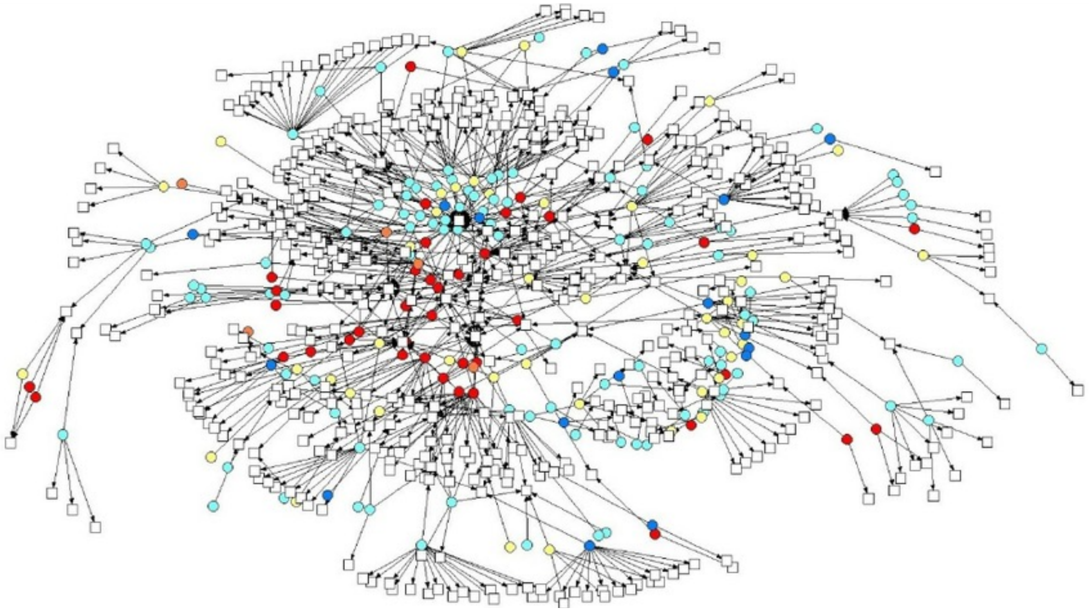
Parallel k-Core Maintenance in Dynamic Graphs

Bin Guo (Speaker), Emil Sekerinski

ICPP 2023

Motivation

- Graphs are important data structures used in many applications:
  - Social Networks: Facebook, Twitter
  - Knowledge Networks: DBpedia
  - Biological Networks and Road Networks
- Data graphs can be large now:
  - Facebook has **2.9 billion** active users
  - DBpedia has **6.6 million** entities and **13 billion** pieces of information

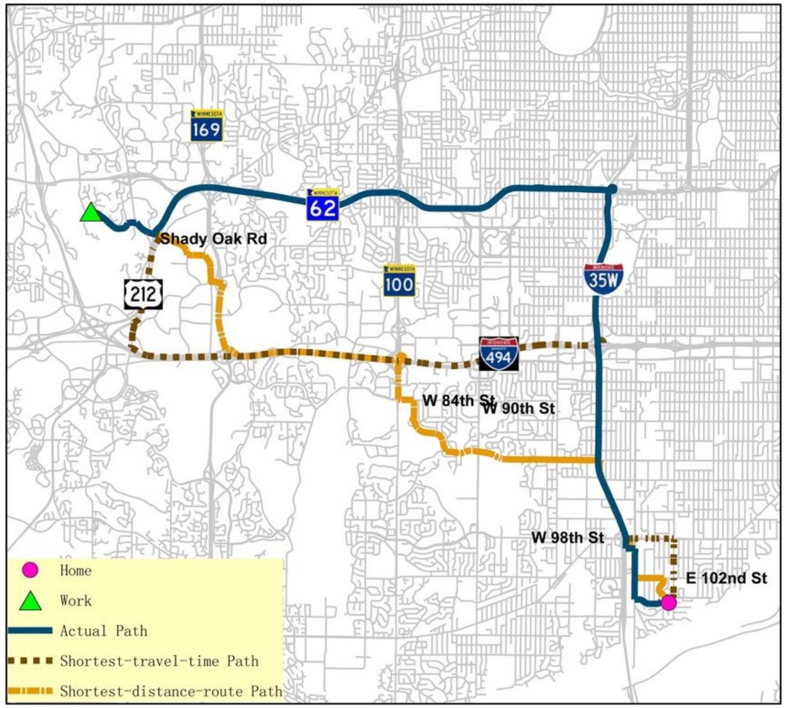


Visualizations of Social Networks show the employee interactions [1]

[1] Kong, Yi-Xiu, et al. "*k*-core: Theories and applications." Physics Reports 832 (2019): 1-32.

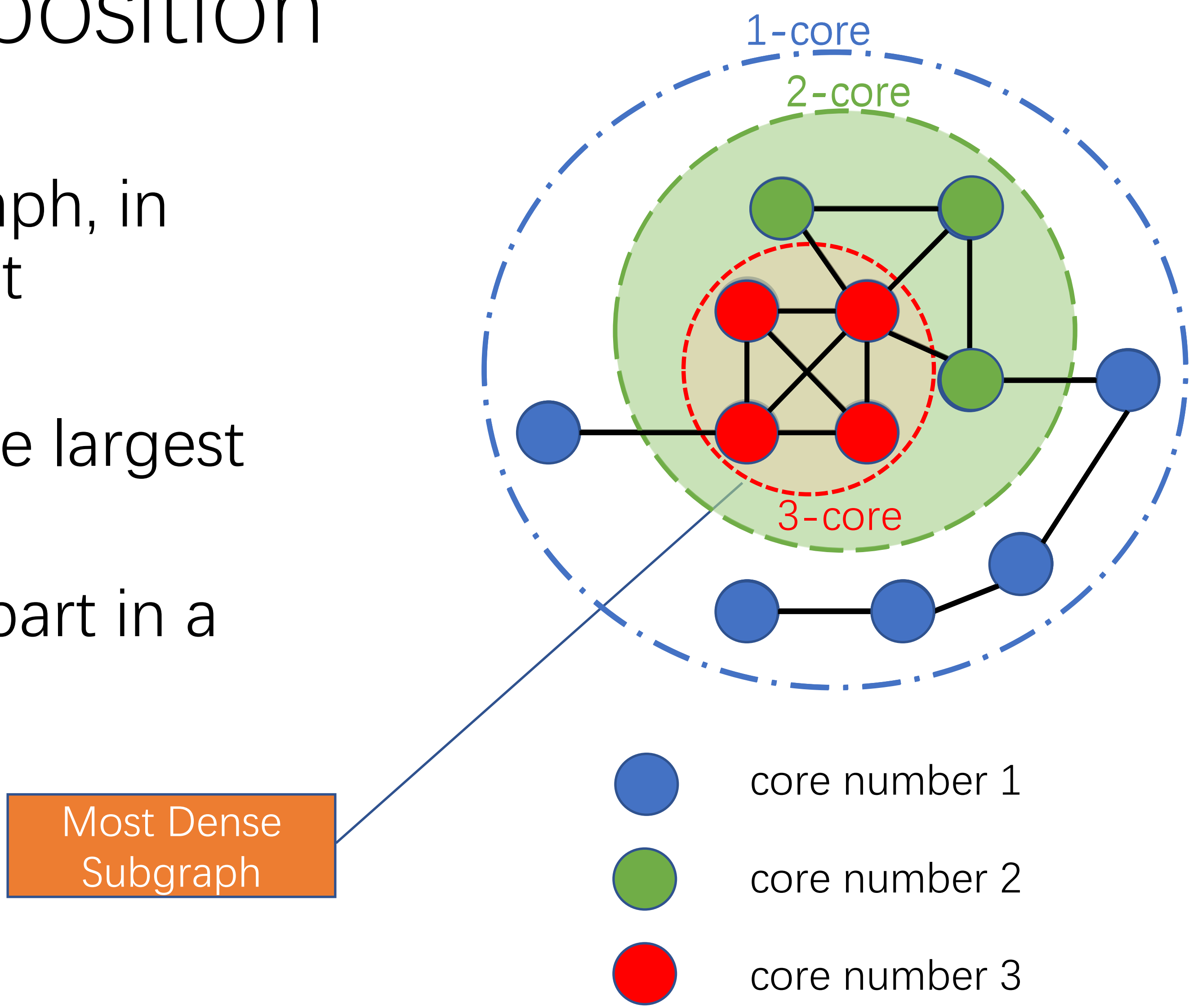
Graph Analytics

- Large data graphs require data analytics
- Graph databases:
  - **Neo4j** https://neo4j.com/
  - Microsoft SQL Server
  - Amazon Neptune
- Graph algorithms:
  - Strongly Connected Components
  - Minimum Spanning Forest
  - Shortest Path Distance
  - k-Core



