# EigenPulse: Detecting Surges in Large Streaming Graph with Row Augmentation

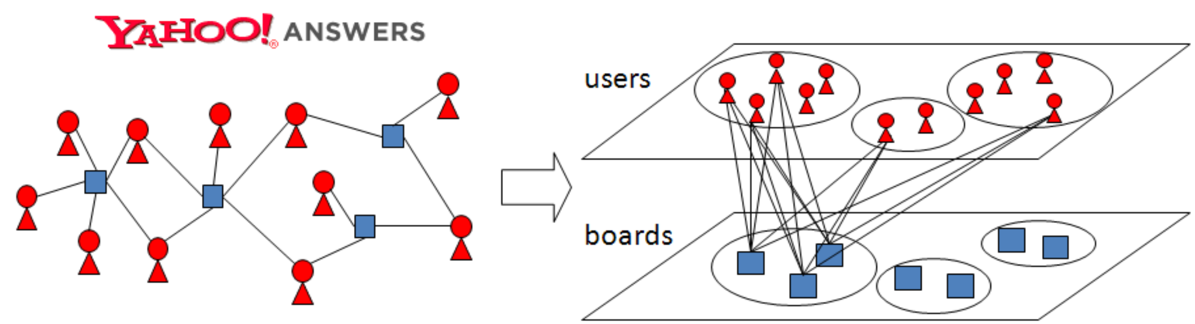
Jiabao Zhang[+], Shenghua Liu[+], Wenjian Yu[*], Wenjie Feng[+] and Xueqi Cheng[+]

*Tsinghua University
+Institute of Computing Technology ICT, CAS

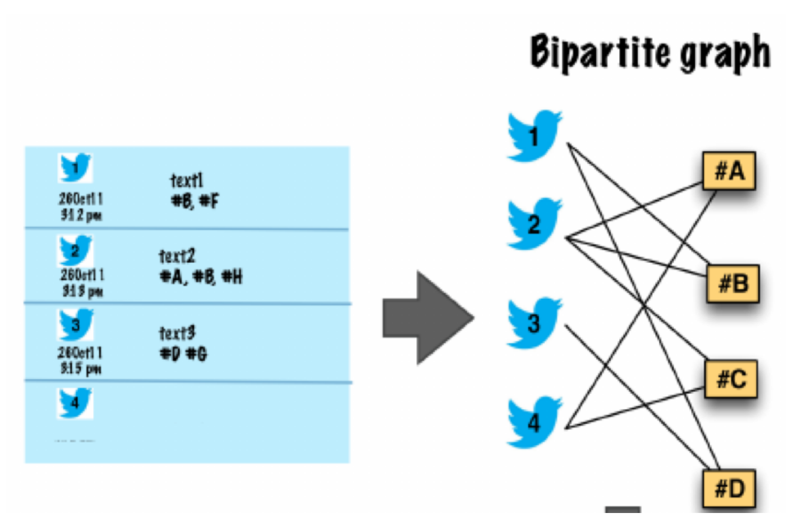
2020/4/20

# Graphs are everywhere.

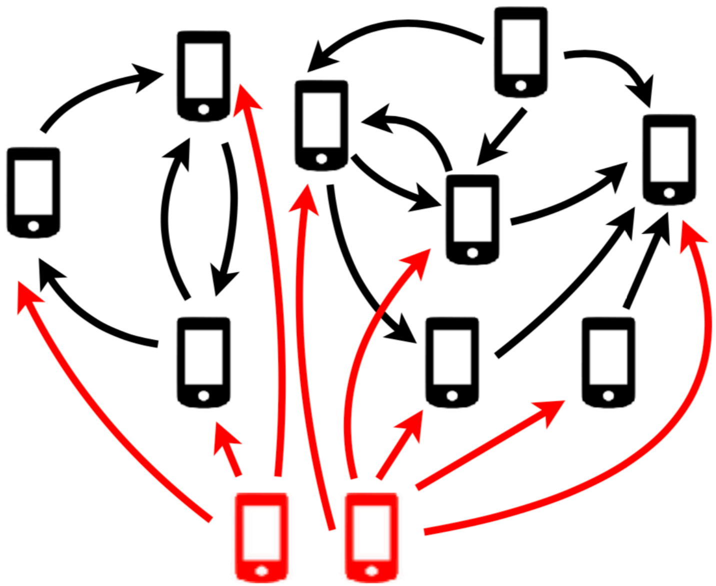
- Yahoo answers




- Twitter



# Density usually indicates unusual events



Dense subgraph

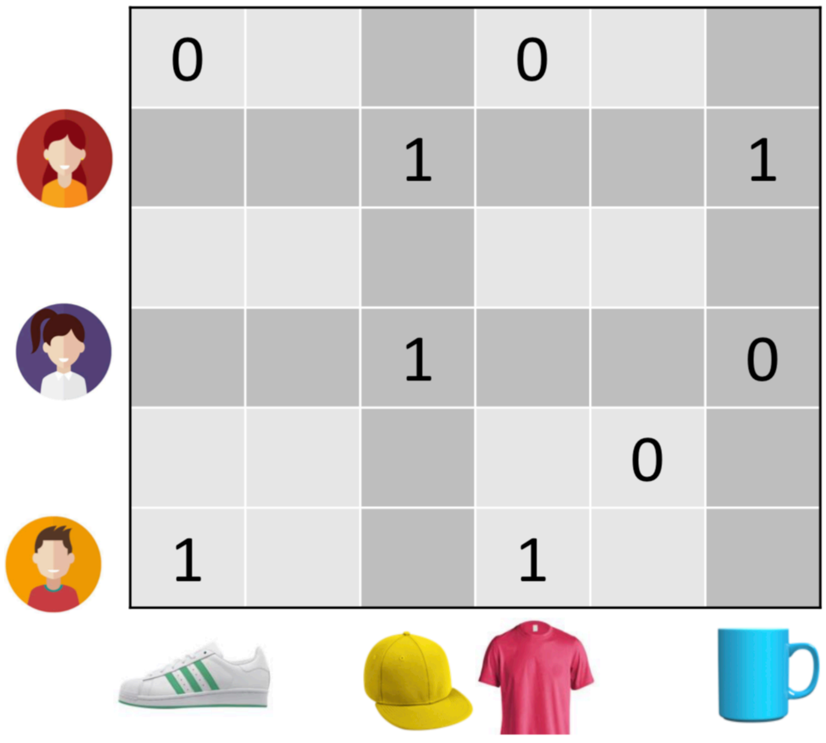
density higher than normal subgraphs!!

Eswaran, D., Faloutsos, C., Guha, S., Mishra, N. Spotlight: Detecting anomalies in streaming graphs. In: SIGKDD. pp. 1378–1386. ACM (2018)

# Adjacency matrix for a graph

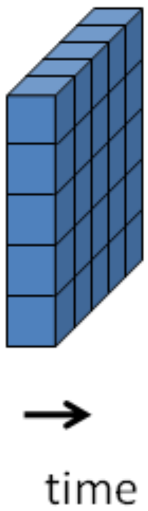
$\mathcal{G}(U,V,E)$

Adjacency Matrix



# Streaming graph

- graphs usually expand with time.

