Graph Computing

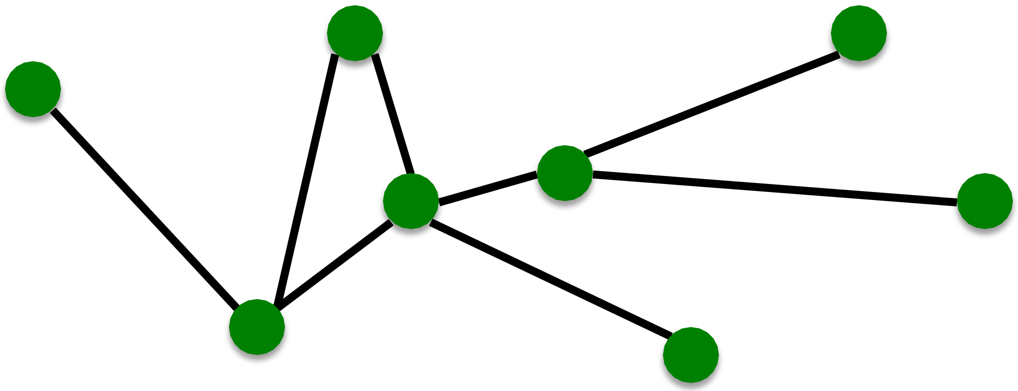
Xiaoyang Wang

My Experience Survey

- The UNSW My Experience survey still open, participation is highly encouraged.

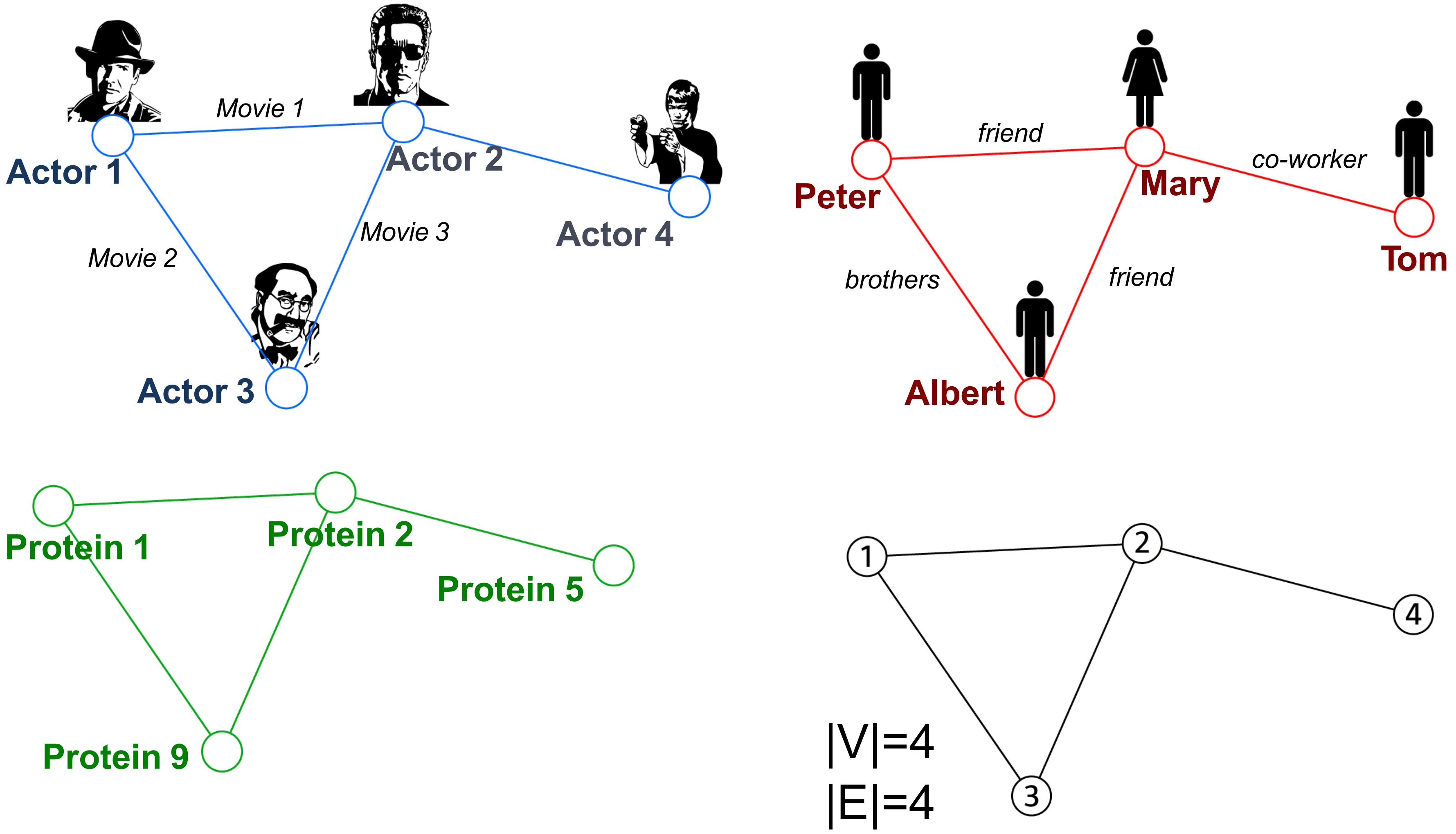
- *“Please participate in the my Experience Survey and take the opportunity to share your constructive thoughts on your 2022 learning experience.*
- *Your contributions help your teachers and shape the future of education at UNSW.”*
- More information: https://www.student.unsw.edu.au/myexperience