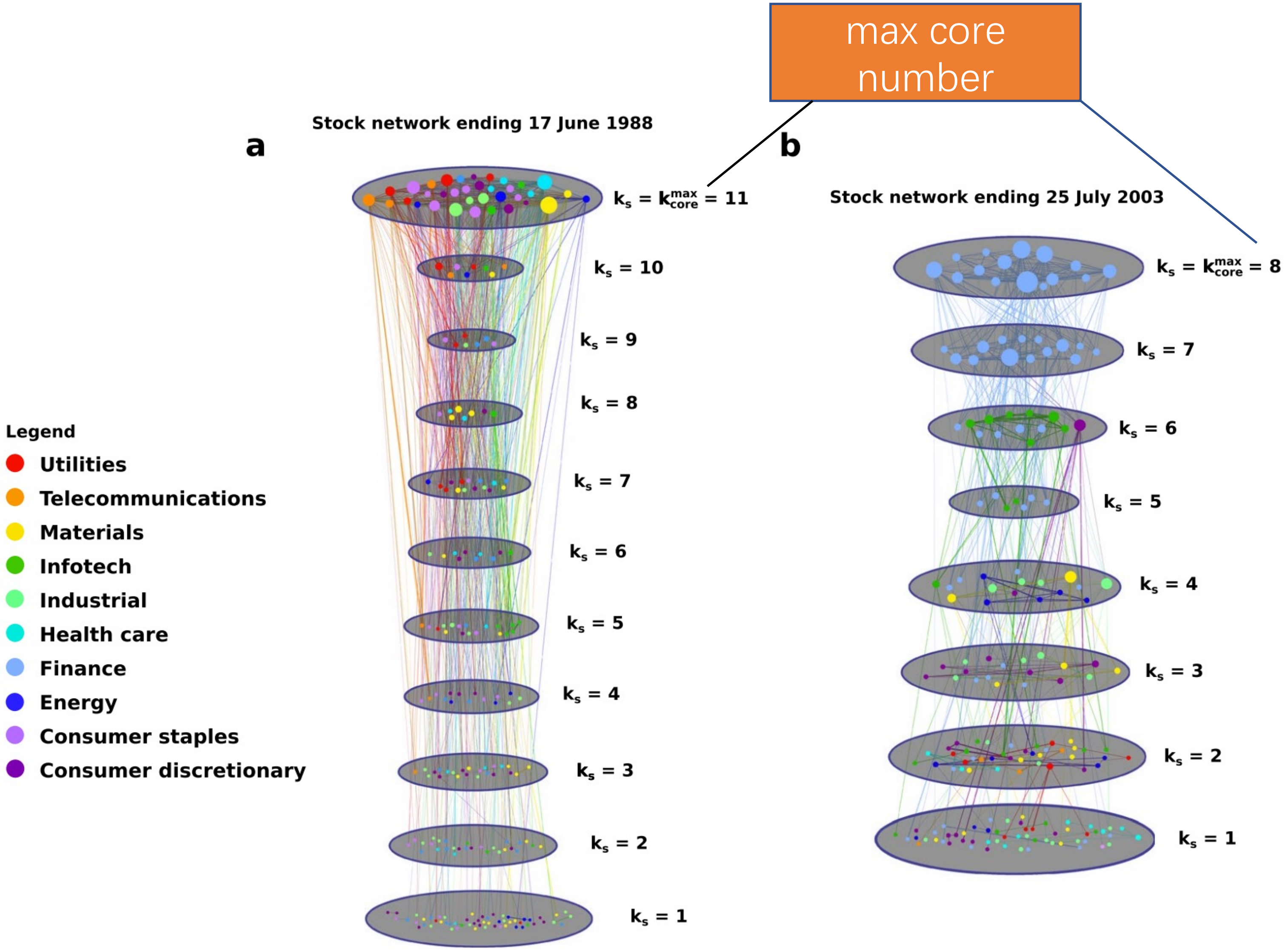
k-Core Decomposition

- Find the largest subgraph, in which each node has at least k neighbours
- The **core number** is the largest value of k
- It is to find the **dense** part in a graph.



Applications in Economy

Stock Networks	
Vertices	Stocks
Edges	Interaction

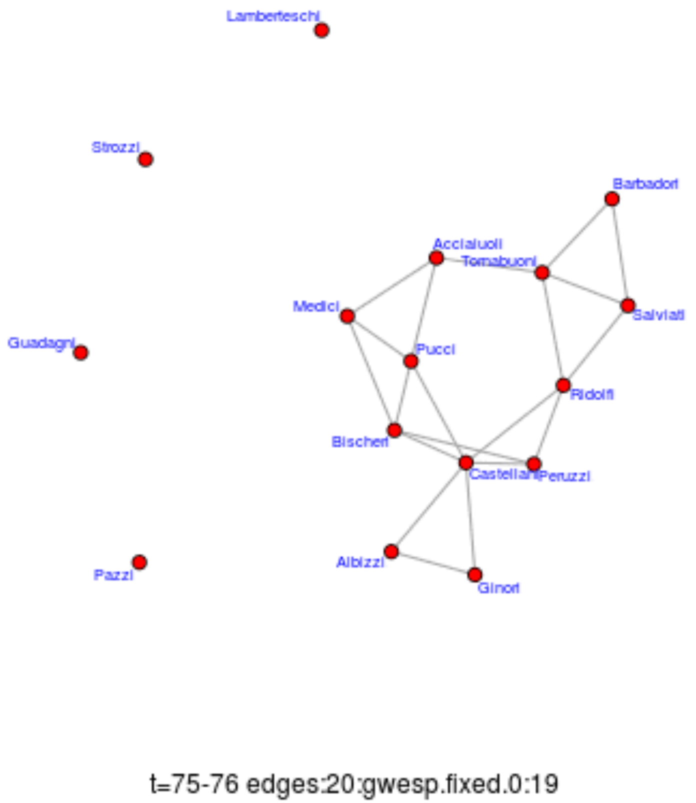


- The **max core** is dominated by the **Finance** in 2003 [2]
- **Finance** has huge effects to economy

[2] Burleson-Lesser, Kate, et al. "K-core robustness in ecological and financial networks." Scientific reports 10.1 (2020): 1-14.

Dynamic Graphs

- In practice, all above graphs can be dynamic
- Dynamic graphs change with new edges inserted or old edges removed, e.g. temporal graphs
- The core numbers have to be updated
- **Recalculate** the core numbers is expensive for large graph



A temporal graph with time-evolving edges [3]. Each edge has a time stamp.

