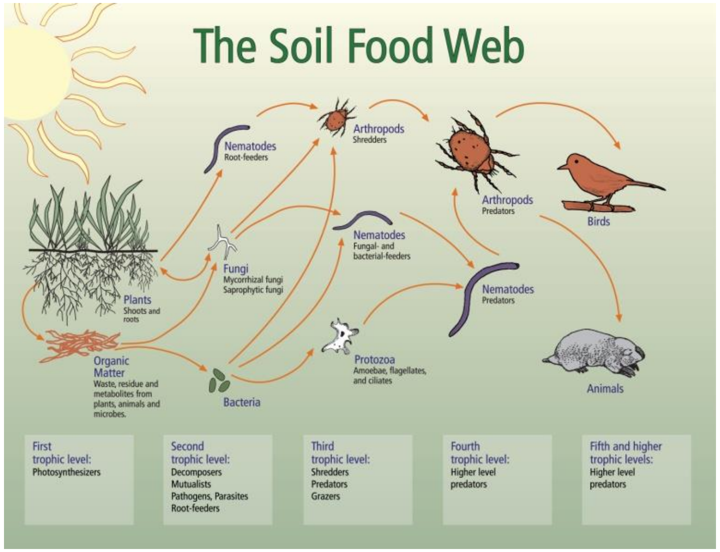
Fast Computation of Dense Temporal Subgraphs

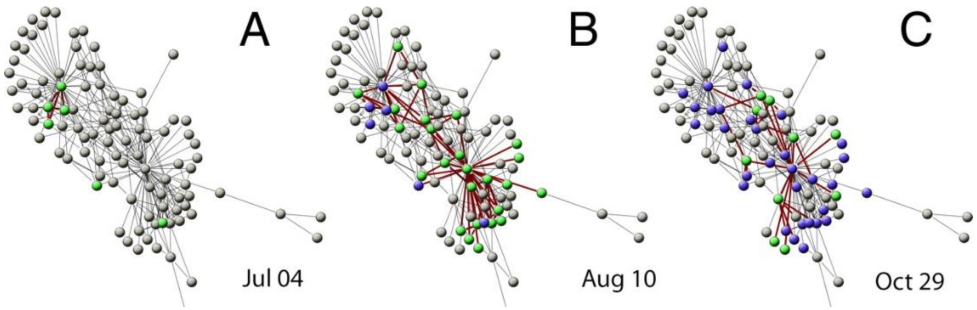
Shuai Ma, **Renjun Hu**, Luoshu Wang, Xuelian Lin, Jinpeng Huai

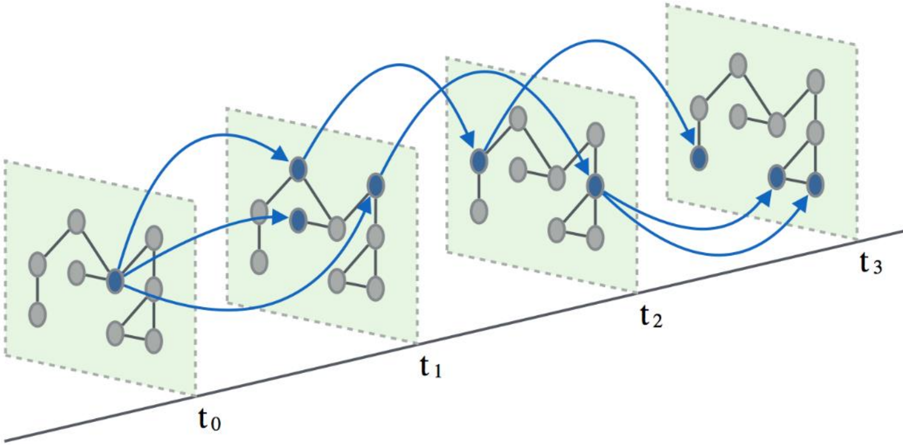
SKLSDE Lab, Beihang University, China

Beijing Advanced Innovation Center for Big Data and Brain Computing

Graphs are dynamic





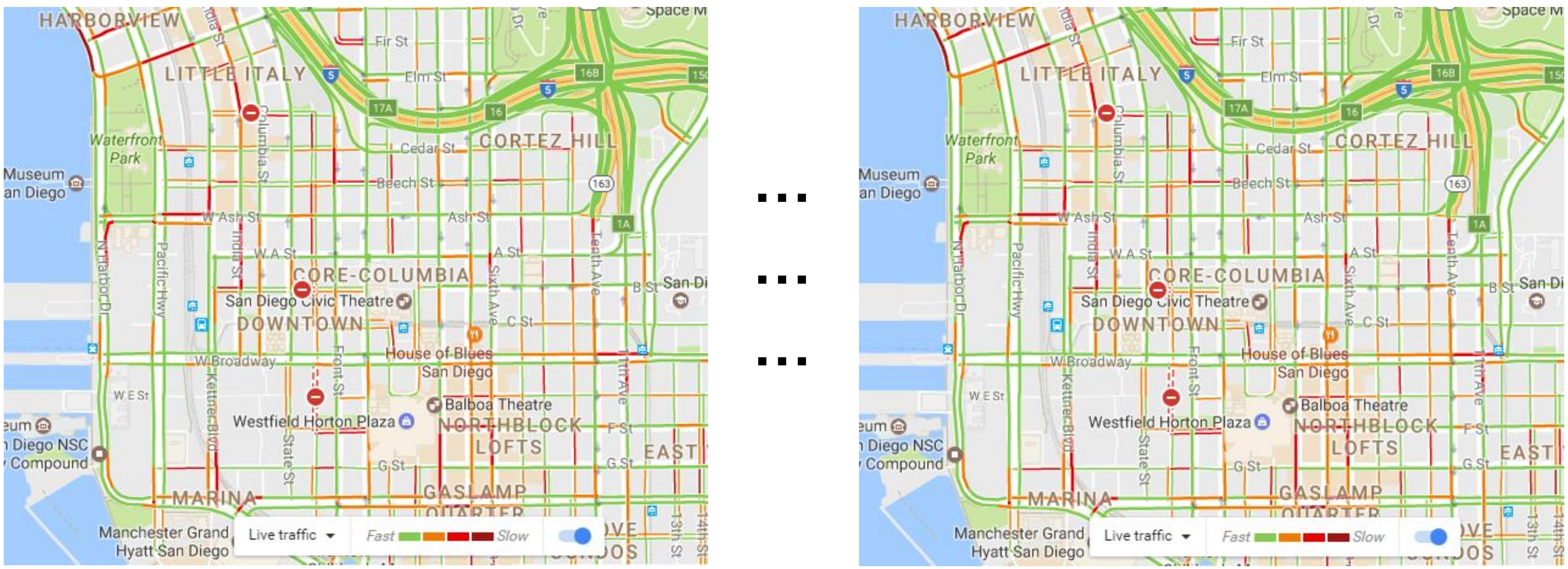


*All images are downloaded from Google.

