Finding Dense Subgraphs with Hierarchical Relations in Real-world Networks

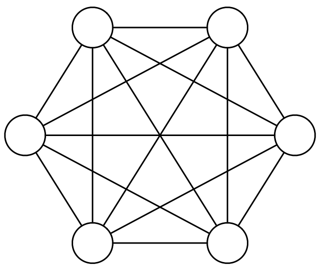
A. Erdem Sarıyüce

Sandia National Laboratories, Livermore CA

http://sariyuce.com

Dense subgraph discovery

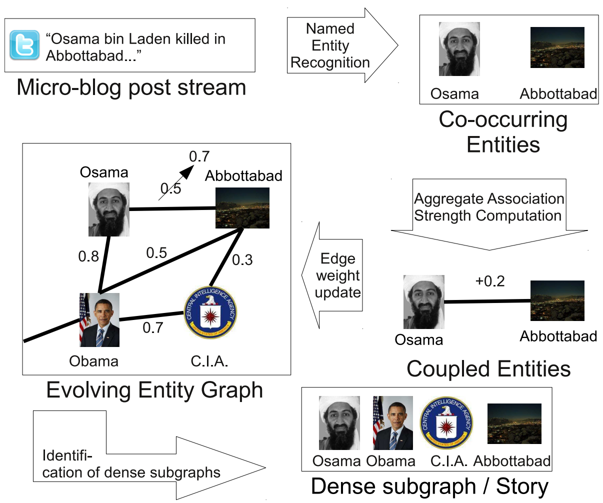
- Measure of connectedness on edges
  - # edge / # all possible
    - |E| / (|V| choose 2), 1.0 for a clique

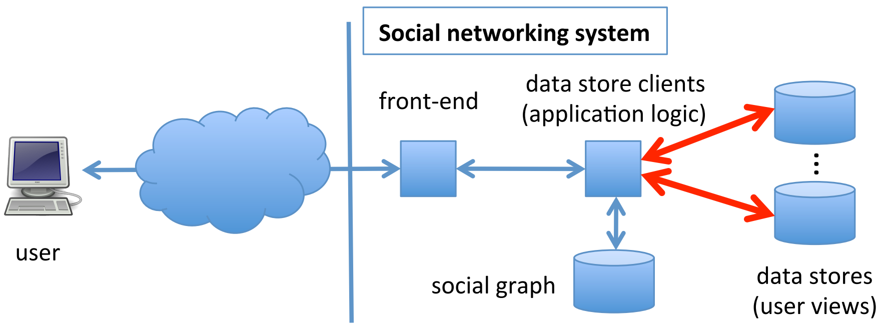


- Globally sparse, locally dense
  - $|E| << |V|^2$, but vertex neighborhoods are dense
    - High clustering coefficients – density of neighbor graph
- Many nontrivial subgraphs with high density
  - And relations among them
- Not clustering: Absolute vs. relative density

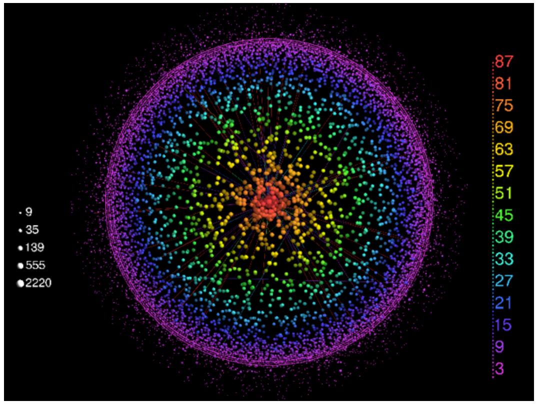
Dense subgraphs matter in many applications

- Significance or anomaly
  - Spam link farms [Gibson et al., '05]
  - Real-time stories [Angel et al., '12]



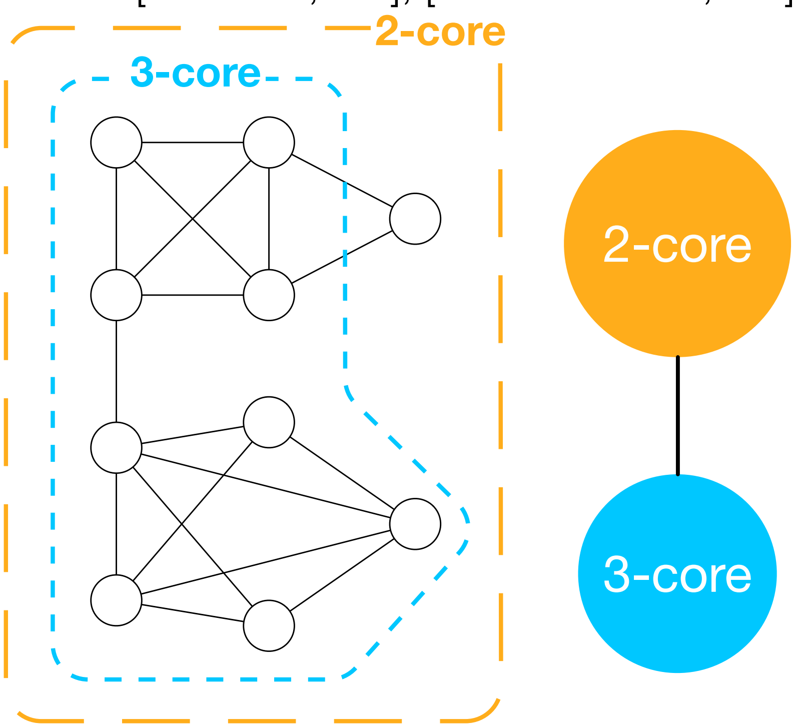


- Computation & summarization
  - System throughputs [Gionis et al., '13]
  - Graph visualization [Alvarez et al., '06]

