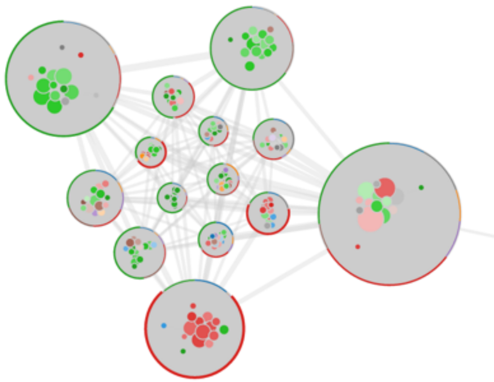
Cheetah: Fast Graph Kernel Tracking on Dynamic Graphs

Presenter: **Liangyue Li**

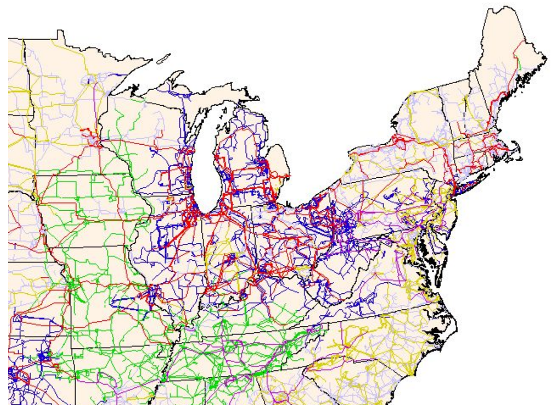
Joint work with

Hanghang Tong (ASU), **Yanghua Xiao** (Fudan), **Wei Fan** (Baidu)

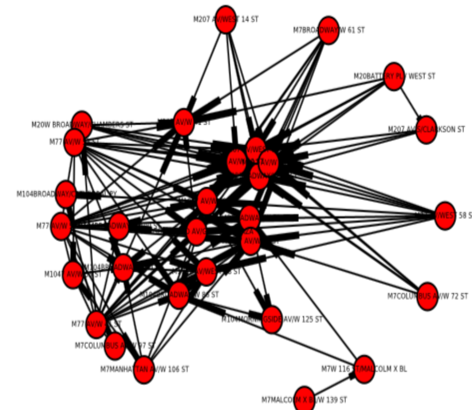
Graphs are Everywhere



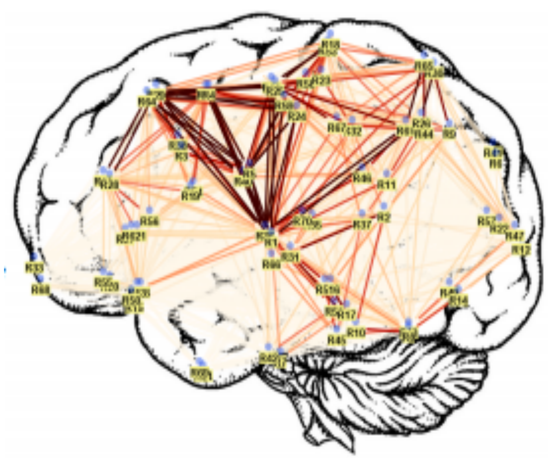
Collaboration Networks



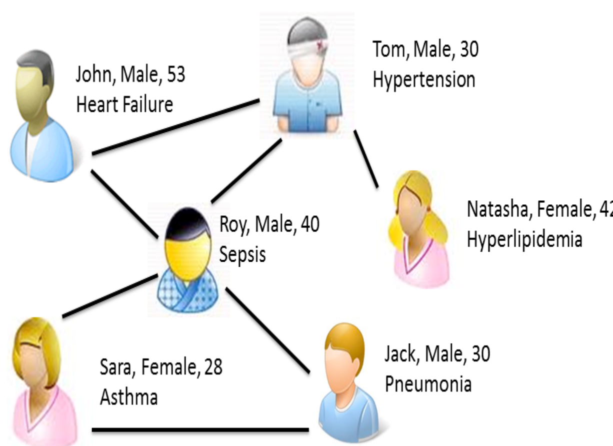
US Power Grid



Bus Network



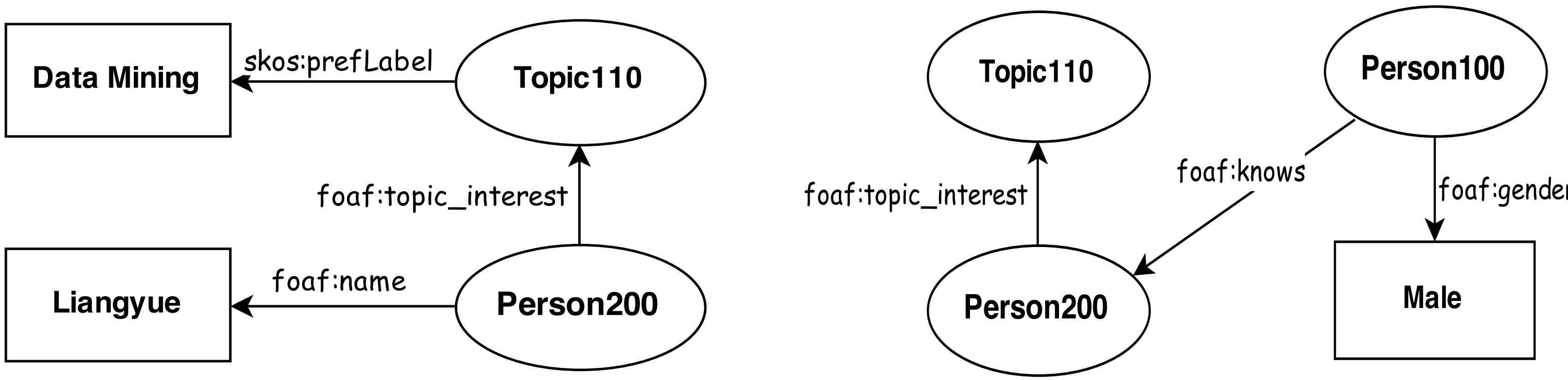
Brain Networks



Patient Networks

Hospital Networks

Application 1: Web Mining



Q: How similar are the two graphs?