What is Graph



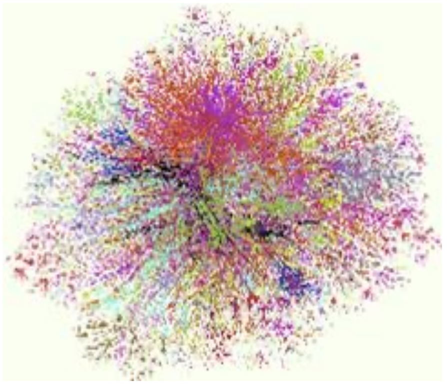
- Objects: nodes, vertices — V
- Interactions: links, edges — E
- Systems: networks, graphs — G(V, E)

Graph is a common language



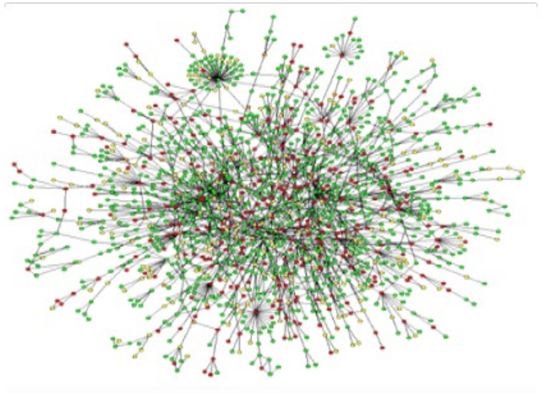
Graph is everywhere

Common model across different fields:



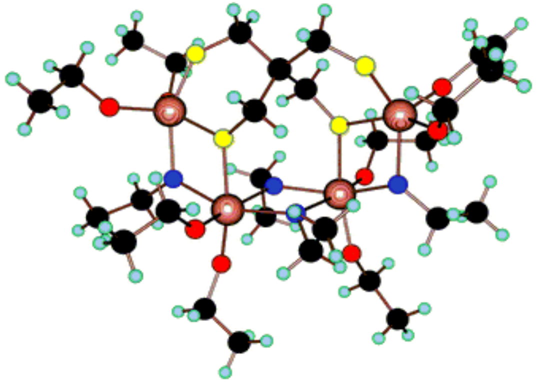
Web Graph

Social Network



Protein Interaction Network

Road Network

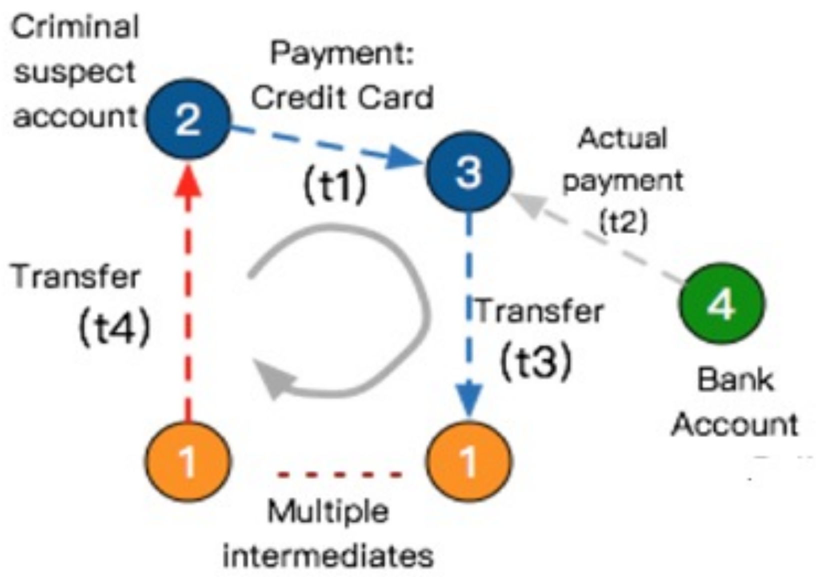


Chemical Compound

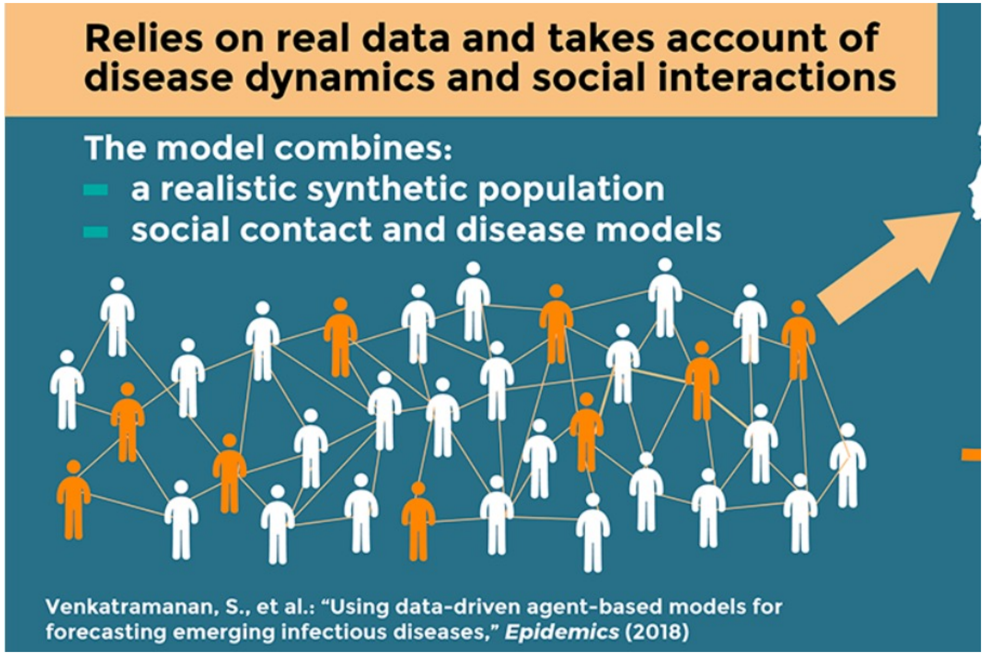
Ontology Graph

Why graphs

- Networks / graphs are everywhere, and we live in a highly-connected world.
- In many applications, we need analyze in the context of networks, not just individuals.



Money laundering detection



Predict the spread of information

Why graphs

- Event Driven Investment

Why graphs

- Can Computer Do It Automatically

Why graphs

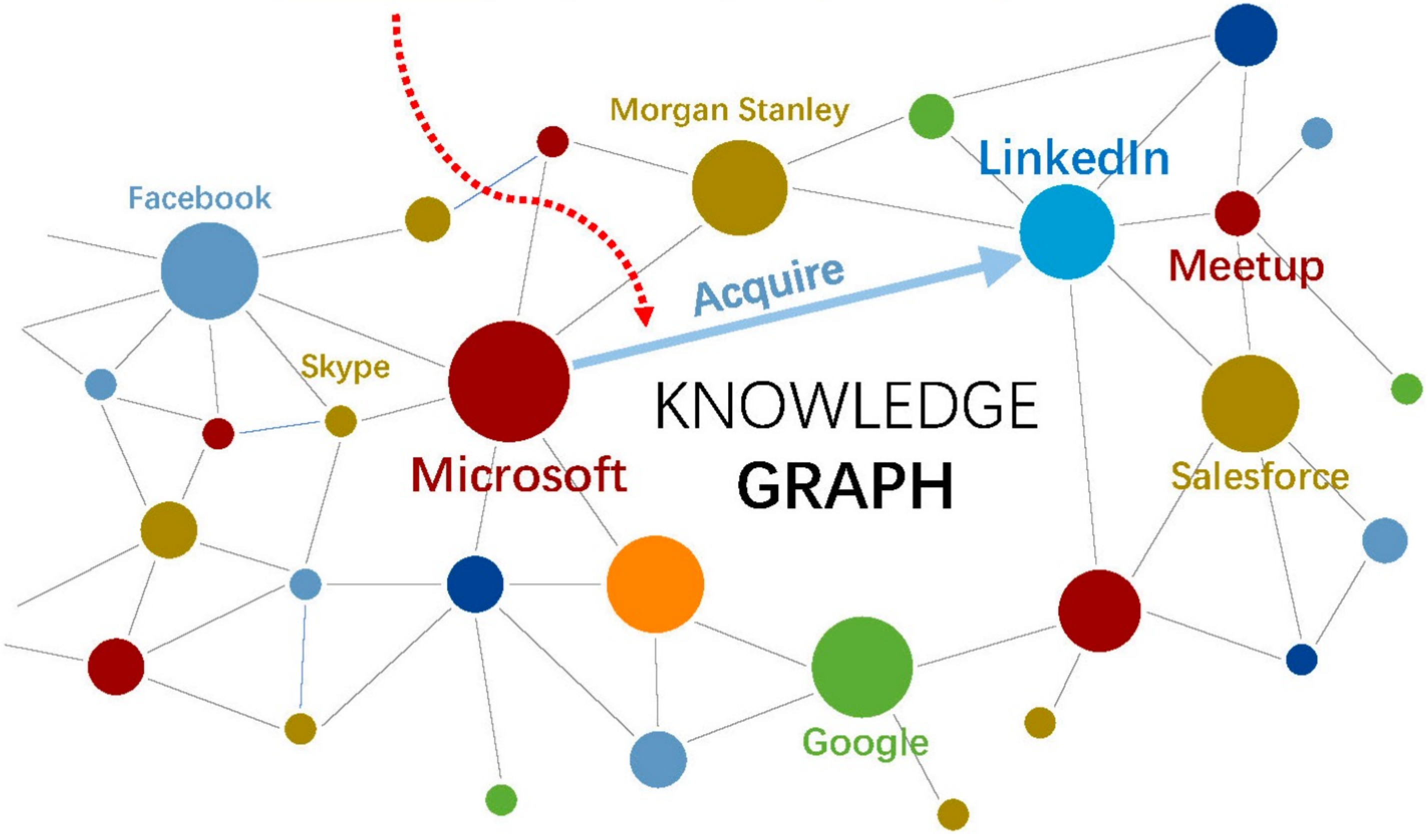
- Can We Do Better
- The financial market reaction of events is also influenced by the lead-lag effect, which is driven by internal relationships.
- For example, the event “Microsoft buys LinkedIn” will also influence the stock movements of upstream and downstream firms and competitors, such as Salesforce and Meetup.
- An event on a raw material will also impact various downstream products in different propagation speed over industrial chains, such as current energy crisis and cold weather.