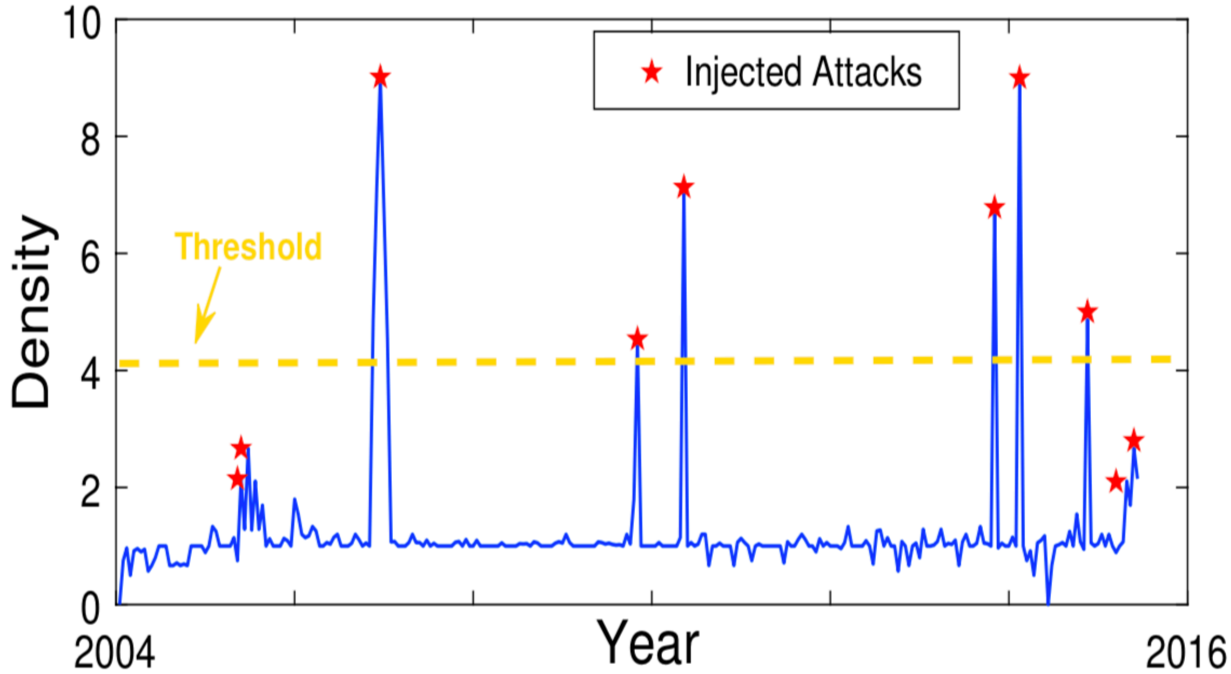


**How do we detect such anomalies in streaming graphs?**

How do we even characterize these anomalies?

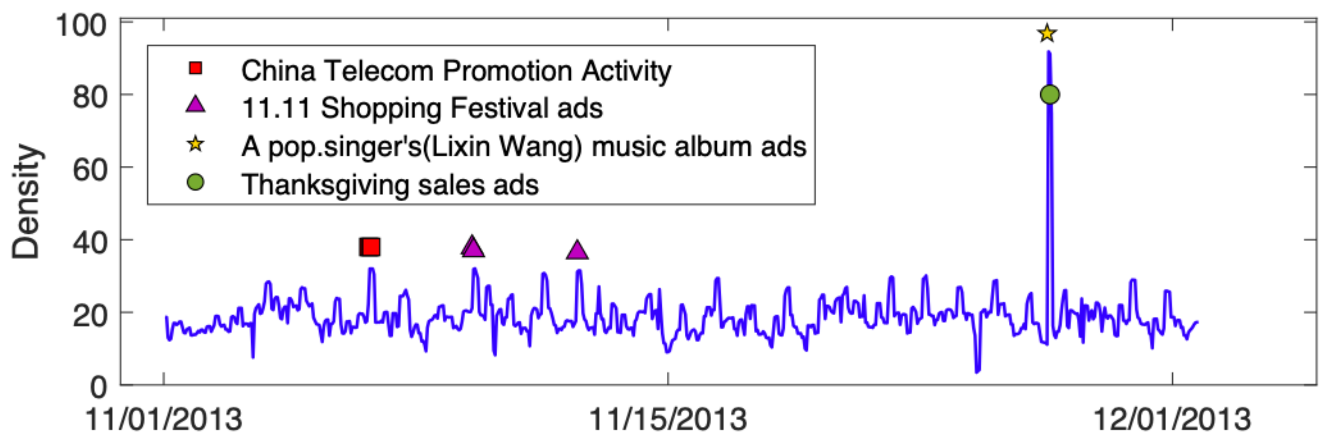
# EigenPulse: detect injection accurately

EigenPulse: “*graph heart-beat*”



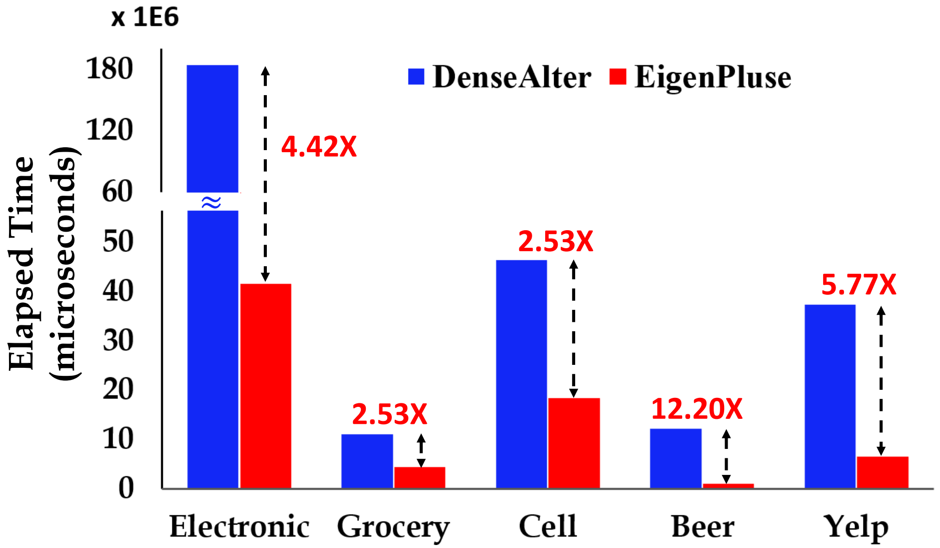
# EigenPulse: detect anomalous surges on real data

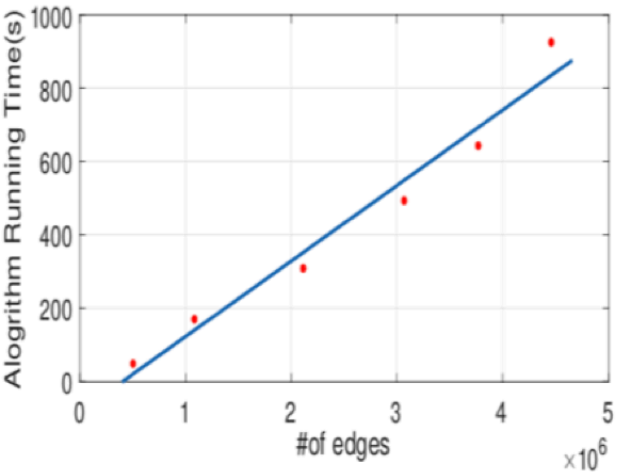
- Microblog: Sina Weibo, Nov. 2013
  - node size: 2.74M x 8.08M, # of edges: 50.06M




EigenPulse: “graph heart-beat”

# EigenPulse: run fast and near-linearly




EigenPulse outperforms DenseAlert by more than 2.53x.