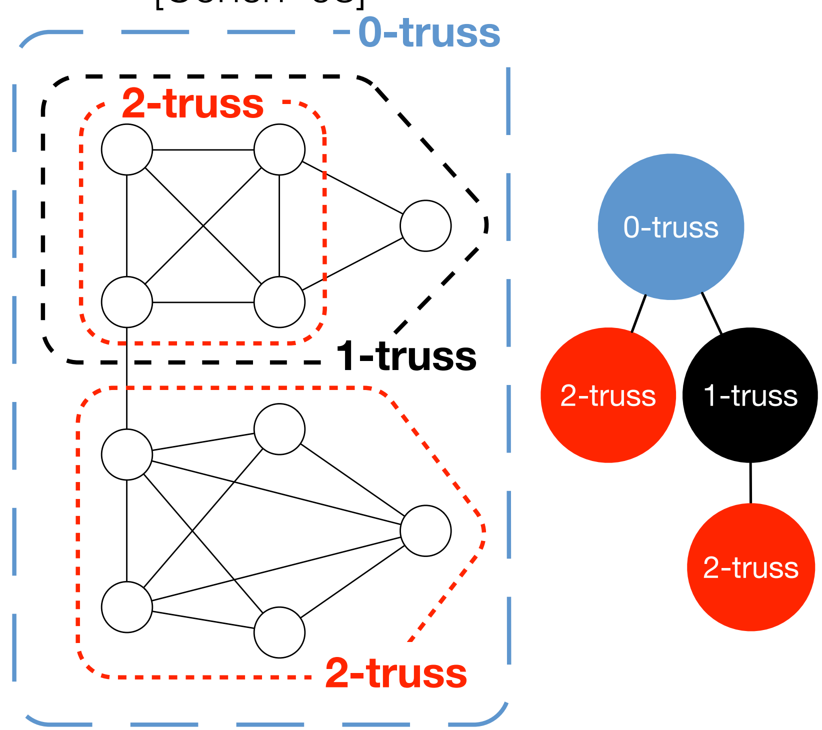


Two effective algorithms to find dense subgraphs with hierarchical relations

- ***k*-core:** Every vertex has at least *k* edges
  - [Seidman, ‘83], [Matula & Beck, ‘83]

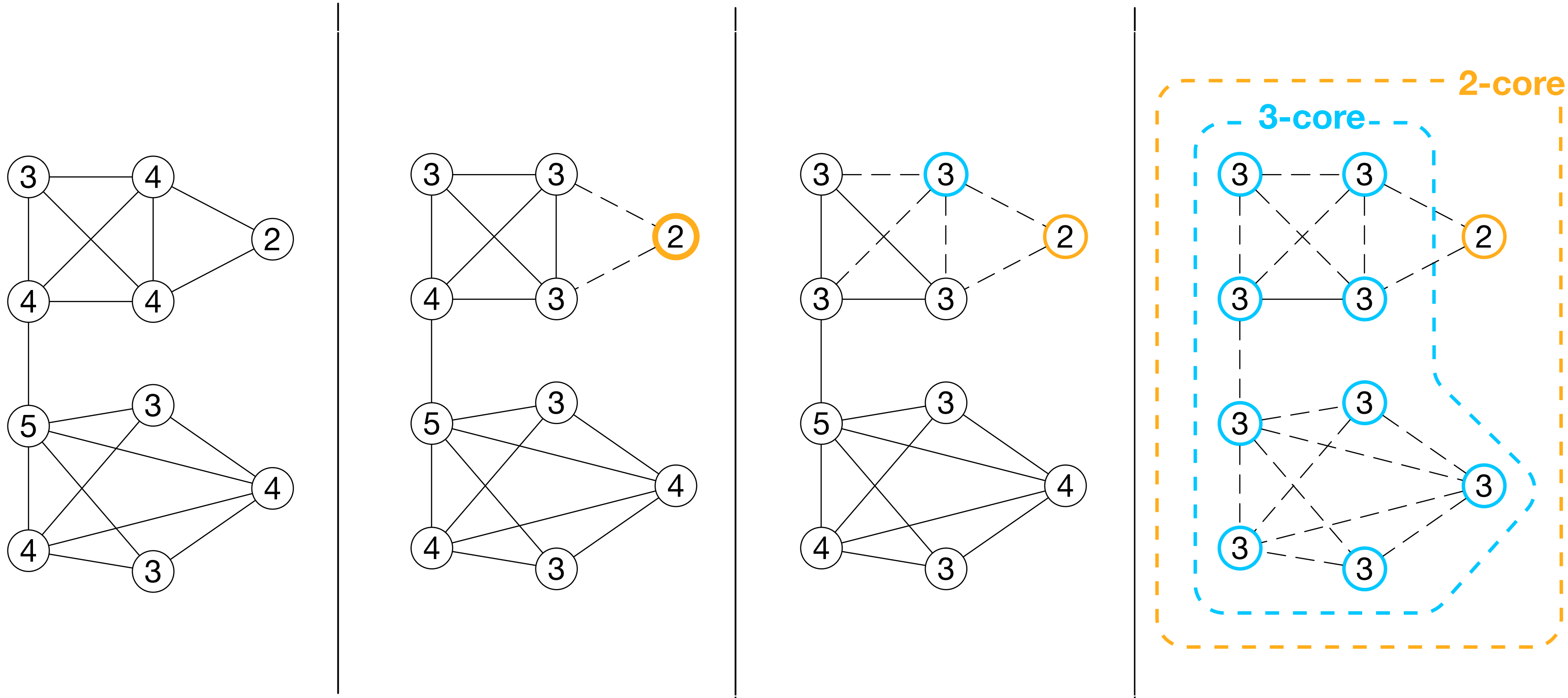


- ***k*-truss:** Every edge has at least *k* triangles
  - [Cohen ‘08]