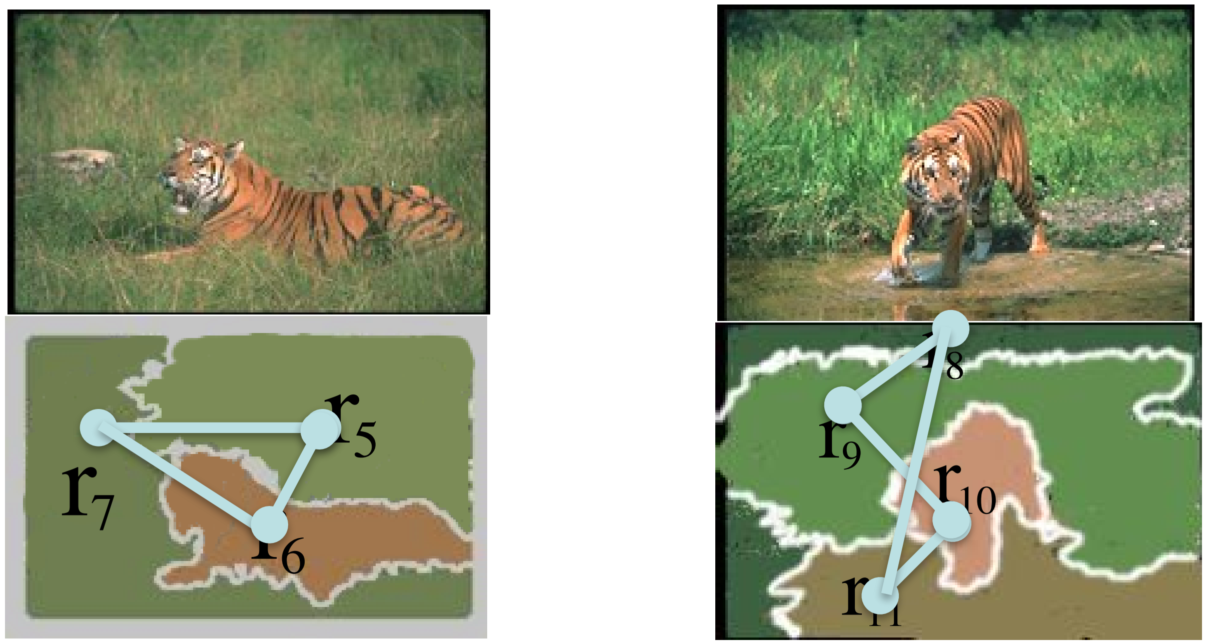
A: Graph Kernel

1. For each entity, construct a neighborhood graph by breadth-first search up to depth k

2. Apply graph kernel in kernel based learning methods

Lösch, Uta, Stephan Bloehdorn, and Achim Rettinger. "Graph kernels for RDF data." The Semantic Web: Research and Applications. Springer Berlin Heidelberg, 2012.

Application 2: Computer Vision



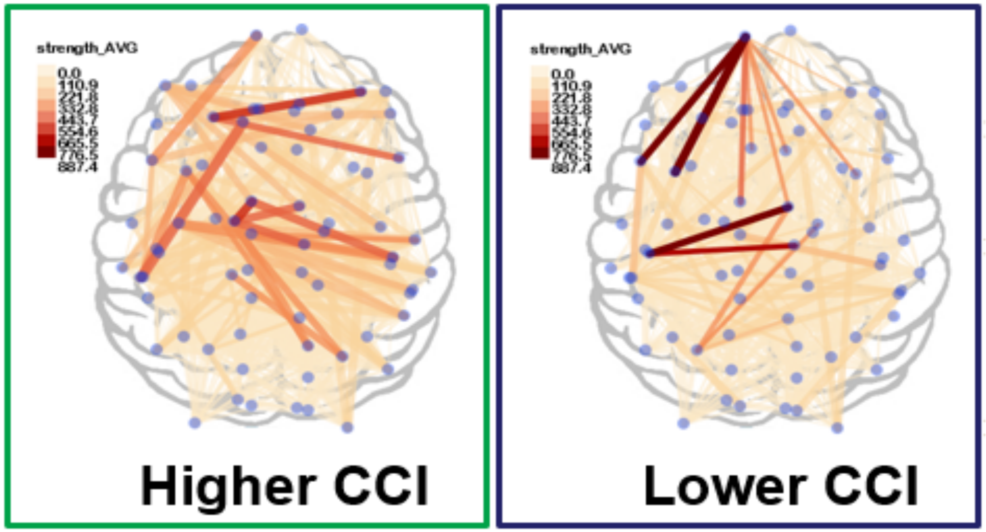
Q: How similar are the two graphs?

A: Graph Kernel

1. For each image, represent it as a segmentation graph
2. Apply graph kernel in kernel based learning methods

Harchaoui, Zaïd, and Francis Bach. "Image classification with segmentation graph kernels." CVPR 2007.

Application 3: Neuroscience



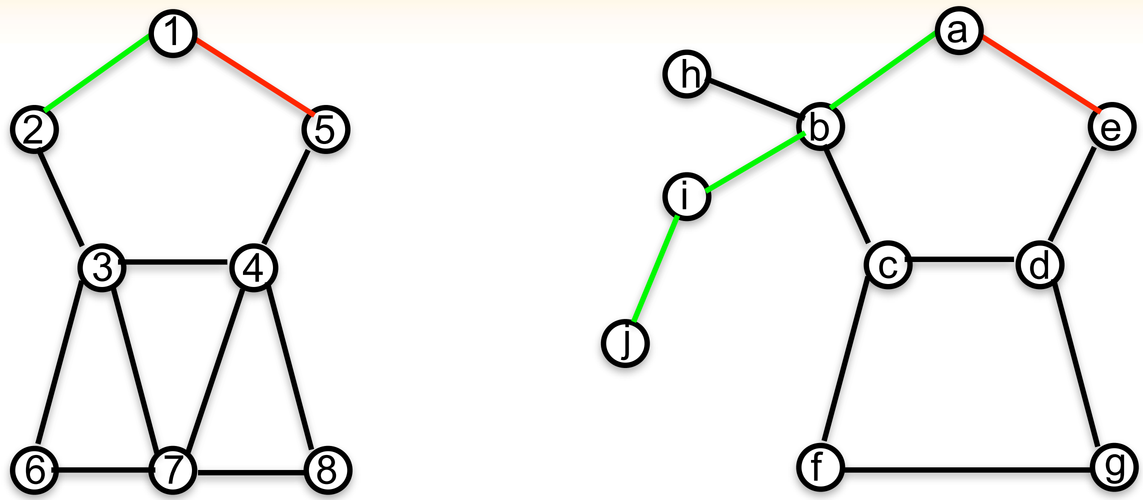
Q: How similar are the two graphs?

A: Graph Kernel

1. For each brain image, represent it as a graph
2. Apply graph kernel in kernel based learning methods

L. Shi, H. Tong, X. Mu: BrainQuest: Perception-Guided Visual Brain Comparison, 2015.

Random Walk based Graph Kernel



Graph 1 **Graph 2**

Intuitions:

1. Compare similarity of every pair of nodes from each graph

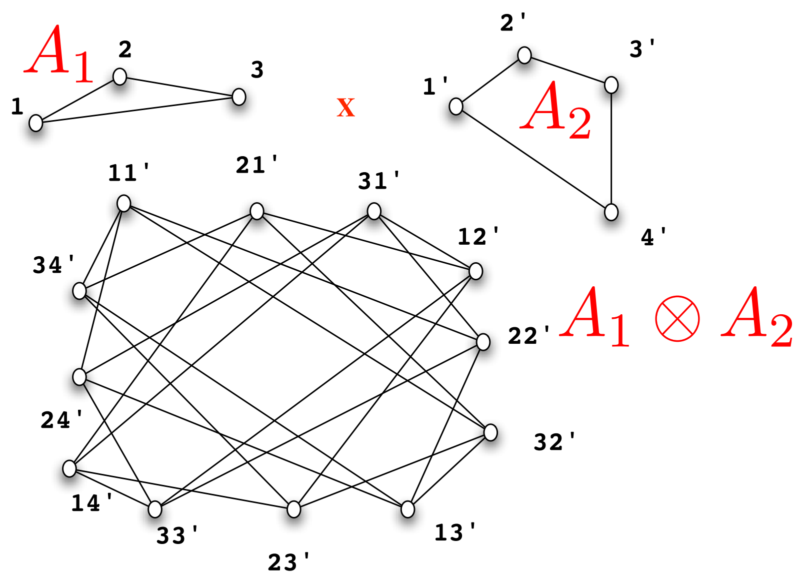
— Eg: (1,2) vs (a, j) → less similar

(1,5) vs (a,e) → more similar

2. Node pair similarity is measured by random walks
3. Two graphs are similar if they share many similar node pairs

Kronecker Product Graph

Graph Illustration



Matrix Description

$$A_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$A_1 \otimes A_2 = \begin{bmatrix} 0&0&0&0&0&1&0&1&0&1&0&1 \\ 0&0&0&0&1&0&1&0&1&0&1&0 \\ 0&0&0&0&0&1&0&1&0&1&0&1 \\ 0&0&0&0&1&0&1&0&1&0&1&0 \\ 0&1&0&1&0&0&0&0&0&1&0&1 \\ 1&0&1&0&0&0&0&0&1&0&1&0 \\ 0&1&0&1&0&0&0&0&0&1&0&1 \\ 1&0&1&0&0&0&0&0&1&0&1&0 \\ 0&1&0&1&0&1&0&1&0&0&0&0 \\ 1&0&1&0&1&0&1&0&0&0&0&0 \\ 0&1&0&1&0&1&0&1&0&0&0&0 \\ 1&0&1&0&1&0&1&0&0&0&0&0 \end{bmatrix}$$

Kronecker product

One Random Walk on A_1

$+$

One Random Walk on A_2

$=$ One Random Walk on $A_1 \otimes A_2 = A_\times$

S. V. N. Vishwanathan, Nicol N. Schraudolph, Imre Risi Kondor, and Karsten M. Borgwardt. Graph Kernels. Journal of Machine Learning Research, 11:1201–1242, April 2010.

RWR Graph Kernel — Formulation

Taking expectations instead of summing

$$
\begin{aligned}
\mathrm{Ker}(G_1, G_2) &= \textstyle\sum_k c^k q_\times' A_\times^k p_\times \\
&= q_\times' (I - cA_\times)^{-1} p_\times
\end{aligned}
$$

Computational challenge:

- $A_\times$ is of size $n^2 \times n^2$
- $O(n^6)$ (Direct computation) or $O(n^3)$ (Sylvester equation)

$\boxed{\text{Time > 1h, n=3328}}$

S. V. N. Vishwanathan, Nicol N. Schraudolph, Imre Risi Kondor, and Karsten M. Borgwardt. Graph Kernels. Journal of Machine Learning Research, 11:1201–1242, April 2010.

Speed up — ARK

Idea: perform low-rank approx on both graphs

$$\boxed{\mathrm{Ker}(\mathbf{G_1},\mathbf{G_2}) = (\mathbf{q_1}'\otimes\mathbf{q_2}')(\mathbf{I}-c\mathbf{A_1}'\otimes\mathbf{A_2}')^{-1}(\mathbf{p_1}\otimes\mathbf{p_2})}$$

Step 1:
Top-r low-rank approx:

$$\mathbf{U_1\Lambda_1V_1'} \qquad \mathbf{U_2\Lambda_2V_2'}$$

Step 2:
Matrix-inverse Lemma:

$$\mathrm{Ker}(\mathbf{G_1},\mathbf{G_2}) \approx (\mathbf{q_1'p_1})(\mathbf{q_2'p_2}) + c(\mathbf{q_1'U_1}\otimes\mathbf{q_2'U_2})\tilde{\mathbf{\Lambda}}(\mathbf{V_1'p_1}\otimes\mathbf{V_2'p_2})$$

$$\tilde{\mathbf{\Lambda}} = ((\mathbf{\Lambda_1}\otimes\mathbf{\Lambda_2})^{-1} - c(\mathbf{V_1'}\otimes\mathbf{V_2'})(\mathbf{U_1}\otimes\mathbf{U_2}))^{-1}$$

- **Matrix of size $r^2 \times r^2$, easy to inverse**
- **Overall complexity:** $O(n^2r^4 + mr + r^6)$

Time = 7.5s, n=3328

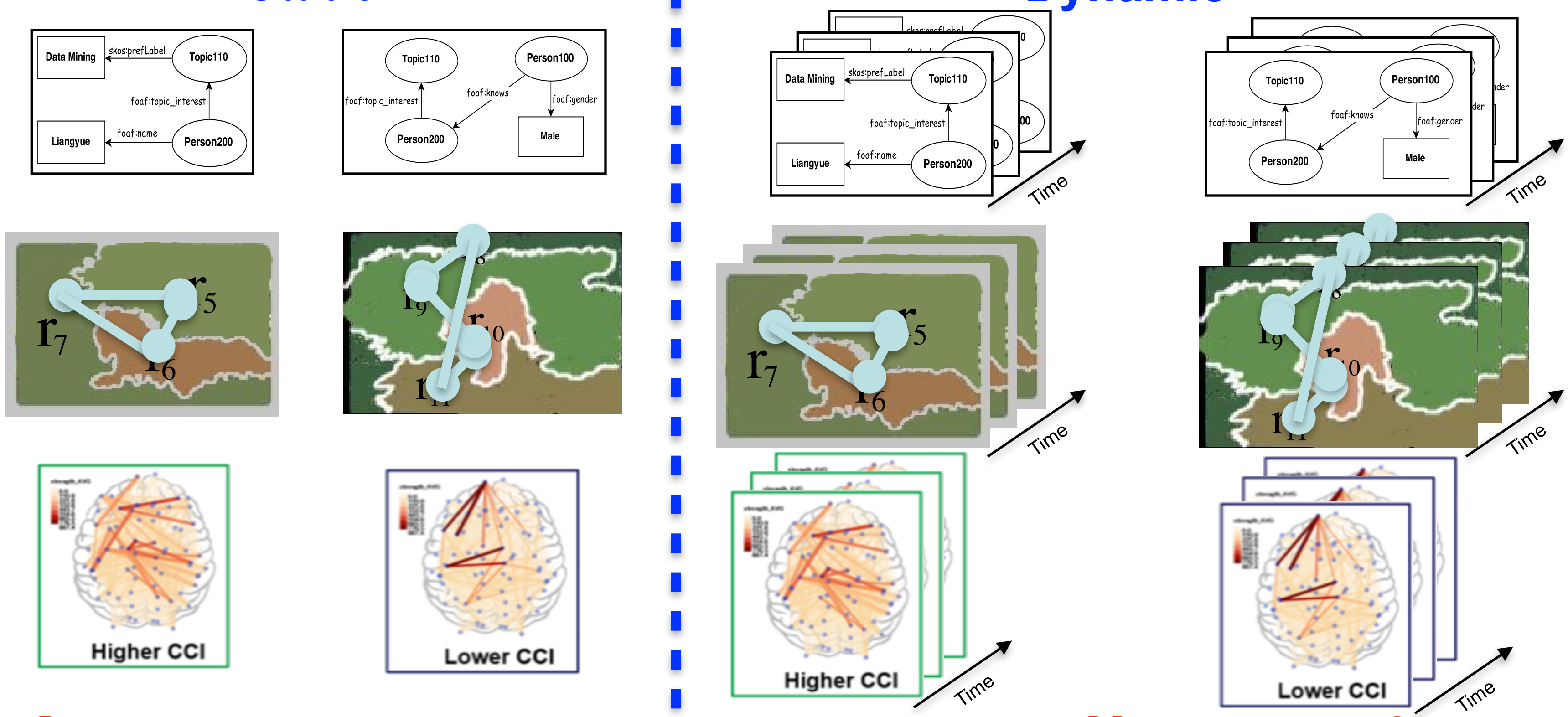
Can be reduced to $O(nr^2 + mr + r^6)$

U. Kang, Hanghang Tong, Jimeng Sun. Fast Random Walk Graph Kernel. SDM 2012

Challenges

- ARK: Good for static graphs
- What if graphs are *evolving* over time

Static **Dynamic**



Q: How to track graph kernel efficiently?