Why graphs

- How

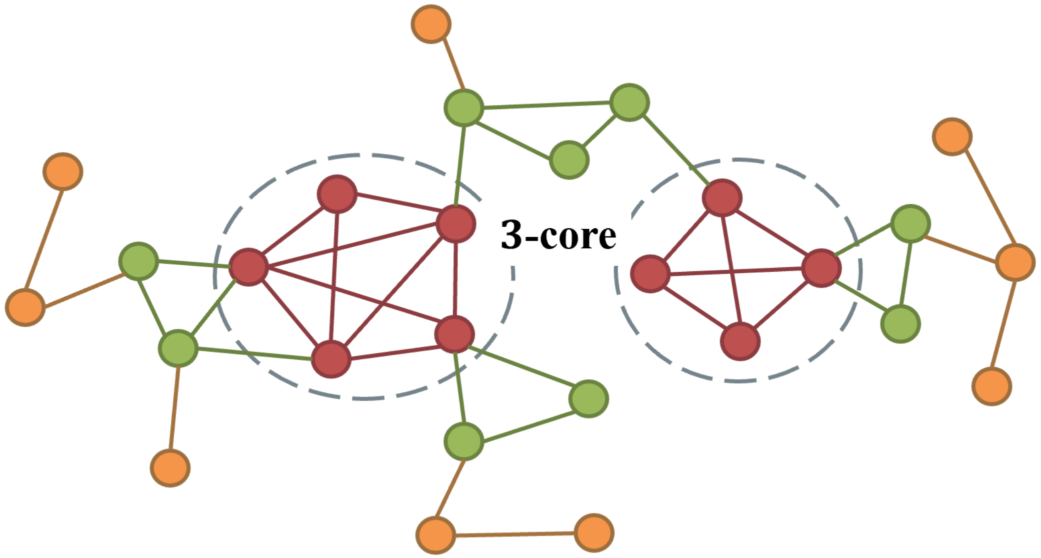
News Text: Microsoft officially closes its $26.2B acquisition of LinkedIn

Event Tuple: (A=Microsoft, P=acquire, O=LinkedIn)



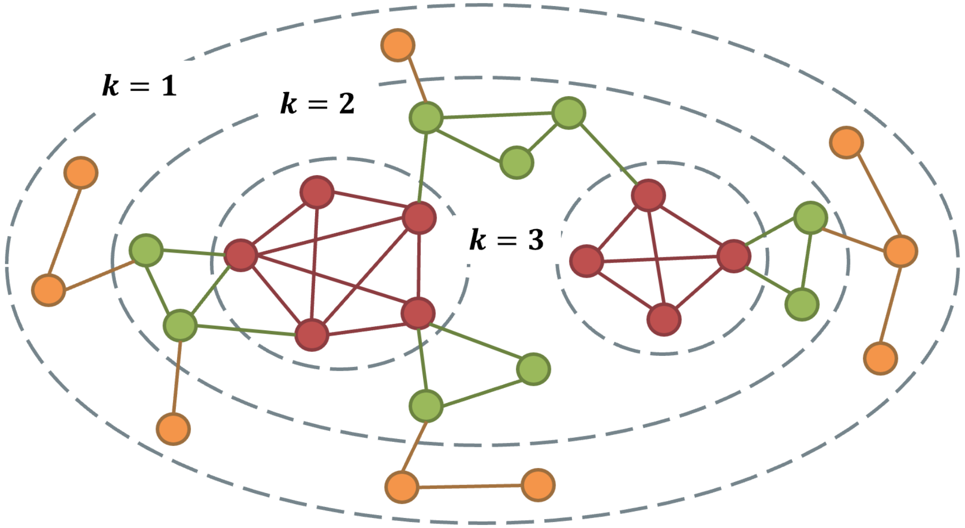
Cohesive Subgraphs

- Clique, k-core, k-truss, k-ECC, k-VCC, ……
- In some models, a value k can be used to capture the cohesiveness of the subgraph.