Motivation

Temporal graph: a continuous sequence of snapshots (each snapshot records the status of a graph at a specific timestamp)

Dense temporal subgraph: a subgraph having heavy edge weights over a continuous time period

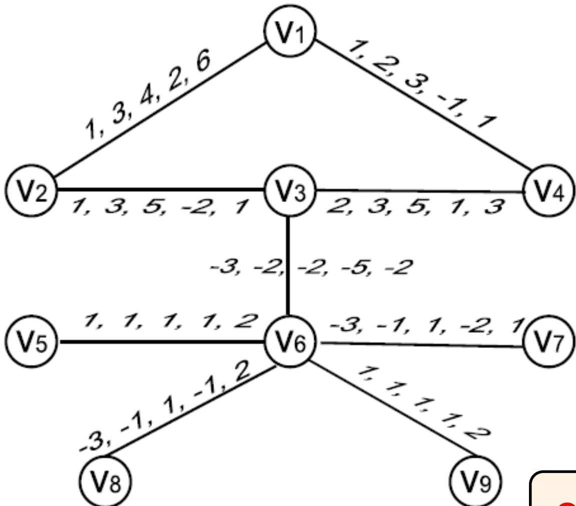


A dense temporal subgraphs corresponds to a crowded area spanning over a continuous time period.

Outline

- **The FDS problem: analyses and challenges**
- A data-driven approach
- Experimental study
- Summary

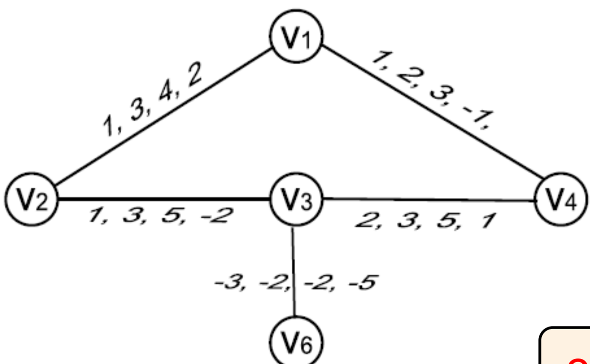
Temporal graphs and subgraphs



Temporal graph G(V, E, F) with T timestamps

- nodes and edges keep unchanged
- edge weights constantly and regularly vary with timestamps
- T snapshots: $G_1(V, E, F^1)$, $G_2(V, E, F^2)$, …, $G_T(V, E, F^T)$

cdensity(G)=36



Temporal subgraph $H(V_s, E_s, F_s, i, j)$

- time interval $[i, j] \subseteq [1, T]$
- subgraph (V_s, E_s) of (V, E)
- denote $H(V, E, F_s, i, j)$ as $G[i, j]$

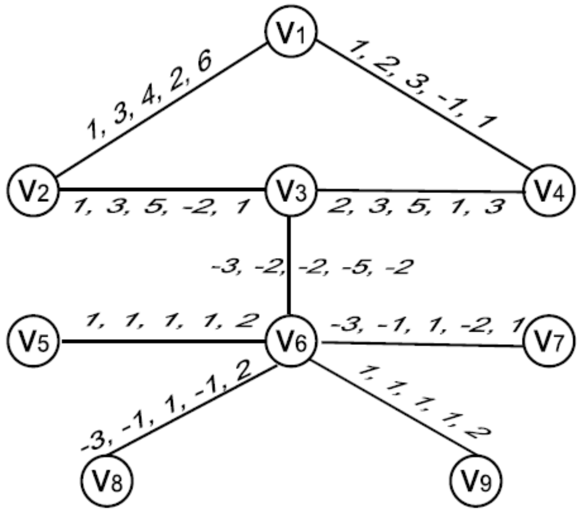
cdensity(H)=21

Cohesive density cdensity(G)

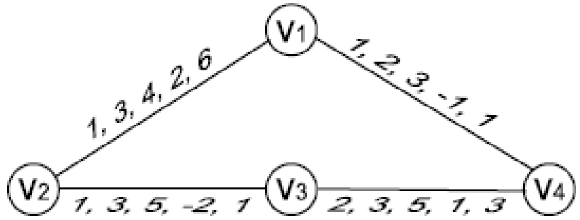
- sum of edge weights among all snapshots

Problem statement and complexity analysis

- FDS: finding dense subgraphs
  - find a connected temporal subgraph with the greatest cohesive density



cdensity(G)=36



cdensity(H)=44

Problem statement and complexity analysis

- FDS: finding dense subgraphs
  - find a connected temporal subgraph with the greatest cohesive density
- Problem hardness[Bogdanov et al. 11]
  - *the FDS problem is NP-complete, even for a temporal network with a single snapshot and with +1 or -1 edge weights only.*
- Our approximation hardness result
  - *the cohesive density of the optimal dense temporal subgraph is NP-hard to approximate within any constant factor.*
    - ✓ proof sketch: building an approximation factor preserving reduction from the net worth maximization problem, which is NP-hard to approximate within any constant factor.

P. Bogdanov, M. Mongiov, A. K. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.

Challenges

find a dense temporal subgraph → determine a time interval [i, j]; find a dense subgraph given [i, j]

Filter-and-Verification[Bogdanov et al. 11]

- consider all time intervals [i, j] and find dense subgraphs by fixing [i, j] each time
- filter [i, j] if its upper bound of cohesive density is worse than the best cohesive density achieved
- prune 99% of a total of T*(T+1)/2 time intervals

T	141	447	1,414	…	14,142
T*(T+1)/2	10^4	10^5	10^6	…	10^8
# unpruned	10^2	10^3	10^4	…	10^6

Filter-and-Verification is insufficient for large temporal graphs!
A new and better algorithm design philosophy is needed.

