



北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS



中國科學院
CHINESE ACADEMY OF SCIENCES



UNIVERSITY OF
SURREY

ATM | TEXAS A&M
UNIVERSITY

NUS
National University
of Singapore



中信銀行
CHINA CITIC BANK

FlowScope: SPOTTING MONEY LAUNDERING BASED ON GRAPHS

Xiangfeng Li¹, Shenghua Liu², Zifeng Li³, Xiaotian Han⁴, Chuan Shi¹, Bryan Hooi⁵, He Huang⁶, Xueqi Cheng²

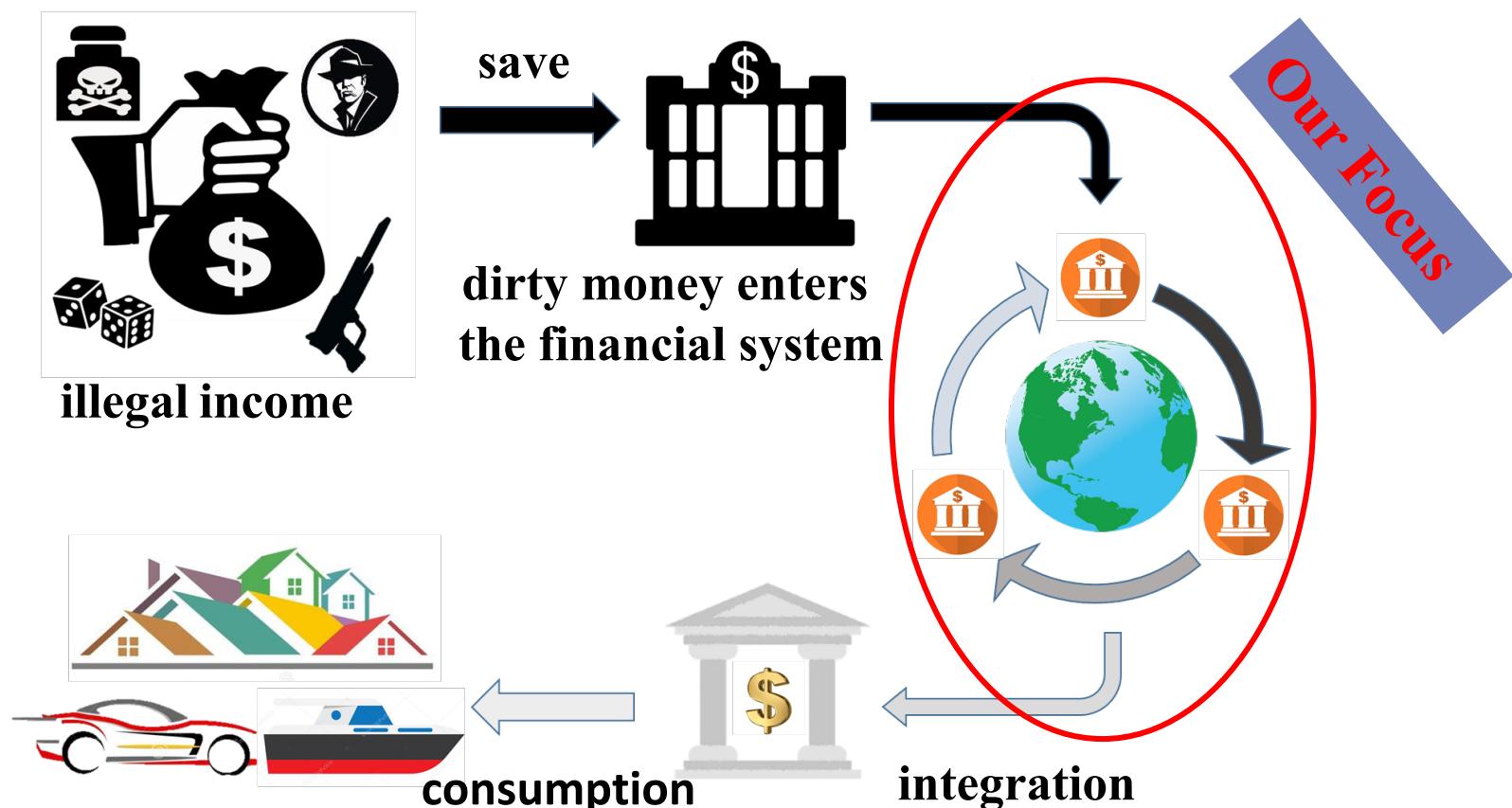
¹Beijing University of Post and Telecommunication ²Institute of Computing Technology, Chinese Academy of Sciences

³University of Surrey ⁴Texas A&M University

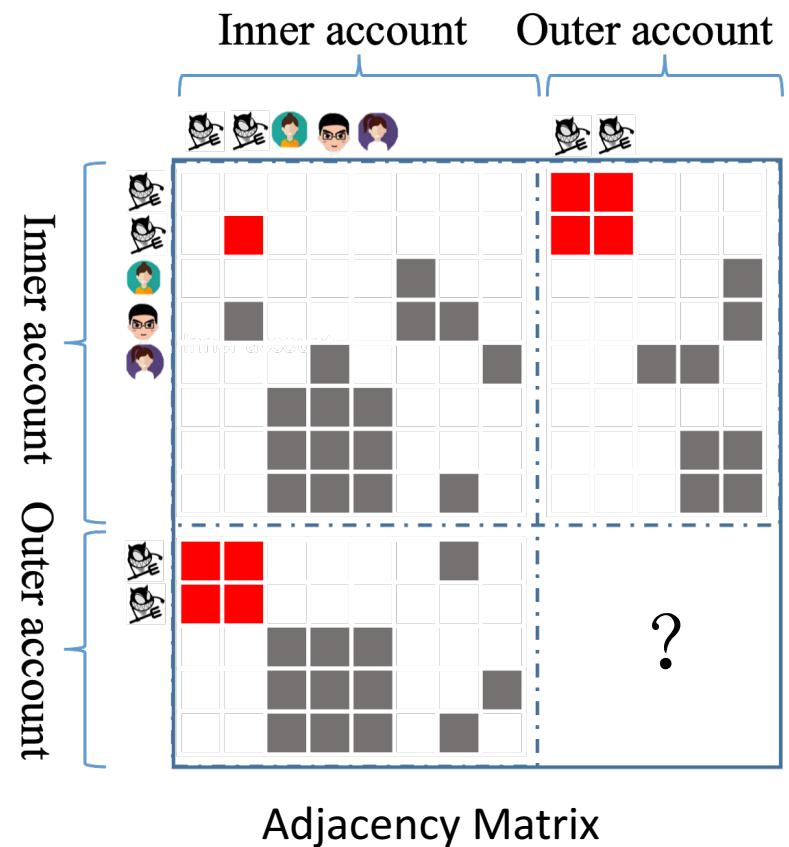
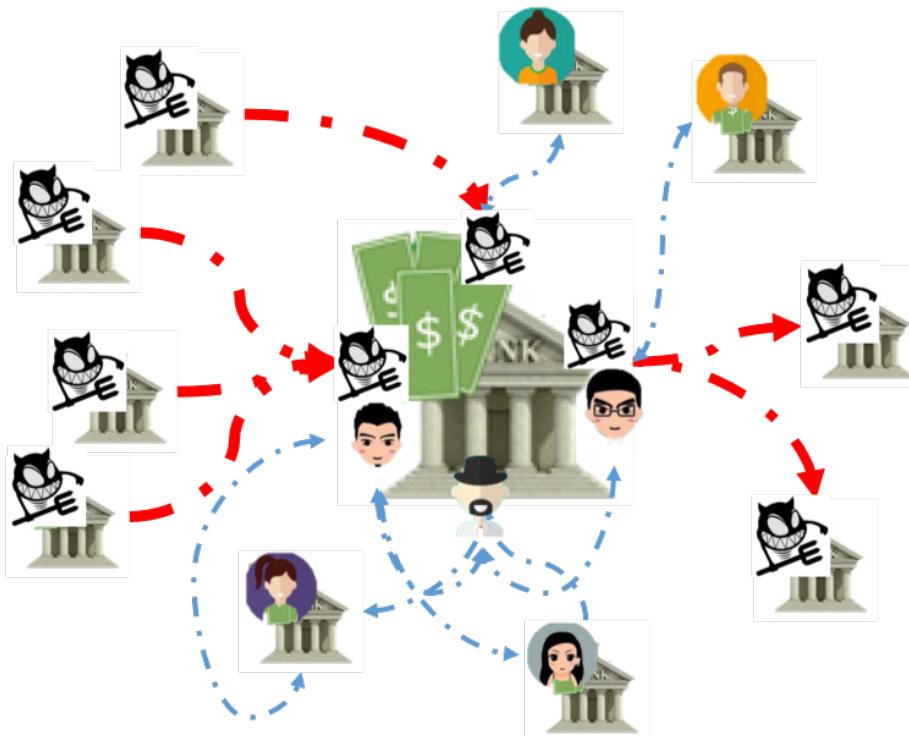
⁵School of Computer Science, National University of Singapore ⁶China Citic Bank

