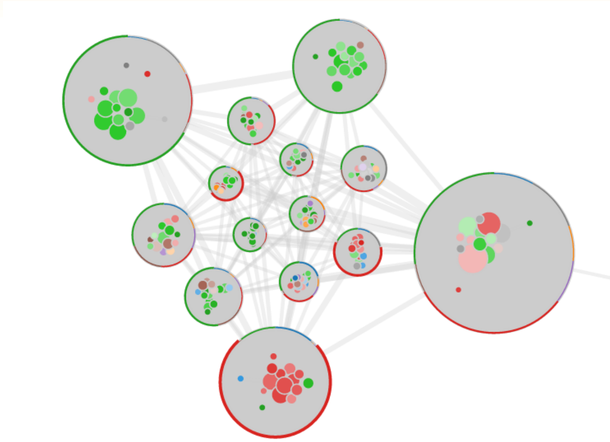
Fast Eigen-Functions Tracking on Dynamic Graphs

Chen Chen and Hanghang Tong

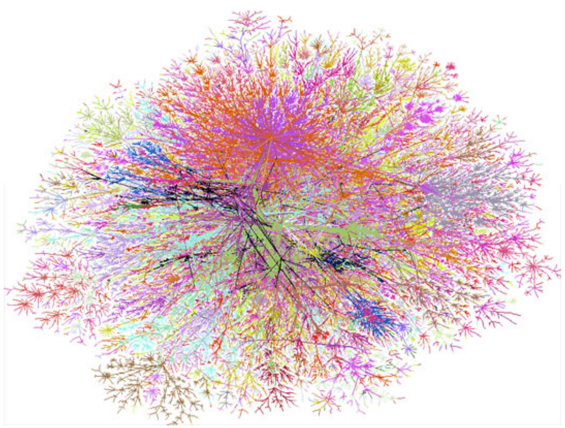
Arizona State University

Graphs are Ubiquitous!

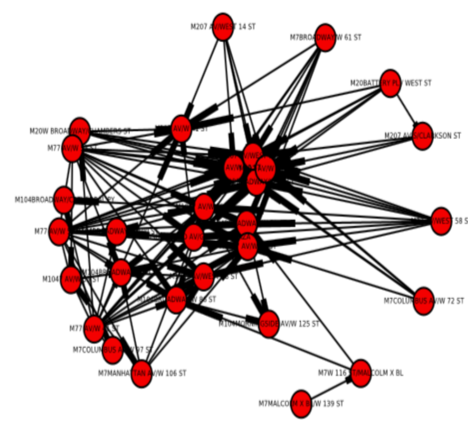


Collaboration Network

Hospital Network



Autonomous Network



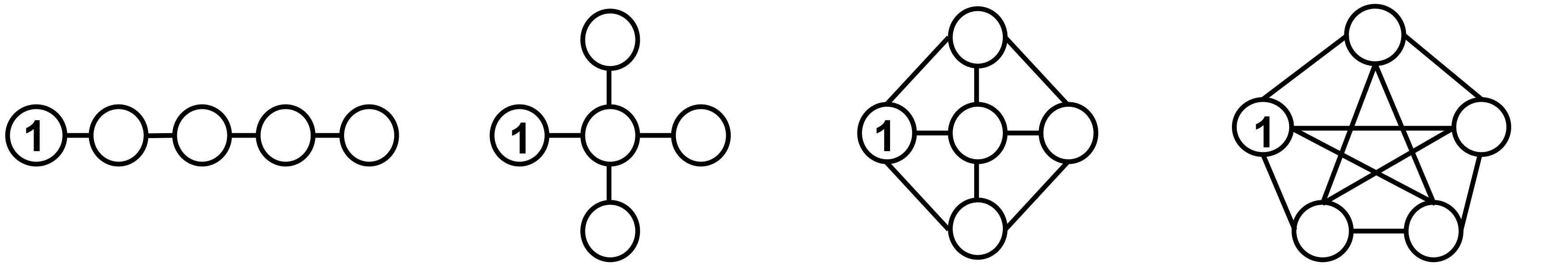
Transportation Network

Key Graph Parameters

- P1: Epidemic Threshold (Propagation network)
- P2: Centrality of nodes (All networks)
- P3: Clustering Coefficient (Social network)
- P4: Graph Robustness (Router/Transportation)

P1: Epidemic Threshold

- **Questions**: How easy is it to spread disease?
- **Intuition**



$\lambda = 1.7$ $\lambda = 2.0$ $\lambda = 2.9$ $\lambda = 4.0$

- **Solution**: Related to the leading eigenvalue (λ) of the adjacency matrix for **ANY** cascade model [ICDM 2011]