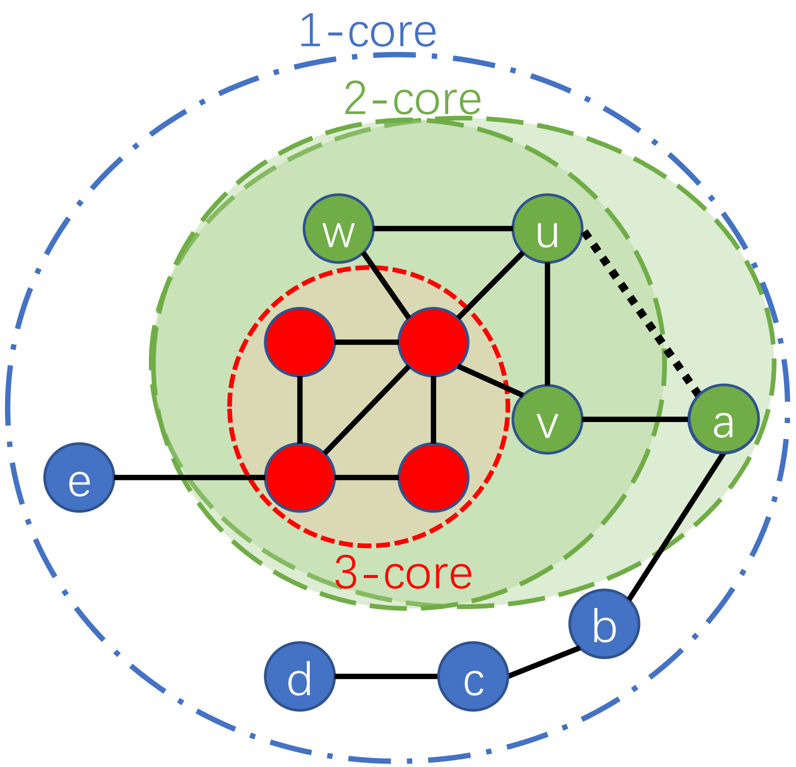
[3] Lotito, Quintino Francesco, and Alberto Montresor. "Efficient Algorithms to Mine Maximal Span-Trusses From Temporal Graphs." *arXiv* (2020).

k-Core Maintenance

- Maintain the core numbers in dynamic graphs when inserting or removing one edge.
- Identify two set: V^* and V^+

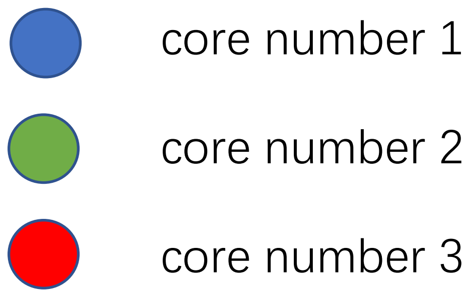
V^*	All vertices with core number changed
V^+	All searched vertices

$$V^* \subseteq V^+$$



$V^* = \{a\}$

$V^+ = \{a, b, c, d\}$



Sequential k-Core Maintenance Algorithms

Insert or remove 100,000 edges

	Insert (seconds)						Remove (seconds)					
Dataset	OrderInsert	Trav-2	Trav-3	Trav-4	Trav-5	Trav-6	OrderRemoval	Trav-2	Trav-3	Trav-4	Trav-5	Trav-6
Facebook	**0.16**	*3.52*	4.07	5.91	10.52	16.95	**0.10**	*0.50*	1.63	4.14	9.70	17.77
Youtube	**0.26**	*2.51*	2.88	4.01	6.13	9.71	**0.28**	*0.61*	1.42	3.19	6.28	11.32
DBLP	**0.16**	1.80	*1.20*	2.31	6.32	17.65	**0.11**	*0.21*	0.61	1.88	5.49	15.78
Patents	**0.88**	2,944.14	1,805.98	1,173.20	845.93	*810.00*	**0.38**	*0.92*	4.22	18.57	75.06	276.37
Orkut	**1.14**	954.36	793.82	*780.69*	996.43	1,576.63	**0.71**	*7.75*	36.80	136.78	428.85	1,089.38
LiveJournal	**0.53**	149.56	90.93	*76.57*	125.29	285.50	**0.33**	*1.66*	6.59	24.56	86.10	233.92
Gowalla	**0.18**	*1.04*	1.37	2.21	3.78	6.38	**0.14**	*0.35*	0.84	1.82	3.45	6.22
CA	**0.52**	15.14	4.20	2.08	1.37	*1.11*	0.16	**0.08**	*0.13*	0.19	0.26	0.33
Pokec	**0.77**	1,726.04	*1,603.80*	1,650.37	1,876.48	2,338.78	**0.32**	*4.86*	53.13	259.93	756.40	1,652.88
BerkStan	**0.37**	*6.37*	7.29	9.37	13.14	16.19	**0.52**	*2.55*	5.04	8.33	12.45	17.34
Google	**0.37**	*1.01*	1.25	2.44	4.81	9.27	**0.25**	*0.46*	0.96	2.08	4.32	8.75

- Existing **Order** algorithm is much faster than the **Traversal** algorithm [4]
- Existing **Order** algorithm maintains an order for all vertices (k-order) to reduce the size of V^+

[4] Yikai Zhang, Jeffrey Xu Yu, Ying Zhang, and Lu Qin. A fast order-based approach for core maintenance. ICDE, 2017.