Motivation

- Typical method of money laundering (ML)

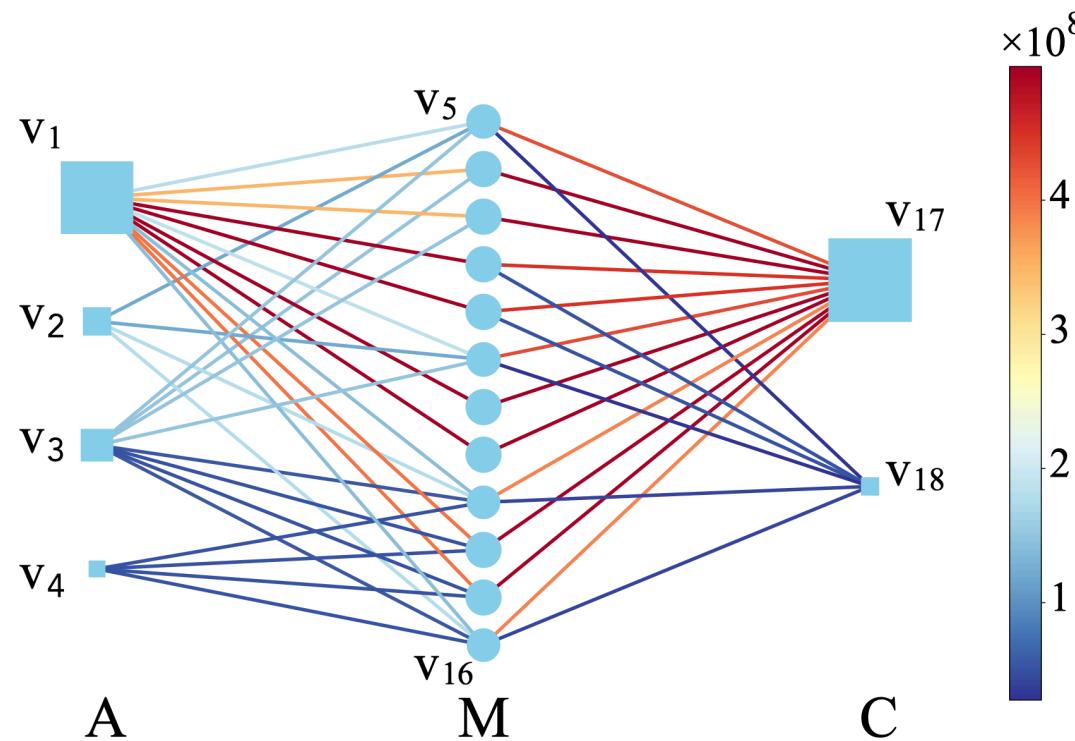


ML Forms a Multipartite Dense Subgraph

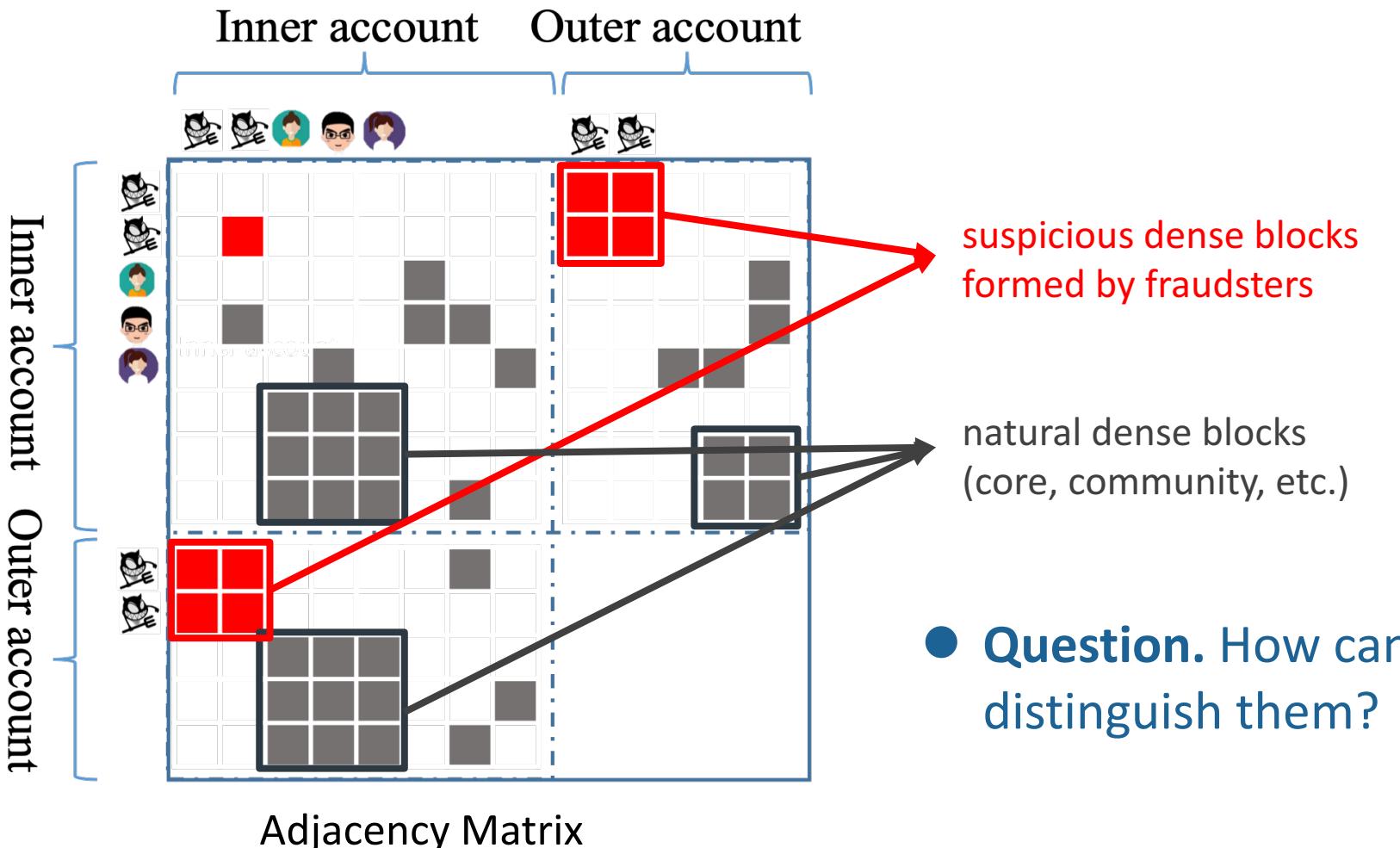


ML Forms a Multipartite Dense Subgraph

- Example: tripartite subgraph formed by money laundering accounts from a real bank.

