique)). *In a bipartite graph* $G = (N \cup M, E)$, *a subgraph* $\mathcal{G} = (\mathcal{N} \cup \mathcal{M}, \mathcal{E})$ *is a half isolated biclique if (1)* $\mathcal{G}$ *is a biclique* $(\forall n \in \mathcal{N}, m \in \mathcal{M},\ (n,m) \in E)$; *(2)* $\{\ \forall (m \in M) \wedge (m \notin \mathcal{M}), n \in \mathcal{N}, then\ (n,m) \notin E\ \}$ *or* $\{\ \forall (n \in N) \wedge (n \notin \mathcal{N}), m \in \mathcal{M}, then\ (n,m) \notin E\ \}$.

PROBLEM DEFINITION

And

Maximal Half Isolated Biclique (MHI Biclique)

Definition 5 (Maximal Half Isolated Biclique (MHI Biclique)). *In a bipartite graph* $G = (N \cup M, E)$, *a subgraph* $\mathcal{G} = (\mathcal{N} \cup \mathcal{M}, \mathcal{E})$ *is a maximal half isolated biclique if (1)* $\mathcal{G}$ *is a HI biclique; (2) there does not exist a HI biclique* $\mathcal{G}_1 = (\mathcal{N}_1 \cup \mathcal{M}_1, \mathcal{E})$ *where* $\mathcal{N}_1 \supseteq \mathcal{N}$ *and* $\mathcal{M}_1 \supseteq \mathcal{M}$.

Maximal Half Isolated Biclique Enumeration Problem (MHIBP): In a bipartite graph $G = (N \cup M, E)$, enumerate all maximal half isolated bicliques.

MHI bicliques can also be found in many other applications such as retweet boosting detection[1], [2], network intrusion[6], and genetic applications[7], [8].

However, existing works:(1) search-based approaches[8,9]; (2) graph inflation[10];(3) association role mining[11,3,13], **cannot** solve **MHIBP**

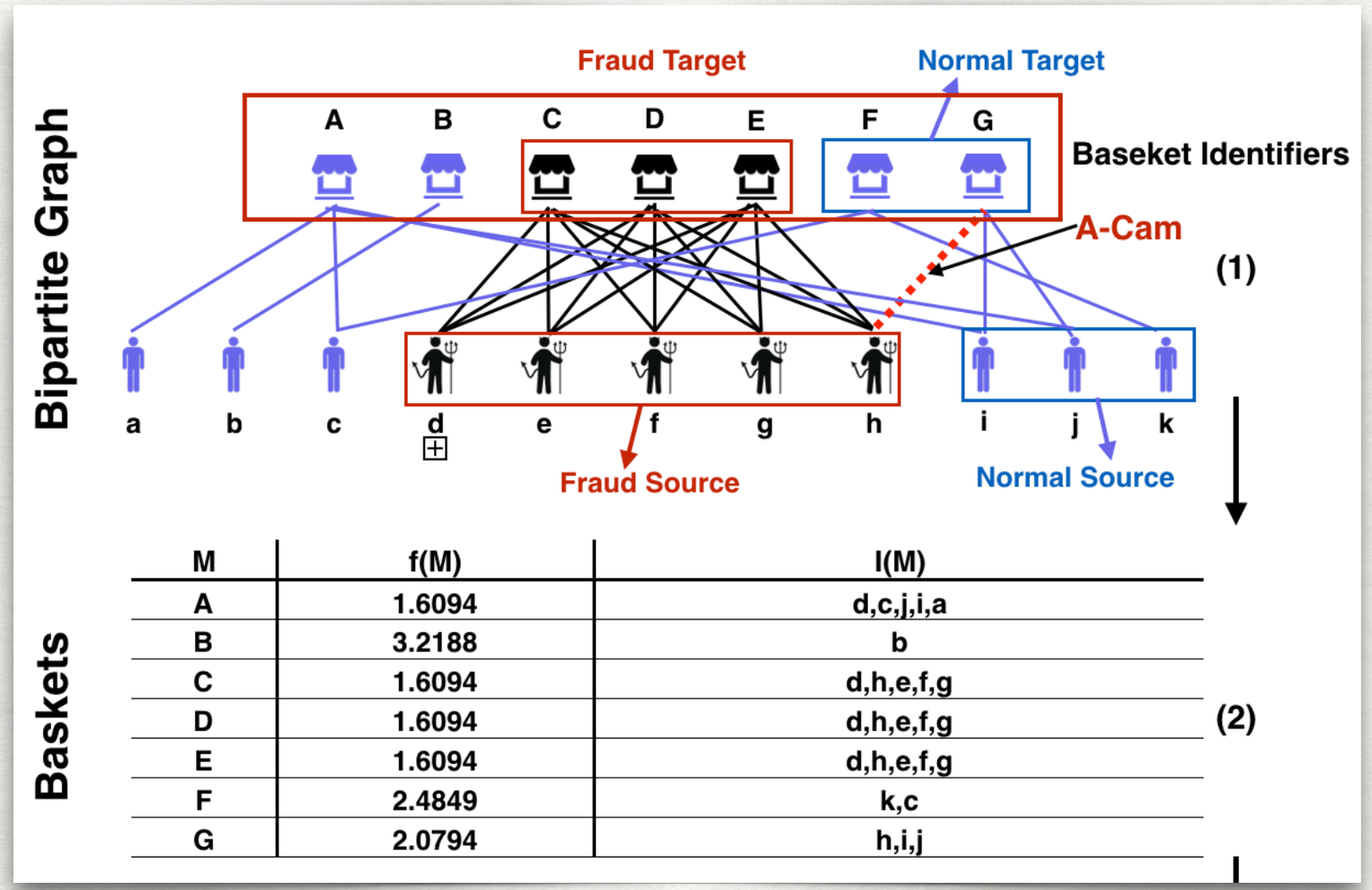
METHODS:SUSPICIOUSNESS TREE

We introduce a novel data structure, denoted by **Suspiciousness Tree** or **S-Tree**, for solving MHIBP and variants, inspired by frequent-pattern tree (FP- tree) [14]

A. Constructing Based

Basket: $B(m) = \{m,\ I(m),\ f(m)\}$

where m is the identifier of $B(m)$. $I(m) = \{n \in N : (n, m) \in E\}$ is the set of source nodes that m is connected with. $f(m)$, called the F-score, is a preliminary estimate of suspiciousness of $B(m)$, using only information local to m.