Hierarchical Graph Decomposition

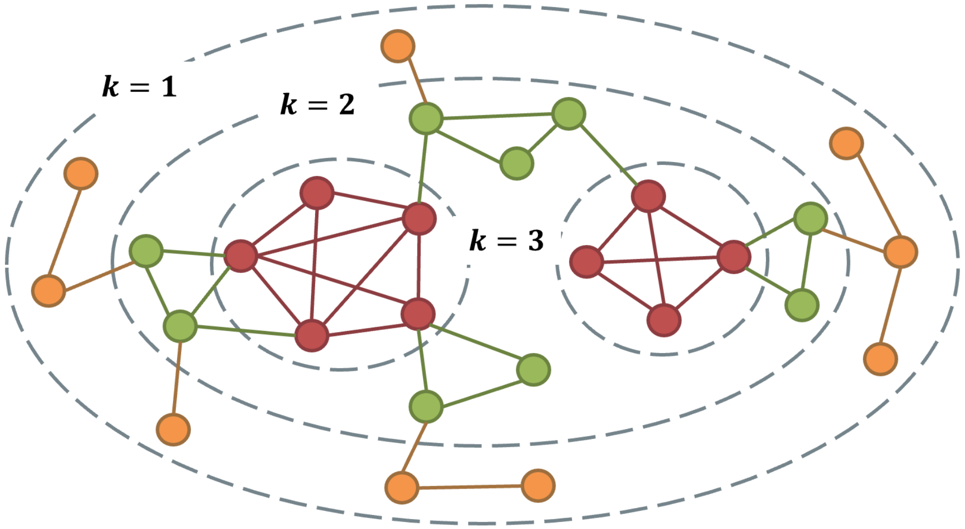
- Varying the possible k values (say $k_1, k_2, \ldots\ldots, k_h$) on the graph G
- h subgraphs $\{S_1, S_2, \ldots\ldots, S_h\}$ with $S_i \supseteq S_j$ for any $i < j$, where $S_1 = G$ and S_i is a subgraph containing a set of connected components.



Hierarchical Graph Decomposition

- Decomposition number of a vertex: $dn(v) = i, v \in S_i$ and $v \notin S_{i+1}$