run near-linearly in # of edges

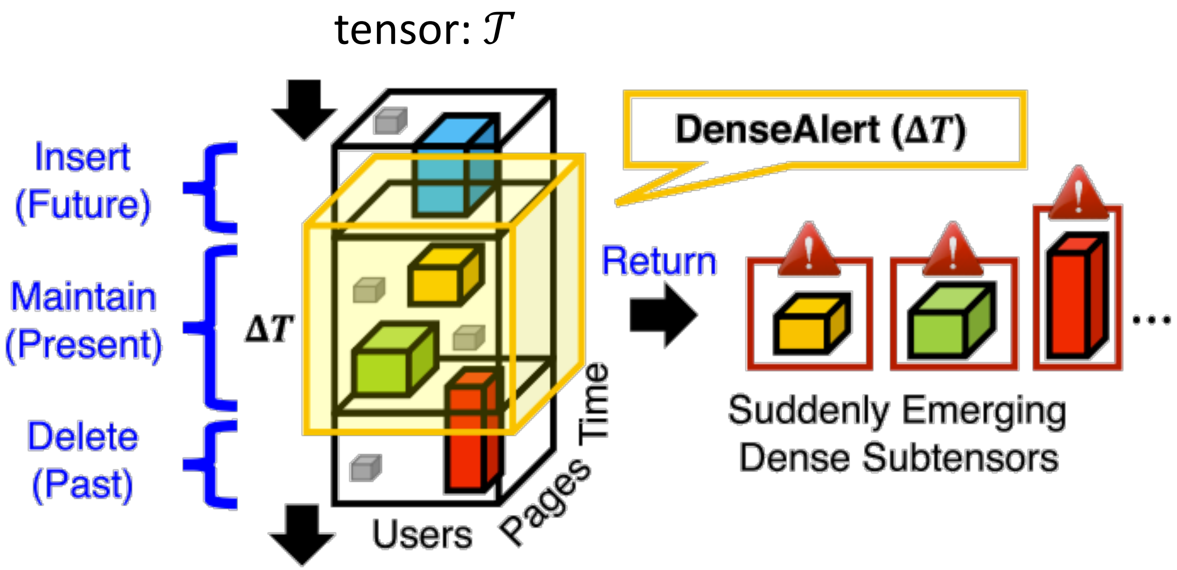
# Outline

- Problem
- Related works
- Our model
- Our algorithm
- Experiments
- Conclusion

# Problem

- **Given:**
  - a stream of triplets (*user, object, timestamp*).
- **Find:**
  - at *each time step*, a group of users and objects who have *densest* edges
  - *detect* suspicious surges of density

# Related work: DenseAlert




Wikipedia edit history

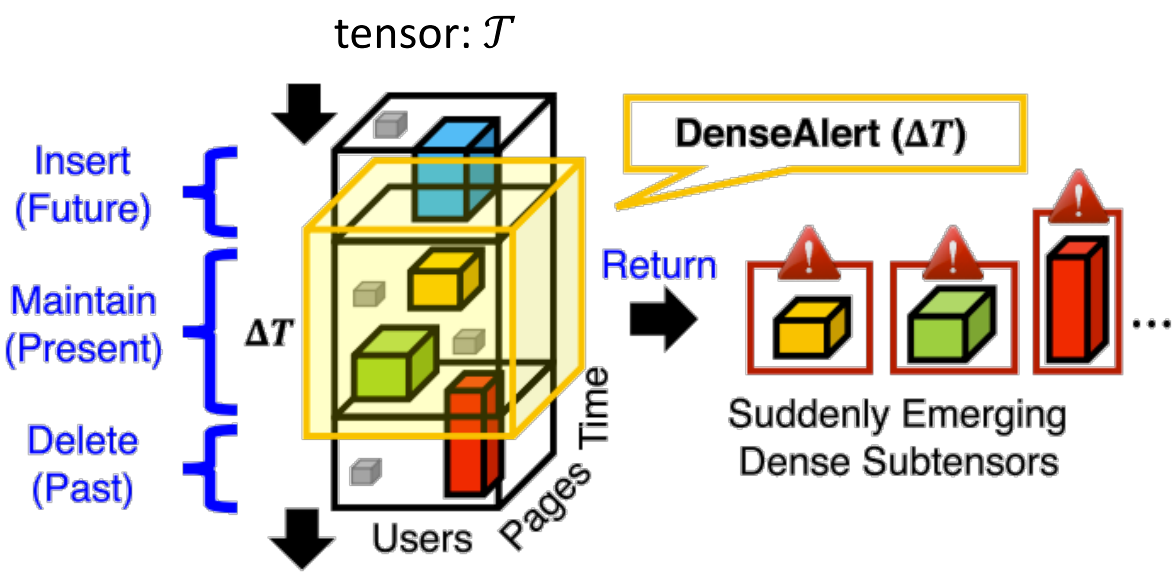
**Given:** a **stream** of changes, e.g. adding/removing edges in tensor $\mathcal{T}$.

- **Maintain:** a subtensor $\mathcal{T}(S)$, where $S$ is set of slice indices.

- **to maximize:** density $\rho(\mathcal{T}(S))$

$$\rho\big(\mathcal{T}(S)\big) = \frac{sum(\mathcal{T}(S))}{|S|}$$

# Related work: DenseAlert



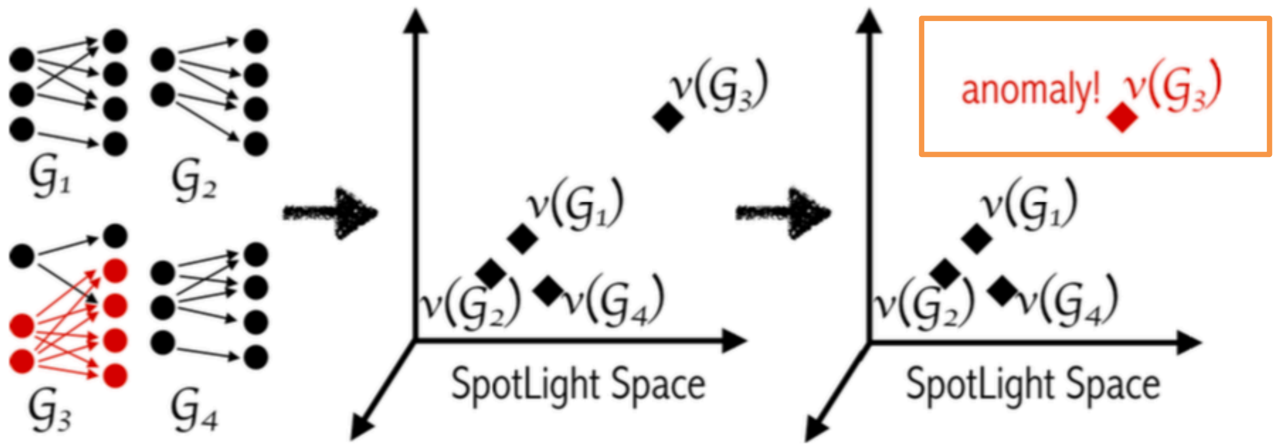

Wikipedia edit history

- Need an updating operation for *every single adding or removing* edge.
- Running a bit *slow*.