Order vs Traversal

Traversal

$V^* = \{a\}$

$V^+ = \{a, b, c, d\}$

Order

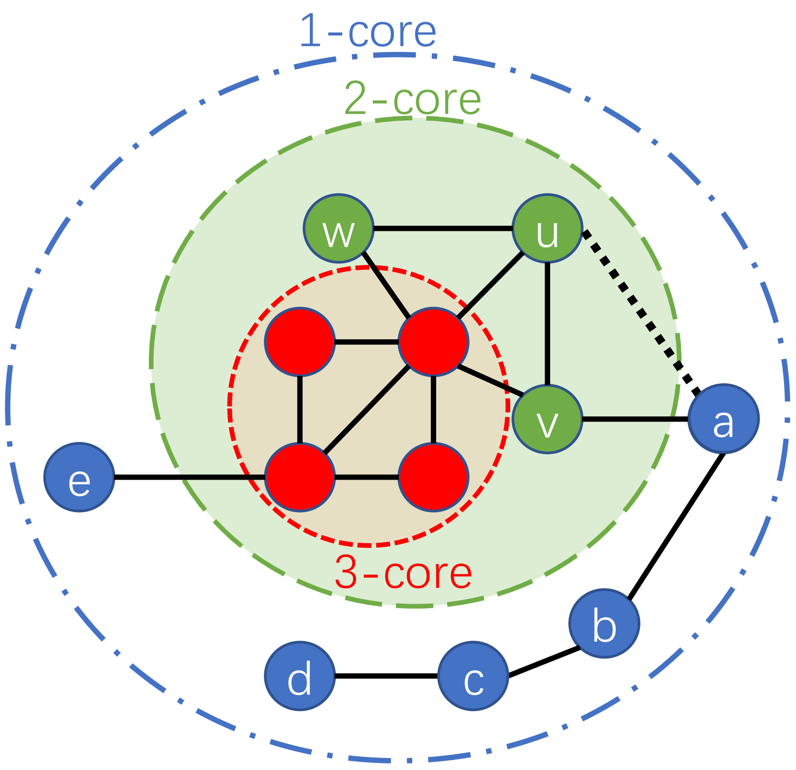
$V^* = \{a\}$

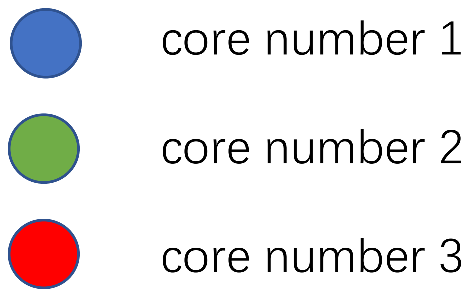
$V^+ = \{a\}$

k-order for vertices with core number 1

$O_1 = \{d, c, b, a, e\}$

Traverse vertices with k-order from a, so b, c, d are omitted

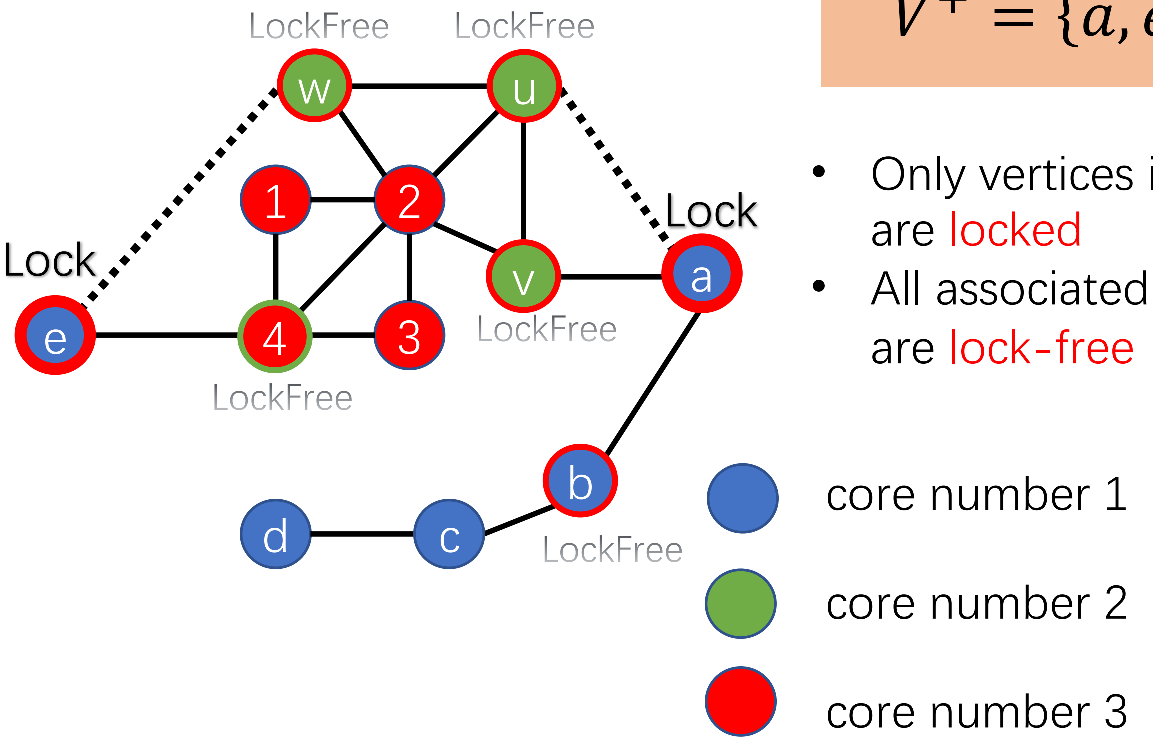




Parallel k-Core Maintenance

- Existing **parallel** methods [7, 8, 9] are based on Traversal algorithm
- We first propose a Simplified-Order algorithm
- Then, we propose a Parallel-Order algorithm by using locks for synchronization

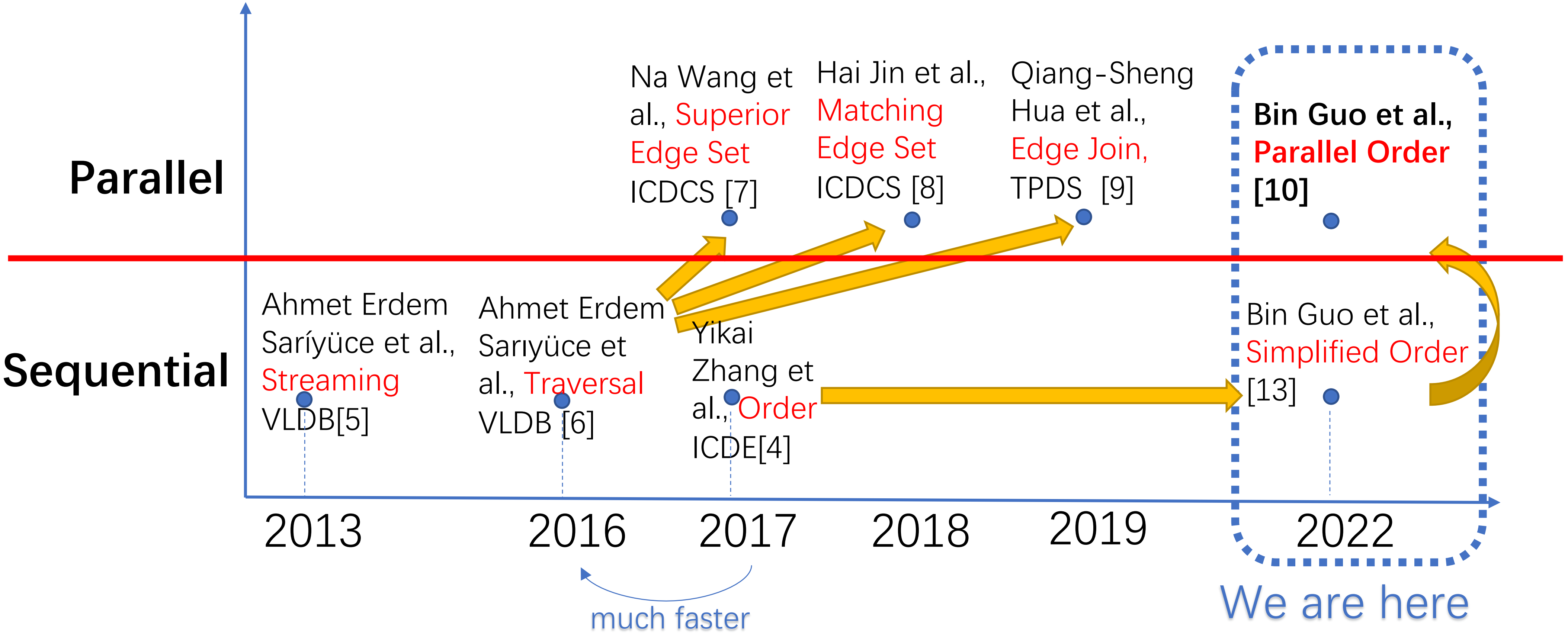
$V^* = \{a, e\}$
$V^+ = \{a, e\}$



- Only vertices in V^+ are locked
- All associated edges are lock-free

[7] Na Wang, Dongxiao Yu, Hai Jin, Chen Qian, Xia Xie, and Qiang-Sheng Hua. Parallel algorithm for core maintenance in dynamic graphs. ICDCS 2017
[8] Hai Jin, Na Wang, Dongxiao Yu, Qiang Sheng Hua, Xuanhua Shi, and Xia Xie. Core Maintenance in Dynamic Graphs: A Parallel Approach Based on Matching. TPDS 2018
[9] Qiang-Sheng Hua, Yuliang Shi, Dongxiao Yu, Hai Jin, Jiguo Yu, Zhipen Cai, Xiuzhen Cheng, and Hanhua Chen. Faster parallel core maintenance algorithms in dynamic graphs. TPDS 2019.