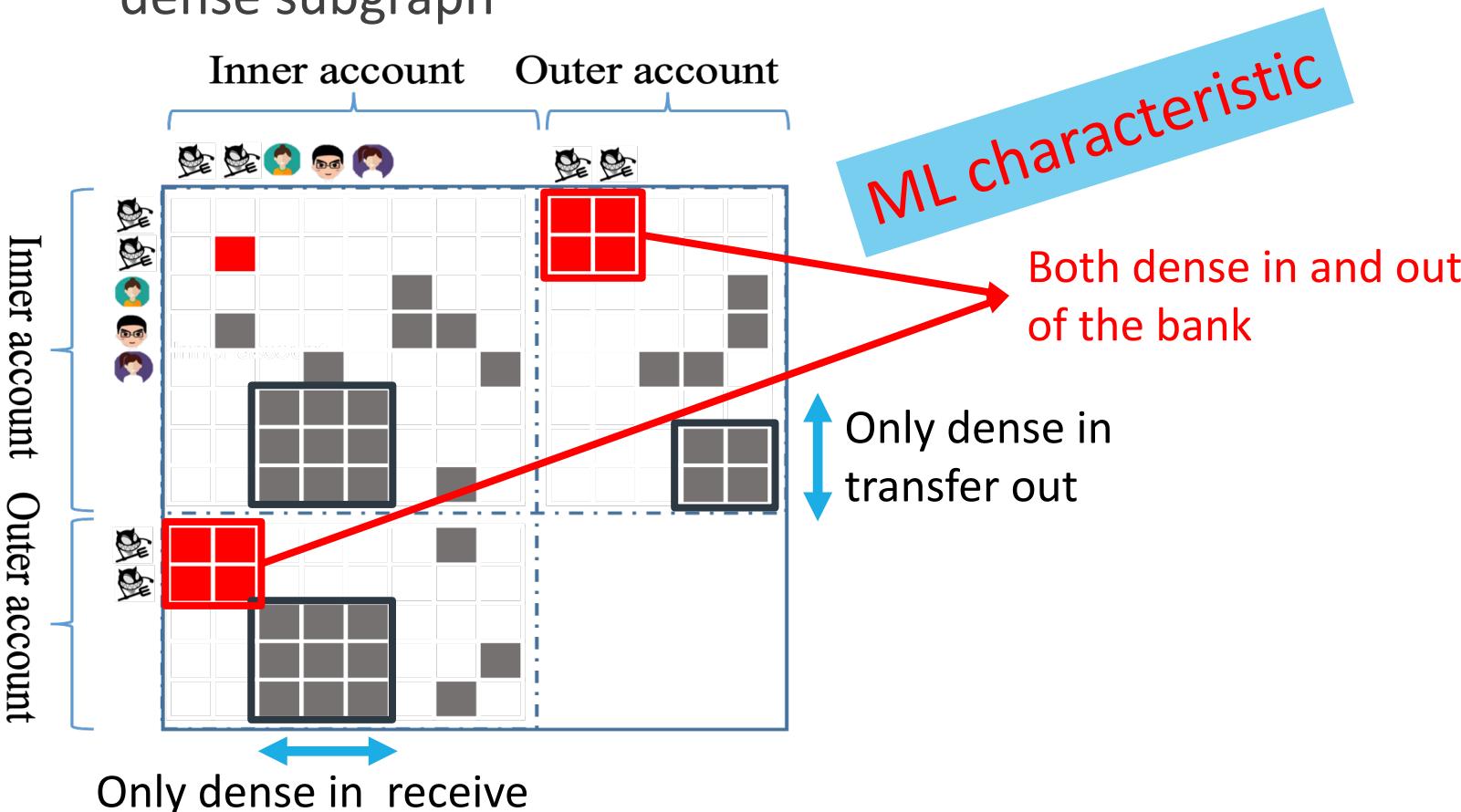


Problem: Natural Dense Subgraph



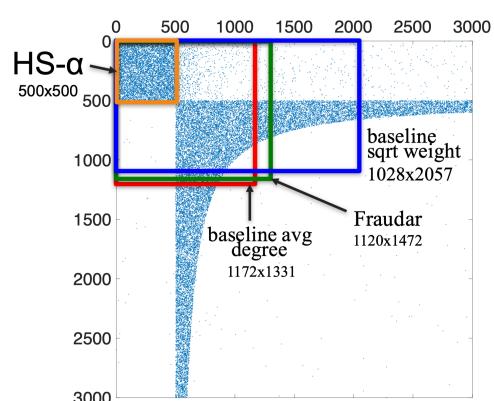
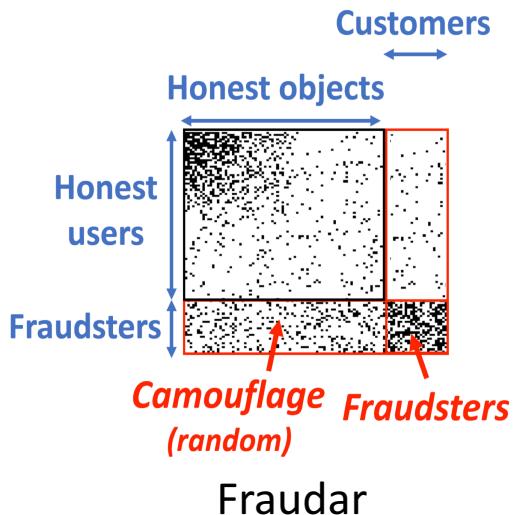
Solution: Multipartite Dense Subgraph

- Natural dense transfer not always form a multipartite dense subgraph

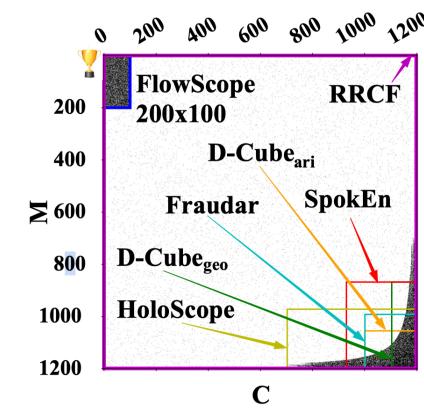
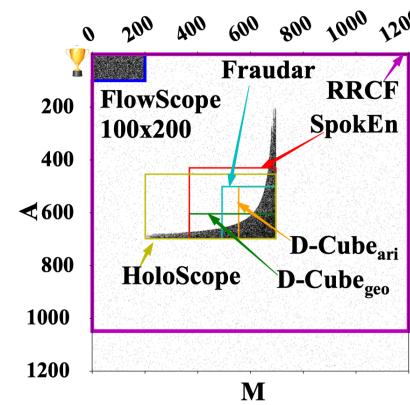


Solution: Multipartite Dense Subgraph (cont.)