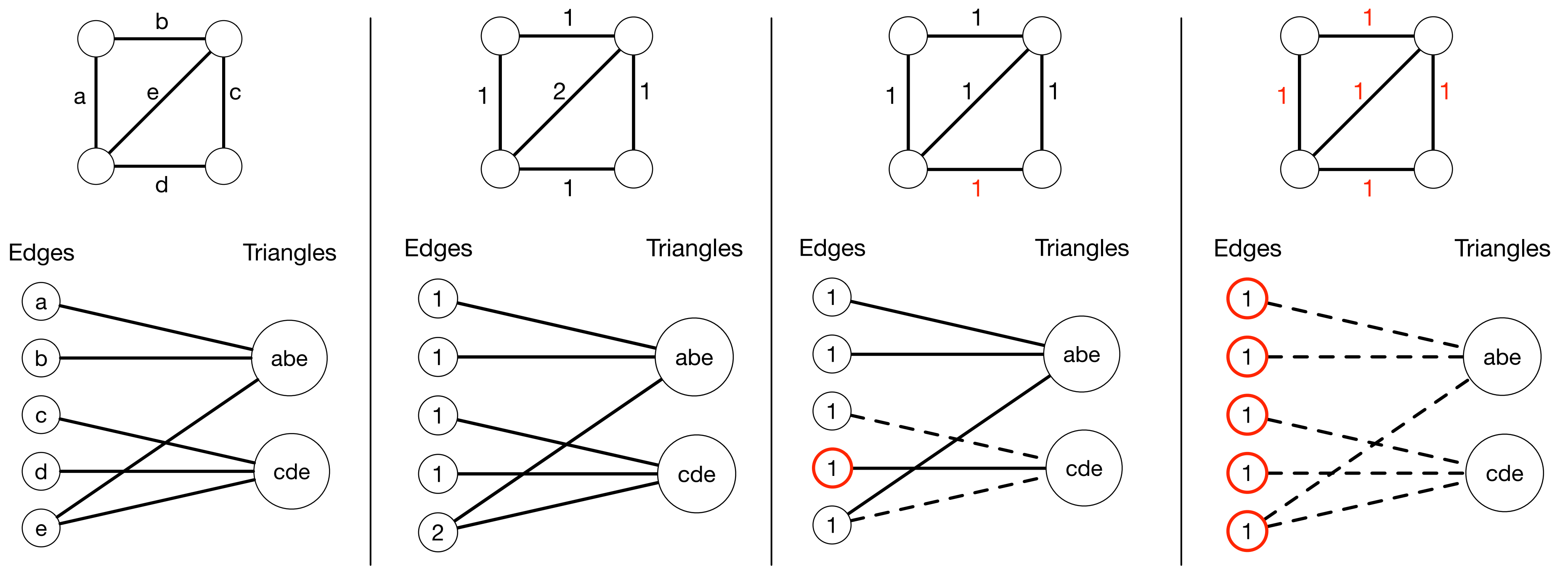
Peeling algorithm finds the *k*-cores & *k*-trusses

- Core numbers of vertices. O(|E|) [Matula & Beck, '83]
- Truss numbers of edges. O(|△|) [Cohen '08]



Observation: *k*-truss IS just *k*–core on the edge-triangle graph!

- Edge and triangle relations
  - Not a binary relation – three edges in a triangle
- Build bipartite graph!



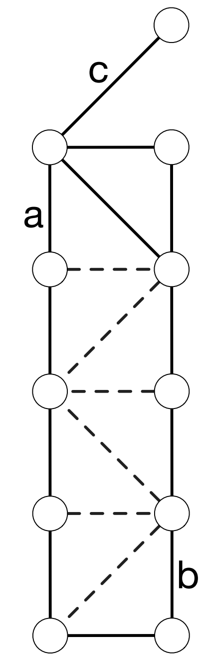
Why limit to *k*-truss?

- Small cliques in larger cliques
  - 1-cliques in 2-cliques (vertices and edges)
  - 2-cliques in 3-cliques (edges and triangles)
- Generalize for any clique
  - *r*-cliques in *s*-cliques ($r < s$)
- Convert to bipartite
  - *r*-cliques → left vertices
  - *s*-cliques → right vertices
  - Connect if right contains left

Nucleus decomposition generalizes *k*-core and *k*-truss algorithms

- Say R is r-clique, S is s-clique ($r < s$)
- k-(r, s) nucleus: Every R takes part in at least k number of S
  - Each R_i, R_j pair is connected by series of Ss

r=1, *s*=2	*r*=2, *s*=3
***k*-core**	***k*-truss**
	(stronger conn.)
d(v) ≥ k	△(e) ≥ k
Simply connected	Triangle connected

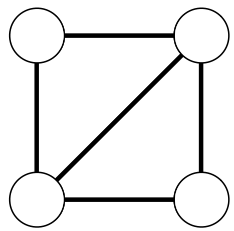


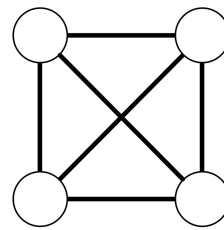
Sariyuce, Seshadhri, Pinar, Catalyurek, WWW 2015 (Best paper runner-up)