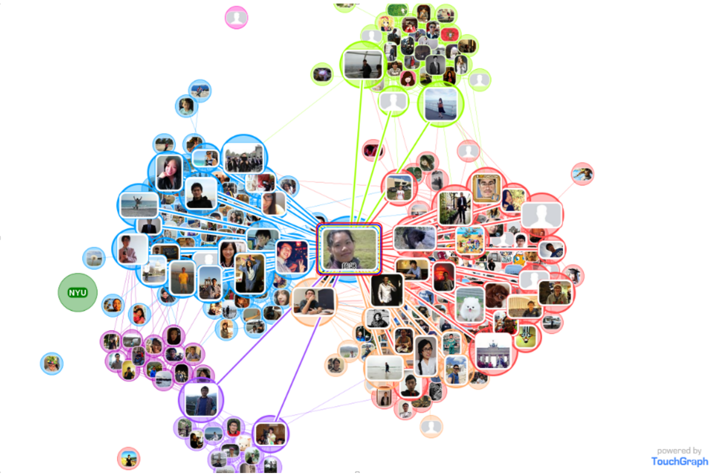
P2: Node Centrality

- **Question**: How important is a node?
- **Intuition**: Having more important friends are considered influential
- **Commonly used**: Eigenvector Centrality

The eigenvector corresponding to the leading eigenvalue $(Au = \lambda u)$

P3: Clustering Coefficient

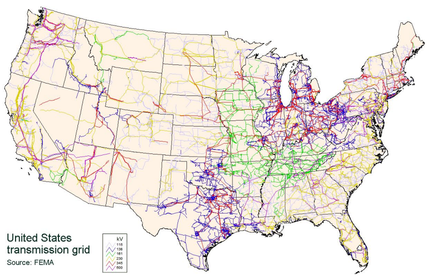
- **Question**: How the nodes in the graph cluster together?
- **Intuition**:



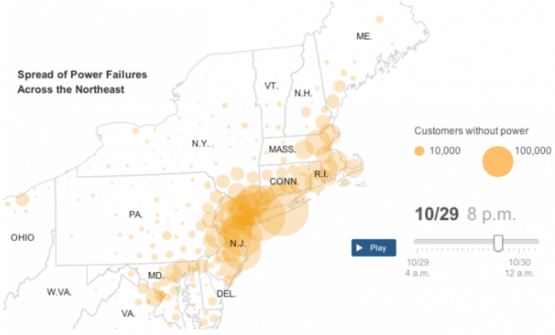
- **Solution**: $C = \dfrac{3 \times \text{number of triangles}}{\text{number of connected triplets}}$

P4: Graph Robustness

- **Question**: How robust is a graph under external attack?
- **Intuition**:



Power Grid
[wikipedia.com]

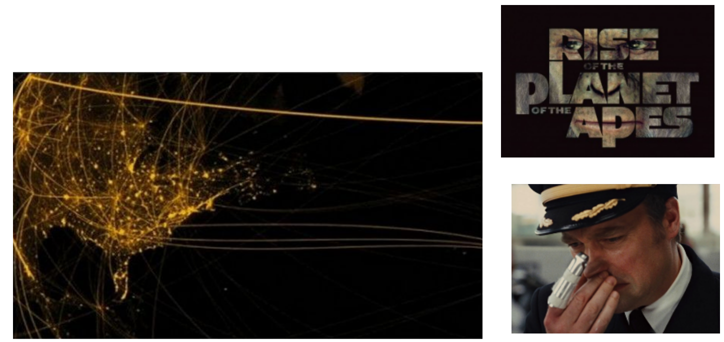


Sandy Aftermath
[forbes.com]

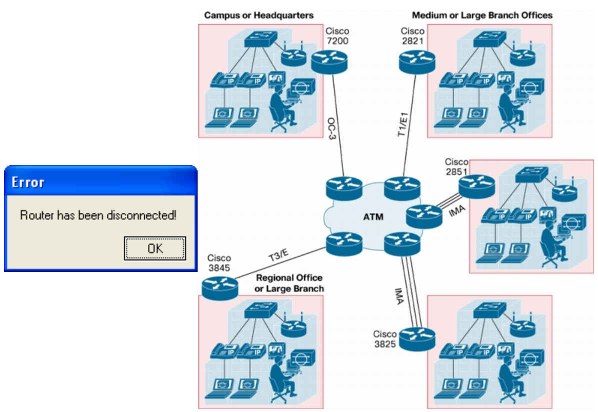
- **Solution**: $$S(G) = \ln\left(\frac{1}{k}\sum_{i=1}^{k} e^{\lambda_i}\right)$$ [SDM2014]

Challenge: Graphs are Dynamic!

Social Networks



Propagation Netoworks



Router Netoworks
[www.cisco.com]

Transportation Netoworks
[www.mapofworld.com]

How to track key graph parameters?

DATA Lab

Eigen-Function Tracking

- Q1. Track key graph parameters
- Q2. Estimate the error of tracking algorithms
- Q3. Analyze attribution for drastic changes

Roadmap

- Motivations
- Q1: Efficient tracking algorithms
- Q2: Error estimation methods
- Q3: Attribution analysis
- Conclusion