- Our FlowScope catches exactly multipartite dense subgraph



HoloScope- α



Our FlowScope

Problem formulation

- **Given**

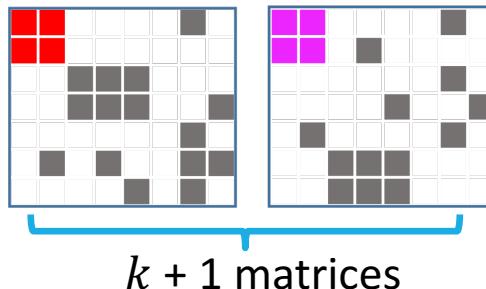
- $G = (V, E)$: a graph of money transfers
- V : accounts as nodes
- E : money amount as edges weight
- k : number of middle layers

- **Find**

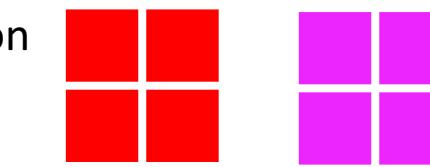
- a dense flow of money transfers (i.e. a subgraph of G),

- **Such that**

- 1) the flow involves high-volume money transfers into the bank, and out of the bank to the destinations;
- 2) it maximizes density as defined in our ML metric.

 $G =$ 

dense flow detection



suspicious flow in the graph

Requirements

- Our goal is to design an algorithm which is

- Fast**: runs in near-linear time
- Accurate**: provides an accuracy guarantee
- Effective**: produces meaningful results in practice

FlowScope, our proposed method, satisfies all the requirements

Model

- Graph

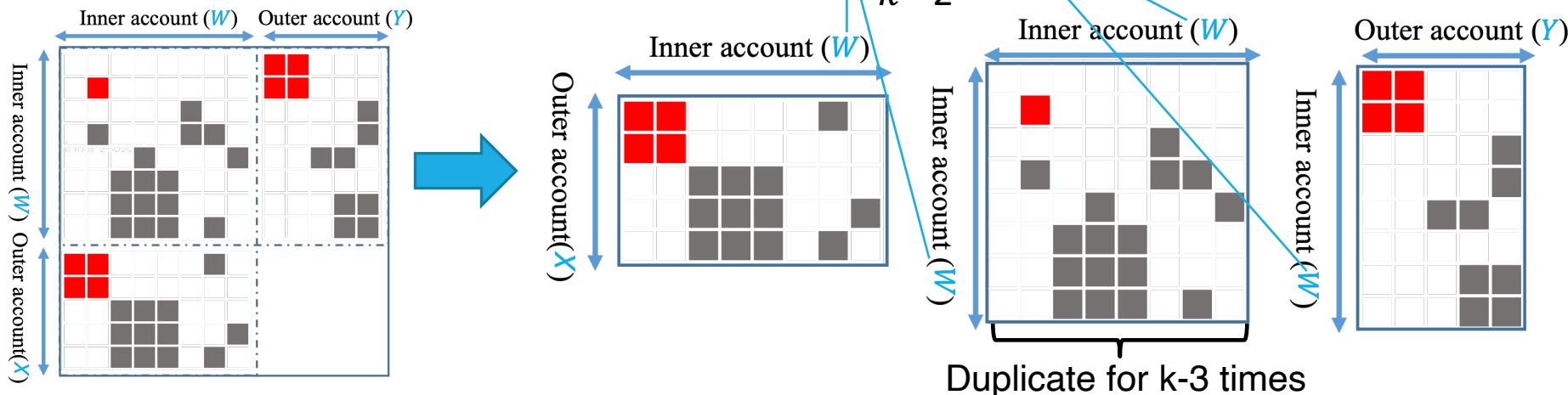
$$G = (V, E), \quad V = X \cup W \cup Y$$

- W is the inner accounts of the bank, and X and Y are sets of outer accounts

- Generate multipartite graph

$$G^k = (V^k, E^k),$$

$$V^k = X \cup W \cup \dots \cup W \cup Y$$



Model (cont.)

- Out/in degree of each middle-layer node

$$d_i^+(S) = \sum_{v_j \in M_{l+1} \wedge (i,j) \in E} e_{ij}$$

$$d_i^-(S) = \sum_{v_k \in M_{l-1} \wedge (k,i) \in E} e_{ki}$$

- Definition of min and max flow

$$f_i(S) = \min\{ d_i^+(S), d_i^-(S) \}, \forall v_i \in M_l$$

$$q_i(S) = \max\{ d_i^+(S), d_i^-(S) \}, \forall v_i \in M_l$$

- Suspicious metric

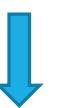
balance parameter

~ flow



$$g^k(S) = \frac{1}{|S|} \sum_{l=1}^{k-2} \sum_{v_i \in M_l} f_i(S) - \lambda(q_i(S) - f_i(S))$$

retention/deficit



$$= \frac{1}{|S|} \sum_{l=1}^{k-2} \sum_{v_i \in M_l} (\lambda + 1)f_i(S) - \lambda q_i(S), k \geq 3$$

Algorithm

- **Input:** Graph $G = (V, E)$
- **Output:** Node set of dense multipartite flow: S
- **Key idea:** priority tree and greedy deletion

Algorithm 1: FlowScope

Input: Graph $G = (\mathcal{V}, \mathcal{E})$

Output: Node set of dense flow: $\hat{\mathcal{S}}$

```

1  $\mathcal{A} \leftarrow \mathcal{X}, \mathcal{M}_1 \leftarrow \mathcal{W}, \dots, \mathcal{M}_{k-2} \leftarrow \mathcal{W}, \mathcal{C} \leftarrow \mathcal{Y}$ 
   // generate  $k$ -partite node
   subsets from  $\mathcal{G}$ 
2  $\mathcal{S} \leftarrow \mathcal{A} \cup \mathcal{M}_1 \cup \dots \cup \mathcal{M}_{k-2} \cup \mathcal{C}$ 
3  $w_i \leftarrow$  calculate node weight as Eq. (5)
4  $\mathcal{T} \leftarrow$  build priority tree for  $\mathcal{S}$  with  $w_i(\mathcal{S})$ 
5 while  $\mathcal{A}, \mathcal{M}_1, \dots, \mathcal{M}_{k-2}$  and  $\mathcal{C}$  is not empty do
6    $v \leftarrow$  find the minimum weighted node in  $\mathcal{T}$ 
7    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{v\}$ 
8   update priorities in  $\mathcal{T}$  for all neighbors of  $v$ 
9    $g(\mathcal{S}) \leftarrow$  calculate as Eq. (3)
10 end
11 return  $\hat{\mathcal{S}}$  that maximizes  $g(\mathcal{S})$  seen during the loop.