- k-core, k-truss, k-ECC, ……

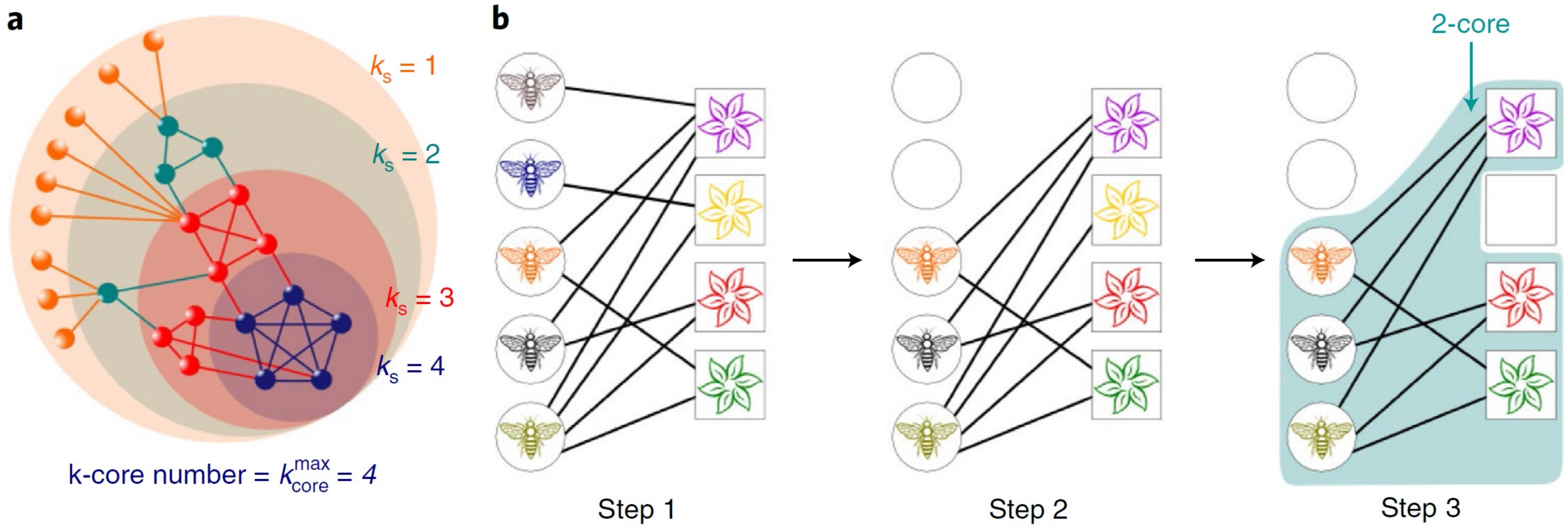


Application Summary

- Network modeling and analysis
- Network Visualization
- Reasoning the collapse of a network
- Prevent Network Unraveling
- Discovering Influential Nodes
- Community Discovery
- Anomaly Detection
- Protein function prediction
- ……

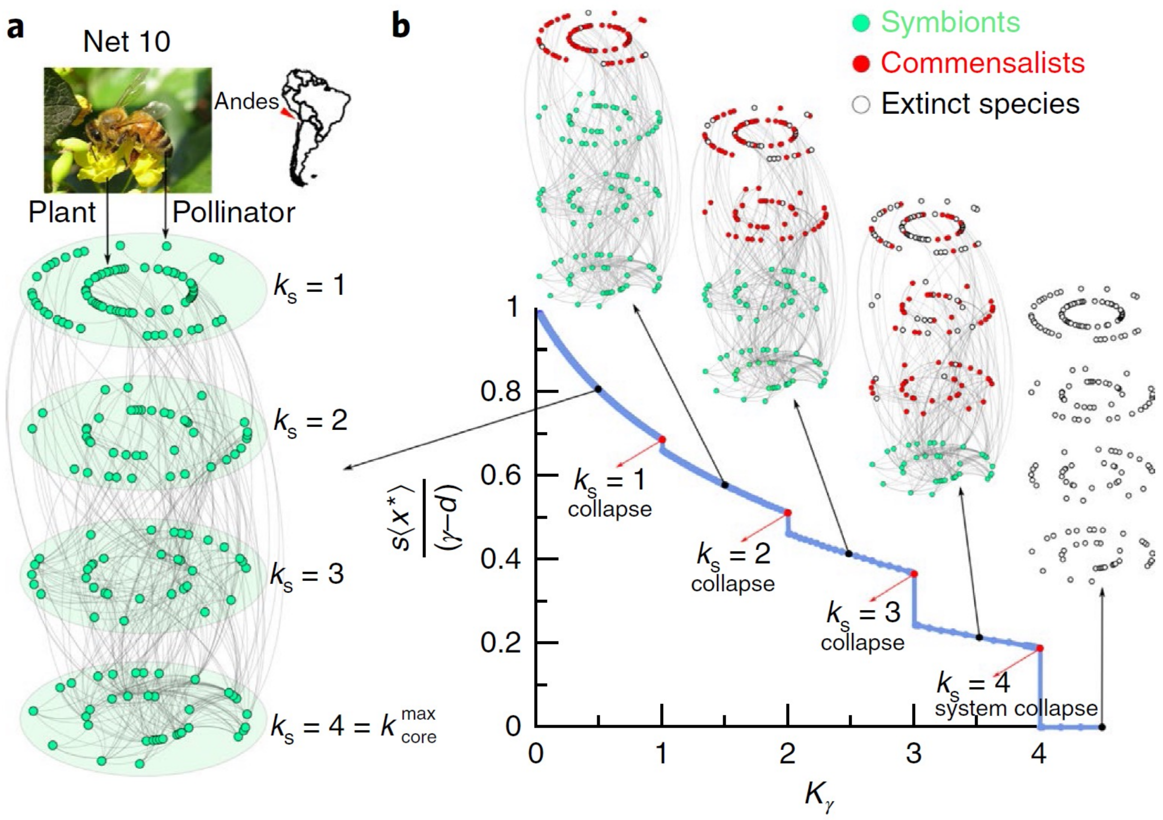
Application – Network Modeling and Analysis

Biology: a cohesive subgraph as a predictor of structural collapse in mutualistic ecosystems [Nature Physics 2018]