Roadmap

- Motivation
- *Cheetah-D* for Directed Graphs
- Experimental Results
- Conclusion

Cheetah-D: graph kernel tracking

$$\boxed{\mathrm{Ker}(\mathbf{G_1}, \mathbf{G_2}) = (\mathbf{q_1}' \otimes \mathbf{q_2}')(\mathbf{I} - c\mathbf{A_1}' \otimes \mathbf{A_2}')^{-1}(\mathbf{p_1} \otimes \mathbf{p_2})}$$

top-r low-rank approx: $\mathbf{U_1 \Lambda_1 V_1'}$ $\mathbf{U_2 \Lambda_2 V_2'}$

$\mathbf{\Delta A_1}$ $\mathbf{\Delta A_2}$

Ideas:

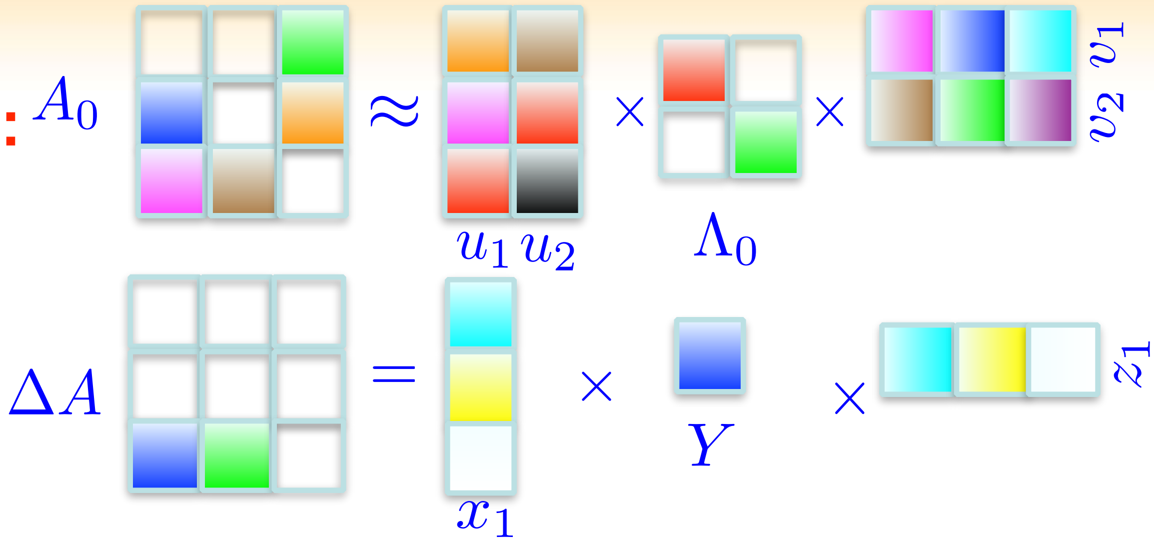
Avoid: re-computing low-rank approx [ARK]

Goal: track low-rank structure efficiently [Cheetah-D]

(SVD in this paper)

Step 0: Low rank approx on ΔA

➡Intuition:



$(m = 5, r = 2)$

$(m' = 2, r' = 1)$

➡Details: $A_0 \approx U_0 \Lambda_0 V_0'$

$\Delta A = XYZ'$

$$\boxed{\begin{aligned} A &= A_0 + \Delta A \\ &= [U_0\ X] \begin{bmatrix} \Lambda_0 & 0 \\ 0 & Y \end{bmatrix} [V_0\ Z]' \end{aligned}}$$

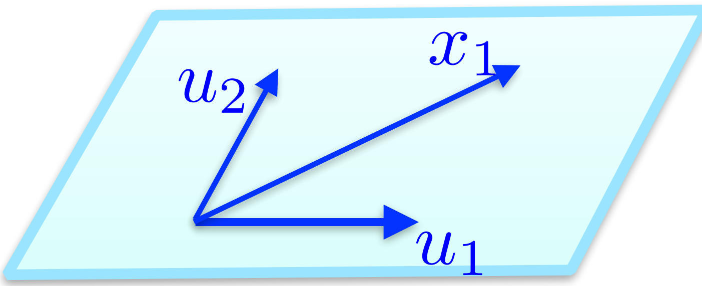
➡Property:

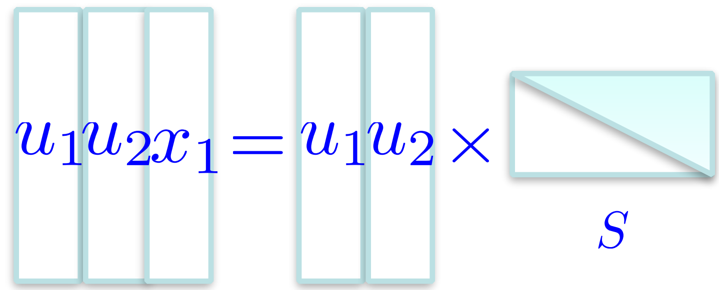
- SVD on A_0 takes: $O(mr + nr)$
- SVD on ΔA takes: $O(m'r' + nr') \ll O(mr + nr)$

  $[m' \ll m, r' \ll r]$

Step 1: Partial QR Decomposition

➡Intuition: **Case 1:** $x_1 \in \text{span}(u_1, u_2)$





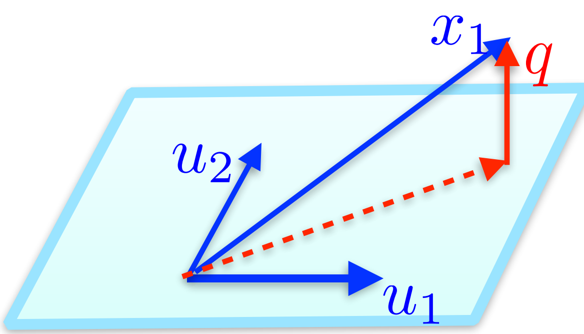
➡Details: $[U_0 \;\; X] = U_0 S$

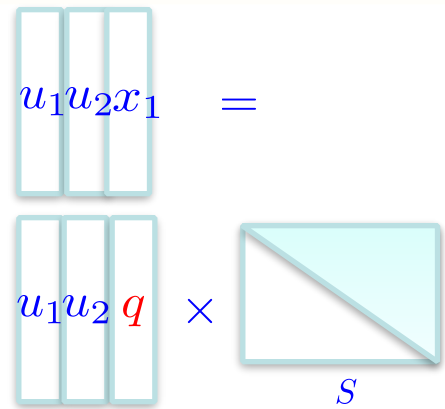
$$
\begin{aligned}
A &= [U_0 \;\; X] \begin{bmatrix} \Lambda_0 & 0 \\ 0 & Y \end{bmatrix} [V_0 \;\; Z]' \\
&= U_0 S \begin{bmatrix} \Lambda_0 & 0 \\ 0 & Y \end{bmatrix} T' V_0'
\end{aligned}
$$

➡Property:
- **Efficiency**: takes $O(nr'^2)$ $[r' \ll r]$
- **Effectiveness**: **No extra error**