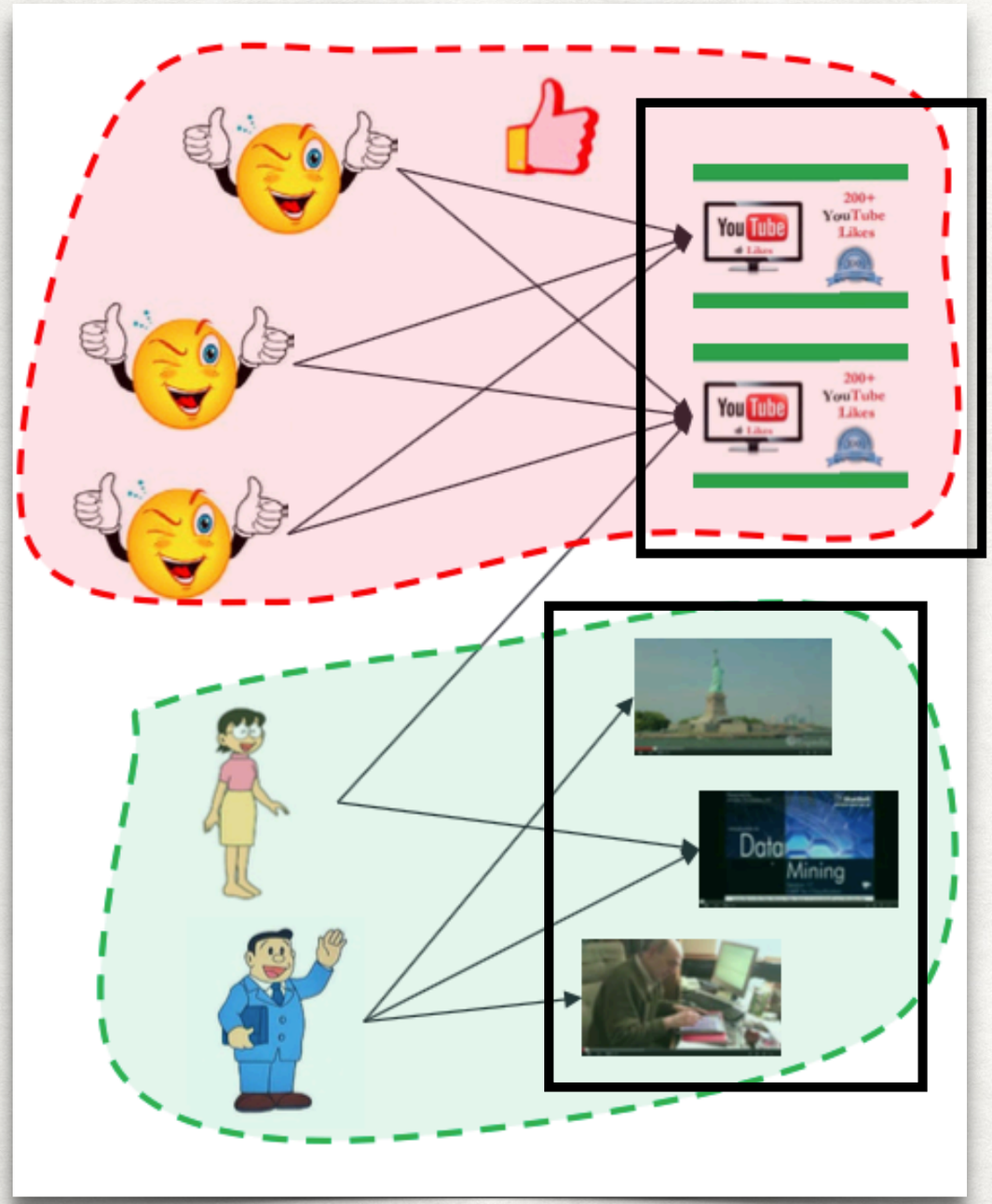


M	f(M)	I(M)
A	1.6094	d,c,j,i,a
B	3.2188	b
C	1.6094	d,h,e,f,g
D	1.6094	d,h,e,f,g
E	1.6094	d,h,e,f,g
F	2.4849	k,c
G	2.0794	h,i,j

METHODS:SUSPICIOUSNESS TREE

Determine f(m):

(1) when m is **objects** (e.g. products on Amazon, followees on Twitter).



"Nobody", low in-degree

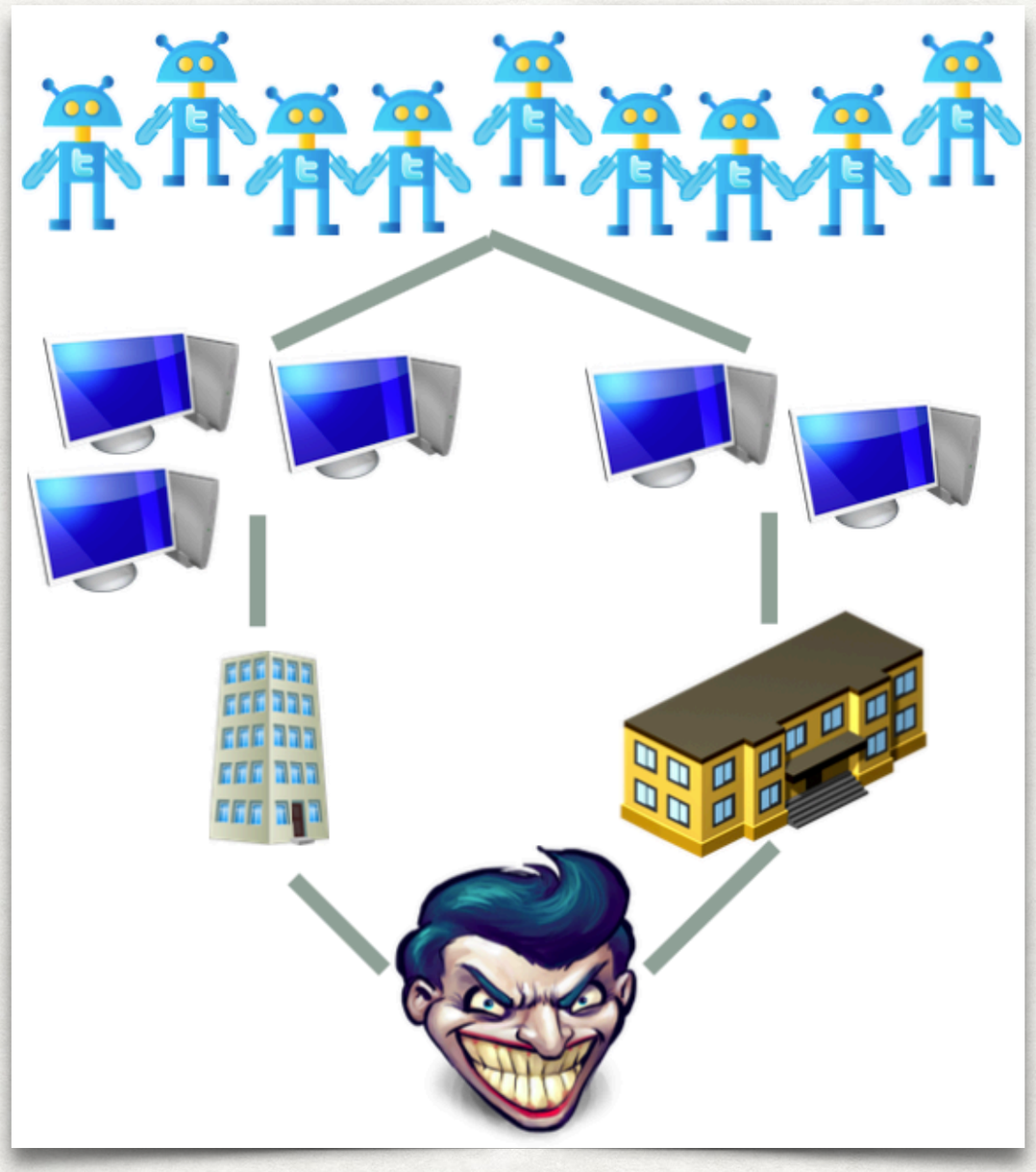
$$f(m) = log(\frac{|E|}{|I(m)| + c}),$$

"Famous", high in-degree

METHODS

Determine f(m):