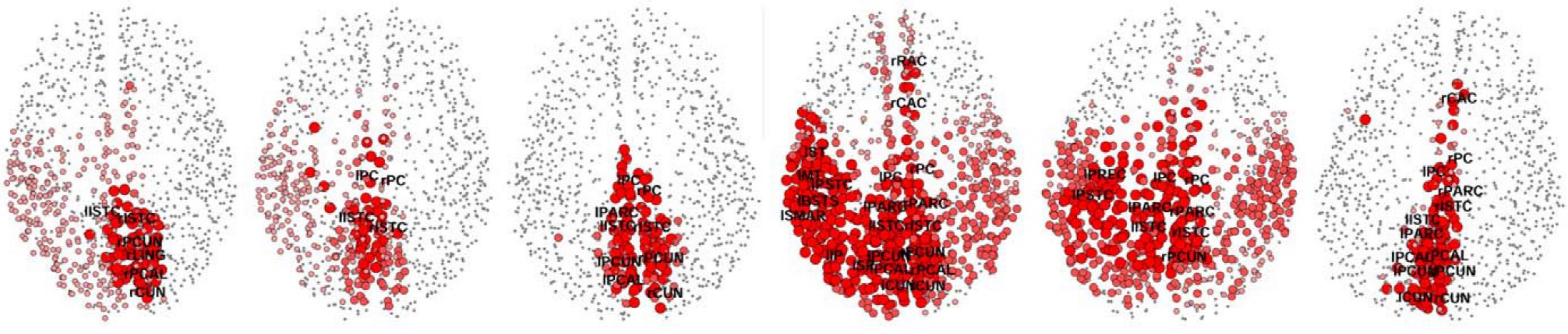
Application – Network Modeling and Analysis

Biology: the hierarchical decomposition reveals the tipping points of structural collapse in mutualistic ecosystems [Nature Physics 2018]



Application – Network Modeling and Analysis

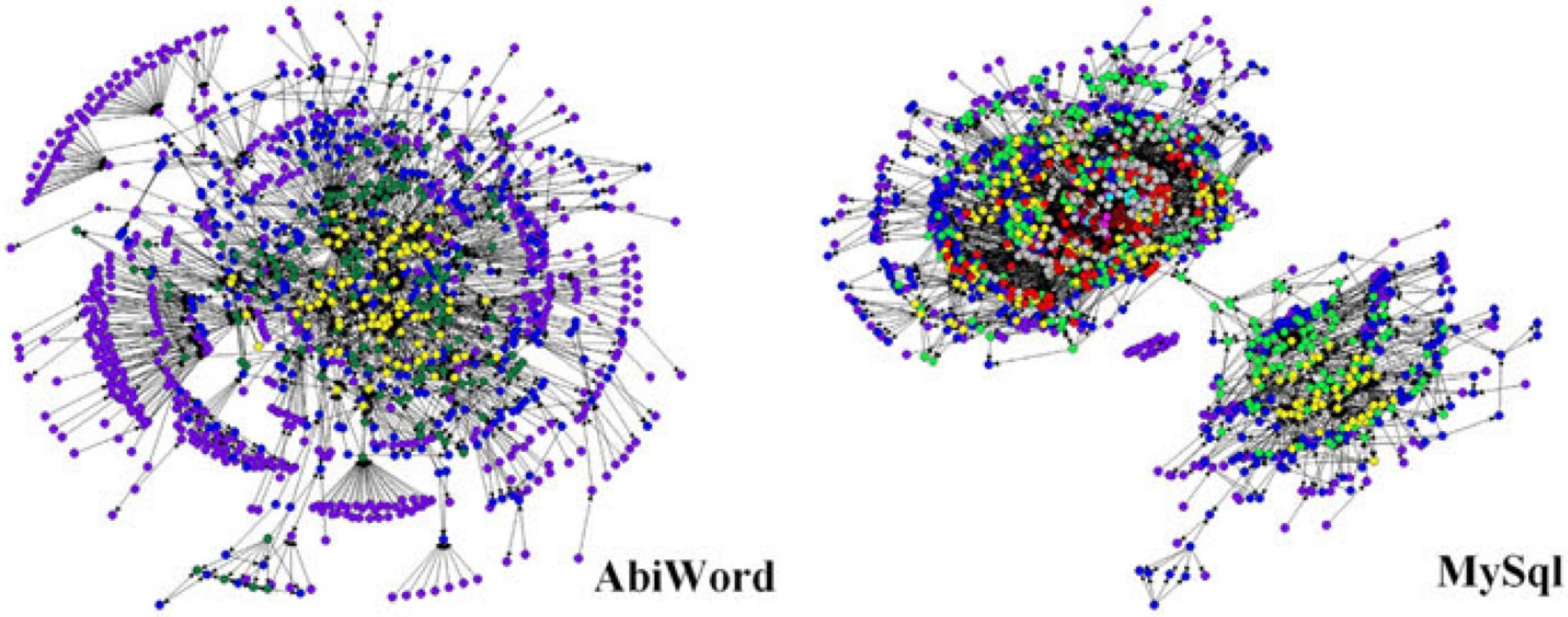
Brain connectivity networks: mapping the Structural Core of Human Cerebral Cortex [PLoS Bio 2008]



scan 1 scan 2
participant A participant B participant C participant D participant E