Some nucleus examples

1 - 1 3 nucleus

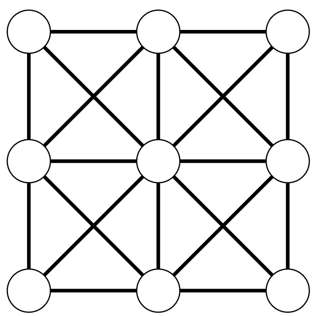
Each vertex has at least one triangle

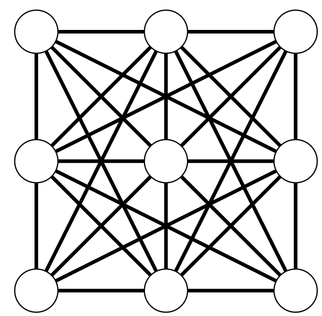




1 - 1 4 nucleus

Each vertex has at least one 4-clique





2 - 2 3 nucleus

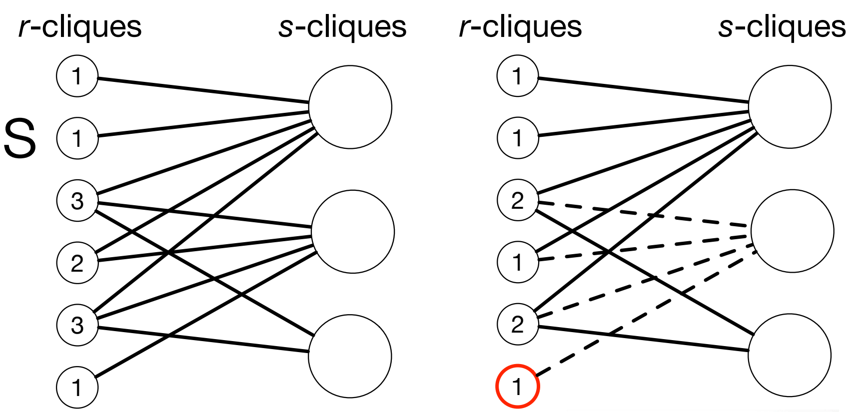
Each edge has at least two triangles

2 - 2 4 nucleus

Each edge has at least two 4-cliques

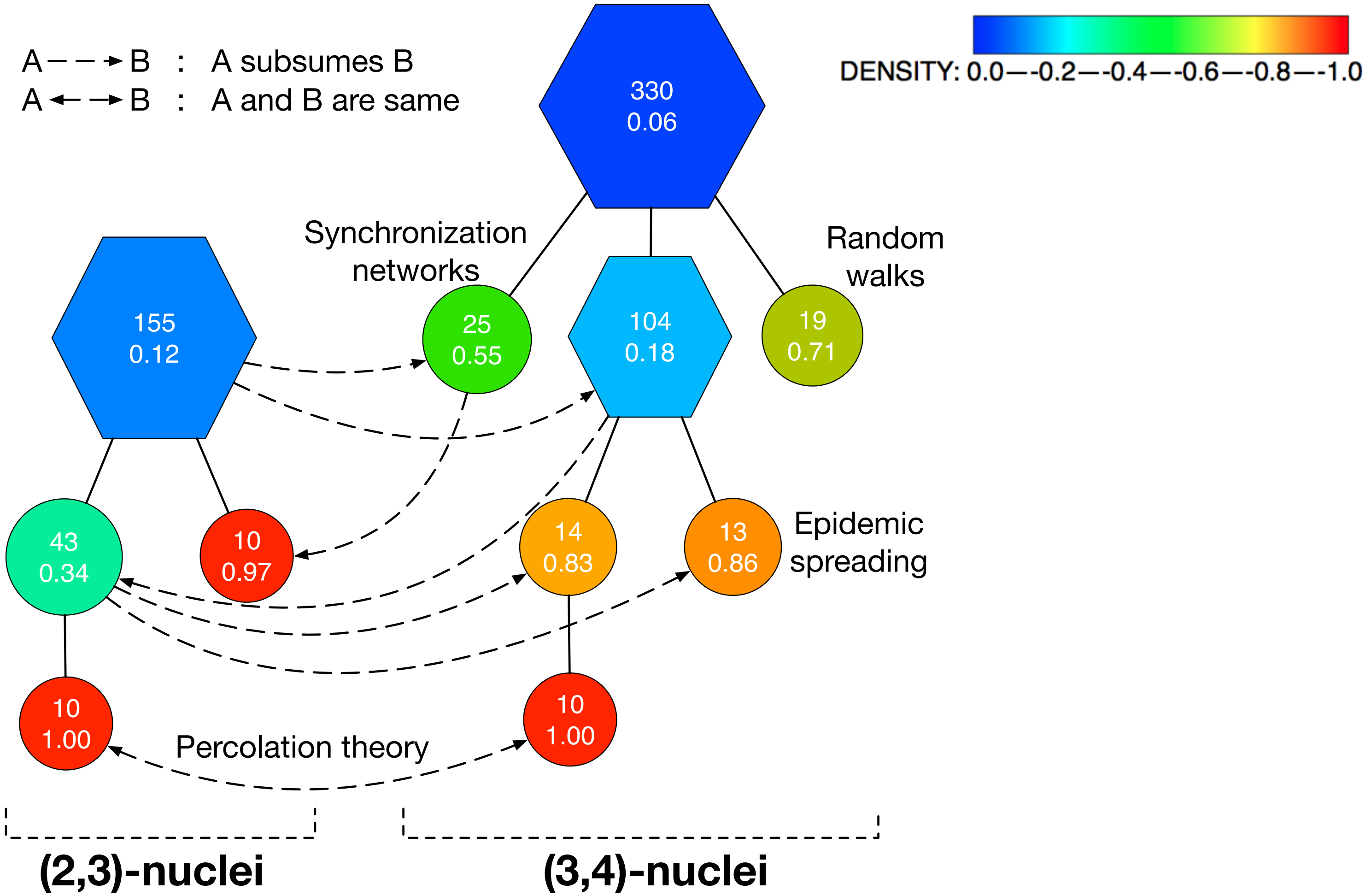
Peeling works for nucleus decomposition as well!

- On the bipartite graph
  - For vertex set of *r*-cliques
  - Degree based



- Sounds expensive?
  - Yes, in theory
  - *r*=3, *s*=4: $O(\sum_v cc(v)d(v)^3)$
  - But practical
    - Clustering coefficients decay with the degree in many real-world networks
  - Can be scaled to tens of millions of edges