Studies of k-Core Maintenance



- Our methodology can also be applied to k-**truss maintenance**, k-**clique maintenance**, **SCC maintenance**

Time Complexity

Parallel	Worst-case (O) $\mathcal{W}$	Worst-case (O) $\mathcal{D}$	Best-case (O) $\mathcal{W}$	Best-case (O) $\mathcal{D}$
Insert	$m'\lvert E^+\rvert \log \lvert E^+\rvert$	$m'\lvert E^+\rvert \log \lvert E^+\rvert$	$m'\lvert E^+\rvert \log \lvert E^+\rvert$	$\lvert E^+\rvert \log \lvert E^+\rvert + m'\lvert V^*\rvert$
Remove	$m'\lvert E^*\rvert$	$m'\lvert E^*\rvert$	$m'\lvert E^*\rvert$	$\lvert E^*\rvert + m'\lvert V^*\rvert$

- Here, m' is total number of inserted edges
- And E^+ is all associated edges for all vertices in V^+, so does V^*

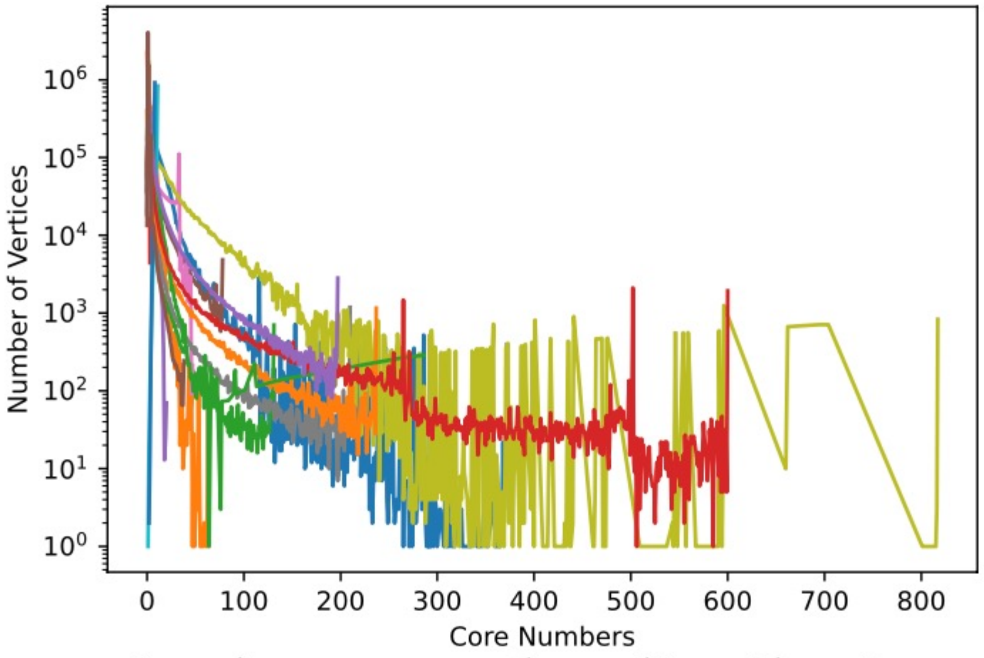
- In the worst case, all workers execute as one blocking chain and reduce to sequential version
- The worst case is unlikely to happen over real graphs
- The best case has high speedups

Tested Graphs

Graph	$n = \|V\|$	$m = \|E\|$	AvgDeg	Max k	Category	Type
livej	4,847,571	68,993,773	14.23	372	Social Networks	Static Graphs
patent	6,009,555	16,518,948	2.75	64	Social Networks	Static Graphs
wikitalk	2,394,385	5,021,410	2.10	131	Social Networks	Static Graphs
roadNet-CA	1,971,281	5,533,214	2.81	3	Road Network	Static Graphs
dbpedia	3,966,925	13,820,853	3.48	20	Social Networks	Static Graphs
baidu	2,141,301	17,794,839	8.31	78	Social Networks	Static Graphs
pokec	1,632,804	30,622,564	18.75	47	Social Networks	Static Graphs
wiki-talk-en	2,987,536	24,981,163	8.36	210	Hyperlink Network	Static Graphs
wiki-links-en	5,710,993	130,160,392	22.79	821	Hyperlink Network	Static Graphs
ER	1,000,000	8,000,000	8.00	11	Synthetic Network	Static Graphs
BA	1,000,000	8,000,000	8.00	8	Synthetic Network	Static Graphs
RMAT	1,000,000	8,000,000	8.00	237	Synthetic Network	Static Graphs
DBLP	1,824,701	29,487,744	16.17	286	Temporal Graphs	Dynamic Graphs
Flickr	2,302,926	33,140,017	14.41	600	Temporal Graphs	Dynamic Graphs
StackOverflow	2,601,977	63,497,050	24.41	198	Temporal Graphs	Dynamic Graphs
wiki-edits-sh	4,589,850	40,578,944	8.84	47	Temporal Graphs	Dynamic Graphs

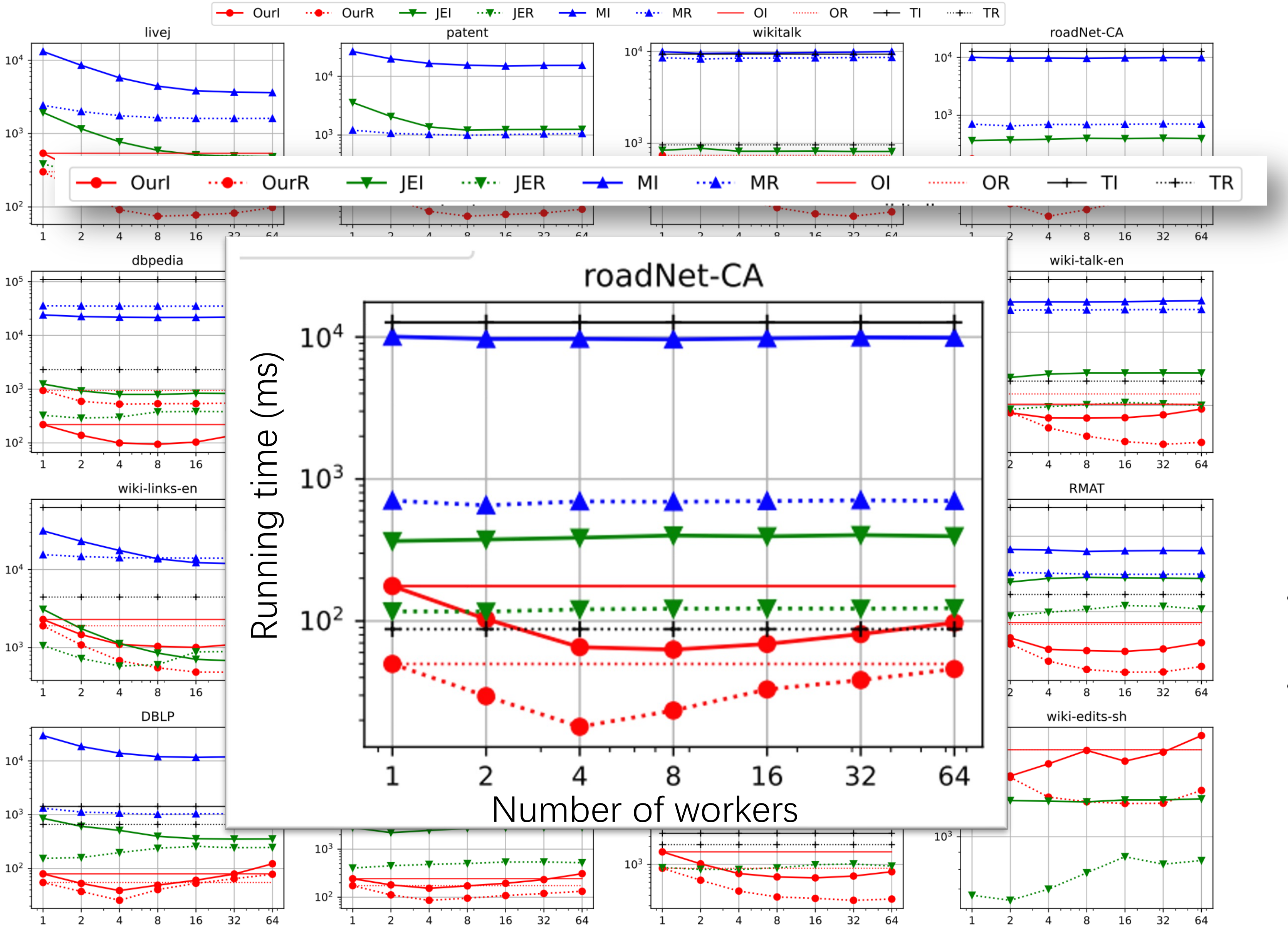
- For static graphs, randomly select 100,000 edges
- For dynamic graphs, select 100,000 edges in a continuous time range