```

Step 1. initialize

Step 2. greedy deletion

Step 3. get the result

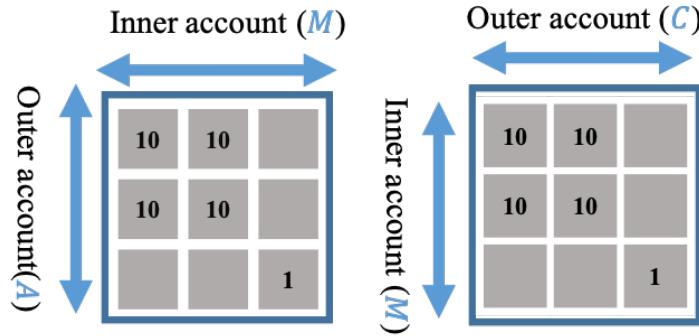
Algorithm (cont.)

- Step 1. initialize

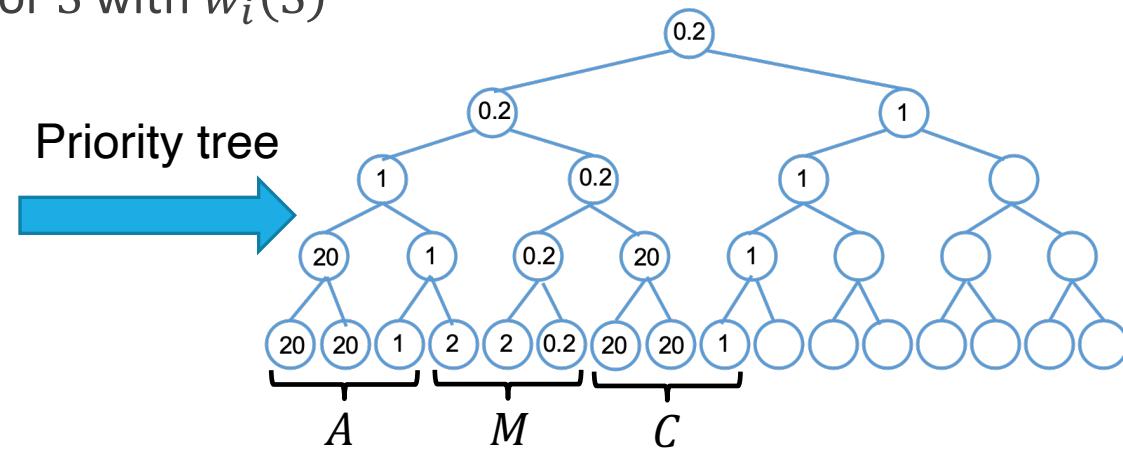
- 1. generate the k -partite graph, $A \leftarrow X, M_1 \leftarrow W, \dots, M_{k-2} \leftarrow W, C \leftarrow Y$
- 2. initialize subset $S \leftarrow A \cup M_1 \cup \dots \cup M_{k-2} \cup C$
- 3. calculate the priority of node

$$w_i(S) = \begin{cases} f_i(S) - \frac{\lambda}{\lambda + 1} q_i(S), & \text{if } v_i \in M_l, l \in \{1, 2, \dots, k-2\} \\ q_i(S) = d_i(S), & \text{if } v_i \in A \cup C \end{cases}$$

- 4. build priority tree for S with $w_i(S)$

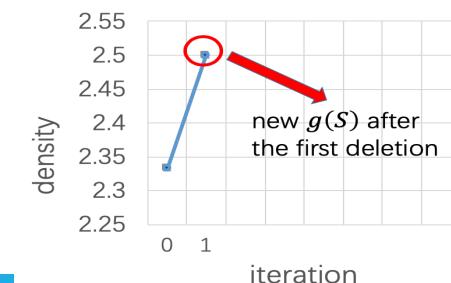
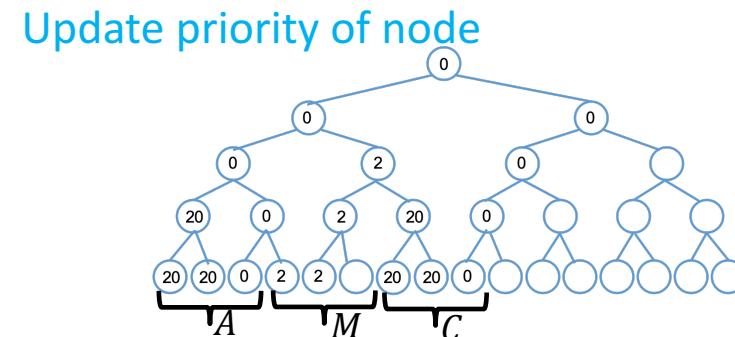
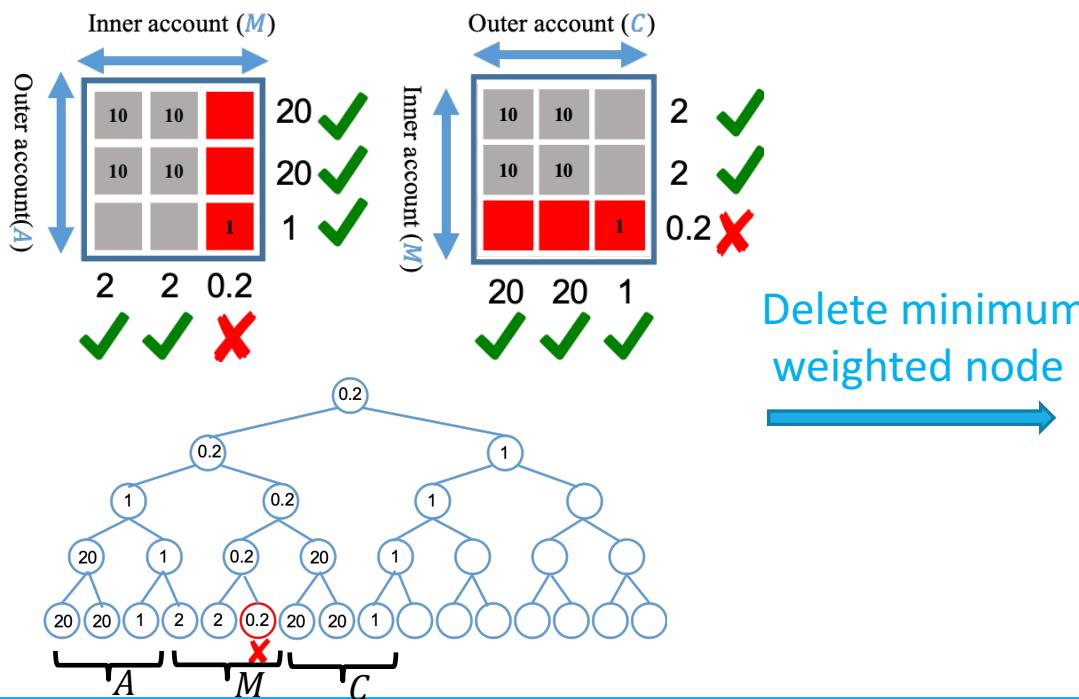