Key Graph Parameters

- Observations: P1-P4 are all eigen-functions

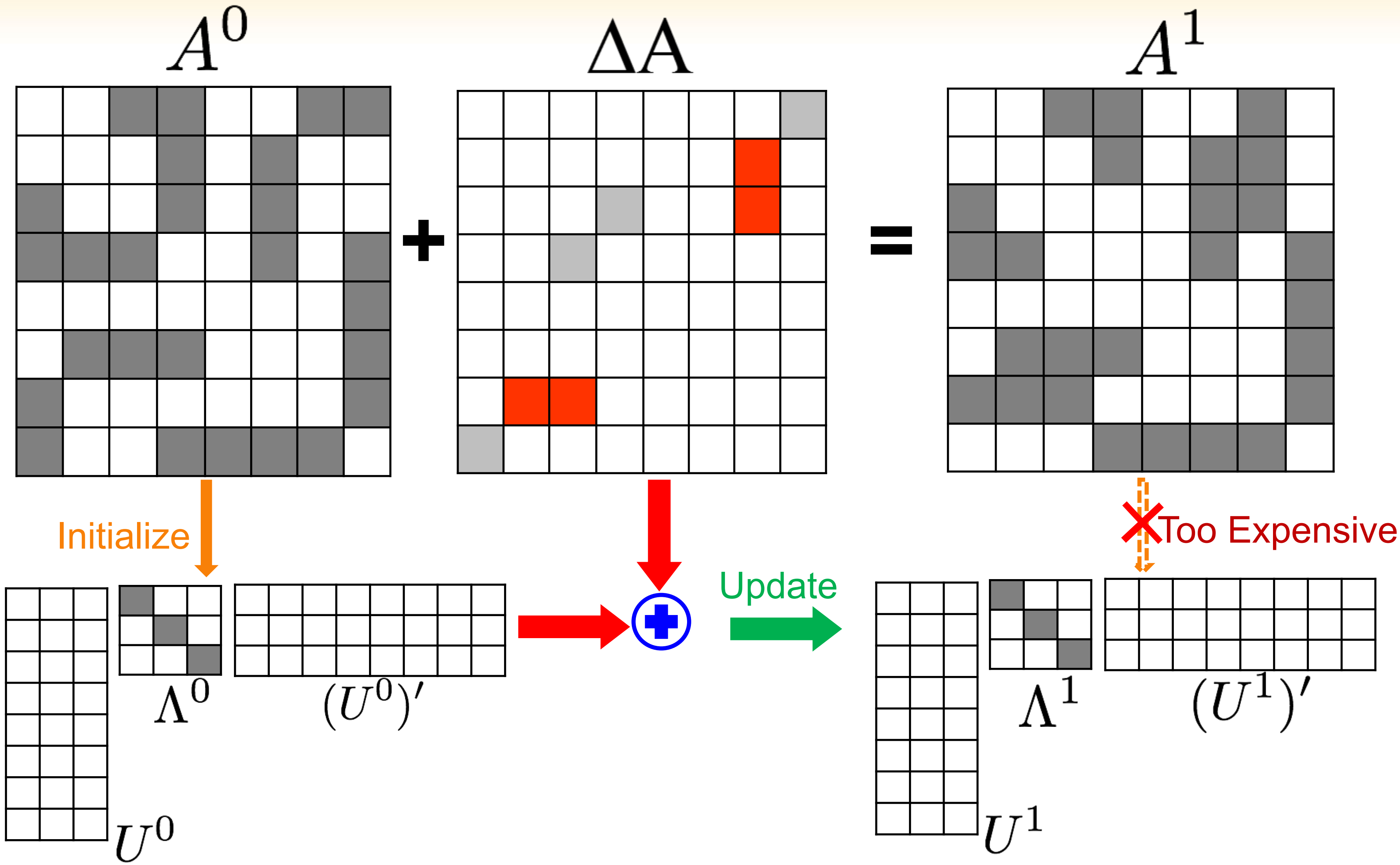
$$f((\Lambda_k, U_k)) = \begin{cases} \lambda_1 & \text{P1. Epidemic Threshold} \\ u_1 & \text{P2. Eigenvector Centrality} \\ \Delta(G) = \frac{1}{6}\sum_{i=1}^{k} \lambda_i^3 & \text{P3. Clustering Coefficient (Triangles)} \\ S(G) = \ln(\frac{1}{k}\sum_{i=1}^{k} e^{\lambda_i}) & \text{P4. Robustness Score} \end{cases}$$

Goal: Tracking Top Eigen-Pairs

- Method 1.
  - Calculate (Λ_k, U_k) **from scratch** whenever the structure changes
  - Lanczos algorithm $O(T(mk + nk^2))$ ✗

Too costly for fast-changing large graphs!