P. Bogdanov, M. Mongiov, A. K. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.

Outline

- The FDS problem: analyses and challenges
- **A data-driven approach**
- Experimental study
- Summary

Main ideas

Big graph friendly

- Employ hidden data statistics to explore k time intervals
  - k is typically a small constant independent of T, e.g., 10

T	141	447	1,414	···	14,142
T*(T+1)/2	10^4	10^5	10^6	···	10^8
# unpruned	10^2	10^3	10^4	···	10^6
our approach	k	k	k	...	k

- Our data-driven approach FIDES
  - **step 1:** identify k time intervals involved with dense subgraphs
    - ✓ employing hidden data statistics and drawing characteristics of targeted time intervals
  - **step 2:** compute dense subgraphs given time intervals
    - ✓ building the connections with the NWM problem, and exploiting effective and efficient optimization techniques

1000x faster while remain comparable quality of dense subgraphs

Step 1: Hidden data statistics

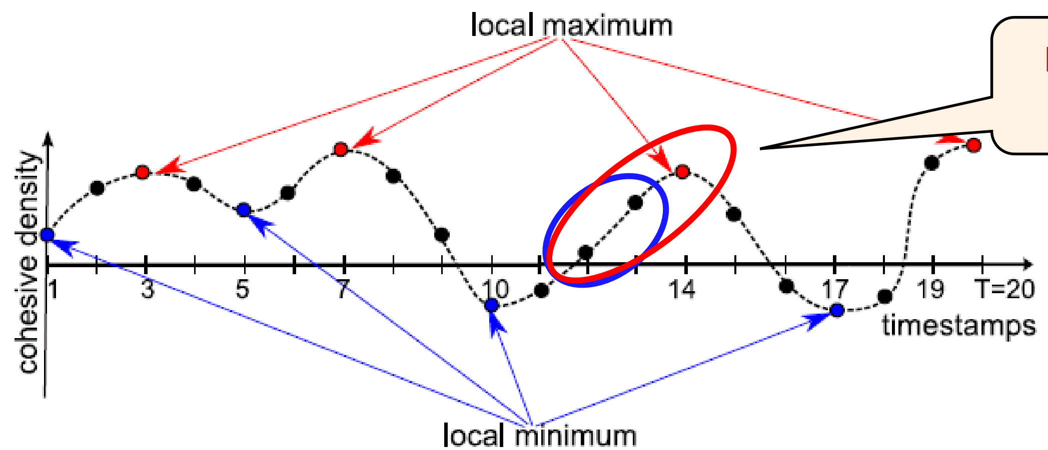
- Convergent evolution
  - organisms not closely related independently evolve similar traits as a result of having to adapt to similar environments

- Evolving convergence phenomenon (ECP)
  - edge weights evolve in a convergent way
  - not completely realistic, but admit a nice mathematical development

ECP assures an important characteristic of targeted time intervals.

Step 1: Characteristics of time intervals

C1: To find the dense subgraph, we only need to consider the time intervals [i, j] such that the cohesive density curve has a local maximum at certain points between i and j under ECP.



must be denser!

top-k for better

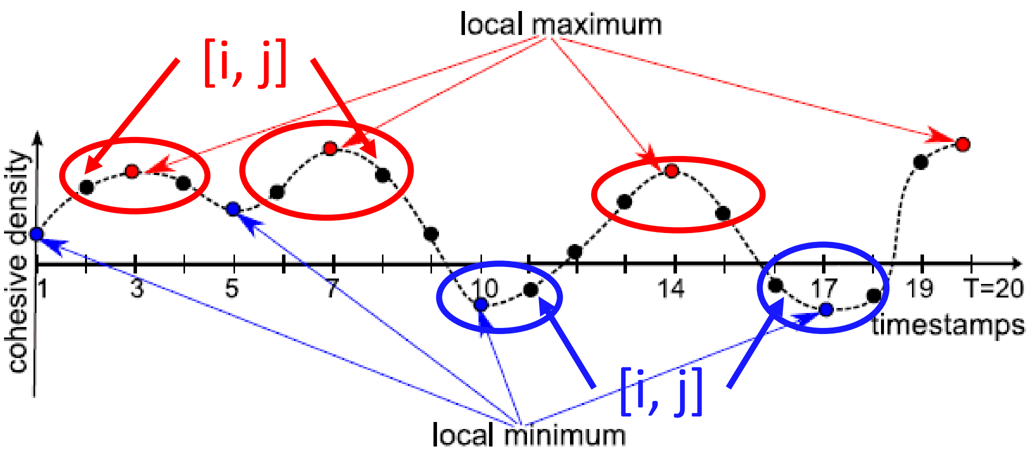
cohesive density curve, $y(x) = \text{cdensity}(G_x)$

C2: All dense subgraph have a non-negative cohesive density.

C3: G[i, j] with a higher positive cohesive density has a higher probability of containing a dense subgraph.

Step 1: Identifying k time intervals

1. compute local maxima/minima of the cohesive density curve;
2. extend local maxima/minima to peaks/valleys of the curve;
3. generate time intervals containing local maxima;
4. find the top-k time intervals having the largest positive cdensity;



Time complexity: $O((T+h^2)|E|)$, h is the # of local maxima/minima

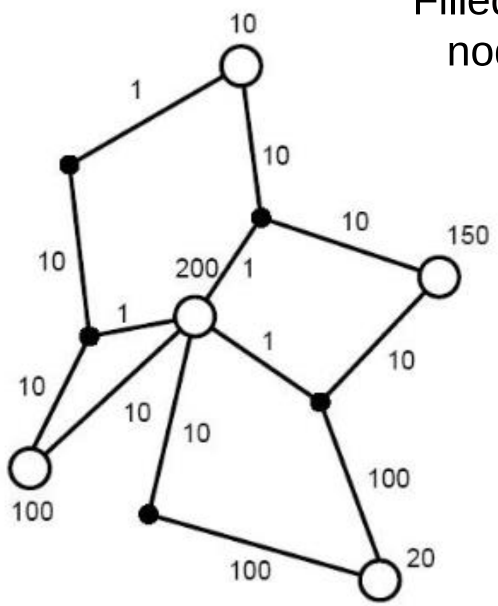
FIDES recap

- **step 1:** identify k time intervals involved with dense subgraphs
  - ✓ hidden data statistics: evolving convergence phenomenon
  - ✓ three characteristics of targeted time intervals
  - ✓ top-k time intervals containing a local maximum and having the largest positive cohesive density