Shin, K., Hooi, B., Kim, J., Faloutsos, C.: Densealert: Incremental dense-subtensor detection in tensor streams. In: KDD. ACM (2017)

# Related work: SpotLight

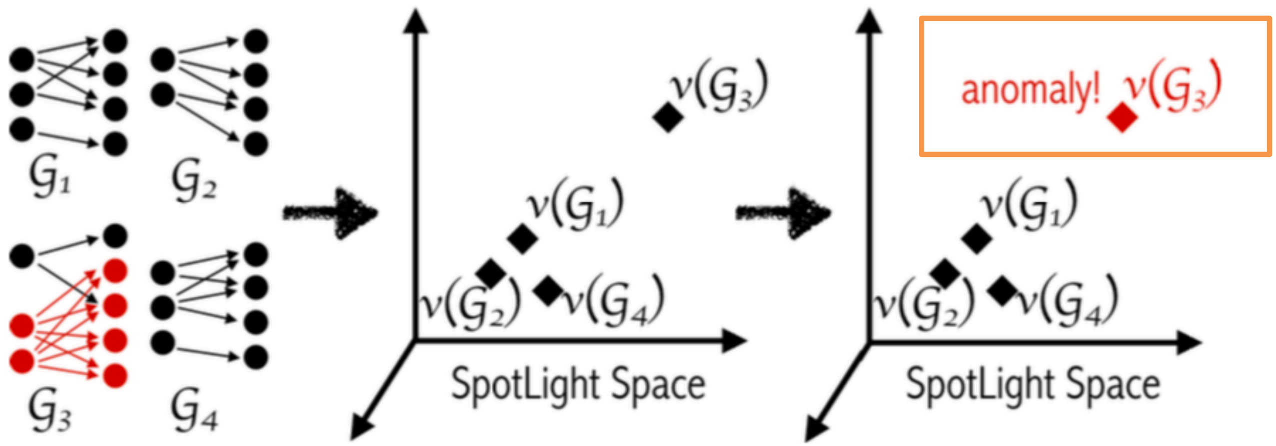
- Given a sequence of *snapshots* of dynamic graphs
  - extract a K-dimensional *sketch* $v(G)$ for every snapshot.
  - *detect* the anomalous snapshot in sketch space




Eswaran, D., Faloutsos, C., Guha, S., Mishra, N. Spotlight: Detecting anomalies in streaming graphs. In: SIGKDD. pp. 1378–1386. ACM (2018)

# Related work: SpotLight

- *Only* spot anomalous snapshots at time windows
- *Not* detect the exactly suspicious groups (subgraphs)




Eswaran, D., Faloutsos, C., Guha, S., Mishra, N. Spotlight: Detecting anomalies in streaming graphs. In: SIGKDD. pp. 1378–1386. ACM (2018)

# Related works: summary of comparisons

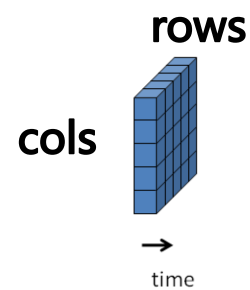
| | Fraudar | HoloScope<br>D-Cube<br>M-Zoom | DenseAlert | SpotLight | **EigenPulse** |
|---|---|---|---|---|---|
| temporal information | | ✓ | ✓ | ✓ | ✓ |
| streaming graphs | | | ✓ | ✓ | ✓ |
| suspicious groups | ✓ | ✓ | ✓ | | ✓ |
| theoretical analysis | ✓ | ✓ | ✓ | ✓ | ✓ |
| scalability | ✓ | ✓ | ✓ | ✓ | ✓ |

# Outline

- Problem
- Related works
- Our model
- Our algorithm
- Experiments
- Conclusion

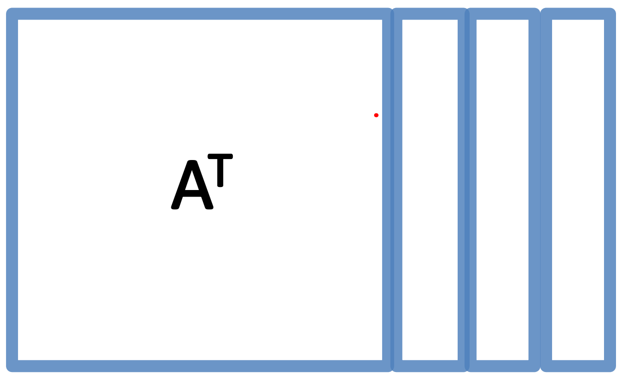
# Model: Row-Augmented Matrix

- **Row-Augmented Matrix (RAM)**
  - rows are augmented with same cols
  - *concatenate*




row: concat(object, ts)

col: user

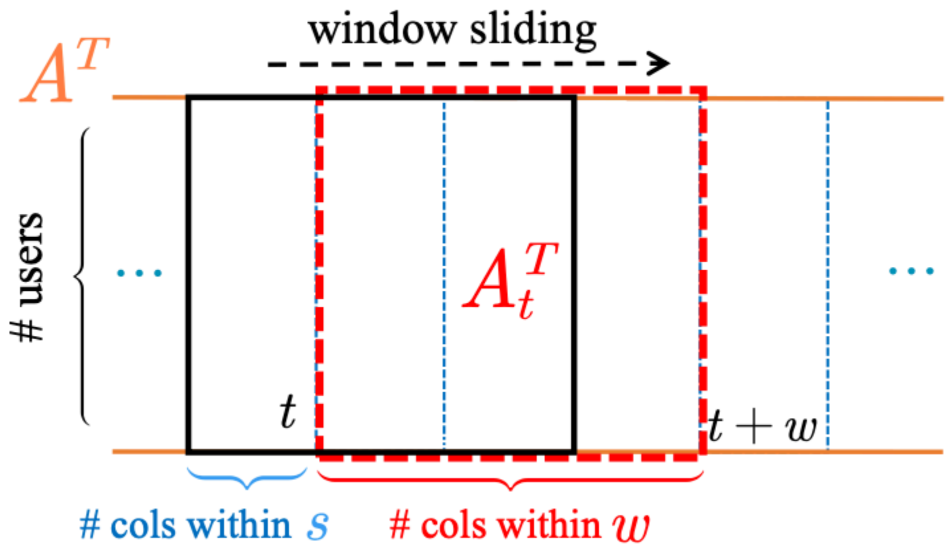



row augmented with timestamp growing!

  - equivalently, unfolding dynamic tensors (removing empty rows)
  - *e.g. same restaurant changes overtime, upgrading of a product, etc.*