(2) when m is **resource features** (e.g. IP, device ID…).



m with **higher in-degree** is **more suspicious** as <u>resource sharing</u> is a key characteristic of group fraud.

$$f(m) = log(|I(m)| + c).$$

METHODS:SUSPICIOUSNESS TREE

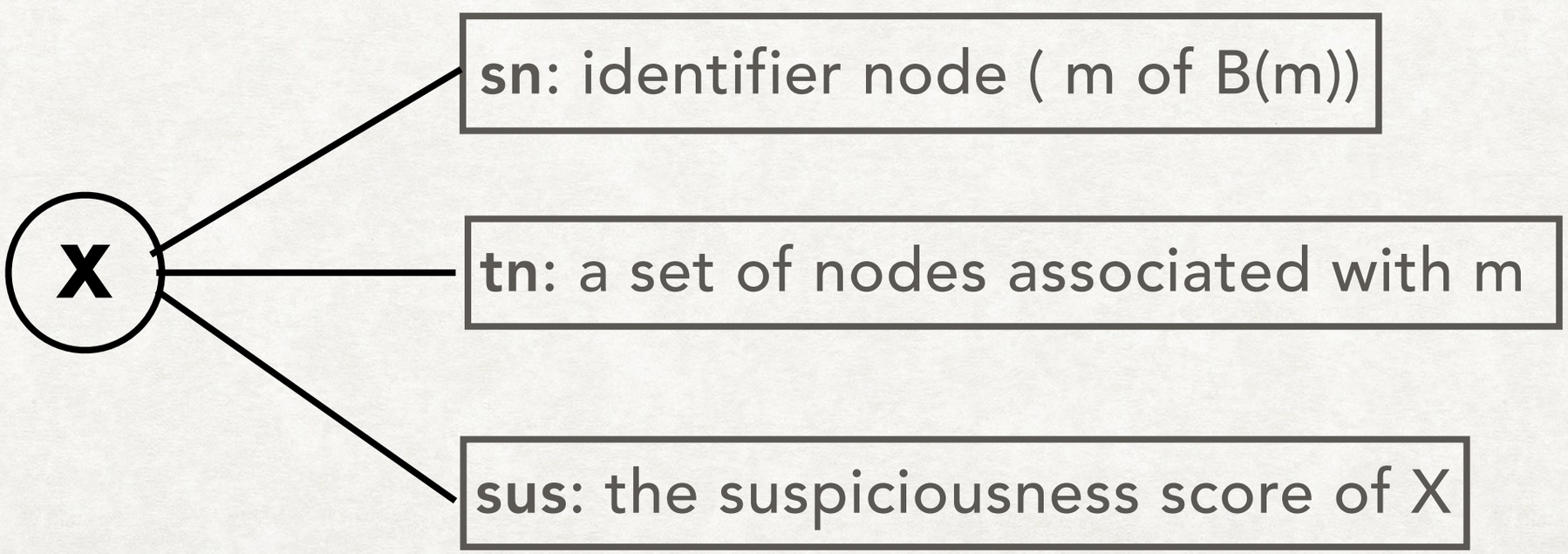
B. Building S-tree

Observation: dense subgraphs —> baskets with similar I(M)s

Baskets

M	f(M)	I(M)
A	1.6094	d,c,j,i,a
B	3.2188	b
C	1.6094	d,h,e,f,g
D	1.6094	d,h,e,f,g
E	1.6094	d,h,e,f,g
F	2.4849	k,c
G	2.0794	h,i,j

Node Construction in S-tree:



METHODS:SUSPICIOUSNESS TREE

S-tree Growth

Sorting metric:

$$g(n) = \sum_{m \in H(n)} f(m), \tag{4}$$

where $H(n) = \{m \in M : (n, m) \in E\}$ is set of target nodes that n has edge with. Metric of this from obeys two basic properties (or 'axioms'):

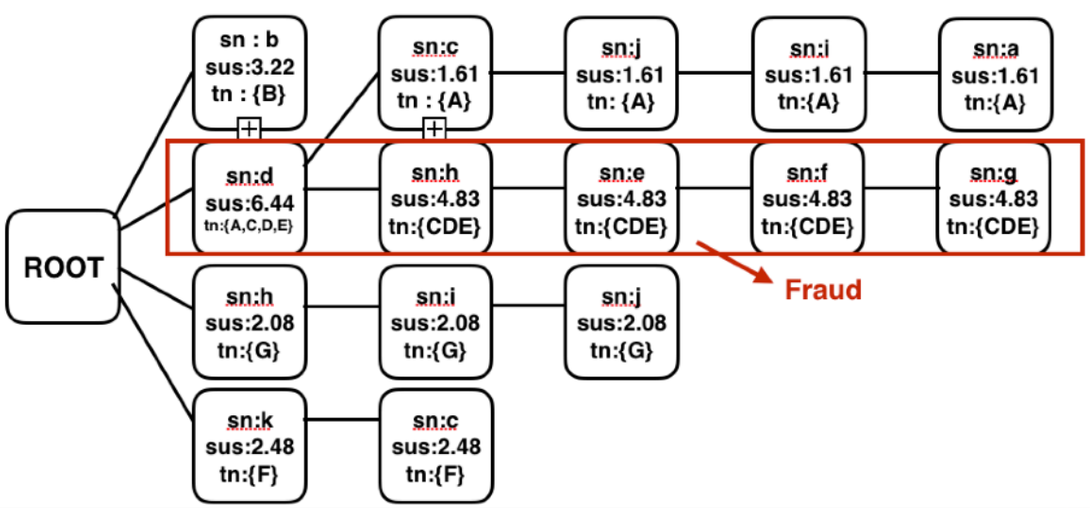
1) All other conditions being equal, the more frequent n is in $\mathbf{B}$, the greater $g(n)$: $|H(n)| \uparrow \Rightarrow g(n) \uparrow$.
2) All other conditions being equal, the more suspicious a basket that n occurs in is, the greater $g(n)$: $f(m) \uparrow \Rightarrow g(n) \uparrow$.