Priority tree



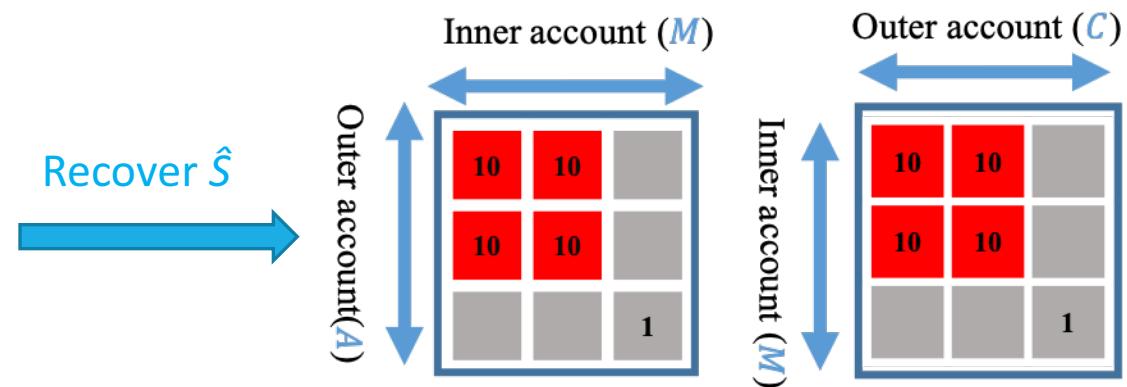
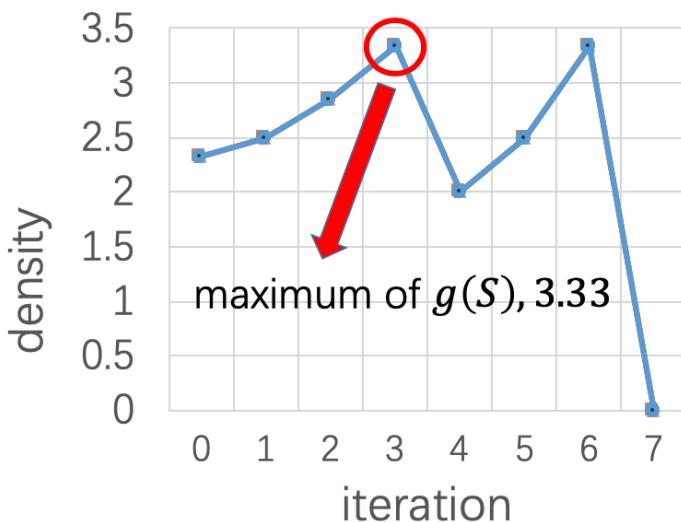
Algorithm (cont.)

- Step 2. greedy deletion
 - 1. get the node v with minimum weight
 - 2. delete the selected node, update the value of $g(S)$ and update node's weight that correlated with v
 - 3. repeat 1 and 2 until one of $A, M_1, \dots, M_{k-2}, C$ is empty



Algorithm (cont.)

- Step 3. get the result
 - 1. find the maximum value of $g(S)$
 - 2. recover correspond node set \hat{S} corresponding to maximum $g(S)$



Result \hat{S} :

$$\hat{A} : \{0,1\}, \hat{M} : \{0,1\}, \hat{C} : \{0,1\}$$

Algorithm (cont.)

- Theorem [Approximation Guarantee]
 - in 3-step ML (tripartite)

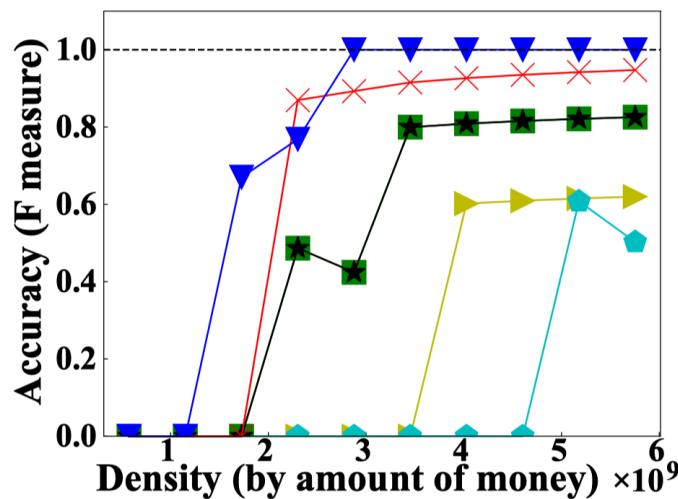
$$g(\hat{S}) \geq \frac{|M'|}{|S'|} (g(S^*) - \lambda\varepsilon)$$

FlowScope → $g(\hat{S})$ → $\frac{|M'|}{|S'|}$ → $(g(S^*) - \lambda\varepsilon)$ →
 ← $\text{node set just before the}$ ← optimal ←
 ← $\text{first optimal node removed}$ ← amount of ←
 ← camouflage ← transfers ←
 ← $\text{middle counts in } S'$

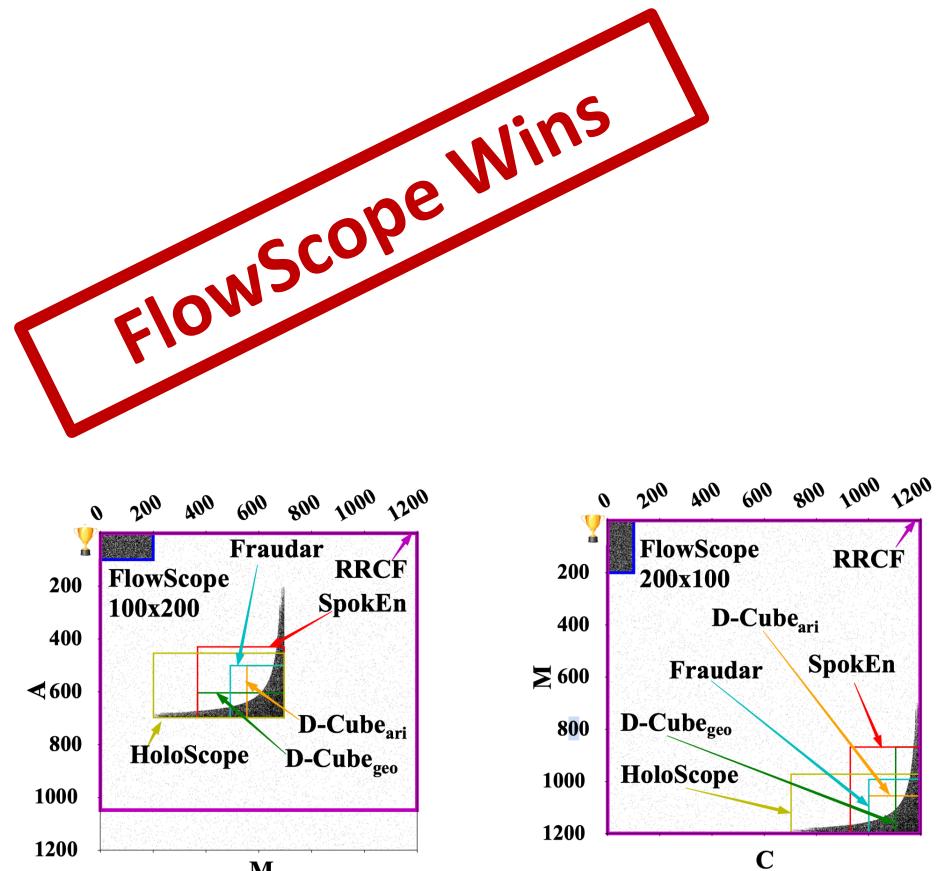
Properties of FlowScope:

- | | |
|-------------------------------------|--|
| <input type="checkbox"/> | Fast : runs in near-linear time |
| <input checked="" type="checkbox"/> | Accurate : provides an accuracy guarantee |
| <input type="checkbox"/> | Effective : produces meaningful results in practice |