# Model: Row-Augmented Matrix

- RAM with sliding window



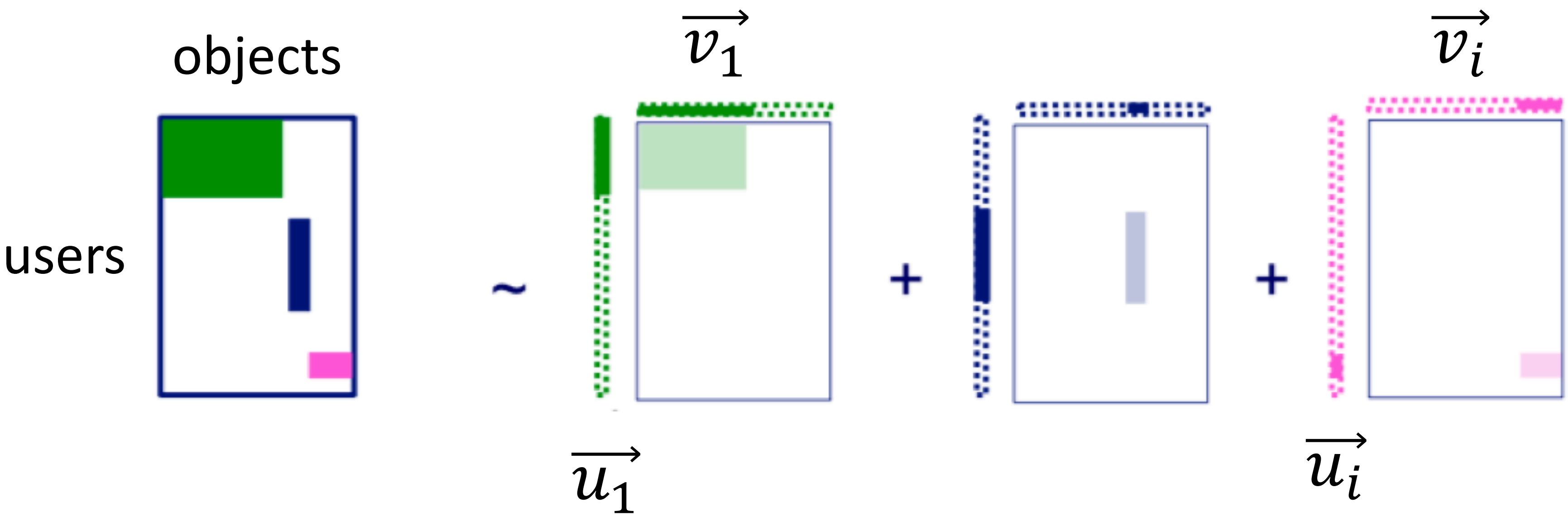

stride

window size

# Crush intro to SVD

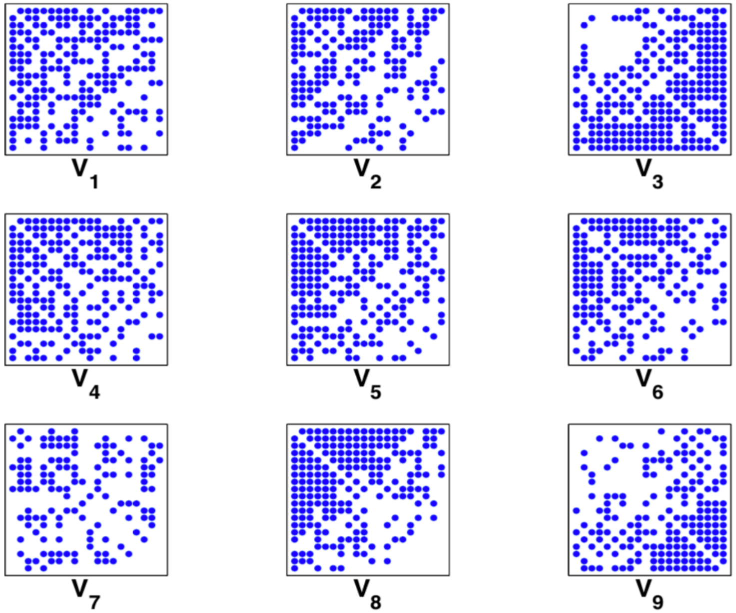
- (SVD) matrix factorization: finds blocks

$$A = U\Sigma V^T$$



# Properties of Singular Vectors

- Find dense groups of users by SVD
  - 20 nodes with the highest magnitude projection along the first 9 singular vectors



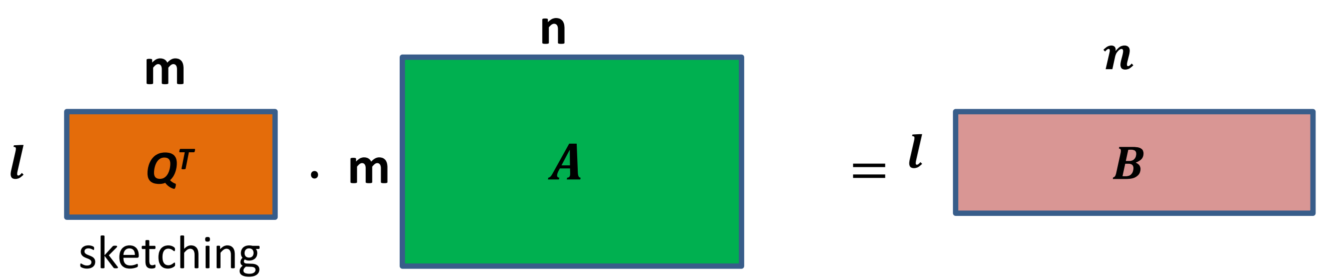

inducing sub-graphs contain near-cliques.

Prakash, B.A., Sridharan, A., Seshadri, M., Machiraju, S., Faloutsos, C. In PAKDD. Springer (2010)

# Outline

- Problem
- Related works
- Our model
- Our algorithm
- Experiments
- Conclusion

# QB approximation




$$\text{(1) } \boldsymbol{\Omega} = \text{randn}(n,\ k+s)$$
$$\text{(2) } \boldsymbol{Q} = \text{orth}(\boldsymbol{A\Omega})$$
$$\text{(3) } \boldsymbol{B} = \boldsymbol{Q}^T \boldsymbol{A}$$

$A: m \times n, l = k + s$

$\Omega: n \times l$

$Q: \mathrm{m} \times l$

$B: l \times n$

$$B = \tilde{U}\tilde{\Sigma}\tilde{V}^T.$$

$$A \approx QB = Q\tilde{U}\tilde{\Sigma}\tilde{V}^T.$$

$$\Rightarrow \quad U = Q\tilde{U}, \Sigma = \tilde{\Sigma}, V = \tilde{V}$$

# AugSVD: incrementally building Q

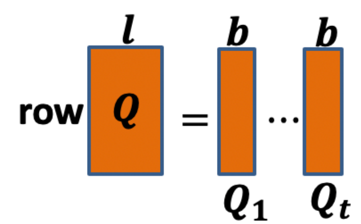
- generate matrices $Q, B$ by $G, H$

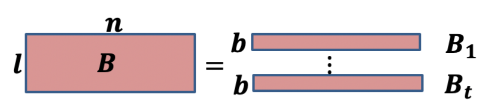
**repeat**
  Read rows $\mathbf{a}$ for next stride $s$ in augmented $A$
  $\mathbf{g} = \mathbf{a}\boldsymbol{\Omega}; \quad \mathbf{h} = \mathbf{a}^T\mathbf{g}$
  $glist.enqueue(\mathbf{g}); \quad hlist.enqueue(\mathbf{h})$
**until** the elements in $glist$ corresponds to a window $w$
**for all** $\mathbf{g}$ in $glist$, $\mathbf{h}$ in $hlist$ **do**
  $\mathbf{G} = [\mathbf{G}, \mathbf{g}]; \quad \mathbf{H} = \mathbf{H} + \mathbf{h}$
**end for**

G: $m$ x $l$
H: $n$ x $l$

$\mathbf{Q} = [\,]; \quad \mathbf{B} = [\,]$
**for** $i = 1, 2, \cdots, t$ **do**
  $\boldsymbol{\Omega}_i = \boldsymbol{\Omega}(:, (i-1)b+1 : ib)$; $\mathbf{Y}_i = \mathbf{G}(:, (i-1)b+1 : ib) - \mathbf{Q}(\mathbf{B}\boldsymbol{\Omega}_i)$
  $[\mathbf{Q}_i, \mathbf{R}_i] = qr(\mathbf{Y}_i)$
  $[\mathbf{Q}_i, \widetilde{\mathbf{R}}_i] = qr(\mathbf{Q}_i - \mathbf{Q}(\mathbf{Q}^T\mathbf{Q}_i))$
  $\mathbf{R}_i = \widetilde{\mathbf{R}}_i\mathbf{R}_i$
  $\mathbf{B}_i = \mathbf{R}_i^{-T}(\mathbf{H}(:, (i-1)b+1 : ib)^T - \mathbf{Y}_i^T\mathbf{Q}\mathbf{B} - \boldsymbol{\Omega}_i^T\mathbf{B}^T\mathbf{B})$
  $\mathbf{Q} = [\mathbf{Q}, \mathbf{Q}_i]; \quad \mathbf{B} = [\mathbf{B}^T, \mathbf{B}_i^T]^T$
**end for**

$Y_i$: $m$ x $b$, $Q_i$: $m$ x $b$






Yu, Wenjian+: Single-Pass PCA, IJCAI. pp. 3350–3356 (2017).