Step 1: Partial QR Decomposition

➡**Intuition:** **Case 2:** $x_1 \notin \mathrm{span}(u_1, u_2)$





➡**Details:** $[U_0 \;\; X] = [U_0 \;\; q]S$

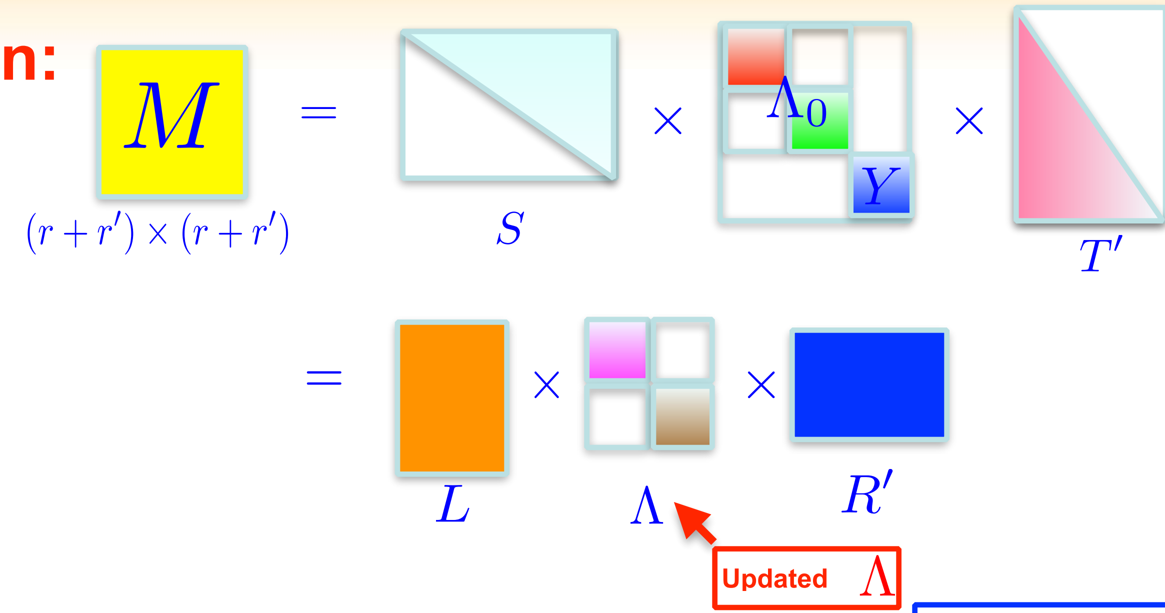
$$\begin{aligned} A &= [U_0 \;\; X] \begin{bmatrix} \Lambda_0 & 0 \\ 0 & Y \end{bmatrix} [V_0 \;\; Z]' \\ &= [U_0 \;\; \Delta Q] S \begin{bmatrix} \Lambda_0 & 0 \\ 0 & Y \end{bmatrix} T'[V_0 \;\; \Delta Z]' \end{aligned}$$

Similar Partial QR decomp on Z

➡**Property:**
- **Efficiency**: takes $O(nr'^2)$ $[r' \ll r]$
- **Effectiveness**: **No extra error**

Step 2: Full SVD on a Small Matrix

➡Intuition:



➡Details:

$$M = S \begin{bmatrix} \Lambda_0 & 0 \\ 0 & Y \end{bmatrix} T' = L \Lambda R'$$

$$\boxed{\begin{aligned} A &= [U_0 \ \Delta Q] S \begin{bmatrix} \Lambda_0 & 0 \\ 0 & Y \end{bmatrix} T' [V_0 \ \Delta Z]' \\ &= [U_0 \ \Delta Q] L \Lambda R' T' [V_0 \ \Delta Z]' \end{aligned}}$$

➡Property:

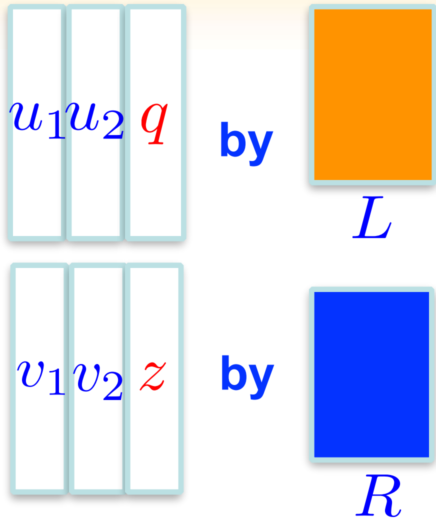
- **Efficiency**: takes $O((r+r')^3)$ $[r' \ll r]$
- **Effectiveness**: **No extra error**

Step 3: Rotate Orthonormal Basis

➡**Intuition:**

Rotate $u_1 u_2\ q$ **by** L

Rotate $v_1 v_2\ z$ **by** R



➡**Details:**

$$U = [U_0\ \Delta Q]L$$
$$V = [V_0\ \Delta Z]R$$

➡**Property:**

- **Complexity:** $O(nr^2)$
- **Overall SVD Update Complexity:** $O(nr^2 + m'r' + nr'^2)$

Re-compute SVD: $O(nr^2 + mr)$

DATA Lab

Analysis and Variants

- Time complexity of *Cheetah-D*: $O(nr^2 + nr'^2 + r^6)$

  ARK: $O(n^2r^4 + mr + r^6)$
- Comparison Example $(n = 3328, r = 500, r' = 5)$
  - ARK:7.5s
  - **Ours:0.4s**
- Variants
  - Undirected graphs
  - Attributed graphs

***Cheetah-D* Algorithm Sketch**

t = *1*, **Initialize** SVD of A1 and A2

for t=2,3,…
    Update SVD for A1 $\leftarrow O(n(r^2 + r'^2))$
    Update SVD for A2 $\leftarrow O(n(r^2 + r'^2))$
    Update Ker(A1,A2) $\leftarrow O(nr^2 + r^6)$
end

