Intelligent Anti-Money Laundering Solution Based upon Novel Community Detection in Massive Transaction Networks on Spark

2017

Introduction

- AML system: rule-based, lack of pattern recognition function
- Divide complex structures into meaningful subgroups and calculate the suspicious degree for each group
- **Temporal-directed Louvain algorithm**
- Implement on Spark GraphX platform

Design and implementation

A. Select effective maximal connected subgraph in massive transaction networks

B. Community detection according to ML characteristics
   - Edge weight optimization by node correction
   - Temporal correction for edge weight
   - Directed optimization for modularity

C. Money laundering risk quantization for Communities

A. Select effective maximal connected subgraph in massive transaction networks

- Focus on the transaction data in a certain time period P_t
- Node: account, edge: transfer
- Merge the edge between node pairs with the same source and destination
  - $w_{Bi} = e^{\omega_m \cdot M_i + \omega_c \cdot C_i}$, where $\omega_m + \omega_c = 1$
- Divide the graph into different maximal connected subgraphs (MCS)
  - Filter the MCSs by the formula: $V_{\theta 1} < V_{mcs} < V_{\theta 2}$
- Hub nodes: the node whose degree exceeds a threshold D_θ
  - MCSs can be further filtered by $N_{hubs} > N_\theta$

B. Community detection according to ML characteristics

A) Edge weight optimization by node correction

- Louvain algorithm: overlook the weight of node -> need corrections
- The suspicious degree of a node is also important
- Node correction for *src*: $\sigma_s = e^{\omega_{Mv} \cdot M_s + \omega_{Cv} \cdot C_s + \omega_{Dv} \cdot D_s}$
- Node correction for *dst*: $\sigma_d = e^{\omega_{Mv} \cdot M_d + \omega_{Cv} \cdot C_d + \omega_{Dv} \cdot D_d}$
- New edge weight: $W_{Ni} = \sigma_s * \sigma_d * W_{Bi}$

B. Community detection according to ML characteristics

B) Temporal correction for edge weight

- t_A^{in} and t_A^{out} : node's average time point of all inbound and outbound transfers
- $T_{s \to d}$: average transfer time point for edge *src* -> *dst*
- Deg_s^{in} and Deg_s^{out} : the number of all inbound and outbound transfers for *src*

B. Community detection according to ML characteristics

B) Temporal correction for edge weight

Pattern 1: “Centralized out after multi-transfer in” (for edge *src* -> *dst*)

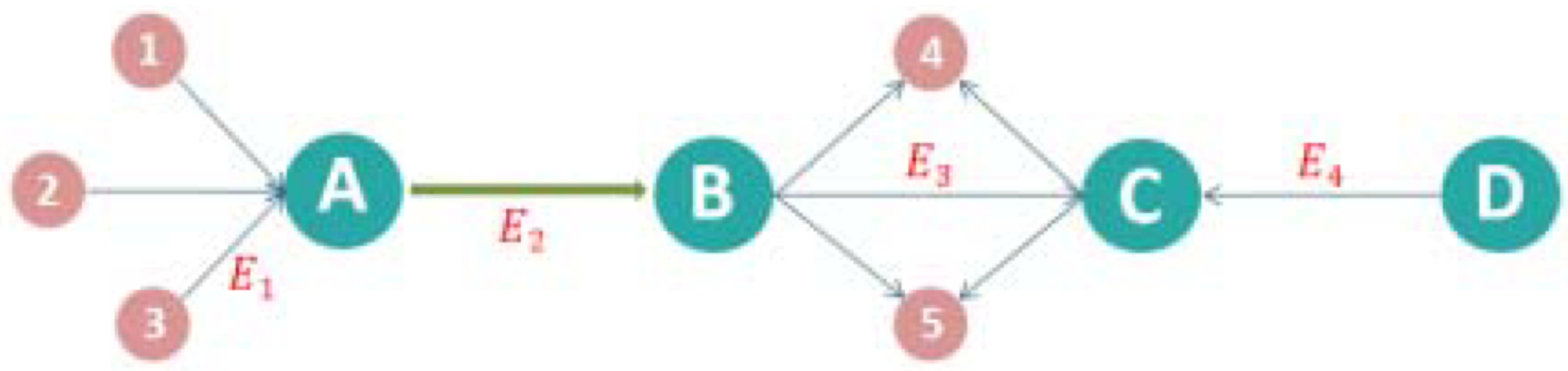
- $Deg_s^{in} > Deg_s^{out}$ and $T_{s\to d} > t_s^{in}$: promote edge weight
- $Deg_s^{in} > Deg_s^{out}$ but $T_{s\to d} < t_s^{in}$: reduce edge weight
- The correction factor $\theta_s = e^{\beta_s \cdot \tau_s^{out}}$
  , where $\beta_s = \dfrac{Deg_s^{in} - Deg_s^{out}}{Deg_s^{in} + Deg_s^{out}}$, $\tau_s^{out} = \dfrac{P_T}{(T_{s\to d} - t_s^{in})}$
- $\theta_s > 1$ if $Deg_s^{in} > Deg_s^{out}$ and $T_{s\to d} > t_s^{in}$, else $\theta_s \leq 1$