Baskets

M	f(M)	I(M)
A	1.6094	d,c,j,i,a
B	3.2188	b
C	1.6094	d,h,e,f,g
D	1.6094	d,h,e,f,g
E	1.6094	d,h,e,f,g
F	2.4849	k,c
G	2.0794	h,i,j

(2)

S-tree



(3)

Algorithm 1 S-tree construction

Require: $\mathbf{B}$, g
Ensure: T

```
for each B(m) ∈ B do
    sort I(m) in a descending order of g (Eq.(4))
R ← T.root
for each B(m) ∈ B do
    x ← R
    while I(m) ≠ ∅ do
        n ← I(m).top() # remove and return the first node in I(m)
        if x has a child x̂ and x̂.sn = n then
            x̂.sus ← x̂.sus + f(m)
            x̂.tn ← x̂.tn ∪ {m}
        else
            x̂ ← new node
            x̂.sn ← n, x̂.sus ← f(m), x̂.tn ← {m}
            x add a child x̂
        x ← x̂
return T
```

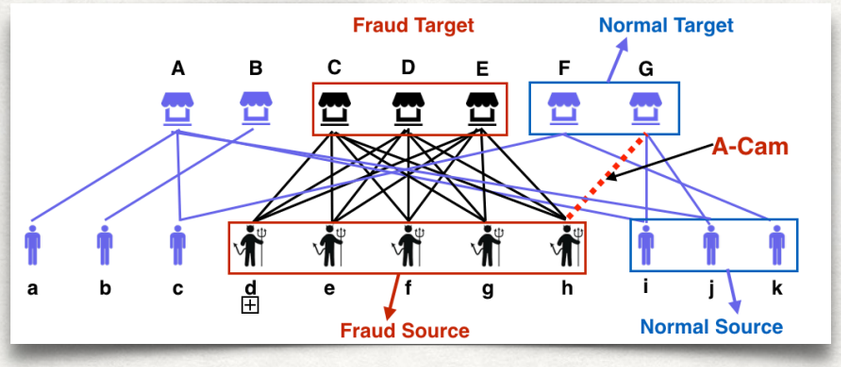
METHODS: ALGORITHM 1 FOR MHIBP

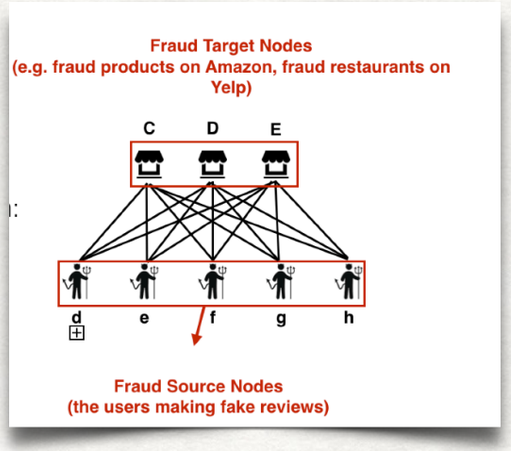
Maximal Half Isolated Bicliques

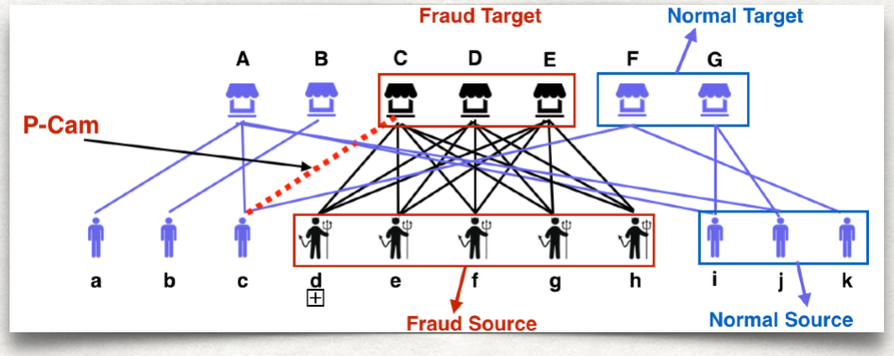
Isolated Bicliques + A-Cam

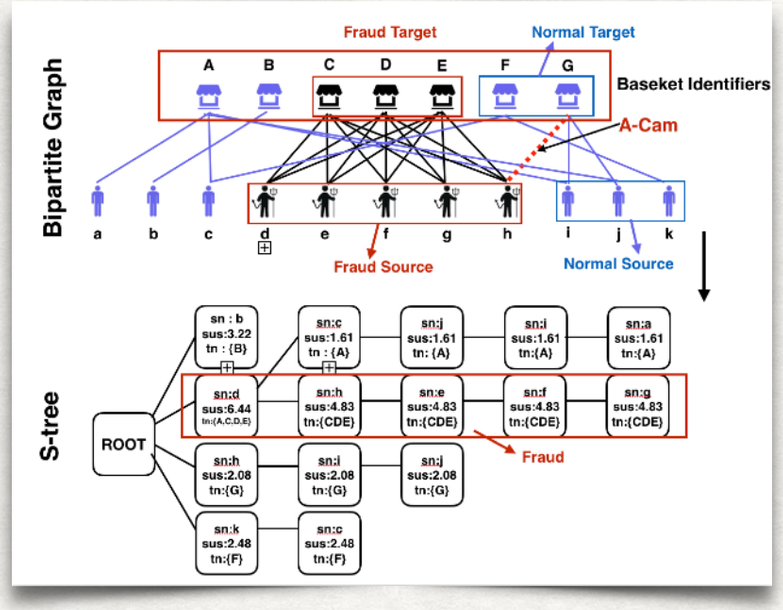
Isolated Bicliques

Isolated Bicliques + P-Cam

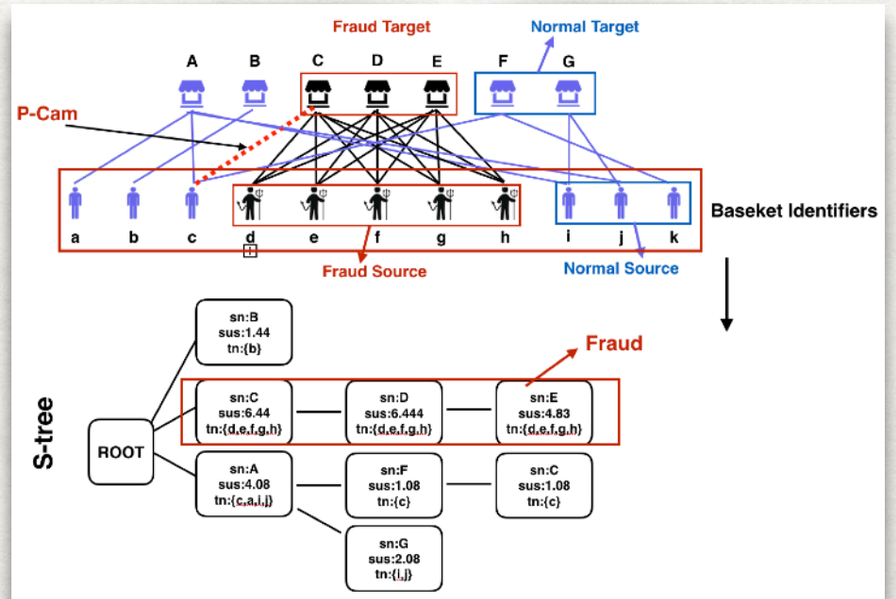








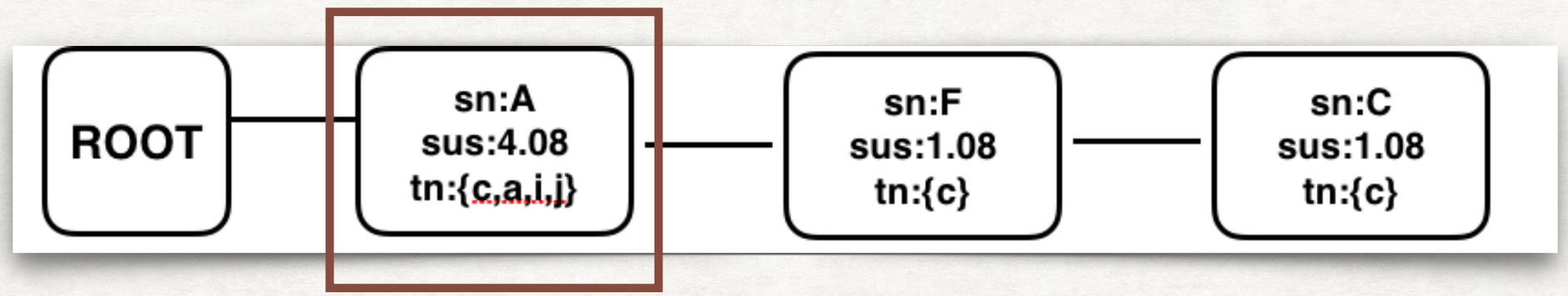
Building S-tree of Target Nodes



Building S-tree of Source Nodes

METHODS: ALGORITHM 1 FOR MHIBP

Definition 7 (Narrow Node). *In an S-tree T, let x be a node that is neither a leaf node nor branch node. And let c denote the only child of x. Then x is a narrow node if $x.tn \supset c.tn$.*



Narrow Node

Leaf Node: a node with any children.

Branch Node: is a node with more than one children.

Definition 8 (Maximal-Shared Path). *Given a path $p(x)$, then we call $p(x)$ is a maximal-shared path if x is a leaf, branch or narrow node.*

METHODS: ALGORITHM 1 FOR MHIBP

Theorem: Maximal Isolated Bicliques (+ A-Cam/P-Cam) must form <u>maximal-shared path</u> in S-tree of target/source nodes. (Proof in Paper)

Algorithm 2 *main*

Require: T
Ensure: $\hat{\mathcal{G}}s$
1: $\hat{\mathcal{G}}s \leftarrow \emptyset$
2: $R \leftarrow T.root$
3: *find MHI bicliques*$(R, \hat{\mathcal{G}}s)$
4: **return** $\hat{\mathcal{G}}s$

Algorithm 3 *find MHI bicliques*

Require: x, $\hat{\mathcal{G}}s$
1: $C \leftarrow x.childern$
2: $\mathcal{M} \leftarrow x.tn$
3: $A \leftarrow \{a.xn, \forall a \in p(x)\}$
4: **if** $|C| > 1$ **then**
5: $\quad \mathcal{M}' \leftarrow \emptyset$
6: $\quad$ **for each** $c \in C$ **do**