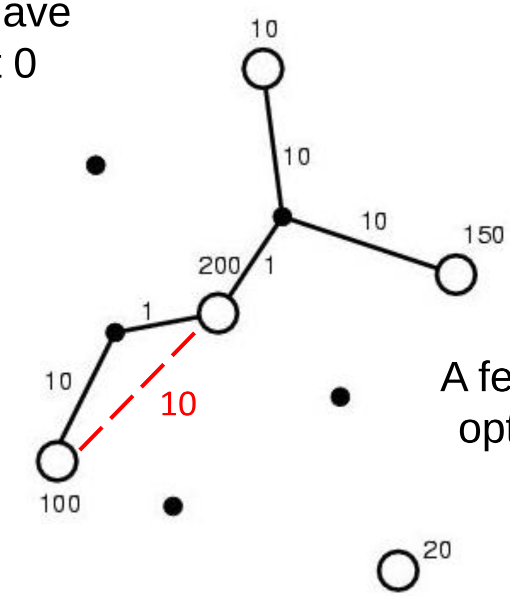
- **step 2:** compute dense subgraphs given time intervals (referred to as computeADS)
  - ✓ building the connections with the NWM problem
  - ✓ exploiting effective and efficient optimization techniques

Step 2: The NWM problem

- Net worth maximization
  - input: a graph with non-negative node and edge weights
  - output: a subtree that maximizes the net worth
    - ✓ net worth = sum of node weights – sum of edge weights



Filled nodes have node weight 0



A feasible but not optimal subtree

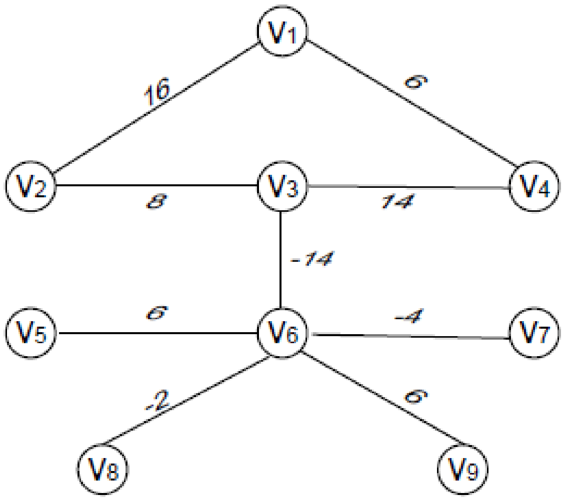
Image credit: Ivana Ljubic, https://homepage.univie.ac.at/ivana.ljubic/research/pcstp/

Step 2: Connections with the NWM problem

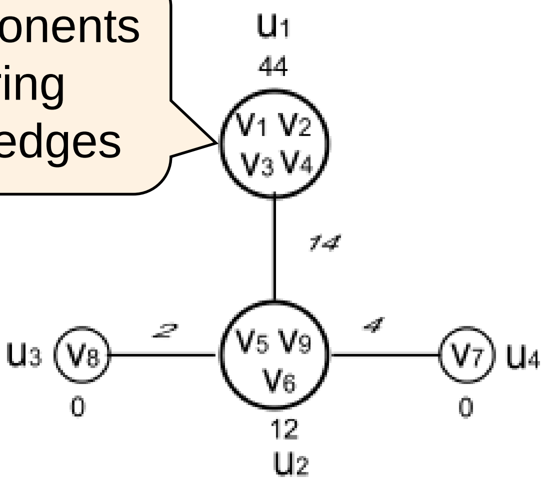
temporal subgraph G[i, j]

aggregate graph G'(V, E, f)
where $f(e)=\sum F^t(e)$, $t \in [i, j]$

converted graph $G'_c(V_c, E_c)$
with node and edge weights



connected components
only considering
positive weight edges



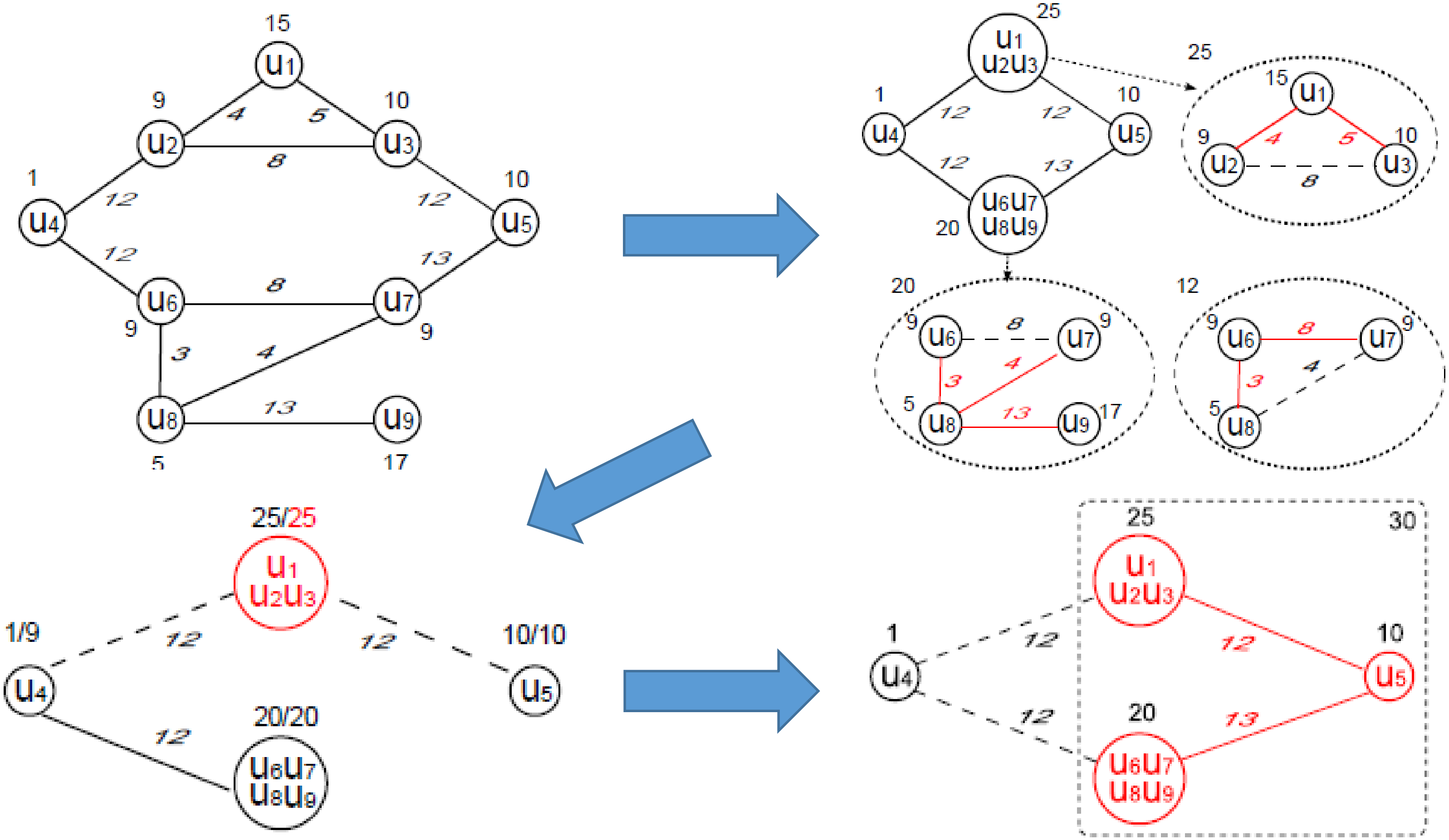
dense subgraph in G' is equivalent to NWM subtree in G'_c