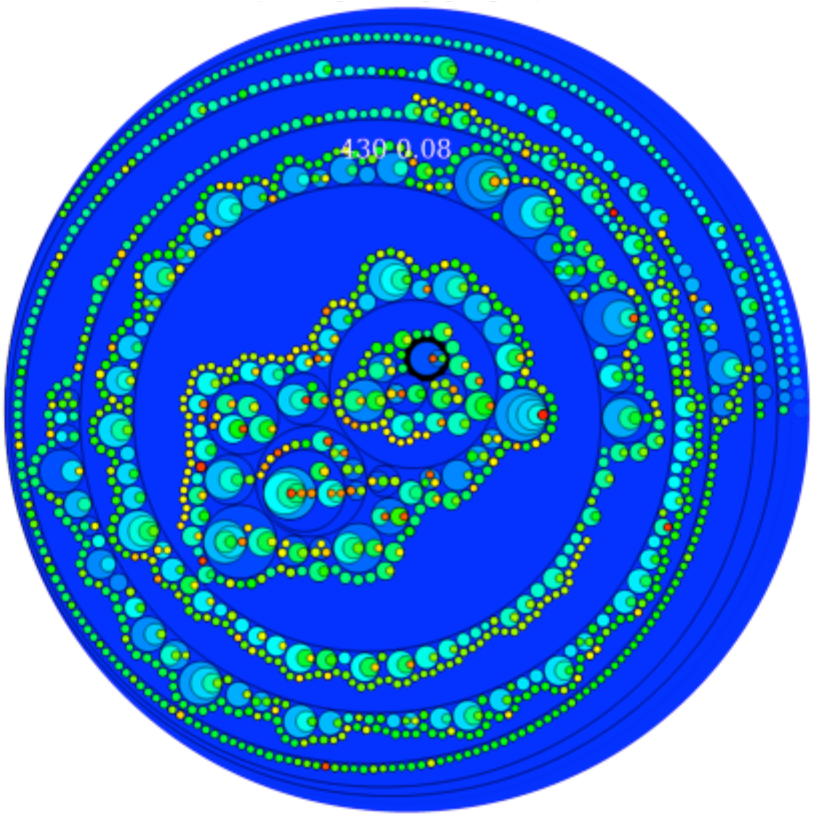
APS Citation Network Analysis

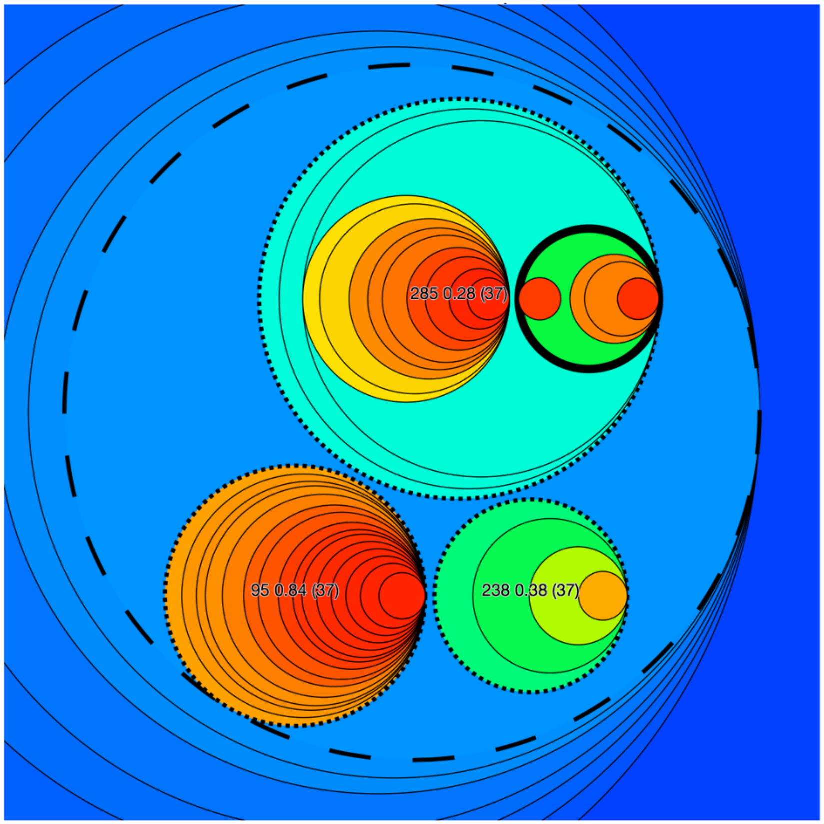


Sariyuce, Seshadhri, Pinar, Catalyurek, TWEB (to appear)

Summarize & visualize the graph with the nucleus hierarchy

- Interactively inspect massive networks
  - Ongoing collaboration with UCSC visualization people





Practical, CAN be made faster

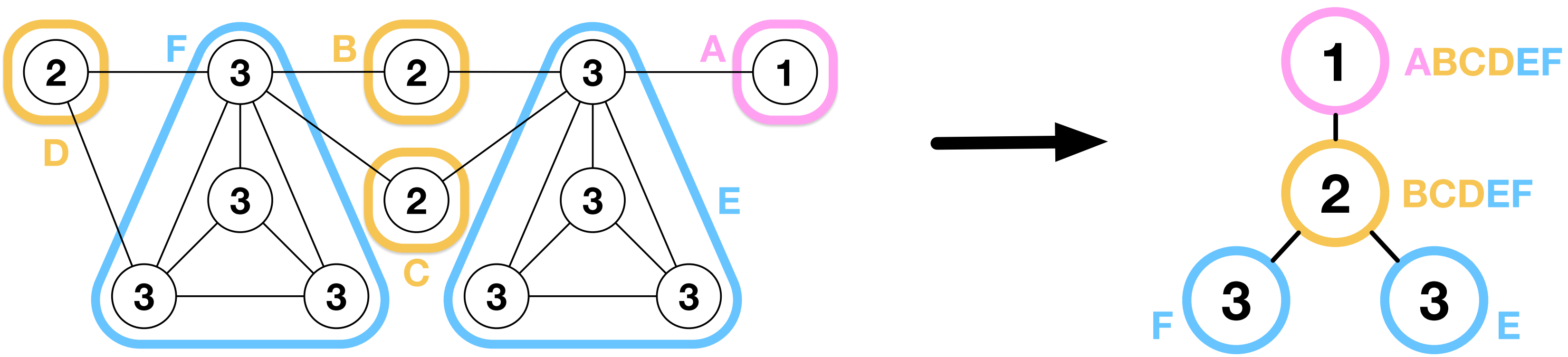
- Sequential
- Less than an hour for 39M edge
  - 6000 nuclei, with ≥ 10 vertex
  - State-of-the-art algorithm reports *one* in a minute
- So many opportunities! (Fun problem)

(in seconds)	|V|	|E|	$\sum_v c_3(v)d(v)$	(3, 4) time
twitter	$81.30K$	$2.68M$	1.8B	396
web-NotreDame	$325.72K$	$1.49M$	33.9B	671
web-Google	$875.71K$	$5.10M$	11.4B	163
as-skitter	$1.69M$	$11.09M$	1.6B	1,036
wikipedia-2005	$1.63M$	$19.75M$	741B	1,312
wiki-Talk	$2.39M$	$5.02M$	136B	605
wikipedia-200611	$3.14M$	$39.38M$	2,197B	3,039

Peeling is not enough to find the entire hierarchy

- Peeling finds core, truss values
- What about constructing connected subgraphs?
  - Easy for *k*-core,O(|E|) -- Challenging for higher-orders!