B. Community detection according to ML characteristics

B) Temporal correction for edge weight

Pattern 2: “centralized in edge before multi-transfer out” (for destination node)

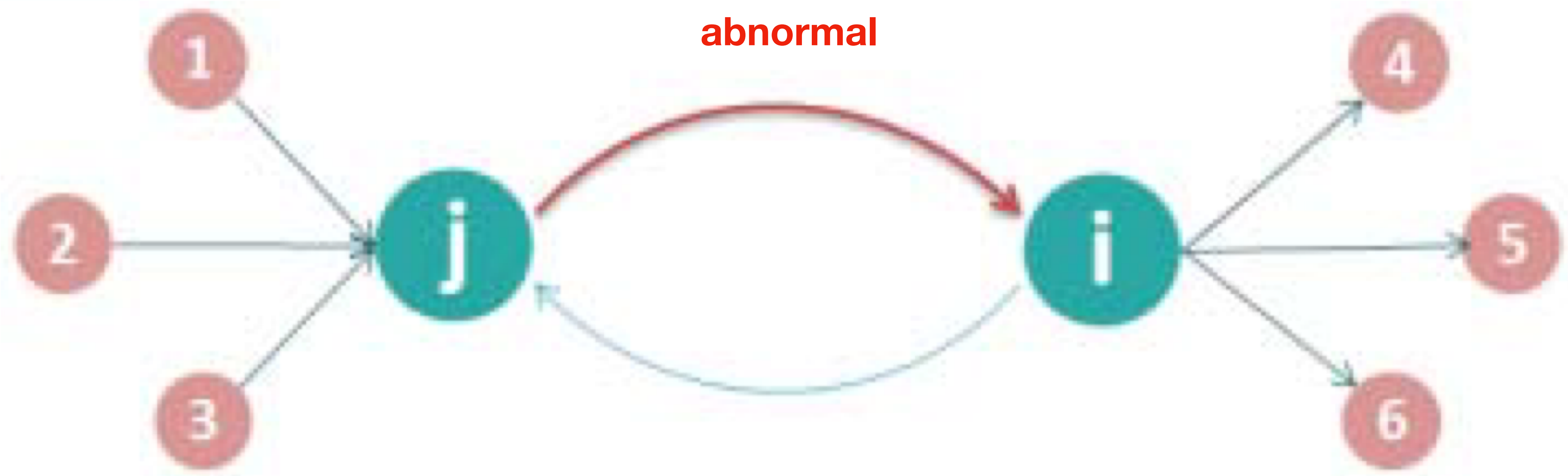
- $Deg_d^{int} < Deg_d^{out}$ and $T_{s\to d} < t_d^{out}$
- The correction factor $\theta_d = e^{\beta_d \cdot \tau_d^{in}}$
  , where $\beta_d = \dfrac{Deg_d^{in} - Deg_d^{out}}{Deg_d^{in} + Deg_d^{out}}, \tau_d^{in} = \dfrac{P_T}{(T_{s\to d} - t_d^{out})}$
- $$W_{Ei} = \begin{cases} \theta_s \cdot \theta_d \cdot W_{Ni}\ , & \beta_s > 0 \text{ and } \beta_d < 0 \\ \theta_s \cdot W_{Ni}\ , & \beta_s > 0 \text{ and } \beta_d > 0 \\ \theta_d \cdot W_{Ni}\ , & \beta_s < 0 \text{ and } \beta_d < 0 \\ W_{Ni}\ , & \beta_s < 0 \text{ and } \beta_d > 0 \end{cases}$$