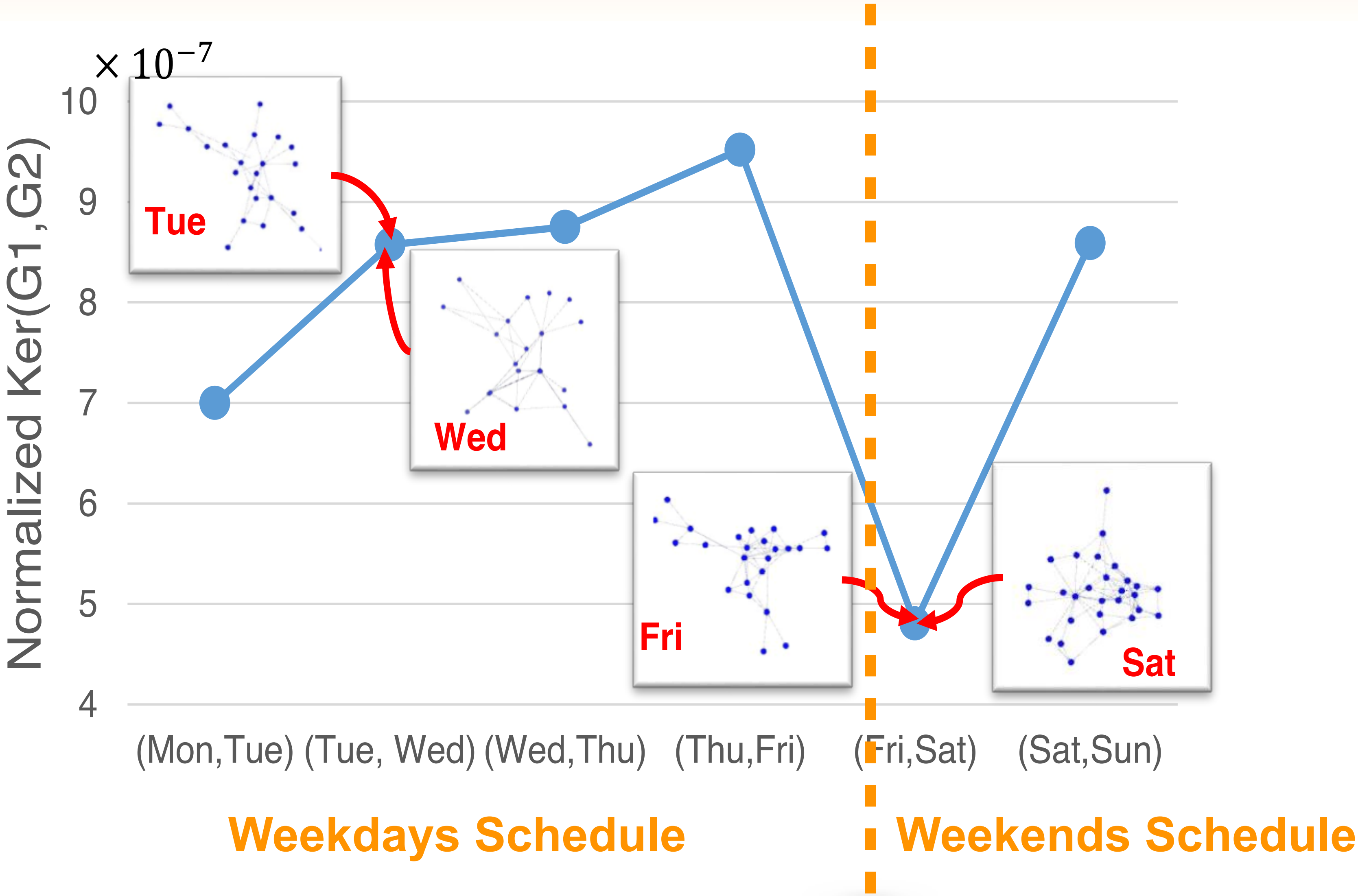
Roadmap

- Motivation
- *Cheetah-D* for Directed Graphs
- Experimental Results
- Conclusion

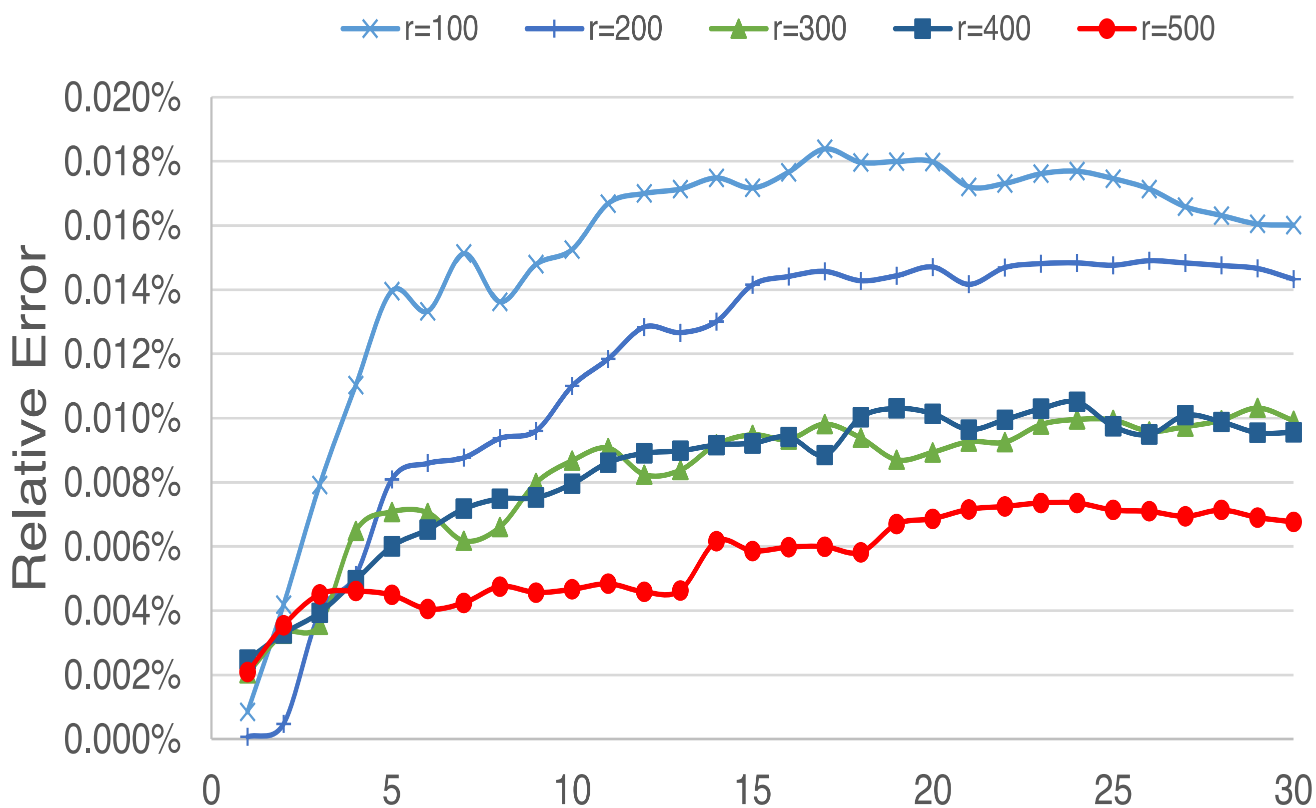
Case Study — MTA Bus Traffic

- **Graph construction**
  - Monitor traffic volume of 30 bus stops on 3 routes, from Monday, 03/24/2014 — Sunday, 03/30/2014
  - Represent each stop as a time series where each timestamp is traffic volume within each hour
  - On each day, build a graph for the 30 stops using Granger causality test
- **Graph kernel computation**
  - Graph kernel is computed between two graphs of two consecutive days