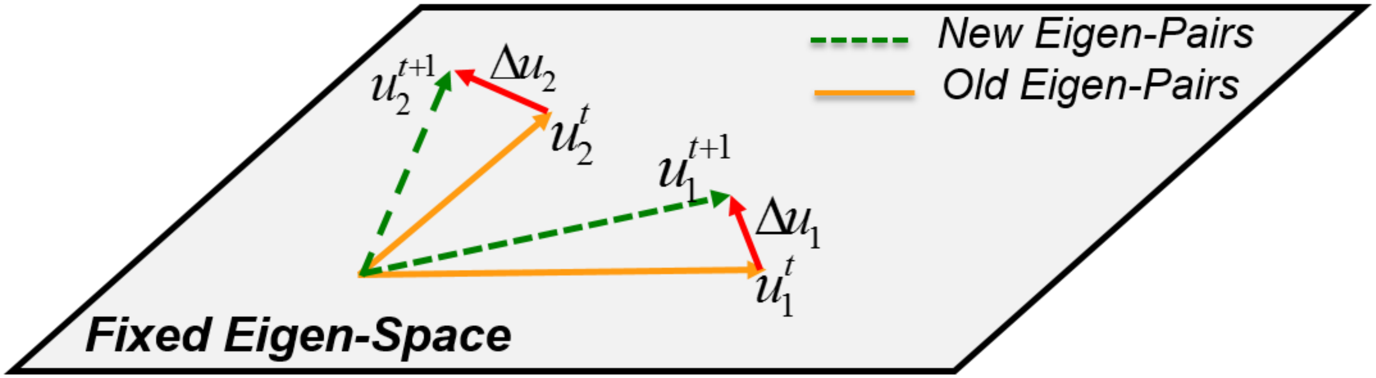
Key Idea



A^0
ΔA
A^1
Initialize
Too Expensive
Update
Λ^0
$(U^0)'$
U^0
Λ^1
$(U^1)'$
U^1

Key Idea: Incrementally Update

- Intuition:



- Solution: Matrix Perturbation Theory

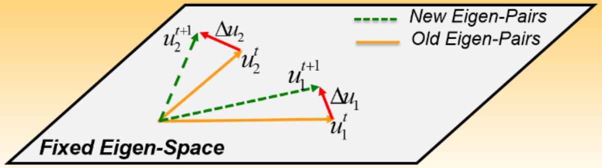
$$A^{t+1}u_1^{t+1} = \lambda_1^{t+1}u_1^{t+1}$$
$$A^{t+1}u_2^{t+1} = \lambda_2^{t+1}u_2^{t+1}$$

$$(A+\Delta A)(u_1+\Delta u_1) = (\lambda_1+\Delta\lambda_1)(u_1+\Delta u_1)$$
$$(A+\Delta A)(u_2+\Delta u_2) = (\lambda_2+\Delta\lambda_2)(u_2+\Delta u_2)$$

$$A^{t+1} = A^t + \Delta A$$
$$\lambda^{t+1} = \lambda^t + \Delta\lambda$$
$$u^{t+1} = u^t + \Delta u$$

Time stamp omitted for brevity.

Details: Step 1



$$(A + \Delta A)(u_1 + \Delta u_1) = (\lambda_1 + \Delta\lambda_1)(u_1 + \Delta u_1)$$

$$(A + \Delta A)(u_2 + \Delta u_2) = (\lambda_2 + \Delta\lambda_2)(u_2 + \Delta u_2)$$

$$Au_1 = \lambda_1 u_1 \quad Au_2 = \lambda_2 u_2$$

$$A\Delta u_1 + \Delta A u_1 + \Delta A \Delta u_1 = \lambda_1 \Delta u + \Delta\lambda_1 u_1 + \Delta\lambda_1 \Delta u_1$$

$$A\Delta u_2 + \Delta A u_2 + \Delta A \Delta u_2 = \lambda_2 \Delta u + \Delta\lambda_2 u_2 + \Delta\lambda_2 \Delta u_2$$

First order perturbation terms

High order perturbation terms

Challenge: two equation with four variables $\Delta\lambda_1 \;\; \Delta\lambda_2 \;\; \Delta u_1 \;\; \Delta u_2$

Solution: Introduce additional constraints and assumptions

$$u_i' u_i = 1 \quad u_i' u_j = 0 \, (\mathrm{i} \neq \mathrm{j})$$

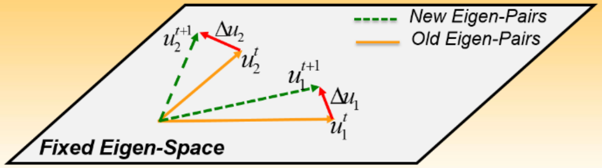
Constraints

$$\Delta u_1 = \alpha_{11} u_1 + \alpha_{12} u_2 \quad \Delta\lambda_i << \lambda_i$$

$$\Delta u_2 = \alpha_{21} u_1 + \alpha_{22} u_2 \quad \Delta u_i << u_i$$

Assumptions

Details: Estimate $\Delta\lambda_i$



- Discard high order term

$$A\Delta u_i + \Delta A u_i + \Delta A \Delta u_i = \lambda_i \Delta u + \Delta\lambda_i u_i + \Delta\lambda_i \Delta u_i \quad (1)$$

$\Delta\lambda_i << \lambda_i$, $\Delta u_i << u_i$

$$A\Delta u_i + \Delta A u_i = \lambda_i \Delta u_i + \Delta\lambda_i u_i \quad (2)$$

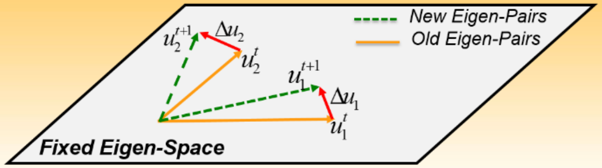
Multiply u_i' on both side

$$u_i' A\Delta u_i + u_i' \Delta A u_i = u_i' \lambda_i \Delta u_i + u_i' \Delta\lambda_i u_i \quad (3)$$

$Au_i = \lambda_i u_i$, $u_i' u_i = 1$

$$\Delta\lambda_i = u_i' \Delta A u_i$$

Estimate Δu_i (Option 1)



$$A\Delta u_i + \Delta A u_i = \lambda_i \Delta u_i + \Delta\lambda_i u_i$$ **(Discard high order)**