B. Community detection according to ML characteristics

C) Directed optimization for modularity

- Consider the asymmetry of information caused by the direction of edges
- Example:
  - Node *i* : low in-degree, high out-degree
  - Node *j* : high in-degree, low out-degree



B. Community detection according to ML characteristics

C) Directed optimization for modularity

- Revision for Louvain algorithm:

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j),$$

  - Define k_n^{in}, k_n^{out}, k_n is the sum of edge weight linked in, out, and with node
  - Revised factor $\delta_n = (k_n^{in} - k_n^{out})/k_n$
  - $$Q_D = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{e^{\delta_i - \delta_j}k_i k_j}{2m}\right]\delta(c_i, c_j)$$

$$= \frac{1}{2m}\left[\sum_{i,j}A_{ij} - \frac{\sum_i e^{\delta_i}k_i \sum_j e^{-\delta_j}k_j}{2m}\right]\delta(c_i, c_j) = \sum_c\left[\frac{\sum W_{ec}}{2m} - \left(\frac{1}{2m}\right)^2\sum M_c\right]$$

B. Community detection according to ML characteristics

* Louvain algorithm:

- Modularity: measures the relative density of edges inside communities with respect to edges outside communities

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

- Compare to a randomly rewired network

- Divide an edge into 2 stubs -> m edges become 2m stubs

- The probability for a stub of node i connecting with node j: $\dfrac{k_j}{2m-1}$

- The probability for node i connecting with node j: $\dfrac{k_i k_j}{2m-1}$

B. Community detection according to ML characteristics

- A_{ij} is the weight of the edge between i and j
- $m = \frac{1}{2}\sum_{i,j} A_{ij}$ is the sum of edge weights in the whole graph.
- $\delta(c_i, c_j) = 1$ if $c_i = c_j$, else $\delta(c_i, c_j) = 0$
- c_i is the community where node i belongs to
- k_i is the sum of all edge weights attached to node i
- M_c is the corresponding matrix for each community
- $\sum M_c$ is the sum of all elements in M_c
- $\sum_c$ means accumulating in the original community

$$M_c = \begin{bmatrix} e^{\delta_1-\delta_1}k_1k_1, & e^{\delta_1-\delta_2}k_1k_2, & \cdots & e^{\delta_1-\delta_N}k_1k_N \\ e^{\delta_2-\delta_1}k_2k_1, & e^{\delta_2-\delta_2}k_2k_2, & \cdots & e^{\delta_2-\delta_N}k_2k_N \\ \vdots & \vdots & \ddots & \vdots \\ e^{\delta_N-\delta_1}k_Nk_1, & e^{\delta_N-\delta_2}k_Nk_2, & \cdots & e^{\delta_N-\delta_N}k_Nk_N \end{bmatrix}$$

B. Community detection according to ML characteristics

- Temporal-directed Louvain algorithm:

Step 1: Initialize the community tag for each node by using its own node tag.

Step 2: For each node, maximize the difference ΔQ_D between allocating it in the original community and the neighbor community →if the maximum $\Delta Q_D > 0$, actualize the allocation. Following is the revised formula for ΔQ_D

$$\Delta \mathrm{Q_D} = \left[\frac{\sum \mathrm{W_{ec}} + \mathrm{k_i^c}}{2\mathrm{m}} - \frac{1}{(2\mathrm{m})^2}\sum \mathrm{M_{cnew}}\right] - \left[\frac{\sum \mathrm{W_{ec}}}{2\mathrm{m}} - \frac{1}{(2\mathrm{m})^2}\sum \mathrm{M_c} - \frac{\mathrm{e}^{\delta_i - \delta_i}\mathrm{k_i k_i}}{(2\mathrm{m})^2}\right] = \frac{\mathrm{k_i^C}}{2\mathrm{m}} - \frac{\sum \mathrm{M_{cnew}} - \sum \mathrm{M_c} - \mathrm{k_i k_i}}{(2\mathrm{m})^2} = \frac{\mathrm{k_i^C}}{2\mathrm{m}} - \frac{\Theta_\mathrm{i}}{(2\mathrm{m})^2} \qquad (4)$$