7: $\quad\quad \mathcal{M}' \leftarrow \mathcal{M}' \cup c.tn$
8: $\quad$ **if** $|\mathcal{M}| > |\mathcal{M}'|$ **then**
9: $\quad\quad \hat{\mathcal{G}}s \leftarrow \hat{\mathcal{G}}s \cup \{(A, \mathcal{M} - \mathcal{M}')\}$
10: **if** $|C| = 1$ **then**
11: $\quad \mathcal{M}' \leftarrow x.child.tn$
12: $\quad$ **if** $|\mathcal{M}| > |\mathcal{M}'|$ **then**
13: $\quad\quad \hat{\mathcal{G}}s \leftarrow \hat{\mathcal{G}}s \cup \{(A, \mathcal{M} - \mathcal{M}')\}$
14: **if** $|C| = 0$ **then**
15: $\quad \hat{\mathcal{G}}s \leftarrow \hat{\mathcal{G}}s \cup \{(A, \mathcal{M})\}$
16: $\quad$ **return**
17: **for each** $\hat{x} \in C$ **do**
18: $\quad$ *find MHI bicliques*$(\hat{x}, \hat{\mathcal{G}}s)$

Algorithm 4 *merge*

Require: $\hat{\mathcal{G}}_S$s, $\hat{\mathcal{G}}_T$s
1: **for each** $\hat{\mathcal{G}}$s $\in \{\hat{\mathcal{G}}_S$s, $\hat{\mathcal{G}}_T$s$\}$ **do**
2: $\quad$ **for each** $\hat{\mathcal{G}} \in \hat{\mathcal{G}}$s **do**
3: $\quad\quad (\mathcal{N}, \mathcal{M}) \leftarrow \hat{\mathcal{G}}$
4: $\quad\quad n \leftarrow \mathcal{N}.top()$ # Return a node of $\mathcal{N}$.
5: $\quad\quad$ let $p(x)$ be the path mapped by $B(n)$ in $\hat{T}$. # If $\hat{\mathcal{G}} \in \hat{\mathcal{G}}_S$s, $\hat{T} \leftarrow T_1$; Else $\hat{T} \leftarrow T$
6: $\quad\quad$ **for each** $c \in p(x)$ **do**
7: $\quad\quad\quad$ **if** c is a leaf, branch, or narrow node **then**
8: $\quad\quad\quad\quad \mathcal{G}' \leftarrow$ biclique represented by c # See Theorem 3-4
9: $\quad\quad\quad\quad$ **if** $\hat{\mathcal{G}} \subseteq \mathcal{G}'$ **then**
10: $\quad\quad\quad\quad\quad$ remove $\hat{\mathcal{G}}$ from $\hat{\mathcal{G}}$s
11: **return** $\hat{\mathcal{G}}_S$s + $\hat{\mathcal{G}}_T$s

METHODS: ALGORITHM 2

we introduce a new algorithm that detects near bicliques and can be used to catch a wide range of fraud groups.

Objective: catching shared prefix of subtree in S-tree

(1) design a suspiciousness boundary:

$$thickness = \frac{\sum_{\forall x \in T} x.sus}{|T|},$$

thickness is average suspiciousness score of each node in S-tree.

$$depth = \frac{|E| - |T|}{|\mathbf{B}|},$$

depth is the average length of shared prefix of each subtree.

suspiciousness boundary: a node x is suspicious if the depth of x in T is more than depth and x.sus is greater than thickness.