Case Study — MTA Bus Traffic



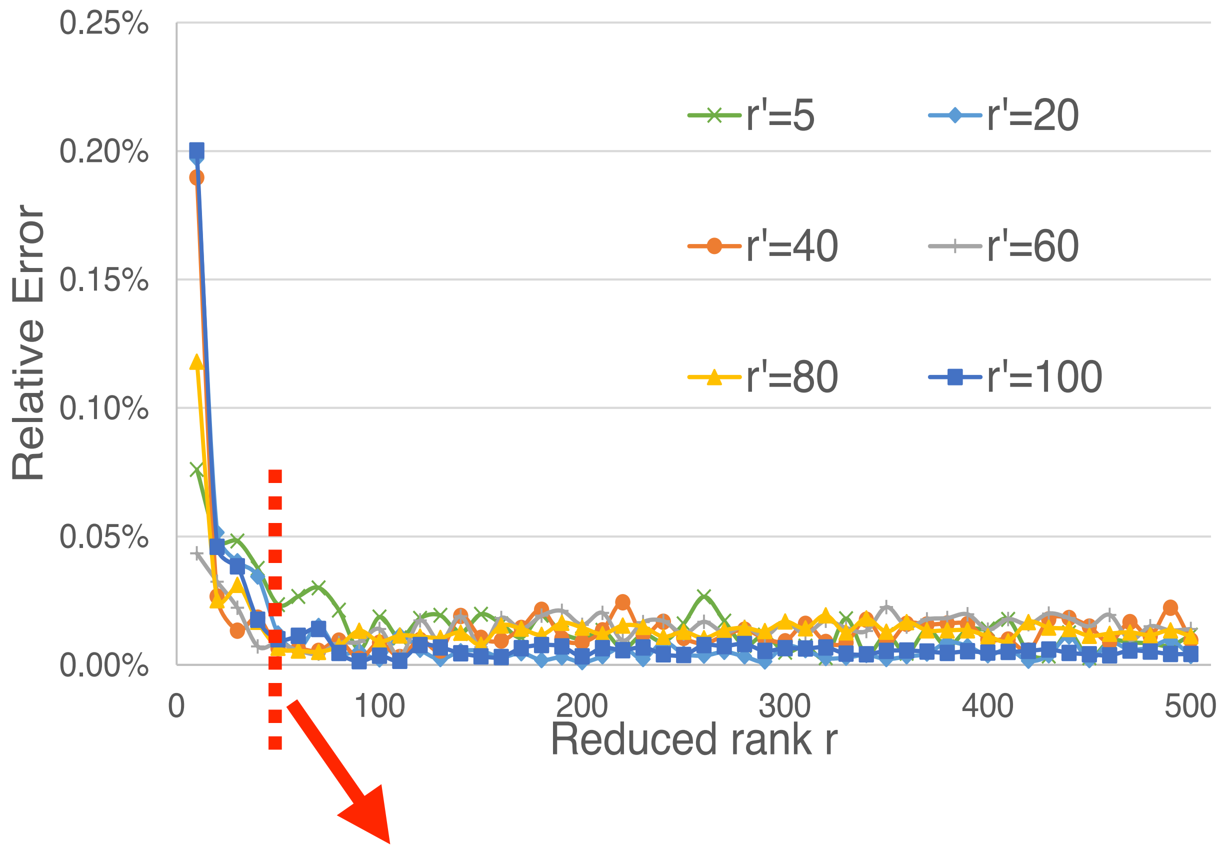
Relative Error



All cases, Err <0.02%

$\boxed{\bar{n} = 4183, \bar{m} = 5692}$

Avg Error vs. Rank

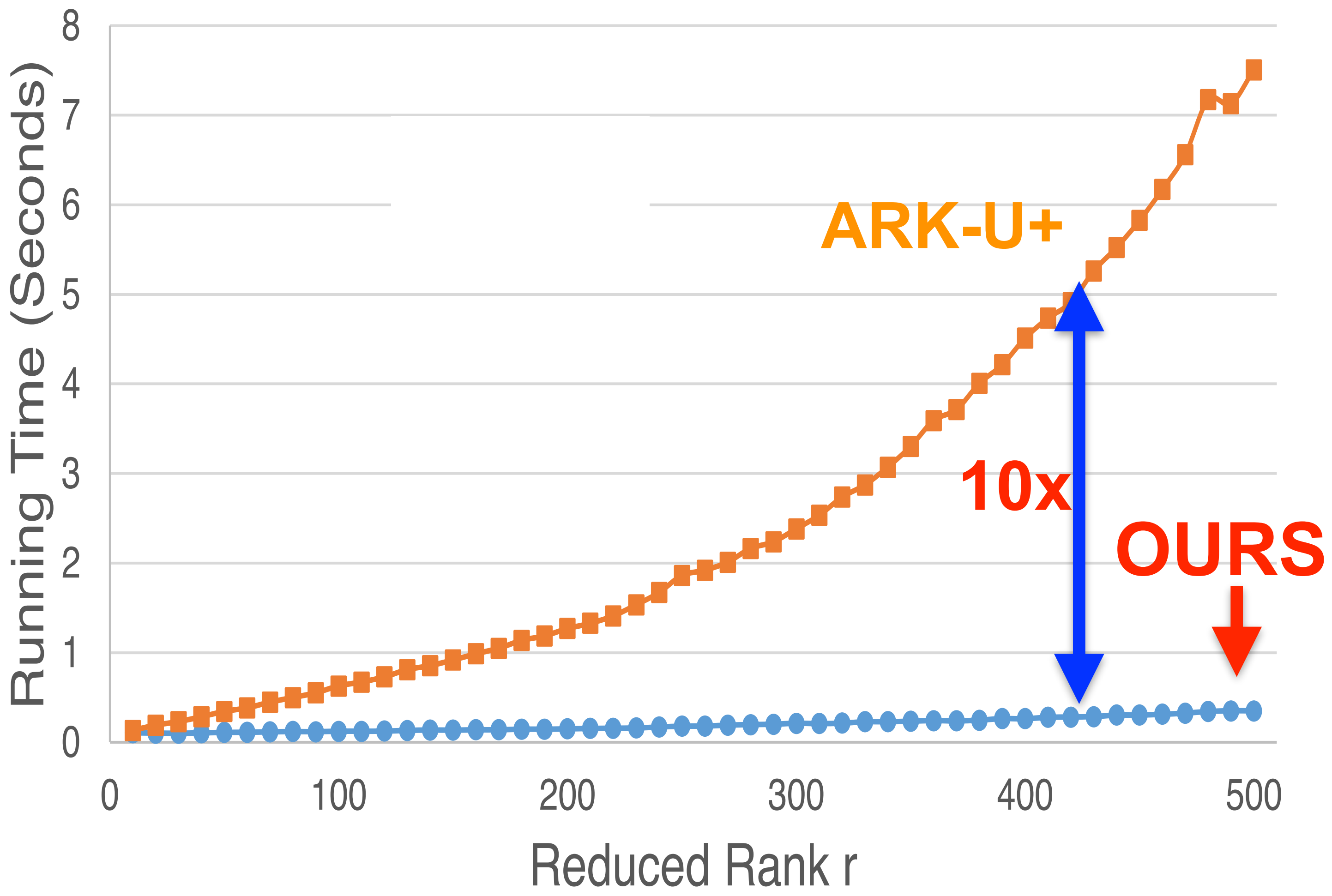


When r>50, Err<0.05%

$\boxed{\bar{n} = 4183, \bar{m} = 5692}$

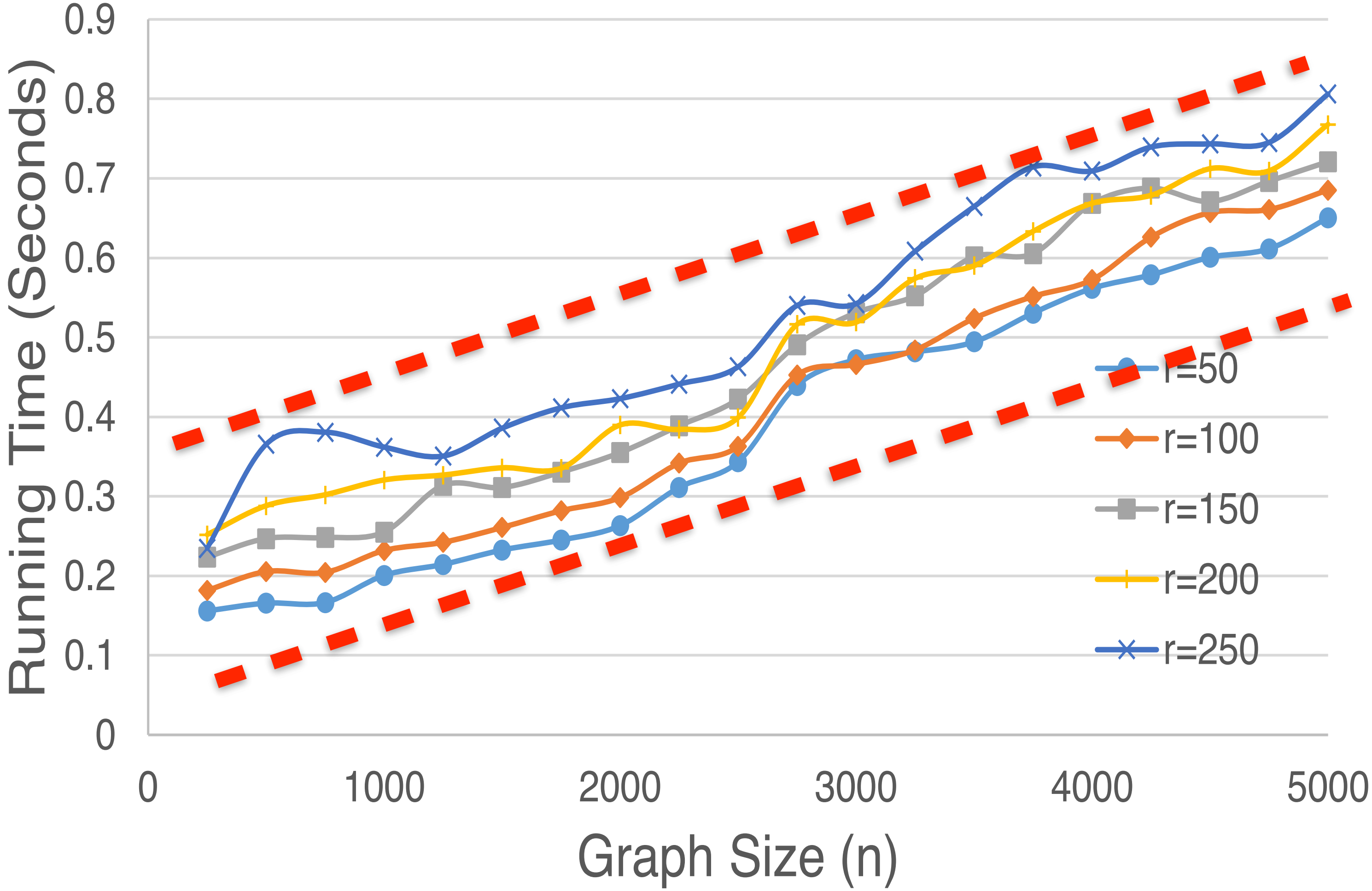
DATA Lab

Running Time vs. Rank



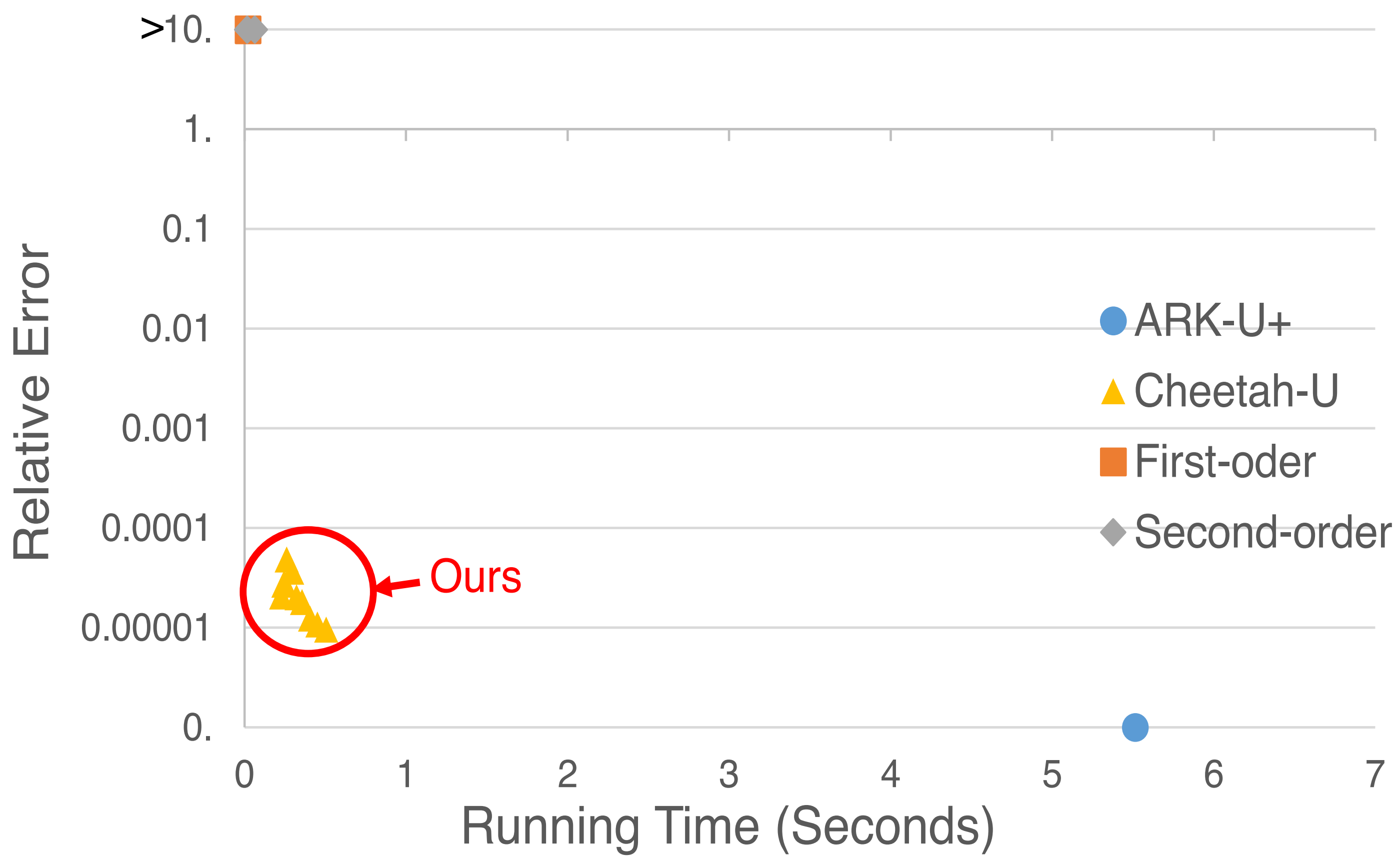
DATA Lab

Scalability



Scale near linearly

Quality vs. Speed



Roadmap

- Motivation
- *Cheetah-D* for Directed Graphs
- Experimental Results
- Conclusion

Conclusion

- **Goal**: track graph kernel of dynamic graphs
- **Our Solution**: *Cheetah-D*
  - **Key idea**: track low-rank approx
  - **Results**:
    - ★ Complexity: $O(nr^2 + nr'^2 + r^6)$
    - ★ In practice: ~15x faster, Err<0.05%
  - More in paper:
    - ★ Cheetah-U for undirected graphs
    - ★ Error bound analysis