The Core Number Distribution of Vertices



- The number of vertices (y-axis) with a same core number (x-axis)

- A large portion of vertices have small core numbers
- All compared methods have **limited parallelism**: vertices with the same core number can only be processed by a single worker at the same time
- Our methods do not have such limitation



OurI	**Our Insert**
OurR	**Our Remove**
JEI	Join Edge Insert
JER	Join Edge Remove
MI	Match Edge Insert
MR	Match Edge Remove
OI	Sequential Order Insert
OR	Sequential Order Remove
TI	Sequential Traversal Insert
TR	Sequential Traversal Remove

- With 1-worker, **OurI** and **OurR** is faster than **JEI** and **JER**
- With 4 to 16-worker, **OurI** and **OurR** always has higher speedups than **JEI** and **JER**

Graph	1-worker vs 16-worker OurI	OurR	JEI	JER	MI	MR	1-worker OurI vs JEI	MI	1-worker OurR vs JER	MR	16-worker OurI vs JEI	MI	16-worker OurR vs JER	MR
livej	3.2	3.9	3.8	0.8	**3.4**	1.5	3.6	24.4	0.7	4.5	3.0	22.7	3.0	**9.5**
patent	3.4	3.4	2.9	0.9	1.8	1.2	11.0	81.2	0.9	**3.7**	13.0	158.3	3.5	10.7
wikitalk	1.8	4.4	1.0	0.7	**1.0**	**1.0**	1.3	15.7	0.5	13.5	2.3	27.6	1.4	24.1
roadNet-CA	2.6	1.5	0.9	1.0	1.0	1.0	2.1	57.1	0.7	4.0	5.7	141.9	1.8	10.1
dbpedia	2.1	1.8	1.5	0.8	1.1	1.0	5.7	109.4	1.5	**162.0**	8.1	208.1	3.7	**337.9**
baidu	3.5	5.2	1.2	0.7	1.2	1.0	1.9	27.6	0.7	11.7	5.7	82.1	3.6	40.7
pokec	4.8	4.7	1.4	0.8	1.3	1.2	6.0	76.9	0.7	4.0	20.9	288.8	4.4	15.9
wiki-talk-en	1.5	4.5	0.8	0.8	1.0	1.0	2.2	25.4	0.8	19.5	4.1	38.2	1.6	29.7
wiki-links-en	2.3	3.9	4.4	1.2	2.6	1.1	1.3	13.7	0.5	6.8	0.7	**12.2**	0.9	13.9
ER	3.8	4.4	0.9	1.0	1.1	1.0	16.8	700.3	1.7	11.8	70.9	2500.7	6.6	45.5
BA	5.4	2.9	0.9	0.9	1.1	1.0	49.6	**2555.3**	0.6	25.9	289.1	**12552.9**	3.5	139.7
RMAT	2.4	4.3	0.7	0.6	1.1	1.2	2.8	**10.2**	1.0	5.2	9.5	21.5	4.1	10.5
DBLP	1.3	1.0	2.4	0.6	2.5	1.3	10.8	371.7	1.9	16.7	5.9	192.4	4.3	17.2
flickr	1.2	1.6	1.0	0.7	1.2	**1.8**	11.6	506.7	1.7	62.3	14.3	542.3	2.8	44.1
StackOverflow	2.8	3.2	1.3	0.9	2.0	1.2	4.3	93.5	0.5	7.1	8.8	130.0	1.7	17.1
wiki-edits-sh	1.1	1.4	1.1	0.8	-	-	0.8	-	0.4	-	0.8	-	0.5	-

Table 2: Compare the speedups.

- For 1-worker vs 16-workers, **OurI** and **OurR** have speedups up to **5.4** and **5.2**, respectively, higher than JEI and JIR
- For 1-worker **OurI** vs **JEI**, **OurI** has speedups up to **49.6**
- For 1-worker **OurR** vs **JER**, **OurR** has speedups up to **1.9**
- For 16-worker **OurI** vs **JEI**, **OurI** has speedups up to **289**
- For 16 –worker **OurR** vs **JER**, **OurR** has speedups up to **6.6**

OurI	**Our Insert**
OurR	**Our Remove**
JEI	Join Edge Insert
JER	Join Edge Remove
MI	Match Edge Insert
MR	Match Edge Remove
OI	Sequential Order Insert
OR	Sequential Order Remove
TI	Sequential Traversal Insert
TR	Sequential Traversal Remove

Summary and Future Work

- We present new parallel core maintenance algorithms:
  - based on the state-of-the-art sequential Order algorithm
  - only the vertices in V^+ are locked and all their associated edges are lock-free, which leads to high parallelism
- My methodology can be applied to:
  - other kinds of graphs, e.g., weighted and probability graphs
  - other sequential graph algorithms, e.g., hierarchical k-core maintenance and k-truss maintenance

Reference

- [1] Kong, Yi-Xiu, et al. "k-core: Theories and applications." Physics Reports 832 (2019): 1-32.
- [2] Burleson-Lesser, Kate, et al. "K-core robustness in ecological and financial networks." Scientific reports 10.1 (2020): 1-14.
- [3] Lotito, Quintino Francesco, and Alberto Montresor. "Efficient Algorithms to Mine Maximal Span-Trusses From Temporal Graphs." *arXiv preprint arXiv:2009.01928* (2020).
- [4] Yikai Zhang, Jeffrey Xu Yu, Ying Zhang, and Lu Qin. A fast order-based approach for core maintenance. In Proceedings - International Conference on Data Engineering, pages 337–348, 2017.
- [5] Ahmet Erdem Saríyüce, Buğra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V Çatalyürek. Streaming algorithms for k -core decomposition. Proceedings of the VLDB Endowment, 6(6):433–444, 2013.
- [6] Ahmet Erdem Sarıyüce, Buğra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V Çatalyürek. Incremental k -core decomposition: algorithms and evaluation. The VLDB Journal, 25(3):425–447, 2016