Real-world performance



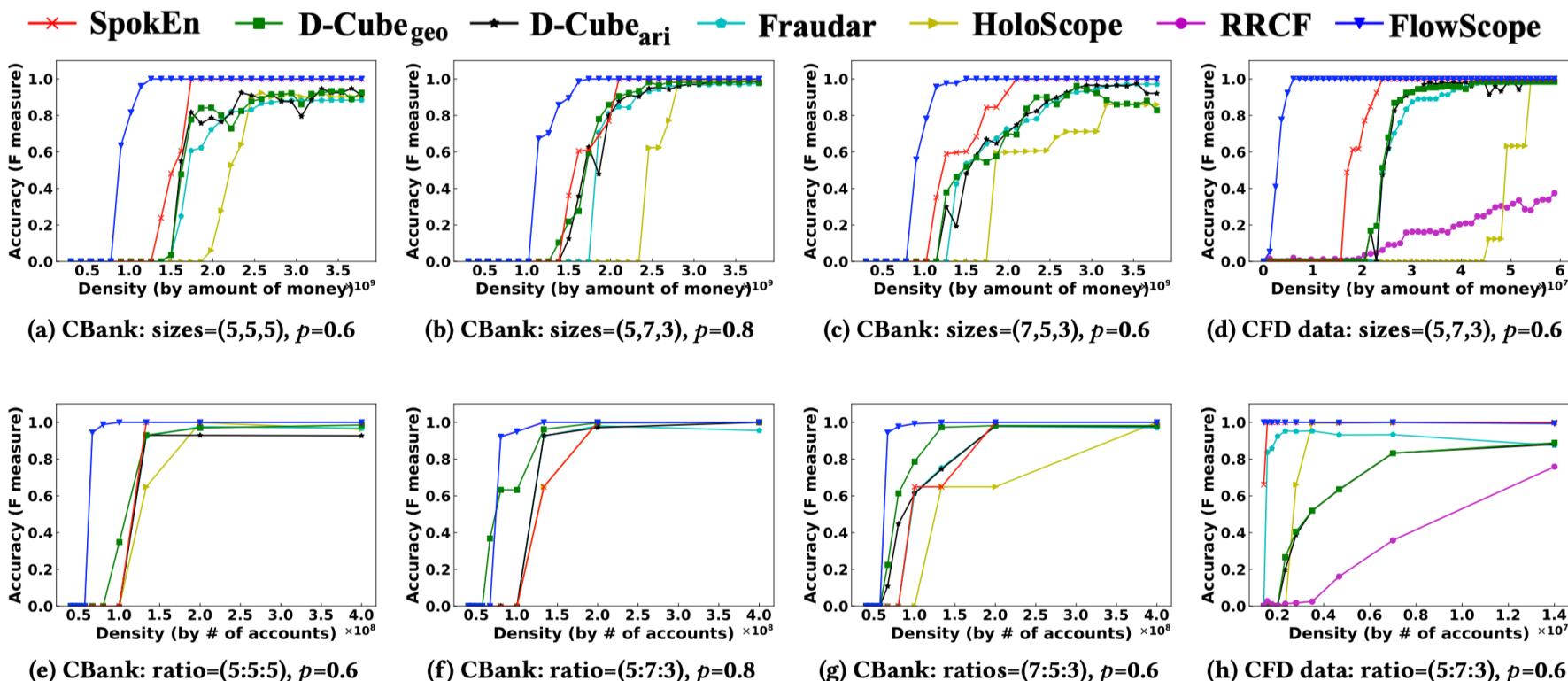
With ground-truth labelled



Performance on synthetic data

Effectiveness: one middle layer

Good performance under variety of topologies



Effectiveness: one middle layer (cont.)

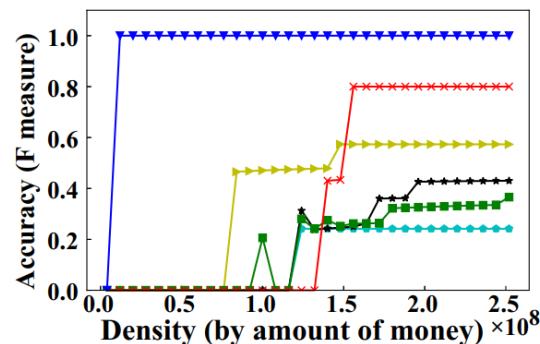
Summary in table

Dataset	metrics*	A:M:C	D-Cube _{ari}	D-Cube _{geo}	Fraudar	HoloScope	SpokEn	RRCF	FlowScope
CBank	FAUC	5:9:1	0.417 / 0.600	0.591 / 0.810	0.347 / 0.634	0.276 / 0.466	0.610 / 0.753	- / -	0.633 / 0.800
		5:5:5	0.502 / 0.658	0.501 / 0.709	0.467 / 0.683	0.379 / 0.655	0.598 / 0.708	- / -	0.757 / 0.843
		7:5:3	0.533 / 0.727	0.522 / 0.779	0.529 / 0.704	0.547 / 0.547	0.633 / 0.708	- / -	0.761 / 0.843
	F1 ≥ 0.9 (million \$ / node size)	5:9:1	190 / 30	- / 45	- / 30	-	154 / 30	- / -	132 / 75
		5:5:5	150 / 45	- / 45	- / 45	-	116 / 45	- / -	84.0 / 90
		7:5:3	175 / 30	166 / 54	- / 15	- / 15	122 / 30	- / -	76.0 / 90
CFD	FAUC	5:9:1	0.498 / 0.577	0.528 / 0.770	0.528 / 0.770	0.125 / 0.773	0.716 / 0.894	0.253 / 0.538	0.939 / 0.877
		5:5:5	0.565 / 0.633	0.549 / 0.867	0.549 / 0.867	0.143 / 0.810	0.716 / 0.897	0.236 / 0.364	0.962 / 0.900
		7:5:3	0.580 / 0.733	0.593 / 0.826	0.593 / 0.826	0.0356 / 0.818	0.728 / 0.898	0.213 / 0.434	0.970 / 0.900
	F1 ≥ 0.9 (million \$ / node size)	5:9:1	2.05 / 15	- / 60	3.52 / 60	1.71 / 120	- / 15	-	0.400 / 150
		5:5:5	1.15 / 30	2.05 / 30	- / 150	- / 75	1.23 / 150	- / 15	0.240 / 150
		7:5:3	- / 30	- / 30	- / 120	- / 60	1.46 / 135	- / 15	0.240 / 150

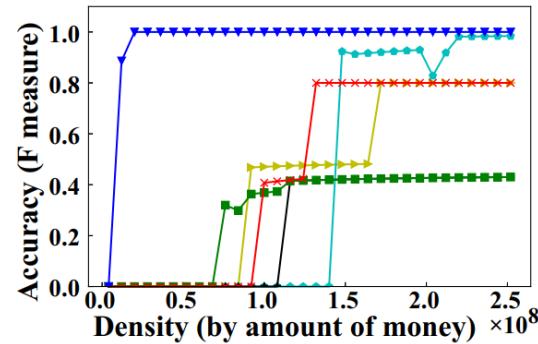
FlowScope Wins

Robustness against longer transfer chains

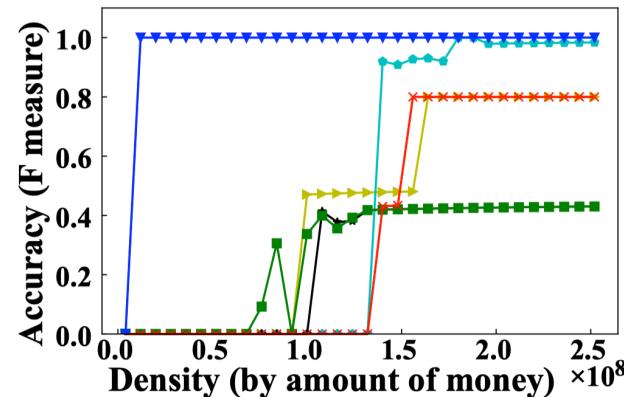
—*— SpokEn —■— D-Cube_{geo} —*— D-Cube_{ari} —●— Fraudar —>— HoloScope —●— RRCAF —→— FlowScope