# AugSVD: incrementally building Q

- generate matrices $Q, B$ by $G, H$

**repeat**
    Read rows $\mathbf{a}$ for next stride $s$ in augmented $A$
    $\mathbf{g} = \mathbf{a\Omega}$; $\quad \mathbf{h} = \mathbf{a}^T\mathbf{g}$
    $glist.enqueue(\mathbf{g})$; $\quad hlist.enqueue(\mathbf{h})$
**until** the elements in $glist$ corresponds to a window $w$
**for all** $\mathbf{g}$ in $glist$, $\mathbf{h}$ in $hlist$ **do**
    $\mathbf{G} = [\mathbf{G}, \mathbf{g}]$; $\quad \mathbf{H} = \mathbf{H} + \mathbf{h}$
**end for**

G: $m$ x $l$
H: $n$ x $l$

- with Q and B

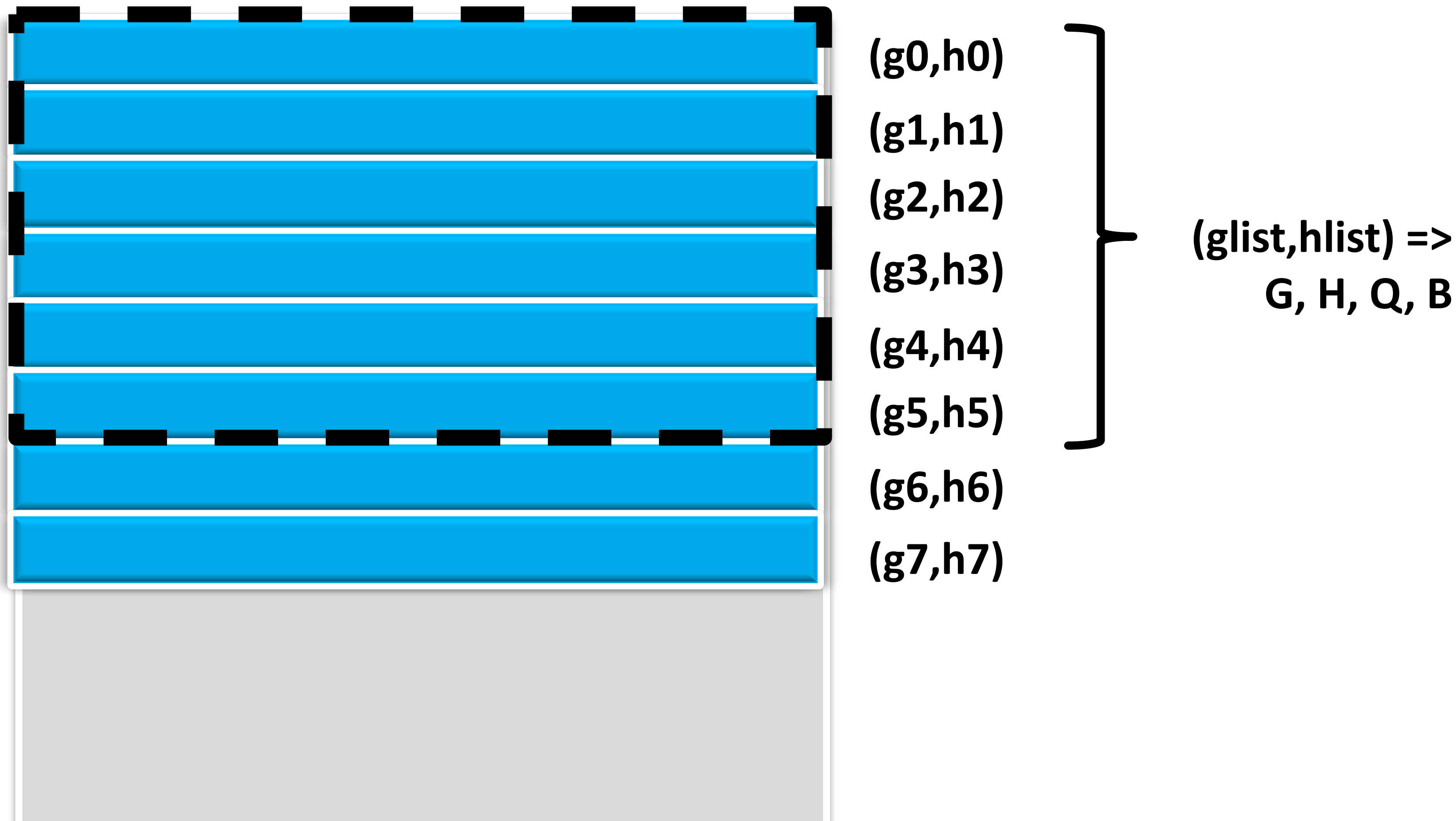
$[\widetilde{\mathbf{U}}, \mathbf{S}, \mathbf{V}] = svd(\mathbf{B})$
$\mathbf{U} = \mathbf{Q}\widetilde{\mathbf{U}}$
$\mathbf{U} = \mathbf{U}(:, 1:k)$; $\quad \mathbf{V} = \mathbf{V}(:, 1:k)$; $\quad \mathbf{S} = \mathbf{S}(1:k, 1:k)$



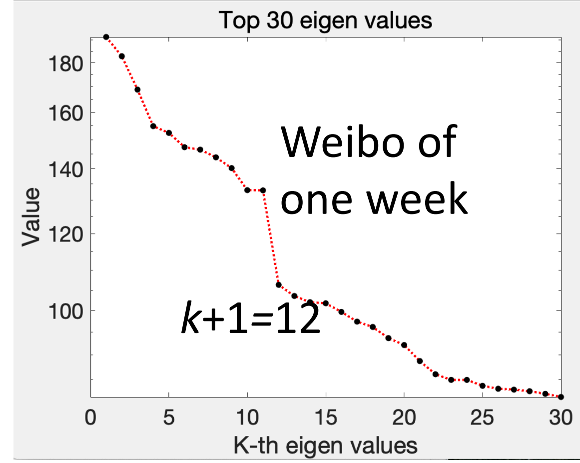
Yu, Wenjian+: Single-Pass PCA, IJCAI. pp. 3350–3356 (2017).

# AugSVD

Combine *Sliding Window*, Change matrices $G, H$ generation.



# Theoretical analysis




- Theorem:
  - Let approx error rate be $\varepsilon_i = (\sigma_i - \hat{\sigma}_i)/\sigma_i$, then

$$\boxed{|\varepsilon_i| \lesssim 2\frac{\sigma_{k+1}}{\sigma_i} + \frac{\mathrm{e}\sqrt{2k+1}}{k}\left(\sum_{j=k+1}^{\min(m,n)}\left(\frac{\sigma_j}{\sigma_i}\right)^2\right)^{1/2}, \quad i = 1,\ldots,k}$$

less than approximately.