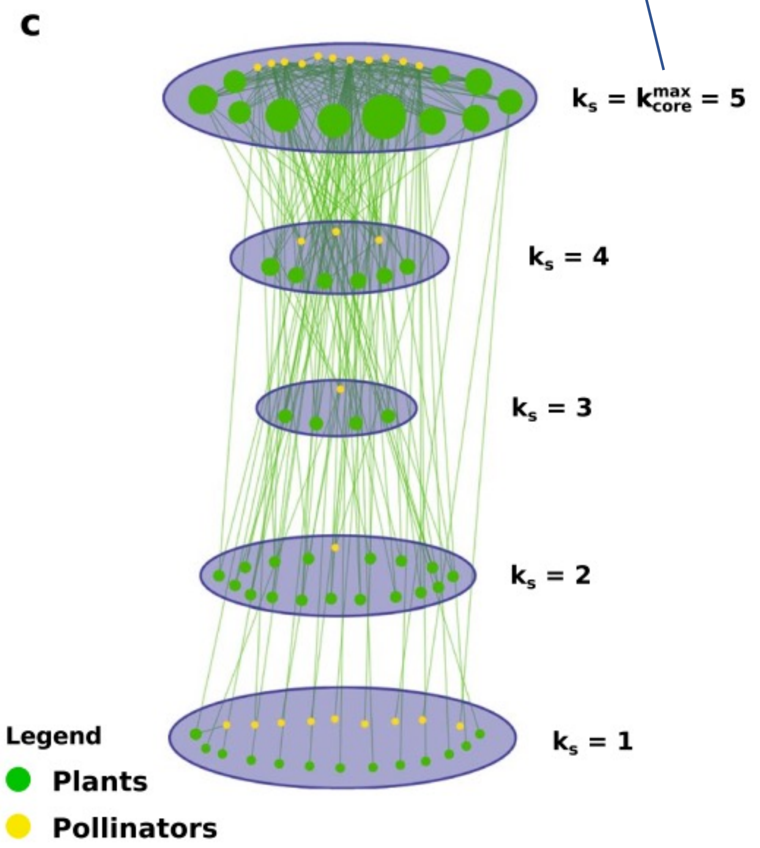
- [7] Na Wang, Dongxiao Yu, Hai Jin, Chen Qian, Xia Xie, and Qiang-Sheng Hua. Parallel algorithm for core maintenance in dynamic graphs. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pages 2366–2371. IEEE, 2017.
- [8] Hai Jin, Na Wang, Dongxiao Yu, Qiang Sheng Hua, Xuanhua Shi, and Xia Xie. Core Maintenance in Dynamic Graphs: A Parallel Approach Based on Matching. IEEE Transactions on Parallel and Distributed Systems, 29(11):2416–2428, nov 2018.
- [9] Qiang-Sheng Hua, Yuliang Shi, Dongxiao Yu, Hai Jin, Jiguo Yu, Zhipen Cai, Xiuzhen Cheng, and Hanhua Chen. Faster parallel core maintenance algorithms in dynamic graphs. IEEE Transactions on Parallel and Distributed Systems, 31(6):1287–1300, 2019.
- [10] Parallel Order-Based Core Maintenance in Dynamic Graphs B Guo, E Sekerinski - arXiv preprint arXiv:2210.14290, 2022 All 2 versions
- [11] Efficient parallel graph trimming by arc-consistency B Guo, E Sekerinski - The Journal of Supercomputing, 2022
- [12] δ-Transitive closures and triangle consistency checking: a new way to evaluate graph pattern queries in large graph databases Y Chen, B Guo, X Huang - The Journal of Supercomputing, 2020
- [13] Guo, Bin, and Emil Sekerinski. "Simplified Algorithms for Order-Based Core Maintenance." *arXiv preprint arXiv:2201.07103* (2022).
- [14] Guo, Bin, and Emil Sekerinski. "New Parallel Order Maintenance Data Structure." arXiv preprint arXiv:2208.07800 (2022).

Applications in Ecology

max core number

Ecological Networks	
Vertices	Plants and Pollinators
Edges	Plant-Pollinator Interaction

- The size of **max-core** are much large than **1-core**
- The extinction of species in **max-core** have huge effect to the mutualistic structure [2]

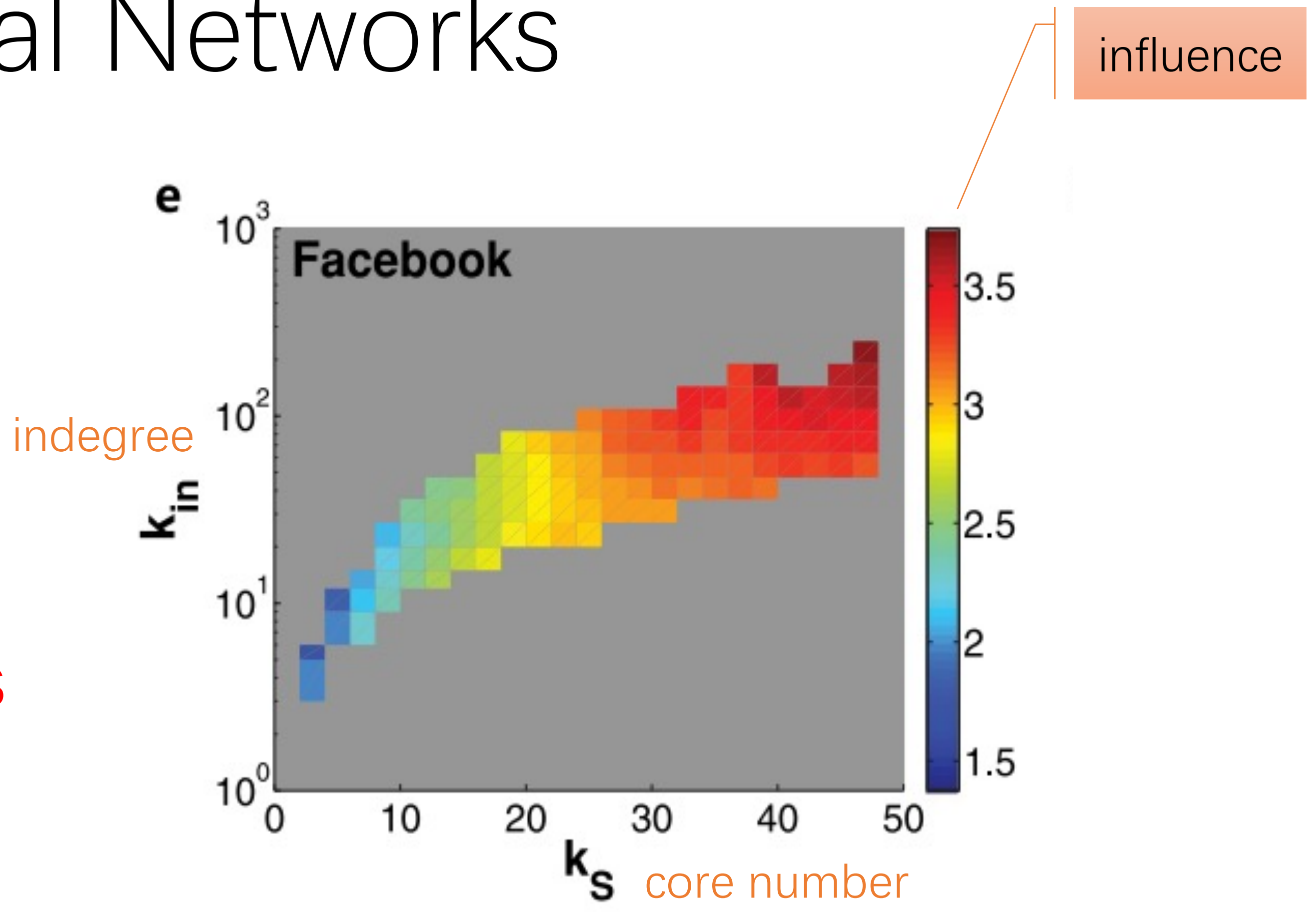


[2] Burleson-Lesser, Kate, et al. "K-core robustness in ecological and financial networks." Scientific reports 10.1 (2020): 1-14.

Applications in Social Networks

Social Networks, e.g. Facebook and Twitter	
Vertices	Individuals
Edges	Relations

- For vertices, larger core numbers and larger indegrees indicate higher influence [1]