(a) CBank: Transfer chains of (5,5,5,5)



(b) CBank: Transfer chains of (2,8,8,2)



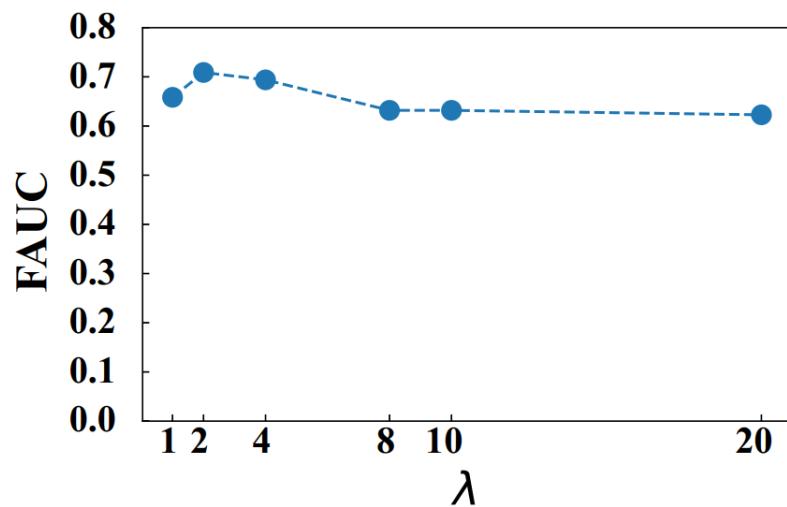
CBank: Transfer chains of (3,7,7,3)

Effectiveness: varies topologies and labelled data

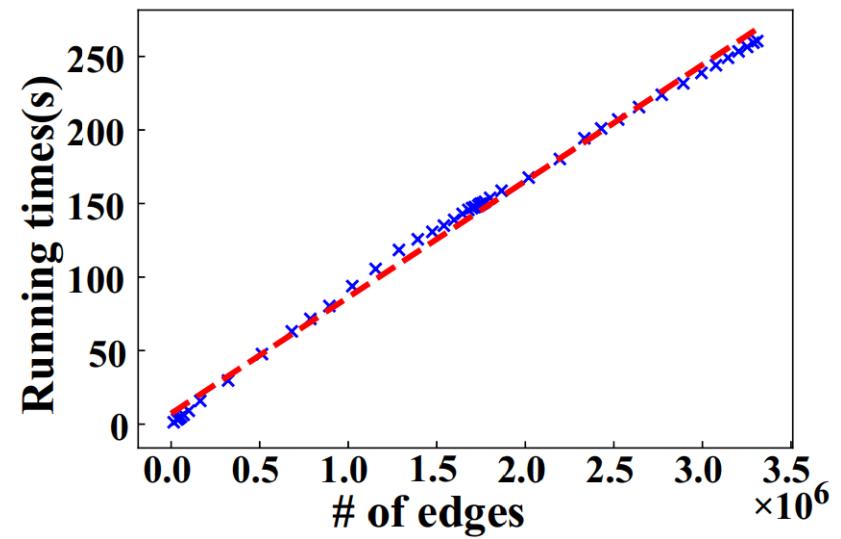
Properties of FlowScope:

- Fast**: runs in near-linear time
- Accurate**: provides an accuracy guarantee
- Effective**: produces meaningful results in practice

Sensitivity and Scalability



FlowScope is robustness to parameter



FlowScope runs in near-linear time with the # of edges

Sensitivity and Scalability(cont.)

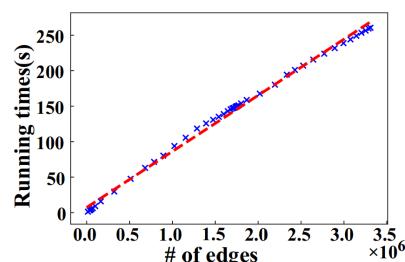
Properties of FlowScope:

- Fast**: runs in near-linear time
- Accurate**: provides an accuracy guarantee
- Effective**: produces meaningful results in practice

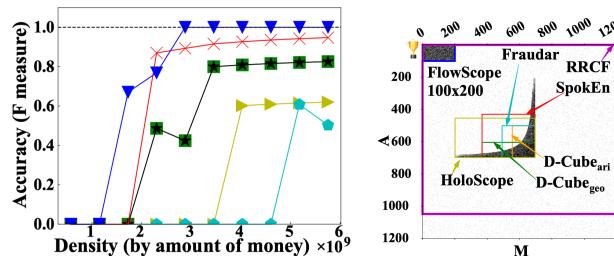
Conclusion

- FlowScope detects money laundering fast and effectively

Fast



Effective



Accurate

$$g(\hat{S}) = \frac{|M'|}{|S'|} (g(S^*) - \lambda \varepsilon)$$

Reproducible



<https://github.com/aplaceof/FlowScope>

More information about FlowScope

- AAAI 2020
- Supplement
 - <https://github.com/aplaceof/FlowScope/blob/master/FlowScope-supplement.pdf>
- Source code
 - <https://github.com/aplaceof/FlowScope>

Reference

- [Charikar M, 2000] Charikar, Moses. "Greedy approximation algorithms for finding dense components in a graph." International Workshop on Approximation Algorithms for Combinatorial Optimization, 2000.
- [Asahiro et al, SWAT'96] Asahiro, Yuichi, et al. "Greedily finding a dense subgraph." Algorithm Theory SWAT'96 (1996): 136-148.
- [B Hooi et al, KDD'16] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: bounding graph fraud in the face of camouflage. KDD 2016
- [M-Zoom] Kijung Shin, Bryan Hooi, and Christos Faloutsos. M-Zoom: Fast Dense- Block Detection in Tensors with ality Guarantees. ECML-PKDD. 2016, 264–280.
- [D-Cube] Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. 2017. D-Cube: Dense- Block Detection in Terabyte-Scale Tensors. WSDM '17. 2017.
- [SpokEn] B Aditya Prakash, Mukund Seshadri, Ashwin Sridharan, Sridhar Machiraju, and Christos Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. PAKDD 2010, 290–295.
- [Holoscope] Liu, S.; Hooi, B.; and Faloutsos, C. 2017. Holoscope: Topology-and-spike aware fraud detection. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 1539–1548. ACM.
- [RRCF] Guha, S.; Mishra, N.; Roy, G.; and Schrijvers, O. 2016. Robust random cut forest based anomaly detection on streams. In International conference on machine learning, 2712–2721.

Future work: using temporal information

- Given
 - tuples (account, account, amount , time stamp)
- Find
 - multi-partite flow
- Such that
 - not only contains high-volume money, but high frequency in a short time
 - new metric

Thank you

Questions and Answers