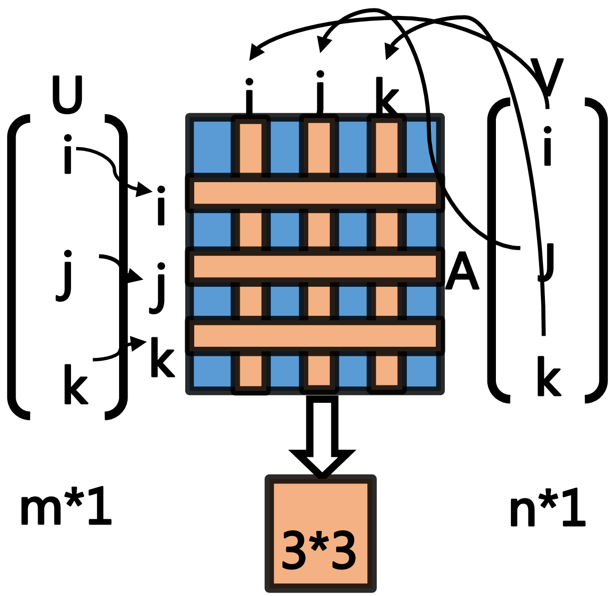
$(k+1)$-th original singular value

$k$ is the truncated length

  - Error is small when $\sigma$ is *highly skewed*

# EigenPulse

- At every time stride,
  - Choose dense blocks based on the first several singular vectors.
    - ✓ above average $\tau_u = \frac{1}{\sqrt{m_t}}; \quad \tau_v = \frac{1}{\sqrt{n}}$



# EigenPulse

- At every time stride,
  - Choose dense blocks based on the first several singular vectors.
  - [Optional] dense block detection in small selected blocks
    - ✓ use Fraudar and HoloScope (HS-α)
  - Calculate density

$$D_t(rowset, colset) = \frac{\sum_{i \in rowset} \sum_{j \in colset} \mathbf{A}_t(i,j)}{|rowset| + |colset|}$$

- plotting EigenPulse, and detecting anomalies.

# Outline

- Problem
- Related works
- Our model
- Our algorithm
- Experiments
- Conclusion

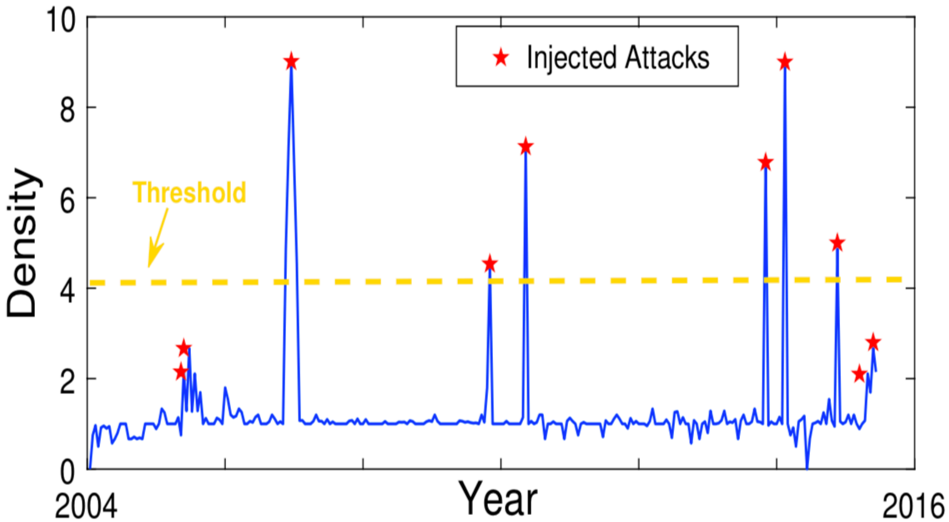
# Data statistics

**Table 1.** Datasets Statistic Information

| Name | nodes | edges | span time |
|---|---|---|---|
| BeerAdvocate | 26.5K $\times$ 50.8K | 1.08M | Jan 08 - Nov 11 |
| Yelp | 686K $\times$ 85.3K | 2.68M | Oct 04 - Jul 16 |
| Amazon Phone & Acc | 2.26M $\times$ 329K | 3.45M | Jan 07 - Jul 14 |
| Amazon Electronics | 4.20M $\times$ 476K | 7.82M | Dec 98 - Jul 14 |
| Amazon Grocery | 763K $\times$ 165K | 1.29M | Jan 07 - Jul 14 |
| Sina Weibo | 2.74M$\times$8.08M | 50.06M | Sep 01 - Dec 01 |

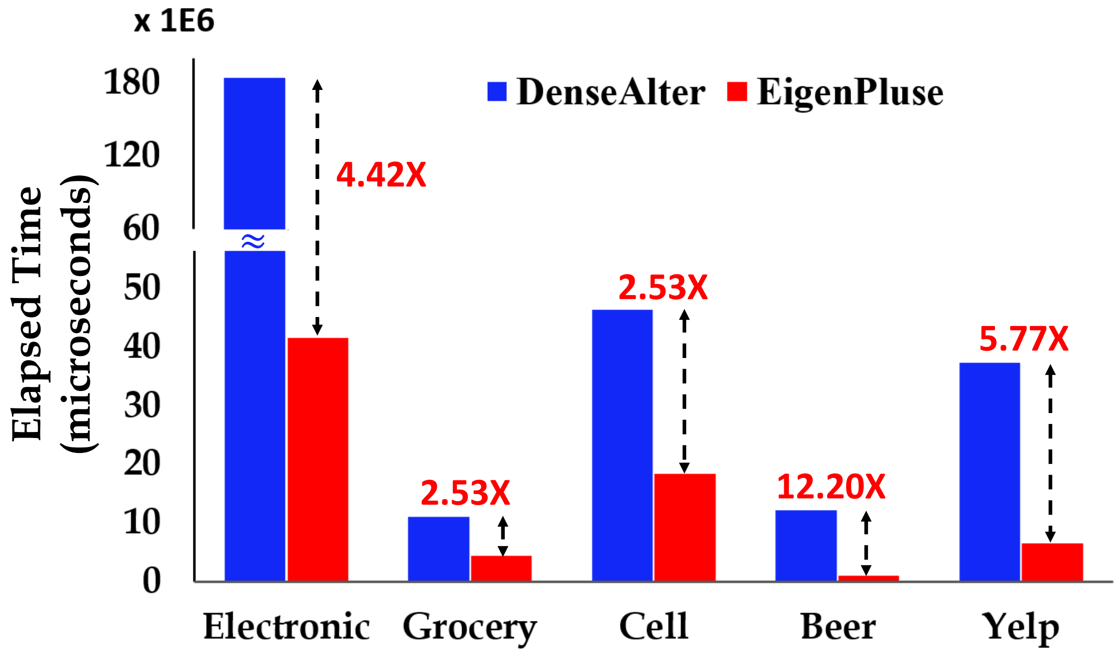
# EigenPluse: detect injection accurately and instantly