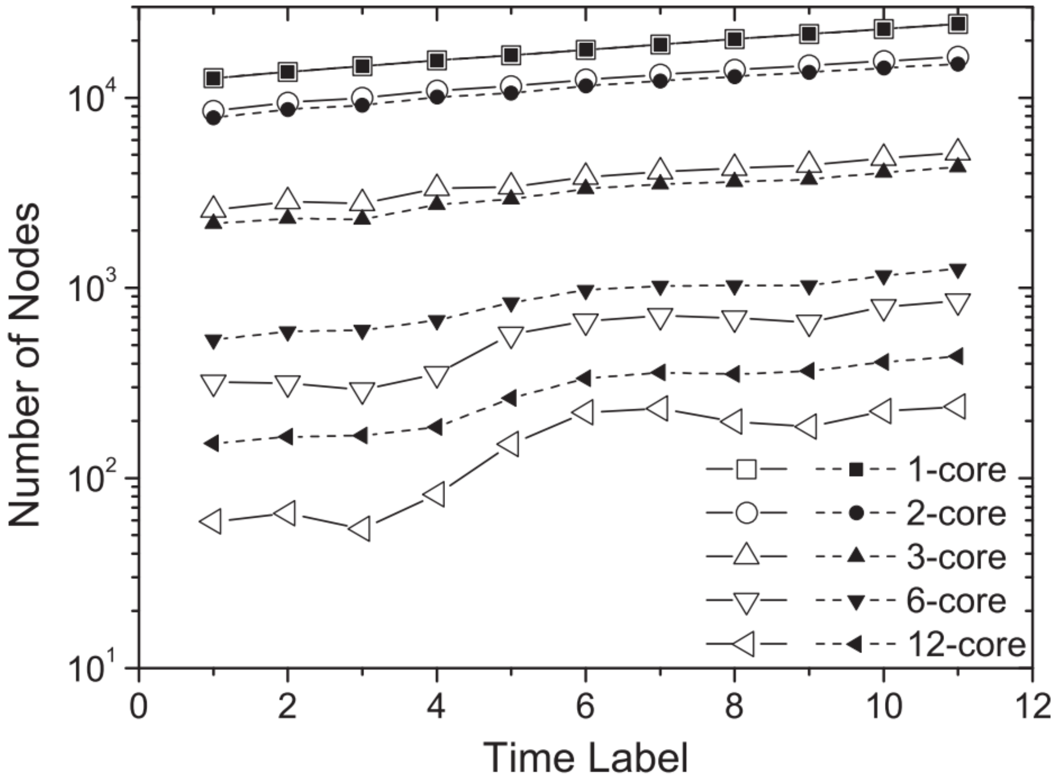
Use core numbers to predicts the influence of spreading in social networks [1]

[1] Kong, Yi-Xiu, et al. "k-core: Theories and applications." Physics Reports 832 (2019): 1-32.

Applications on Analyzing Internet Networks

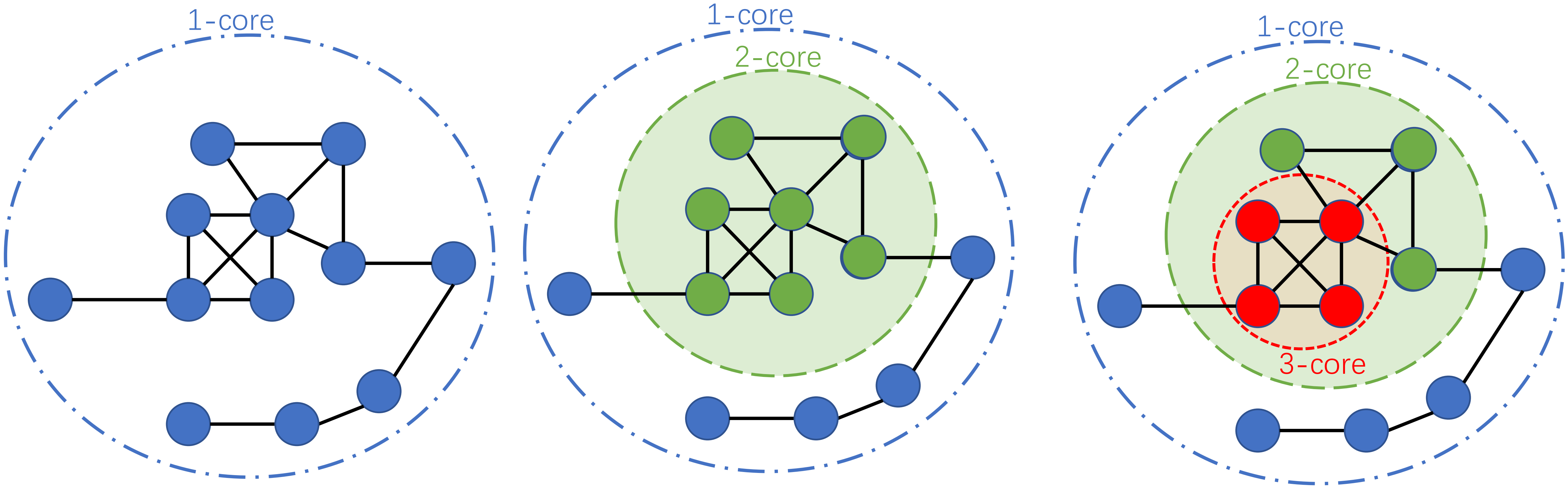
Internet Networks	
Vertices	Websites
Edges	Links Between Websites

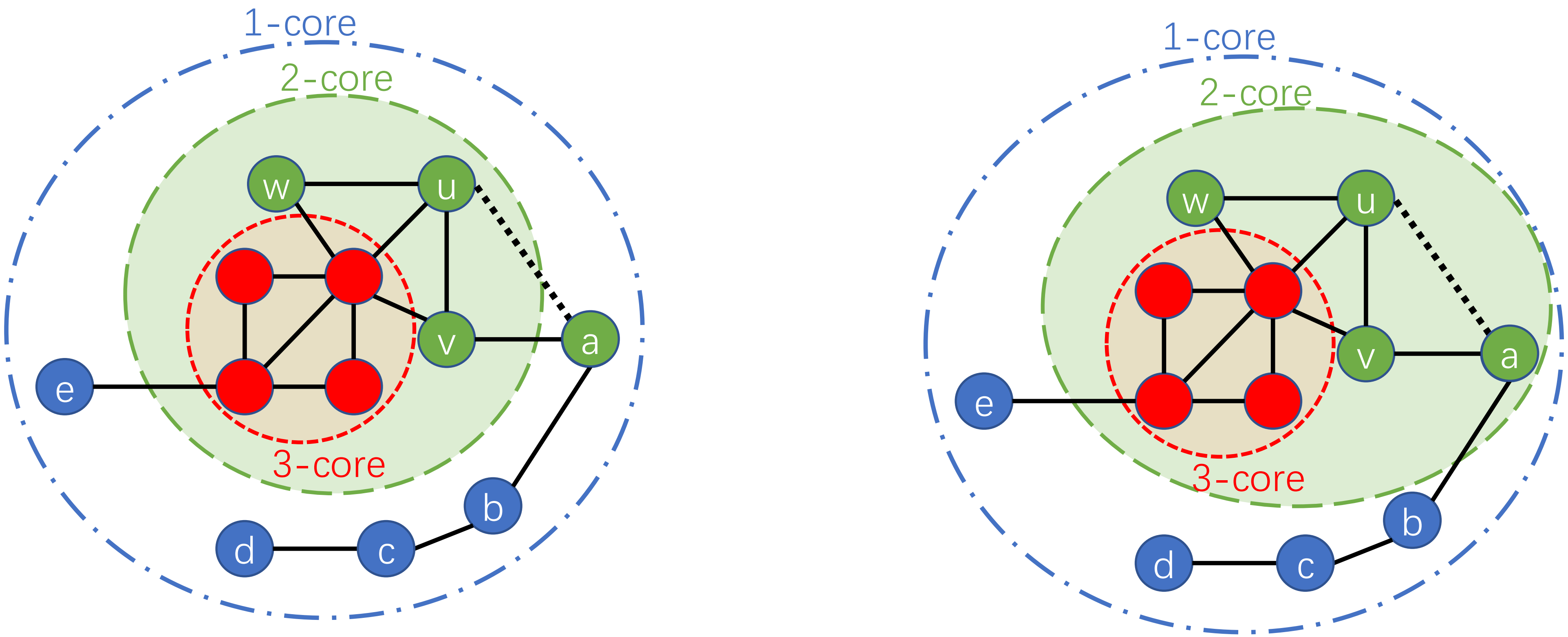
- The sizes of k-cores change with time
- The size of the k-core with a larger k is basically unchanged [1]



From Dec. 2001 to Dec. 2006 with six months interval

[1] Kong, Yi-Xiu, et al. "k-core: Theories and applications." Physics Reports 832 (2019): 1-32.


1-core
1-core
2-core
1-core
2-core
3-core


1-core
2-core
w
u
v
a
e
3-core
b
d
c
1-core
2-core
w
u
v
a
e
3-core
b
d
c