- k_i^c: the total weight of edges formed between node *i* and all nodes in the community
- $$\Theta_i = k_i e^{\delta_i} \sum_c k_j e^{-\delta_j} + k_i e^{-\delta_i} \sum_c k_j e^{\delta_j} \qquad \mathrm{M_{cnew}} = \begin{bmatrix} e^{\delta_1-\delta_1}k_1k_1, & e^{\delta_1-\delta_2}k_1k_2, & \cdots & e^{\delta_1-\delta_N}k_1k_N, & e^{\delta_1-\delta_i}k_1k_i \\ e^{\delta_2-\delta_1}k_2k_1, & e^{\delta_2-\delta_2}k_2k_2, & \cdots & e^{\delta_2-\delta_N}k_2k_N, & e^{\delta_2-\delta_i}k_2k_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ e^{\delta_N-\delta_1}k_Nk_1, & e^{\delta_N-\delta_2}k_Nk_2, & \ddots & e^{\delta_N-\delta_N}k_Nk_N, & e^{\delta_N-\delta_i}k_Nk_i \\ e^{\delta_i-\delta_1}k_ik_1, & e^{\delta_i-\delta_2}k_ik_2, & \cdots & e^{\delta_i-\delta_N}k_ik_N, & e^{\delta_i-\delta_i}k_ik_i \end{bmatrix}$$

B. Community detection according to ML characteristics

- Temporal-directed Louvain algorithm:

Step 3: Repeat Step 2 until the community tag of all nodes does not change.

Step 4: Compress nodes with the same community label into a new node.

Original total edge weight → self-link weight of new node

Step 5: Repeat Step 2 until the modularity of the whole graph does not change

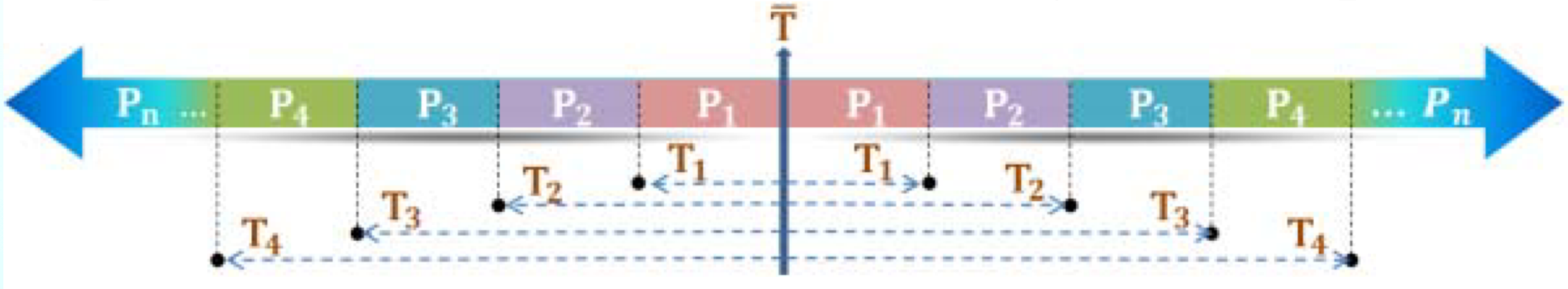
C. Money laundering risk quantization for Communities

- ML risk score for the community *k*: $\Psi_k = e^{w_k \cdot V_k + w_E \cdot E_k + w_M \cdot M_k + w_D \cdot D_k + w_H \cdot H_k}$
- V_k : total nodes number
- E_k : total edge number
- M_k : total money amount
- D_k : average node suspicious degree
- Temporal entropy $H_c = -\sum_{i=1}^{n} P_i * log_2 P_i$

  (to measure the transfer time concentration)