1. Multiply u_j' on both side

2. $$u_j' A\Delta u_i + u_j' \Delta A u_i = u_j' \lambda_i \Delta u_i + u_j' \Delta\lambda_i u_i$$

$Au_j = \lambda_j u_j, u_j' u_i = 0$

3. $$u_j' \Delta A u_i = (\lambda_i - \lambda_j) u_j' \Delta u_i$$

(Trip-Basic)

$$\Delta u_i = \sum_{p=1}^{k} \alpha_{ip} u_p$$

4. $$u_j' \Delta A u_i = (\lambda_i - \lambda_j)\left(\sum_{p=1, p\neq j}^{k} \alpha_{ip} u_j' u_p + \alpha_{ij} u_j' u_j\right)$$

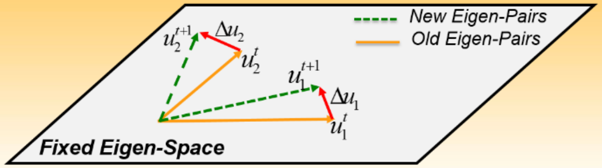
$u_i' u_j = 0, u_i' u_i = 1$

$$\alpha_{ij} = \frac{u_j' \Delta A u_i}{\lambda_i - \lambda_j}$$

Time Complexity: $O(Tk^2(s+n))$ $(s = |\Delta A|)$

Lanczos: $O(T(mk + nk^2))$

Estimate Δu_i (Option 2)



- Keep high order perturbation terms

$$A\Delta u_i + \Delta A u_i + \Delta A \Delta u_i = \lambda_i \Delta u + \Delta\lambda_i u_i + \Delta\lambda_i \Delta u_i$$

$$\Delta\lambda_i = u_i' \Delta A u_i, \Delta u_i = \sum_{j=1}^{k} \alpha_{ij} u_j$$

(Trip)

$$\alpha_i = (D - X)^{-1} X(:, i) \quad (\alpha_i = [\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ik}])$$

where $X = U_k' \Delta A U_k$, $D = diag(\lambda_i + \Delta\lambda_i - \lambda_j)$ $(j = 1, \ldots, k)$

(Trip-Basic) $\alpha_{ij} = \dfrac{u_j' \Delta A u_i}{\lambda_i - \lambda_j}$ ⟺ $\alpha_i = D^{-1} X(:, i)$

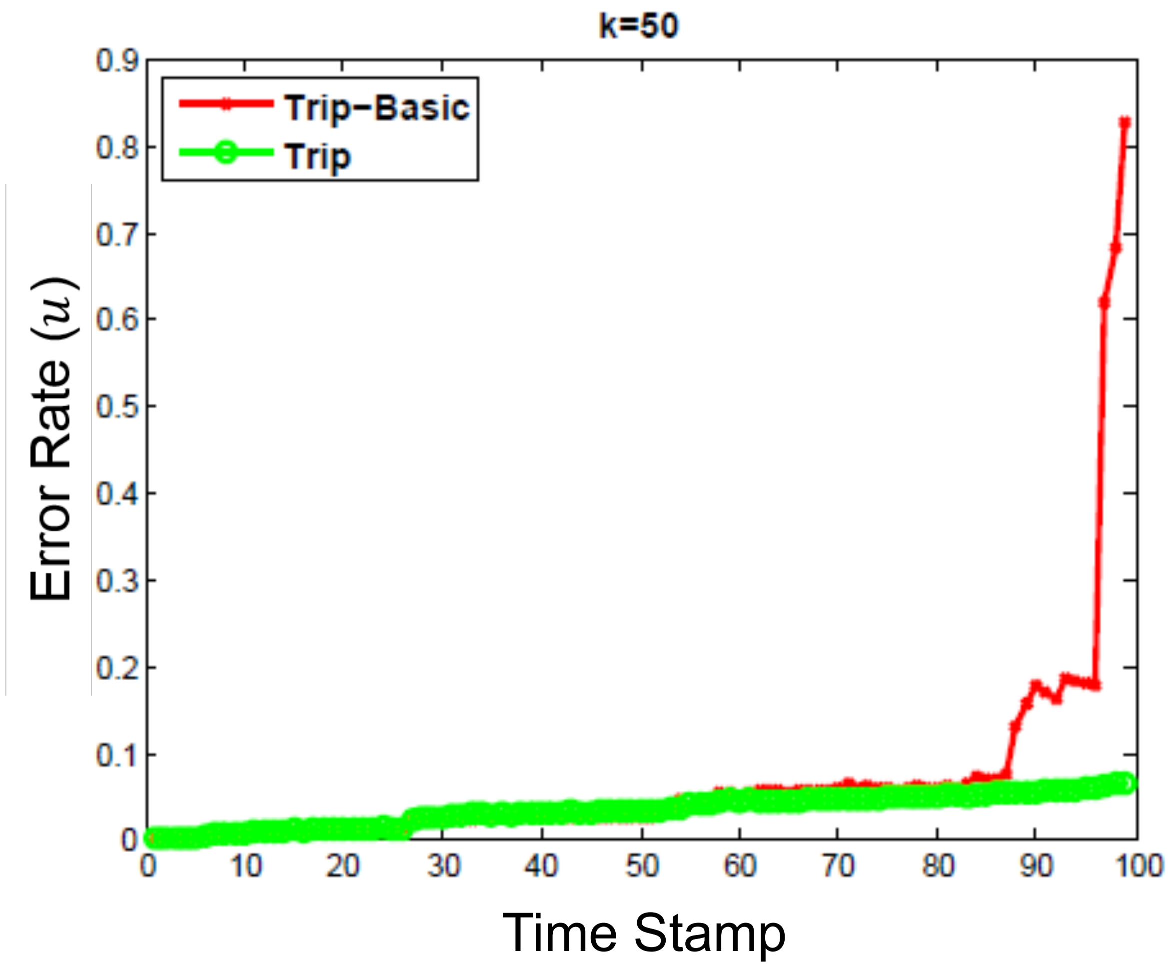
where $X = U_k' \Delta A U_k$, $D = diag(\lambda_i - \lambda_j)$ $(j = 1, \ldots, k)$