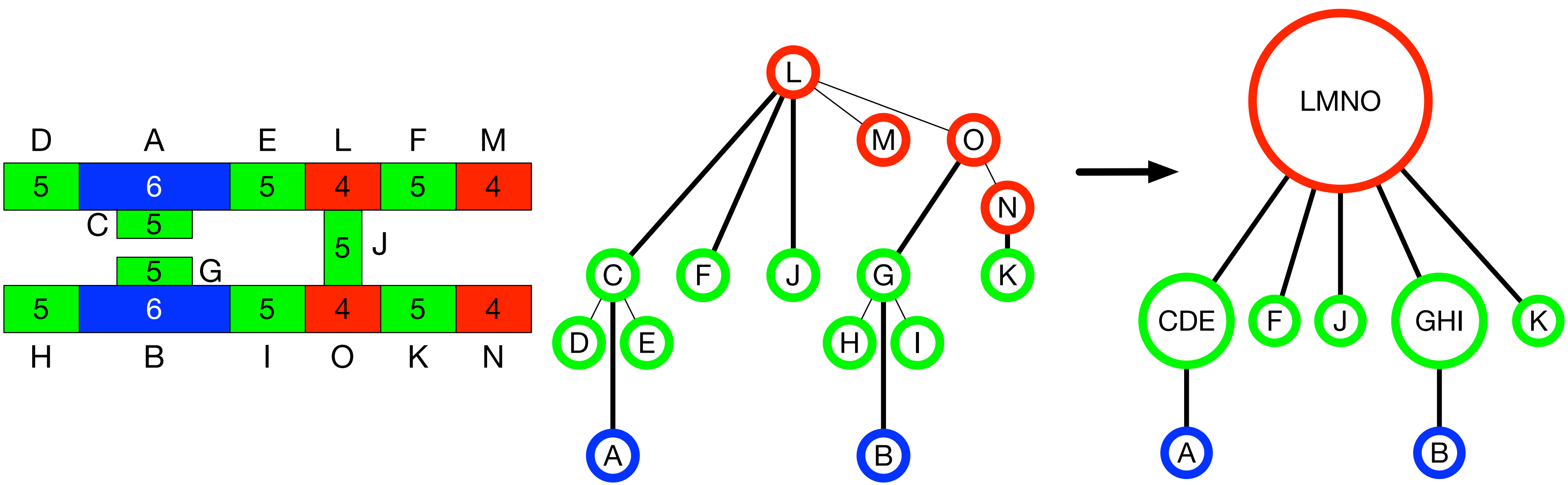
– Construct subgraphs while peeling
  - And build the hierarchy (VLDB'17)



Sariyuce & Pinar, VLDB 2017 (to appear)