Step 2: Optimization techniques

strong merging: merge nodes that belong to the same NWM subtree;

strong pruning: compute an optimal subtree ST on the MST;

bounded probing: optimize ST by probing nodes in bounded distance;

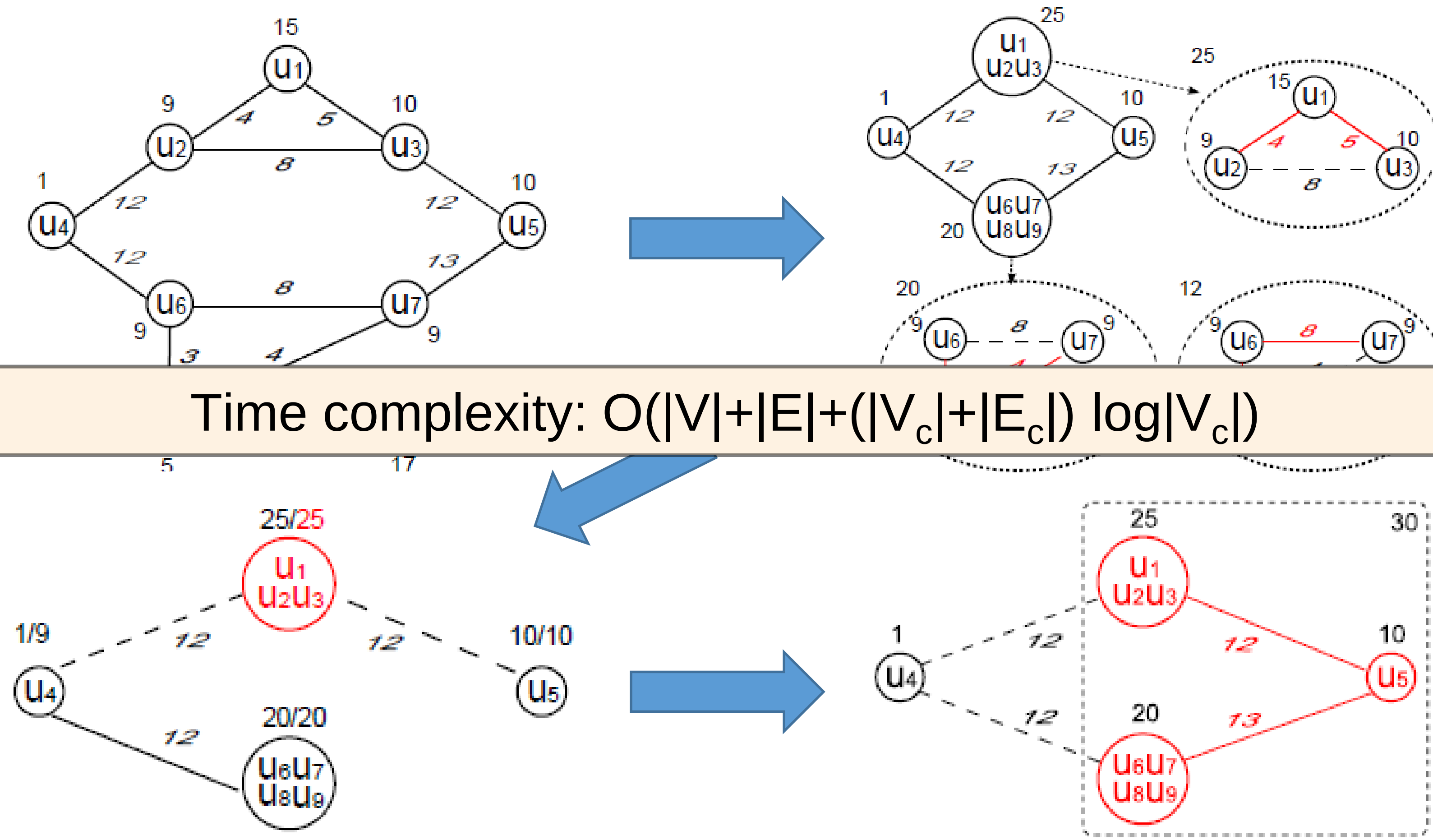


Step 2: Optimization techniques

strong merging: merge nodes that belong to the same NWM subtree;

strong pruning: compute an optimal subtree ST on the MST;

bounded probing: optimize ST by probing nodes in bounded distance;



Time complexity: $O(|V|+|E|+(|V_c|+|E_c|) \log|V_c|)$

Outline

- The FDS problem: analyses and challenges
- A data-driven approach
- **Experimental study**
- Summary