Time Complexity: $O(T(k^4 + k^2(n + s)))$ $(s = |\Delta A|)$

Evaluation

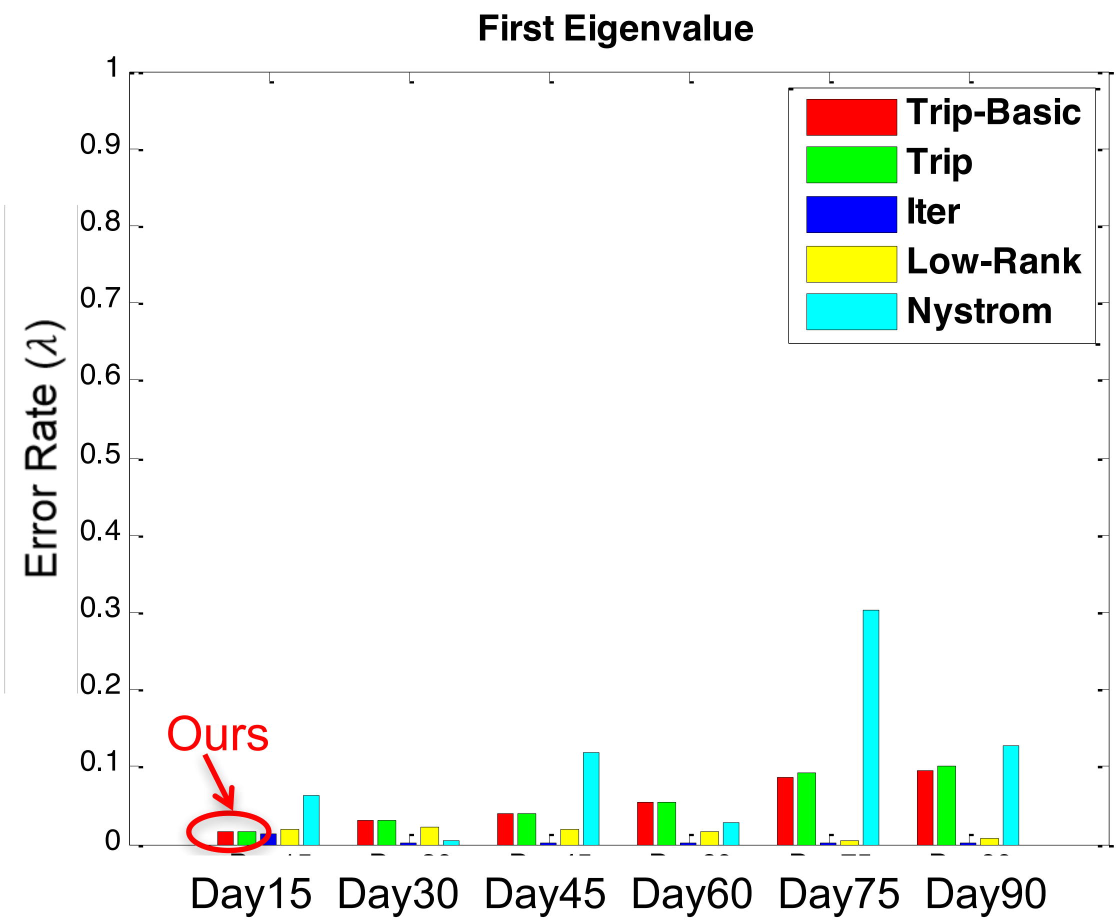
- Data set:
  - Autonomous systems AS-733 (https://snap.stanford.edu/data/as.html)
  - 100 days time spans
    - (11/08/1997-02/16/1998)
    - (03/15/1998-06/26/1998)
  - Maximum #nodes = 4,013
  - Maximum #edges = 14,399