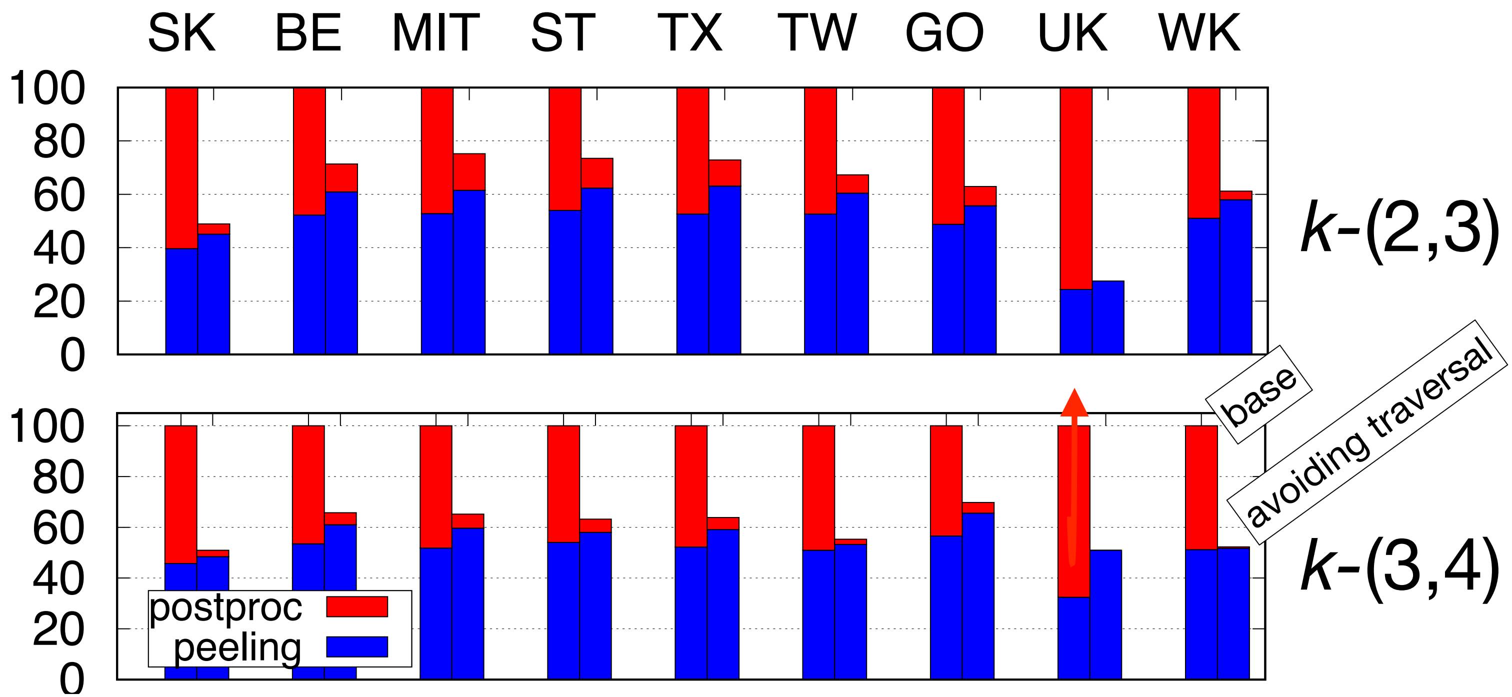
Adapting Union-Find

- We adapt Union-Find for multiple levels
  – Merge disjoint trees and hierarchy tree
- Sub-nucleus: r-cliques with same *K* value
- Find subnuclei in decreasing order of *K* values



We still traverse, any way to avoid?

- Construct subgraphs during peeling



What about other graph types? Bipartite networks?

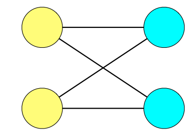
- Author-paper, word-document, actor-movie…
  - Bipartite in nature, no triangle
- Most project bipartite to unipartite
  - Author-paper → Co-authorship
  - |E| explodes! 100x observed
  - Information lost!
    - Projections are not bijective

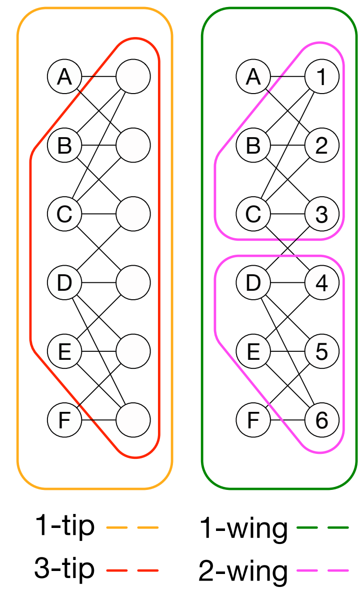
$\lvert E \rvert$	$\lvert E_p \rvert$
58.6K	95.1K
30.7K	84.8K
440.2K	44.5M
96.7K	336.5K
5.6M	157.5M
92.8K	2.0M

- Find dense regions directly on bipartite graph!

Bipartite networks can be peeled as well, if you devise a 'triangle'

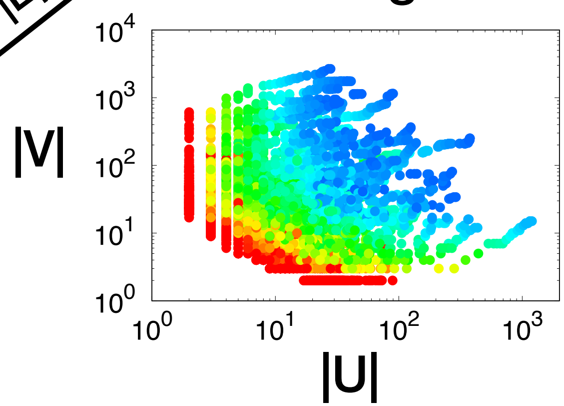
- Focus on the smallest non-trivial structure
  - (2, 2)-biclique, or butterfly
- Vertex-butterfly, edge-butterfly relations
  - *k*-tip: Each vertex has ≥ *k* butterflies
  - *k*-wing: Each edge has ≥ *k* butterflies