Experimental setups

- Data sets

Data sets	|V|	|E|	T	ad_r	Description
BJData	82,093	108,238	289	0.44	real-life Beijing road network with traffic status
SYNData	50,000 ~ 400,000	2|V|	200 ~ 2,000	0.05 ~ 0.35	synthetic temporal network used in [Bogdanov et al. 11]

  - activation density ad_r: ratio of positive weight edges

- Algorithms
  - FDS given time intervals: computeADS & topDown [Bogdanov et al., 11]
  - FDS on temporal graphs: FIDES & MEDEN [Bogdanov et al., 11]

- Experimental goals
  - verify the rationale behind evolving convergence phenomenon
  - test the quality of dense subgraph found by computeADS and FIDES
  - test the efficiency of computeADS and FIDES

P. Bogdanov, M. Mongiov, A. K. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.

Verification of ECP

- Edge weights evolve in a convergent way
  - to what degree does ECP hold in temporal graphs?

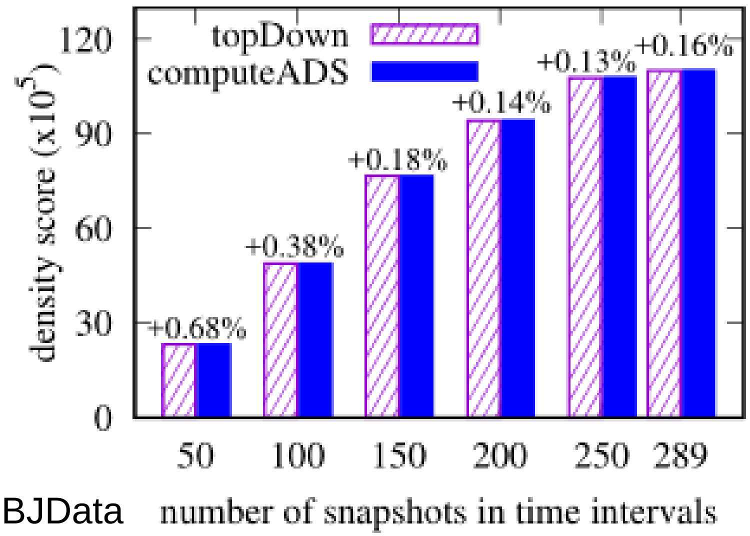
- Proportion of edges that satisfy ECP
  - 96% on BJData
  - 90% on average on SYNData

$$p_{EC} = \frac{\sum_{t=2}^{T} \max(|E^{\geq}(t)|, |E^{\leq}(t)|)}{|E|(T-1)}$$

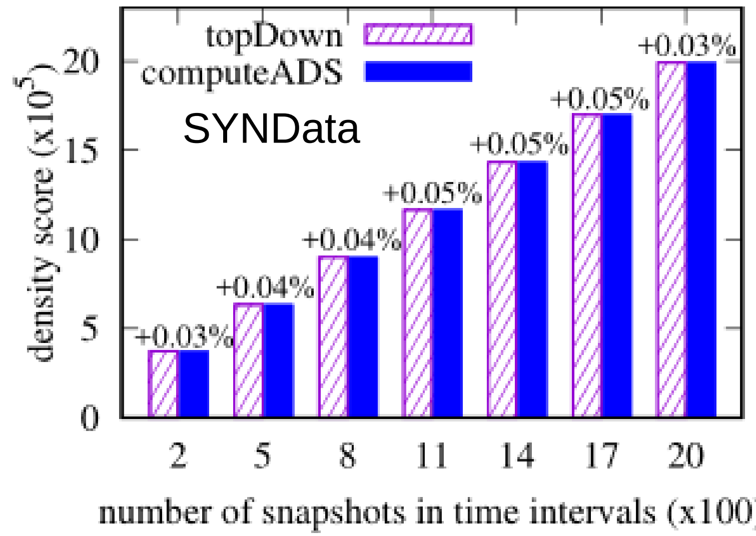
ECP is quite common on both real-life and synthetic temporal graphs.

The characteristics based on ECP work, though ECP is not completely satisfied.

Algorithms computeADS vs. topDown

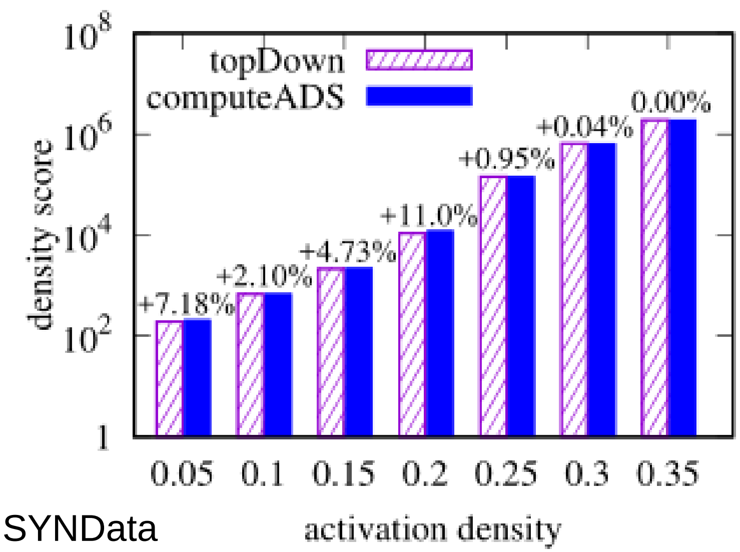


BJData

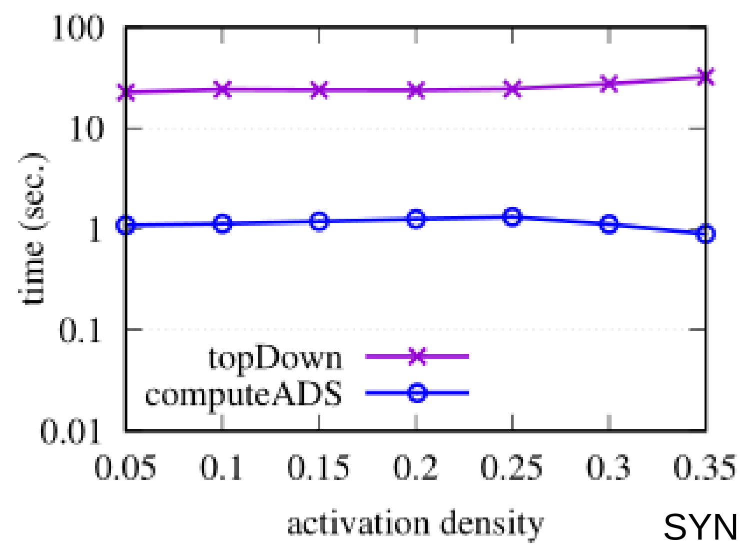


SYNData

Algorithm computeADS is better than topDown in both quality and efficiency.