- Injected 10 dense blocks for *Yelp* dataset



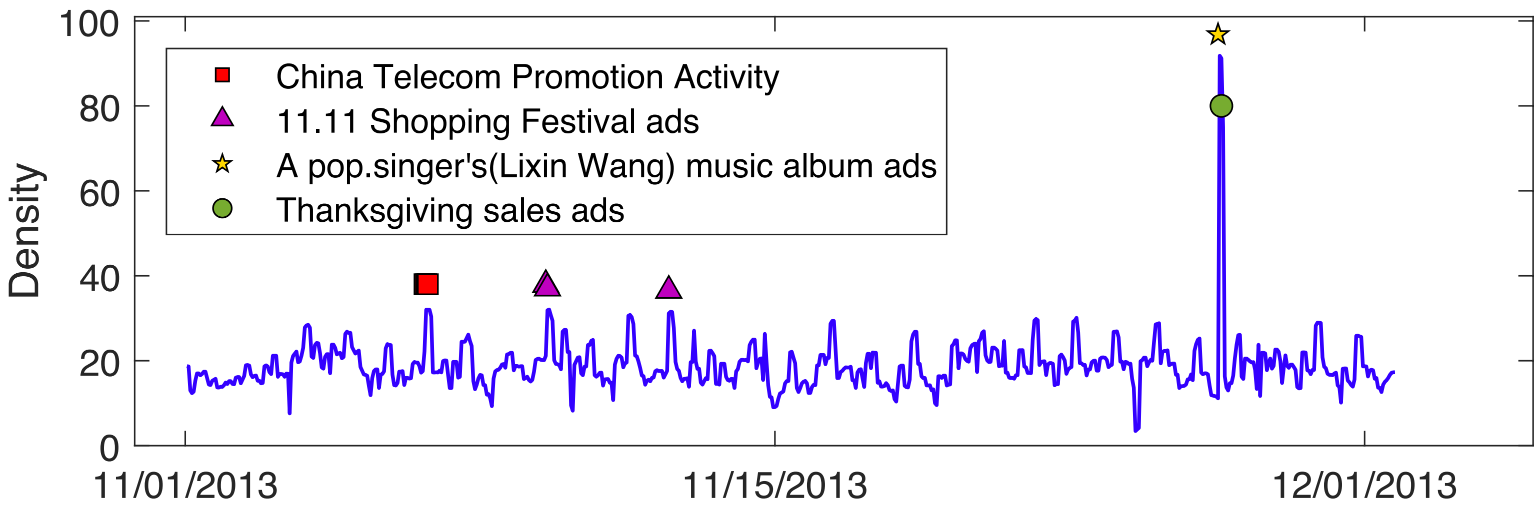

$$threshold = 99.7 \text{ percentile}$$

# EigenPluse: run faster than state-of-art methods




EigenPulse achieves more than 2.53× speed up.

# EigenPluse: detect anomalous surges on Microblog data



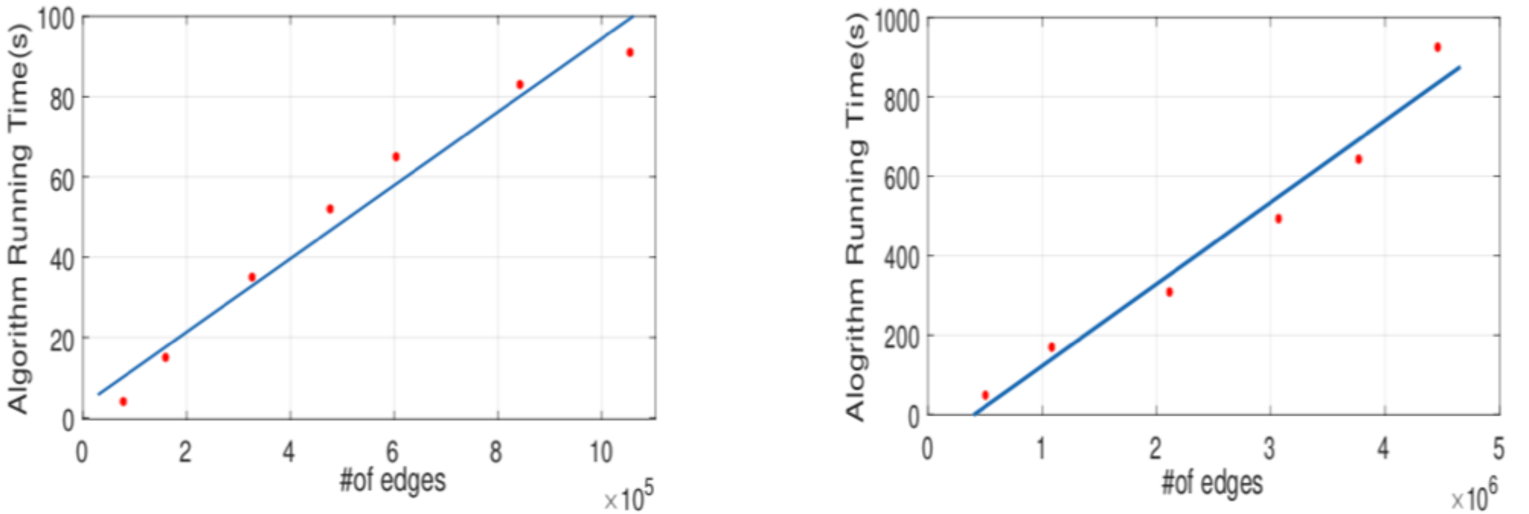

set $w = 2h$ and $s = 1h$ on Sina Weibo data

# Detected Blocks in Sina Weibo

| Message Topic | Size | Time range | #Edges | |
|---|---|---|---|---|
| China Telecom Promotion Activity | 39 × 57 | 6:00∼ 8:00, Nov 7 | 2,004 | |
| | 78 × 58 | 7:00∼9:00, Nov 7 | 4,051 | |
| | 151 × 119 | 8:00∼10:00, Nov 7 | 8,295 | |
| 11.11 Shopping Festival ads | 201 × 139 | 6:00∼8:00, Nov 10 | 7,012 | |
| | 196 × 111 | 7:00∼9:00, Nov 10 | 9,668 | $\approx 100\ rt/user$ |
| | 126 × 93 | 8:00∼10:00, Nov 13 | 638 | |
| A pop. singer's (Lixin Wang) music album ads. | **7** × **8** | 22:00∼24:00, Nov 26 | **953** | $\approx 140\ rt/user$ |
| Thanksgiving sale ads | 26 × 36 | 23:00, Nov 26∼1:00, Nov 27 | 629 | |
| | 43 × 34 | 1:00∼3:00, Nov 27 | 263 | |

**7 users** × **8 messages, 953** edges in **2** hours means every user retweeted **more than once per minute**.

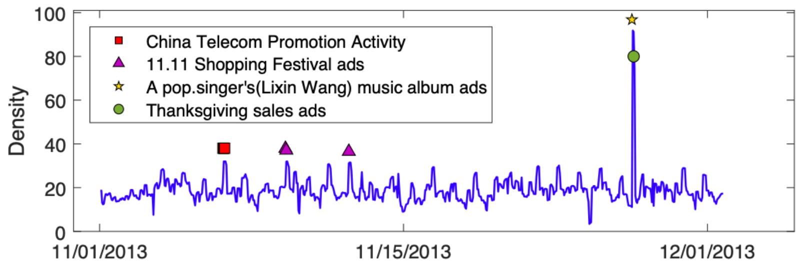
# EigenPulse: Run linear with # of edges

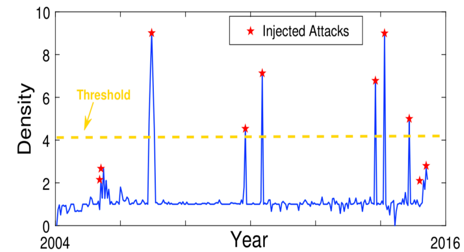



(a) **BeerAdvocate dataset** (b) **Amazon Electronic dataset**

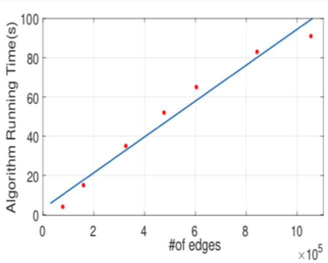
# Conclusion

- Detect dense blocks given a streaming graph in form of triplet (*user, object, timestamp*)
- Robust and effective
  - theoretical robust approximation to batch SVD.

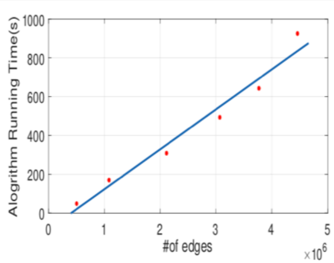


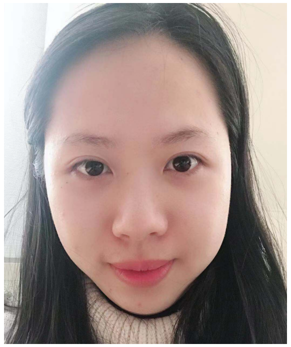

- Scalable




(a) BeerAdvocate dataset




(b) Amazon Electronic dataset



Questions and Answers

# THANK YOU!