METHODS: ALGORITHM 2

Algorithm:

Algorithm 5

Require: T, *thickness*, *depth*

$X_{sus} \leftarrow \emptyset$
Start search from the root node R.
Find all nodes X of which depth $=$ *depth*.
for x in X **do**
    if $x.sus \geq$ *thickness* **then**
        $X_{sus} \leftarrow X_{sus} + p(x) + x.descendant$
return X_{sus}

Computing Suspiciousness Score:

$$s(n) = \sum_{\forall x \in X_{sus}} x.sus \ \text{ if } x.sn == n.$$

METHODS: SUSPICIOUSNESS FOREST

Motivation:

Focus on **multi-dimensional data**

My boss wants me to **catch fraud** in such a big table – **billions of records, tens of columns**!!! How?!

	ID	USER_NAME	CREATED_AT	TEXT	HASH_TAGS
1	251	SpiritSofts	Dec 14, 2013	SAP HANA ONLINE TRAINING COURSE CONTENT http://t.co/2DefOMC0Vi	
2	252	Blue net studiO	Dec 14, 2013	sap hana online training and placenet 2 http://t.co/S1wGh8n5Kk	
3	253	Hana Kingham	Dec 14, 2013	Right film fest today: love actually, elf, gravity, training day. #dayyym	dayyym,
4	254	Nora Apnila J...	Dec 14, 2013	Alhamdulilaaahhhh...selesai ikutin kelanjutan training dadakan mb Hana ...	
5	255	ZaranTech	Dec 14, 2013	I added a video to a @YouTube playlist http://t.co/O3qD9wfi8K SAP BUSI...	
6	256	ZaranTech	Dec 14, 2013	I added a video to a @YouTube playlist http://t.co/XxrfuCUqAS SAP BUSI...	
7	257	Helmich op t...	Dec 14, 2013	Reserveer alvast 15 januari 2014 training HANA Essentials #SAP #HANA	SAP,HANA,
8	258	Social News	Dec 13, 2013	sap hana online training and placenet 2 http://t.co/JlaA41IdnV	
9	259	Nurianah	Dec 13, 2013	Baca notif fb... ada training dadakaann dari eyang kita.... ayo wara wiri ca...	
10	260	Nora Apnila J...	Dec 13, 2013	lanjutt di rumaah dulu ikutan trainingnyaaa..mau buru buru pulang see u...	
11	261	madhu	Dec 13, 2013	SAP HANA TRAINING | SAP HANA PLACEMENT | SAP HANA INSTITUTE I...	
12	262	Hana O'Neill	Dec 13, 2013	@sarahsilvanator no I have life guard training Saturday and my final test t...	
13	263	arjun	Dec 13, 2013	sap grc online training |sap hana sap security online training@YEKTEK - A...	

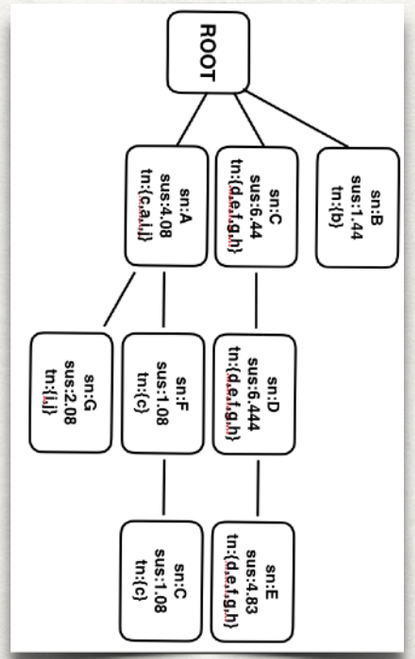
fraud

METHODS: SUSPICIOUSNESS FOREST

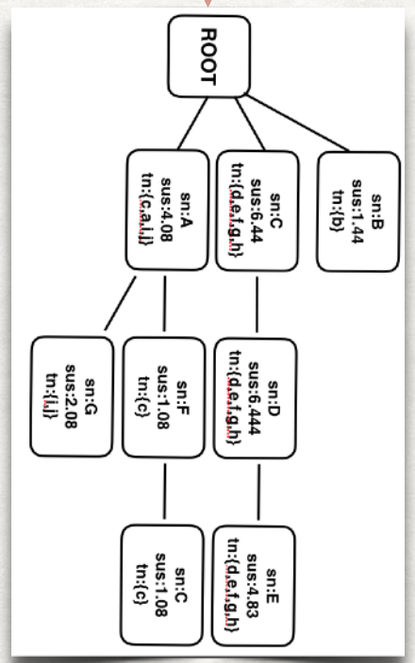
Source Node

Target Node

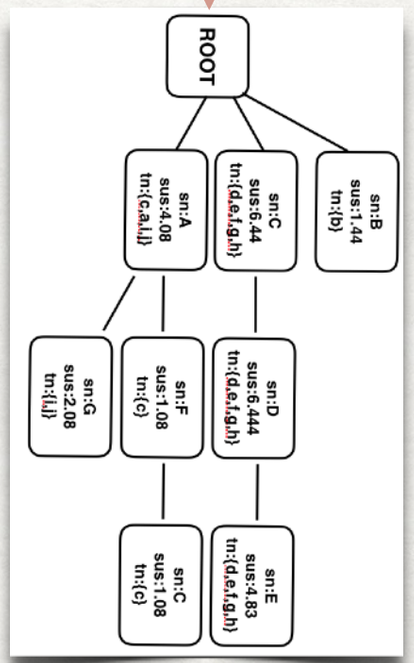
	Dim 1	Dim 2	Dim 3	Dim 4	...
Entry 1	user 1	product 1	date 1	IP 1	...
Entry 2	user 2	product 1	date 1	IP 2	
Entry 3	...				
⋮					



S-tree 1:
Weight 1



S-tree 2
Weight 2



S-tree 3
Weight 3

S-forest

EXPERIMENTS AND EVALUATIONS

Baselines:

- Fraudar[2], the best paper, KDD 2016;
- CatchSync[1], the best paper finalist, KDD 2014;
- CrossSpot[15], TKDE 2016
- D-Cube[16], WSDM 2017
- M-Zoom[6], PKDD 2017
- M-Biz[17], TKDD 2018

Datasets: thirteen datasets.

S-tree			S-forest		
datasets	edges	dims	datasets	entries	dims
AmazonOffice[23]	53K	2	YelpChi[24]	67K	4
AmazonBaby[23]	160K	2	YelpNYC[24]	359K	4
AmazonTools[23]	134K	2	YelpZip[24]	1.14M	4
AmazonFood[23]	1.3 M	2	DARPA[25]	4.55M	4
AmazonVideo[23]	583K	2	AirFore[26]	4.89 M	8
AmazonPet[23]	157K	2	Registration	26k	6
Twitter[27]	1.47B	2			

✦ EXPERIMENTS AND EVALUATIONS

Accuracy.