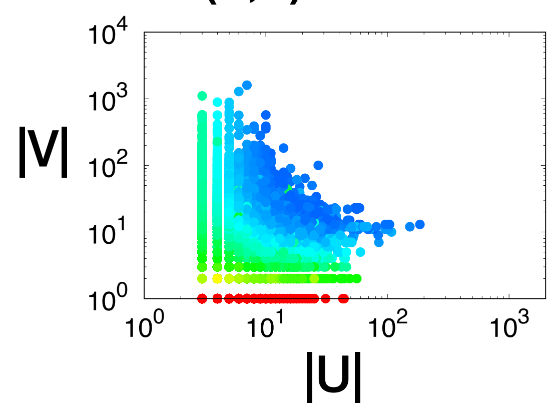


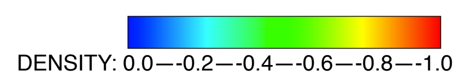
IMDb
|V|: 1.6M |E|: 5.6M

_k_-wing



(2,3) nuclei

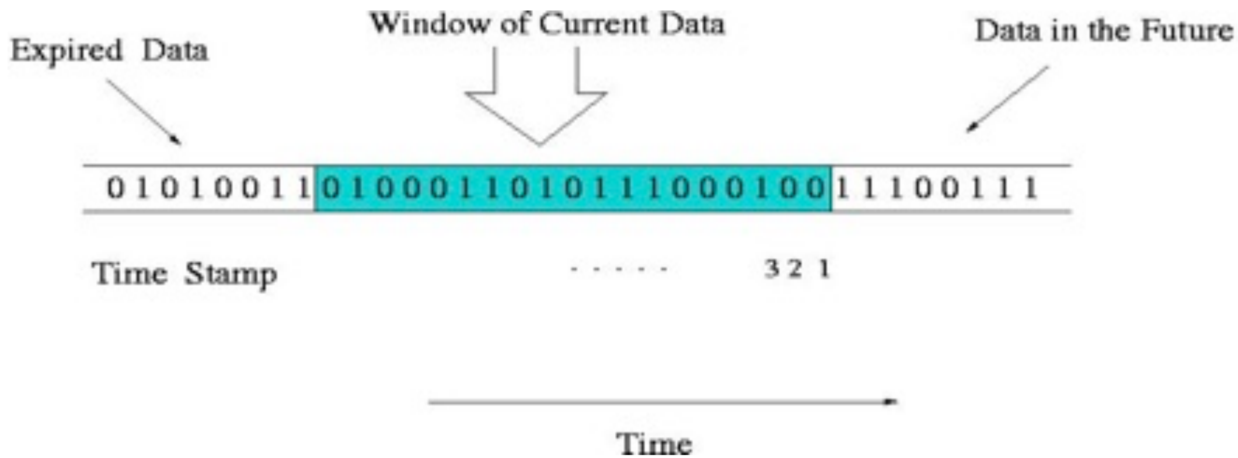




Sariyuce & Pinar, arXiv: 1611.02756

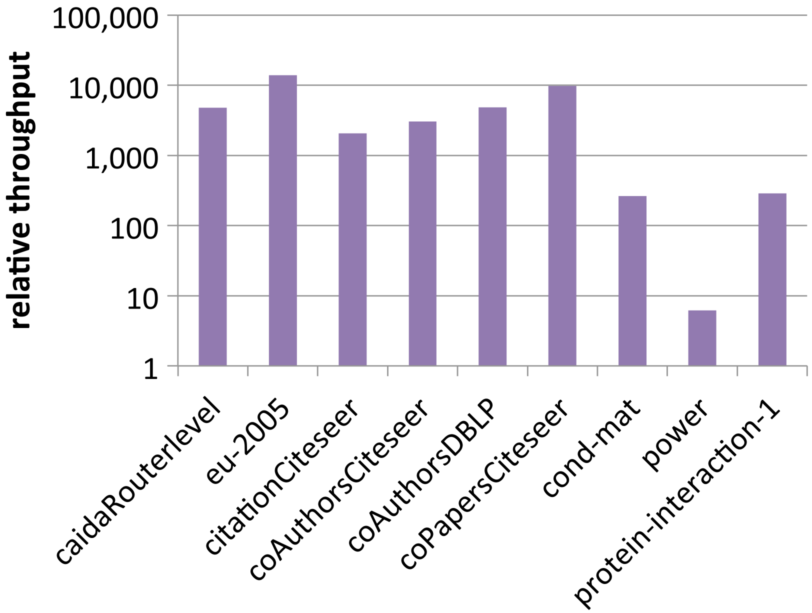
Streaming *k*-core decomposition

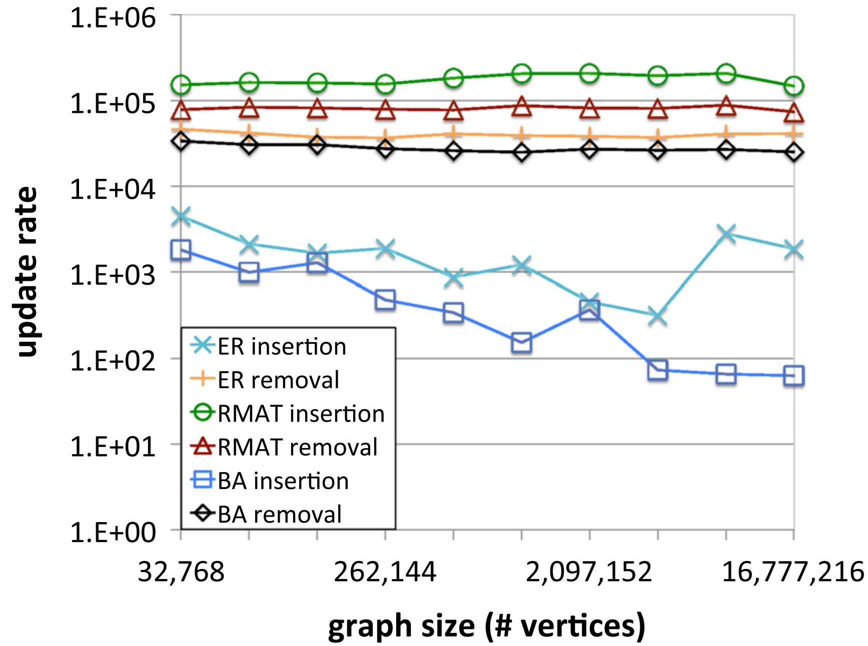
- Considers a sliding-window scenario
  - Count-based or time-based



- Single edge insertion & removal algorithms
  - Should have high processing rate
  - Should be way faster than from-scratch solution

10K edges processed per second





- What about *k*-truss?
  - Or other nucleus decompositions?

Sariyuce, Gedik, Jacques-Silva, Wu, Catalyurek, VLDB'13, VLDBJ

Conclusion

- Introduced the nucleus decomposition
  - Generalizes *k*-core and *k*-truss, and extend
  - Network analysis by the nucleus hierarchy

- Hierarchy construction embedded into peeling
- Bipartite networks
- Incremental algorithms
  - Maintain dense subgraphs, temporal analysis

References

- **A. E. Sariyuce**, C. Seshadhri, A. Pınar, U. V. Catalyurek; Nucleus Decompositions for Identifying Hierarchy of Dense Subgraphs, ACM Transactions on the Web (TWEB), to appear
- **A. E. Sariyuce**, B. Gedik, G. Jacques-Silva, K. Wu, U. V. Catalyurek; Incremental k-core Decomposition: Algorithms and Evaluation, Very Large Data Bases Journal (VLDBJ), 25(3): 425-447, 2016
- **A. E. Sariyuce**, A. Pinar; Fast Hierarchy Construction for Dense Subgraphs, International Conference on Very Large Data Bases (VLDB), 2017, to appear arXiv: 1610.01961
- **A. E. Sariyuce**, C. Seshadhri, A. Pınar, U. V. Catalyurek; Finding the Hierarchy of Dense Subgraphs using Nucleus Decompositions, International World Wide Web Conference (WWW), 2015 (AR: 14.1%) (Best Paper Runner-up)
- **A. E. Sariyuce**, B. Gedik, G. Jacques-Silva, K. Wu, U. V. Catalyurek; Streaming Algorithms for k-core Decomposition, International Conference on Very Large Data Bases (VLDB), 2013 (AR: 22.7%)
- **A. E. Sariyuce**, A. Pınar; Butterfly Effect: Peeling Bipartite Networks, arXiv: 1611.02756

http://sariyuce.com