Trip-Basic vs. Trip: Effectiveness

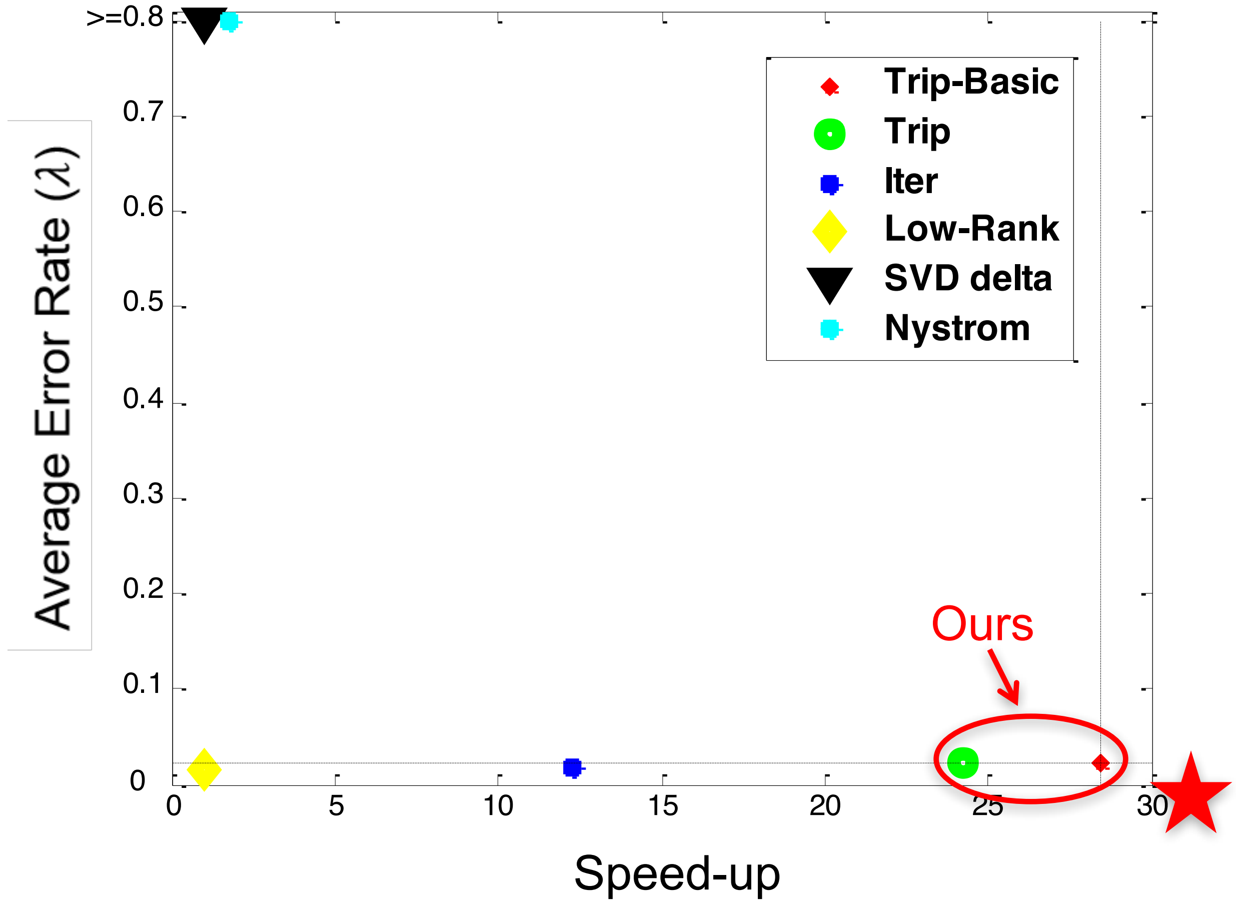


DATA Lab

Effectiveness Comparison



Effectiveness vs. Efficiency

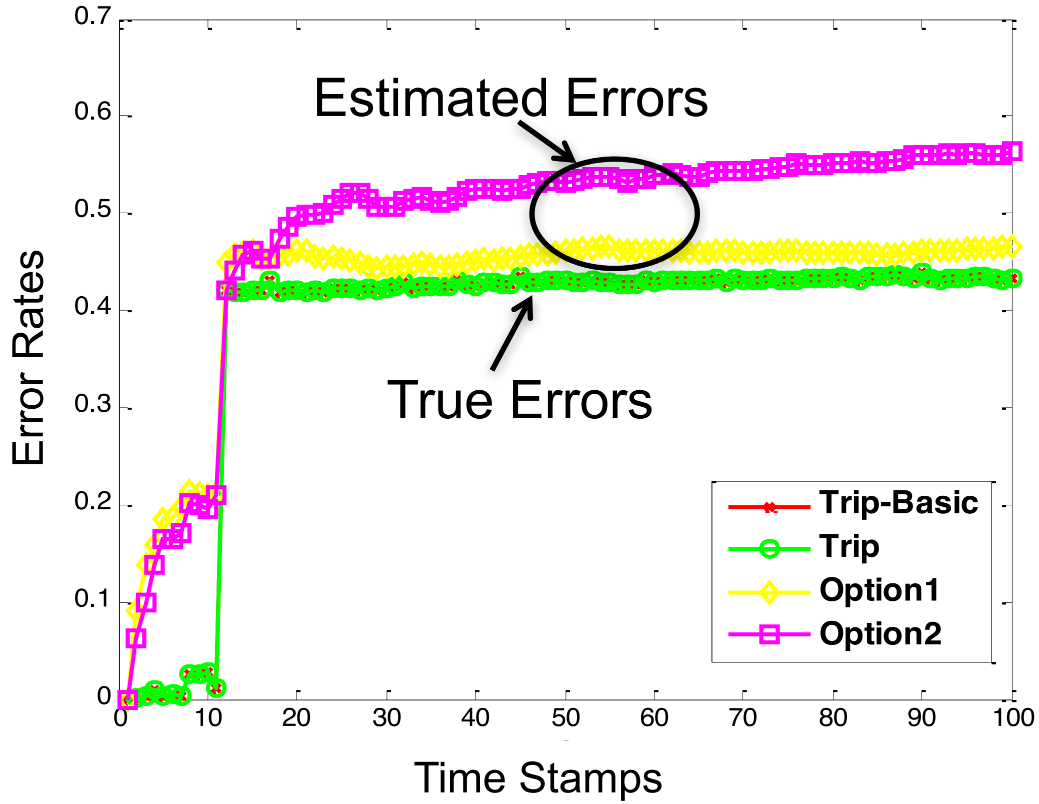


Roadmap

- Motivations
- Q1: Efficient tracking algorithms
- Q2: Error estimation methods
- Q3: Attribution analysis
- Conclusion

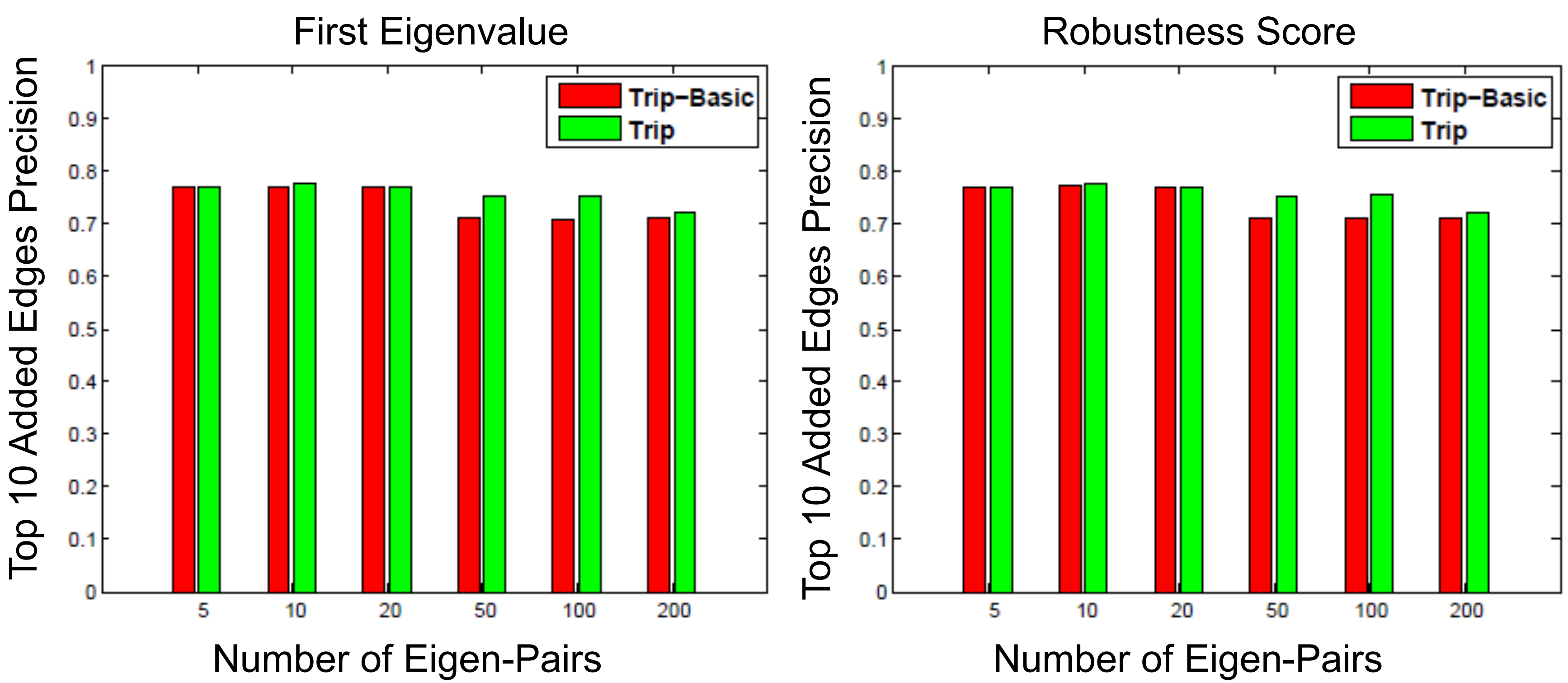
Q2.Error Estimation

- Setting: $c_1 = 0.005, c_2 = 0.025$

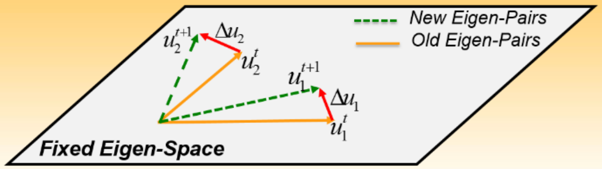


Q3. Attribution Analysis

- Precision (Edge Addition)



Conclusion



- Goal: Tracking key graph parameters
- Solutions:
  - Key idea:
    - Fixed eigen-space, Matrix perturbation theory
  - Algorithms: Trip-Basic, Trip
- More Details:
  - Error Estimation
  - Attribution Analysis