Fig. 3: Performance on AmazonOffice ($\rho = 1.0$)

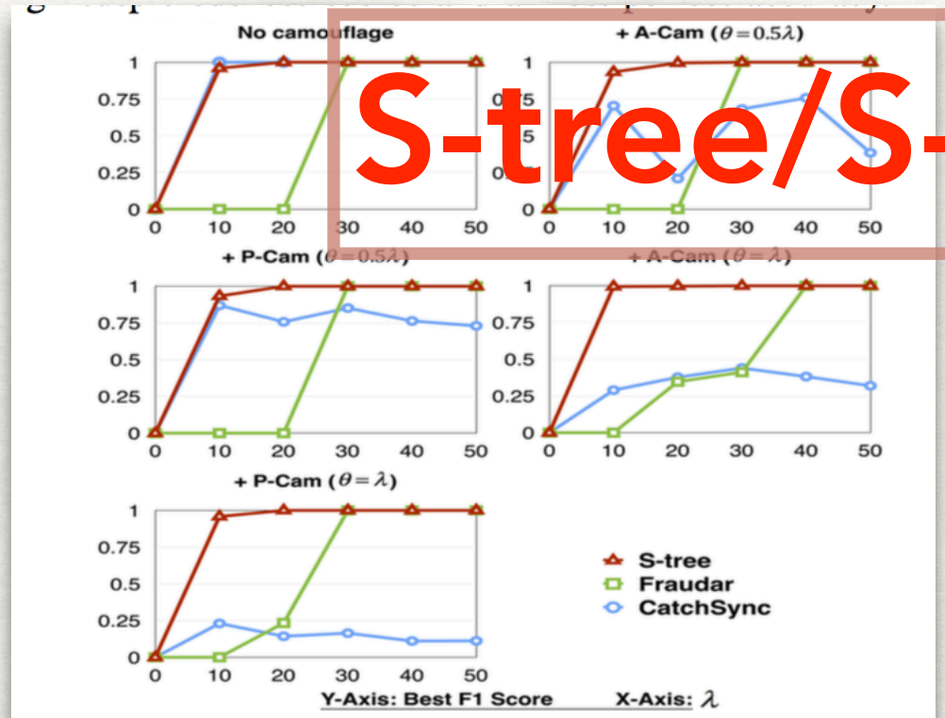


Fig. 4: Performance on AmazonOffice ($\rho = 0.6$)

S-tree/S-forest all wins!

TABLE IV: Performance(AUC) on the [Amazon] datasets

	AmazonOffice	AmazonBaby	AmazonTools
Fraudar	0.9415	0.8674	0.9264
CatchSync	0.8412	0.8190	0.7907
S-tree	**0.9987**	**0.9595**	**0.9389**
	AmazonFood	AmazonVideo	AmazonBook
Fraudar	0.6915	0.7361	0.8923
CatchSync	0.7412	0.6990	0.7634
S-tree	**0.8558**	**0.8756**	**0.9435**

EXPERIMENTS AND EVALUATIONS

Accuracy.

TABLE VI: Performance(AUC) on the [YELP] datasets

	YelpChi	YelpNYC	YelpZip
CrossSpot	0.9905	0.9328	0.9422
M-Zoom	0.9850	0.9255	0.9391
M-Biz	9850	0.9229	0.9302
Dcube	0.9857	0.9232	0.9391
S-forest	**0.9945**	**0.9406**	**0.9456**

TABLE V: Performance(AUC) on the [DARPA] and [Air-Force] dataset

Dataset	DARPA	AirForce
CrossSpot	0.923	0.924
M-Zoom	0.923	0.975
M-Biz	0.923	0.975
D-Cube	0.930	0.987
S-forest	**0.942**	**0.991**

S-tree/S-forest all wins!

TABLE VII: Performance (AUC) on [Registration] dataset. 'C' represents 'critical feature' and 'N' represents 'noise feature'.

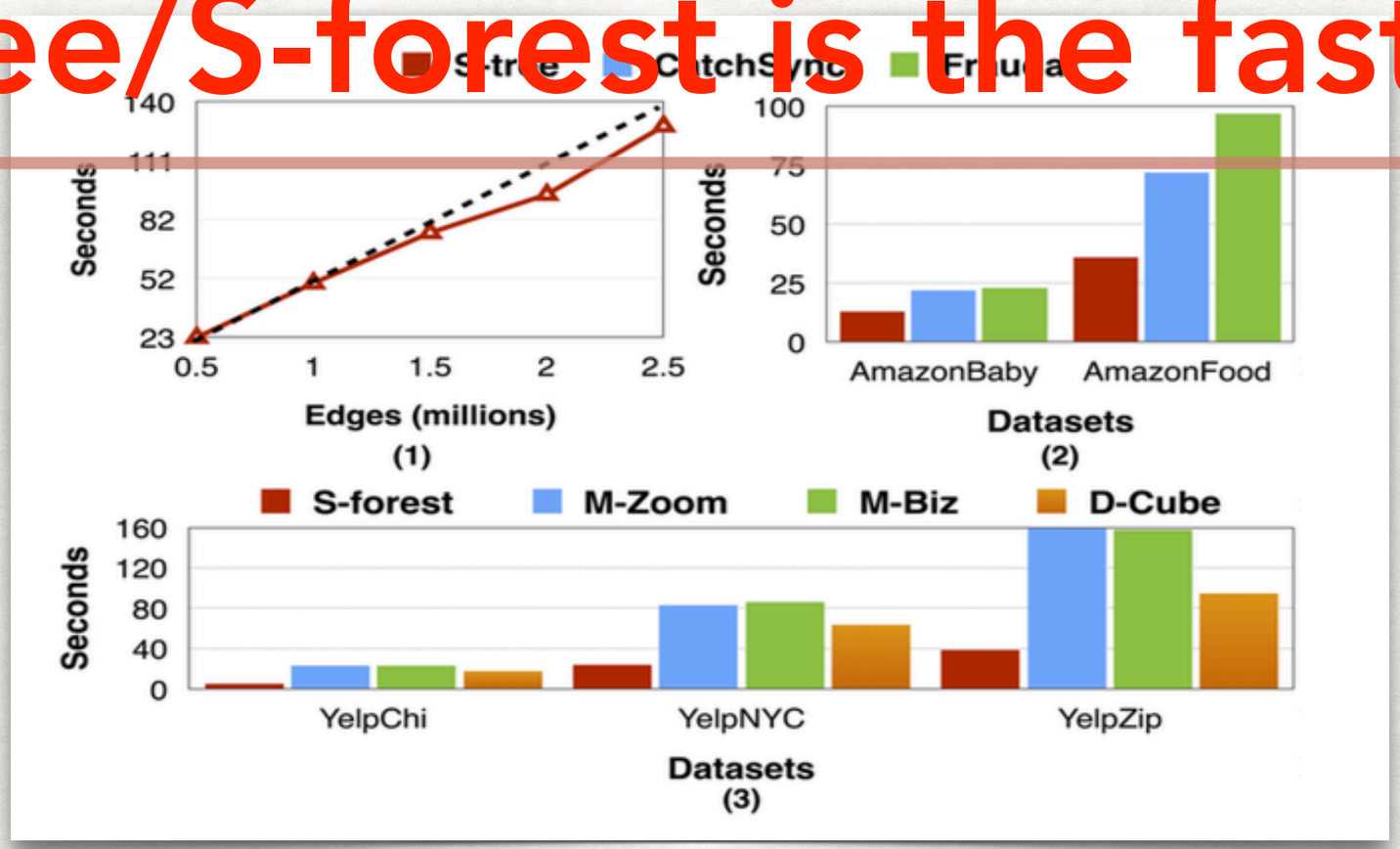
	1 C	2 C	2C +1N	2C + 2N	2C+3N
CrossSpot	0.7471	0.7312	0.7307	0.7539	0.8099
M-zoom	0.7544	0.8880	0.8409	0.8754	0.6802
M-Biz	0.7577	0.8842	0.8409	0.8674	0.6812
D-Cube	0.7601	0.9201	0.8532	0.8723	0.7102
S-forest	**0.7678**	**0.9961**	**0.9964**	**0.9876**	**0.9976**