C. Money laundering risk quantization for Communities

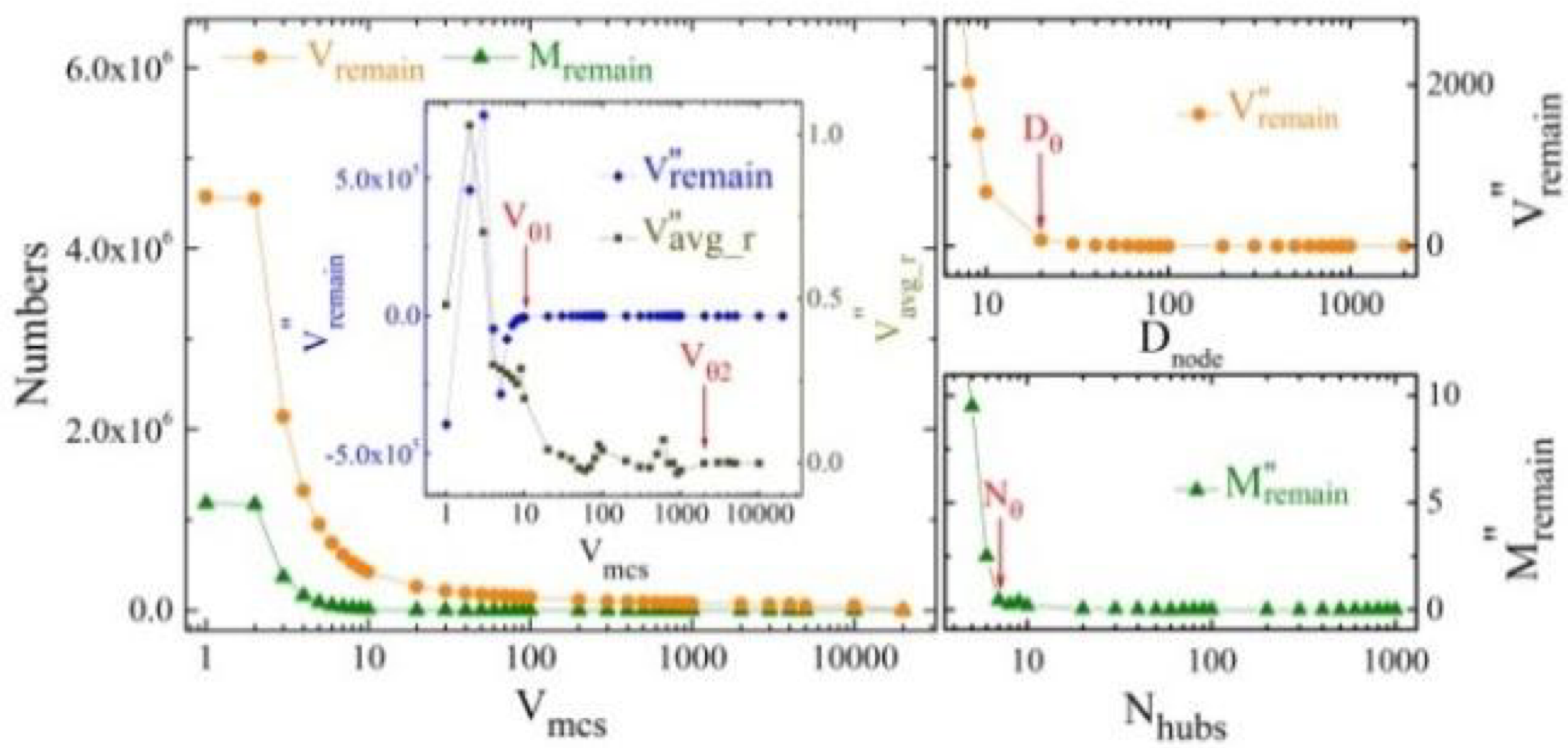
- $\bar{T}$: the average time point of a community
- ΔT : the absolute interval between each transfer time point and $\bar{T}$
- P_1 is the percentage of transactions in the segment of $0 \leq \Delta T \leq T_1$
- P_2 is the percentage of transactions in the segment of $T_1 \leq \Delta T \leq T_2$ ....
- All segments should obey $\sum_{i=1}^{n} P_i = 1$



III. EXPERIMENTS AND RESULTS

A. Determination of Filter's Parameter

- Data: around 10 million of real transfer records in the first week of November 2016 -> filtered -> scale down to 45% of original size
- Filter's Parameter: $V_{\theta 1}$, $V_{\theta 2}$, D_θ , N_θ
- $V_{avg_r} = V_{remian}$ / M_{remain}