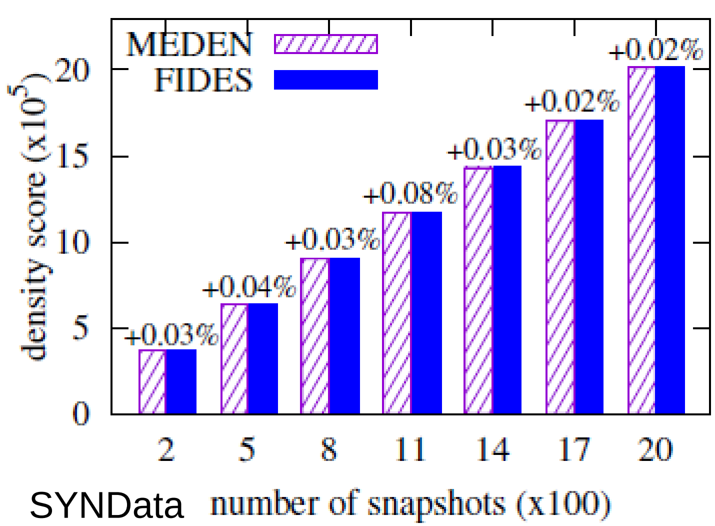


SYNData

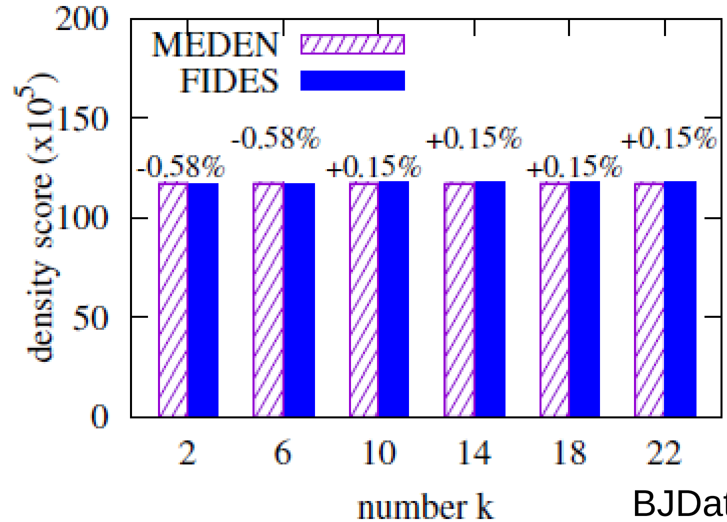


SYNData

Algorithms FIDES vs. MEDEN: quality

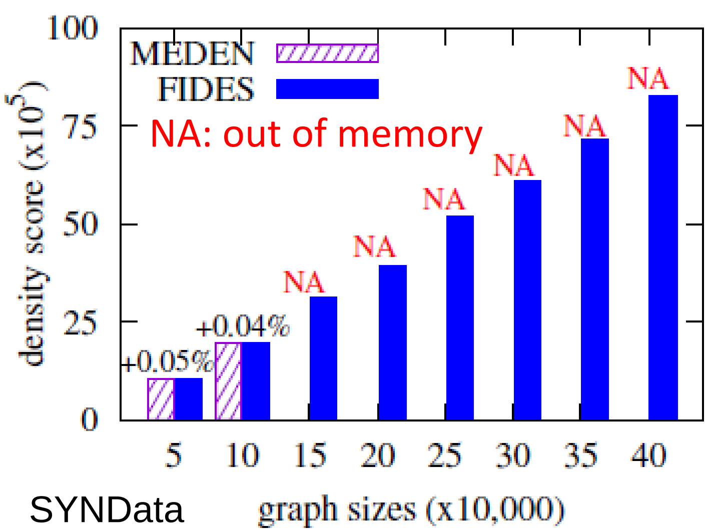


SYNData

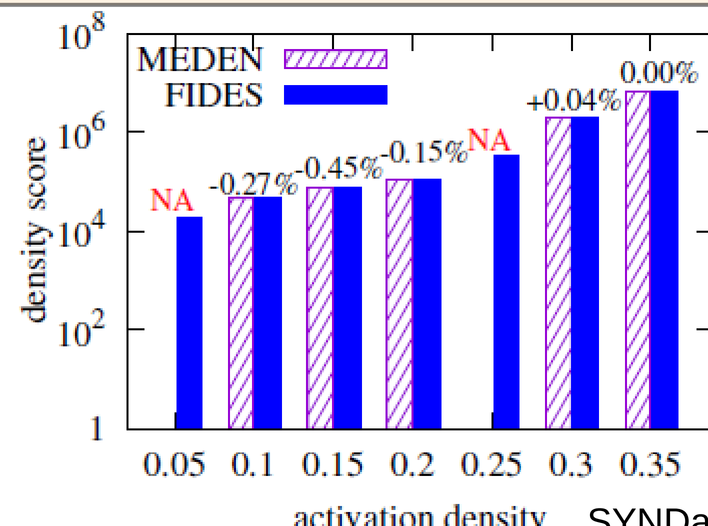


BJData

Dense subgraphs found by FIDES are (+0.28%, -0.16%) better than those found by MEDEN on (BJData, SYNData).