✦ EXPERIMENTS AND EVALUATIONS

Speed: **Linear Time Complexity:**

building S-tree takes O(2|E|) time and all algorithms only need one scan of S-tree.

S-tree/S-forest is the fastest!



SUMMARY

- [A New Graph Problem]. We introduce a new restricted biclique problem, Maximal Half Isolated Biclique (MHI Biclique), and show that the problem finds immediate applications in fraud detection.
- [A Linear Algorithm]. We propose a novel data structure, S-tree, and a mining algorithm, that solves the MHI Biclique problem in linear time. The algorithm is effective for detecting unbalanced dense sub-bipartite graphs. We provide theoretical proofs regarding the algorithm's effectiveness and robustness against adversarial perturbations.
- [Practical Algorithms for Fraud Detection]. Based on the S-tree-based MHI Biclique problem solver, we introduce a new algorithm that detects near bicliques and can be used to catch a wide range of fraud groups.
- [Automatic Feature Prioritization]. We further extend S-tree to S-forest to handle multimodal data. Furthermore, S-forest can automatically combine and prioritize multiple features, reducing the need for feature engineering.
- [Effectiveness on Real-world Data]. We conducted extensive experiments on thirteen real-world datasets. Our solution outperforms strong rivals across all configurations, becoming the new state of the art in fraud detection.

FUTURE RESEARCH INTERESTS

- Mining patterns/insights from biological data collection by designing algorithms.
- Integrating various types of data using machine learning.
- Building classification/inference models with application to biomedical fields.

THANK YOU FOR YOUR TIME AND CONSIDERATION!

ACKNOWLEDGEMENTS

Thank present
advisors: Wei Xu, Yitao Duan, Ling Huang, and Xue Liu,
teammates: Xin Liu and Jiao Sun,
for valuable suggestions, helpful discussions, and modifications to these papers.

REFERENCE

[1] Jiang, Meng, et al. "Catchsync: catching synchronized behavior in large directed graphs." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.
[2] Hooi, Bryan, et al. "Fraudar: Bounding graph fraud in the face of camouflage." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016.
[3] Zhang, Yun, et al. "On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types." BMC bioinformatics 15.1 (2014): 110.
[4] G. W. Flake, S. Lawrence, and C. L. Giles, “Efficient identification of web communities,” in ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 150–160.
[5] D. Eppstein, “Arboricity and bipartite subgraph listing algorithms,” Inf. Process. Lett., vol. 51, no. 4, pp. 207–211, 1994.
[6] K. Shin, B. Hooi, and C. Faloutsos, “M-zoom: Fast dense-block de- tection in tensors with quality guarantees,” in ECML PKDD, 2016, pp. 264–280.
[7] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X. N. Zhang, “Dense subgraphs with restrictions and applications to gene annotation graphs,” in International Conference on Research in Computational Molecular Biology, 2010, pp. 456–472.
[8] M. J. Sanderson, A. C. Driskell, R. H. Ree, O. Eulenstein, and S. Lang- ley, “Obtaining maximal concatenated phylogenetic data sets from large sequence databases,” Molecular biology and evolution, vol. 20, no. 7, pp. 1036–1042, 2003.
[9] R. A. Mushlin, A. Kershenbaum, S. T. Gallagher, and T. R. Rebbeck, “A graph-theoretical approach for pattern discovery in epidemiological research,” IBM systems journal, vol. 46, no. 1, pp. 135–149.
[10] K. Makino and T. Uno, “New algorithms for enumerating all maximal cliques,” in Scandinavian Workshop on Algorithm Theory. Springer, 2004, pp. 260–272.
[11] J. Li, G. Liu, H. Li, and L. Wong, “Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspon- dence and mining algorithms,” IEEE Trans. Knowl. Data Eng., vol. 19, no. 12, pp. 1625–1637, 2007.
[12] J. Wang, J. Han, and J. Pei, “Closet+: Searching for the best strategies for mining frequent closed itemsets,” in Proceedings of the ninth ACM SIGKDD. ACM, 2003, pp. 236–245.
[13] G. Grahne and J. Zhu, “Efficiently using prefix-trees in mining frequent itemsets.” in FIMI, vol. 90, 2003.
[14] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in ACM SIGMOD, 2000, pp. 1–12.

[15] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, “Spotting suspicious behaviors in multimodal data: A general metric and algorithms,” IEEE TKDE, vol. 28, no. 8, pp. 2187–2200, 2016.
[16] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, “D-cube: Dense-block detection in terabyte-scale tensors,” in ACM WSDM, New York, NY, USA, 2017, pp. 681–689.
[18] K. Shin, B. Hooi, and C. Faloutsos, “Fast, accurate, and flexible algorithms for dense subtensor mining,” Acm Transactions on Knowledge Discovery from Data, vol. 12, no. 3, pp. 1–30, 2018.
[19] Jaccard Paul. [n. d.]. THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. New Phytologist 11, 2 ([n. d.]), 37–50.
[20] G. Cordasco and L. Gargano. 2010. Community detection via semi-synchronous label propagation algorithms. In 2010 IEEE International Workshop on: BASNA. 1–8.
[21] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. Physical review E 76, 3 (2007), 036106.
[22] Koji Maruhashi, Fan Guo, and Christos Faloutsos. 2011. MultiAspectForensics: Pattern Mining on Large-Scale Heterogeneous Networks with Tensor Analysis. In ASONAM. 203–210.
[23] C.EShannon.1948.A mathematical theory of communication. BellLabs Technical Journal 27, 4 (1948), 379–423.
[24] Moses Charikar. 2000. Greedy Approximation Algorithms for Finding Dense Com- ponents in a Graph. Springer Berlin Heidelberg. 84–95 pages.