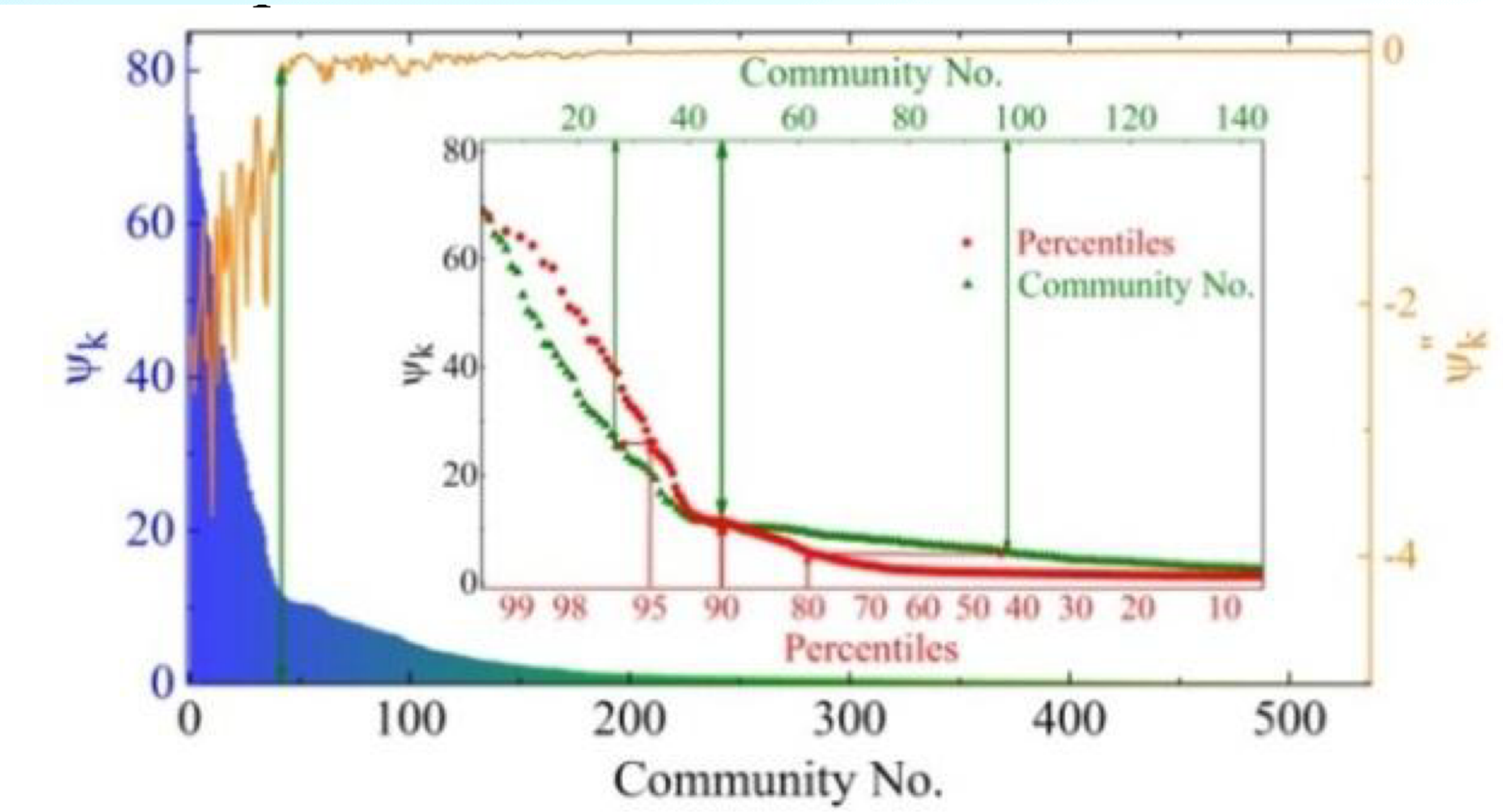
A. Select effective maximal connected subgraph in massive transaction networks

Recall

- Focus on the transaction data in a certain time period P_t
- Node: account, edge: transfer
- Merge the edge between node pairs with the same source and destination
  - $w_{Bi} = e^{\omega_m \cdot M_i + \omega_c \cdot C_i}$, where $\omega_m + \omega_c = 1$
- Divide the graph into different maximal connected subgraphs (MCS)
  - Filter the MCSs by the formula: $V_{\theta 1} < V_{mcs} < V_{\theta 2}$
- Hub nodes: the node whose degree exceeds a threshold D_θ
  - MCSs can be further filtered by $N_{hubs} > N_\theta$

B. ML risk level partition

- Orange line: first derivative for the rick score
- 95% ~ 100%: level 1, 90% ~ 95%: level 2, 80% ~ 90%: level 3
- After reporting level 1, 2 -> 9/13 were reconfirmed suspicious



C. TD Louvain algorithm v.s. Louvain algorithm

- Nodes ascribed to different communities are drawn in distinct colors
- Edges are also drawn in progressive color with increasing time