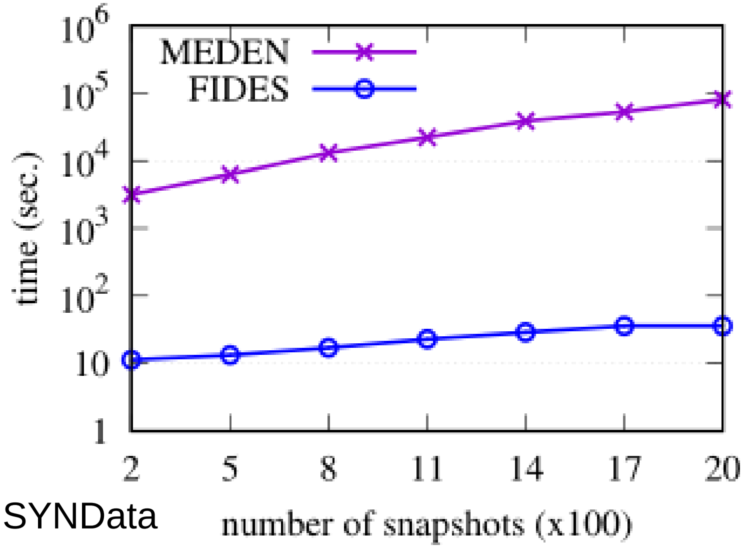


SYNData

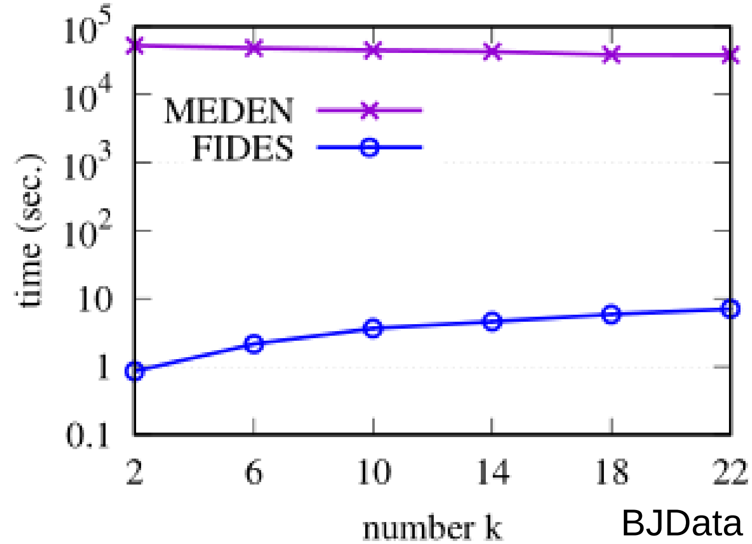


SYNData

Algorithms FIDES vs. MEDEN: efficiency

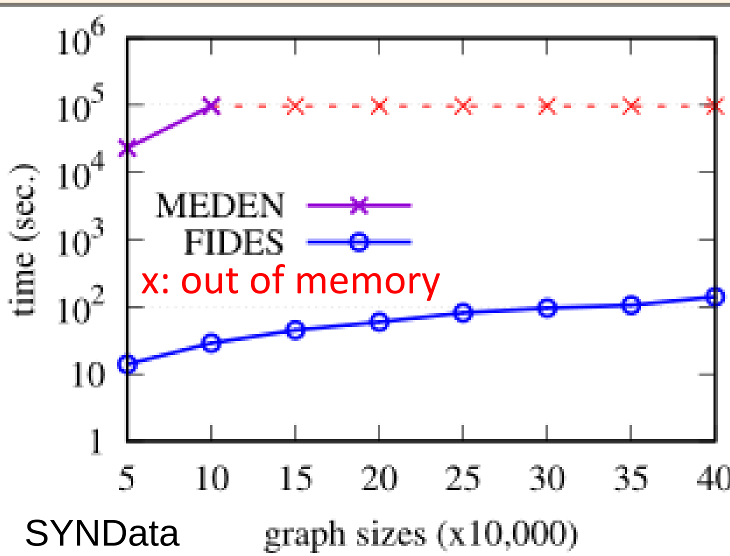


SYNData

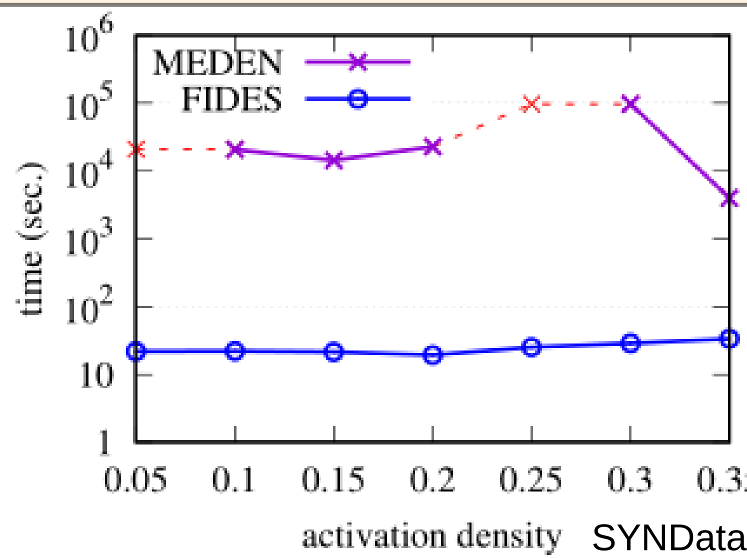


BJData

FIDES is (2,980, 1,079) times faster than MEDEN on (BJData, SYNData).



SYNData



SYNData

Outline

- The FDS problem: analyses and challenges
- A data-driven approach
- Experimental study
- **Summary**

Summary

- **Find dense subgraphs on large temporal graphs**
  - NP-complete and NP-hard to approximate
  - Filter-and-Verification is insufficient for large temporal graphs

- **A data-driven approach (big graph friendly)**
  - identify k time intervals by employing hidden data statistics
  - build the connection between FDS and NWM
  - three algorithm optimization techniques

- **Comparison with the state-of-the-art solution on both real-life and synthetic data**
  - comparable in quality of dense subgraphs found
  - three orders of magnitude faster

Thanks!

Q & A

Welcome to tomorrow afternoon’s poster session!

Synthetic data generator[Bogdanov et al., 11]

- Random graphs as underlying graphs

- Step 1: all edges in every snapshots have weight -1
- Step 2: randomly activate an edge in a specific snapshot (+1), and activate its neighboring edges and the same edge in the next snapshots with certain probabilities. Later activated edge will perform the same activation process.
- Step 3: repeat step 2 until a prefixed ad_r is satisfied

P. Bogdanov, M. Mongiov, A. K. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.