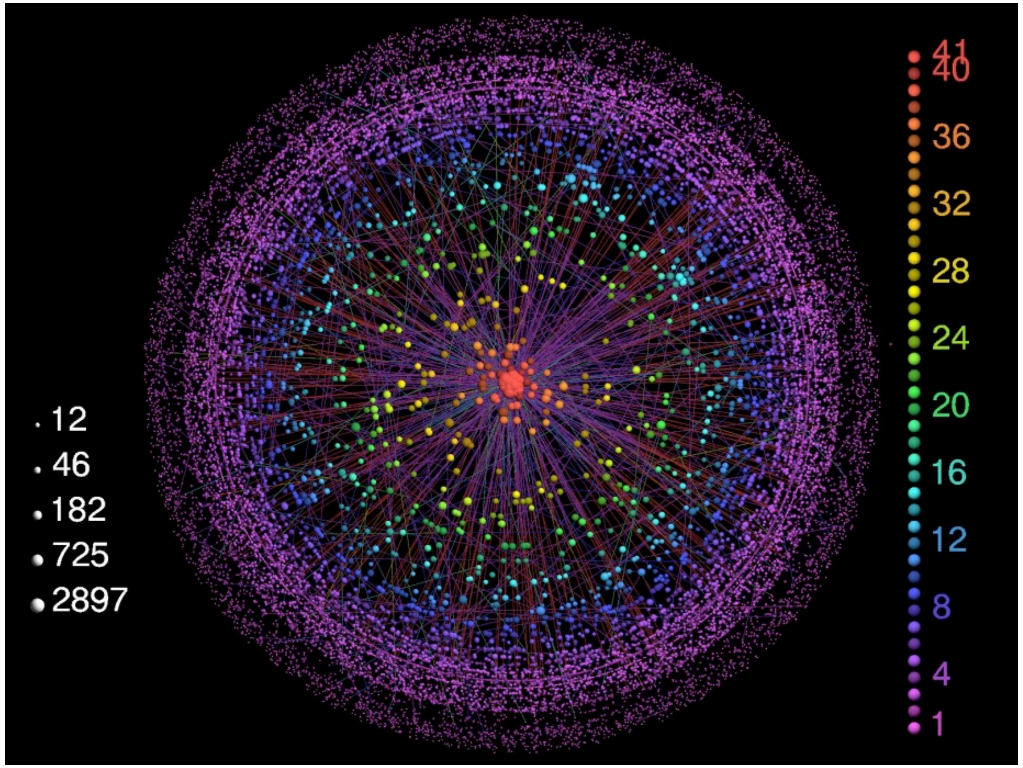
Application – Network Modeling and Analysis

Software systems: analyzing the static structure of large-scale software systems [J Supercomput 2010]



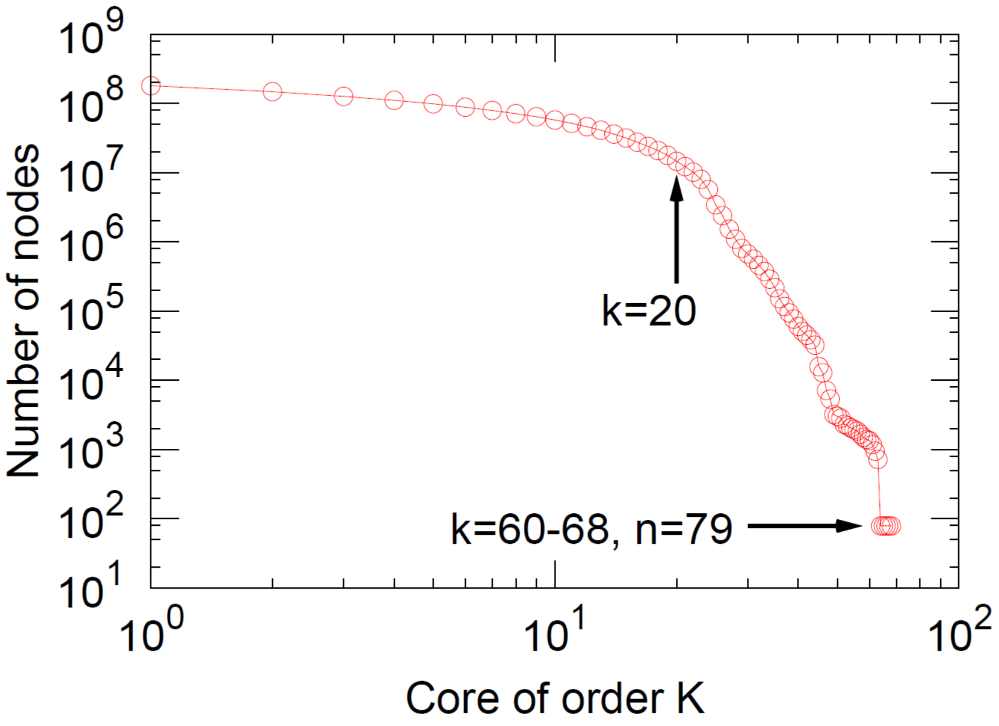
Application – Network Modeling and Analysis

Internet networks: modeling Internet topology using k-shell decomposition [PNAS 2007] [NHM 2008]



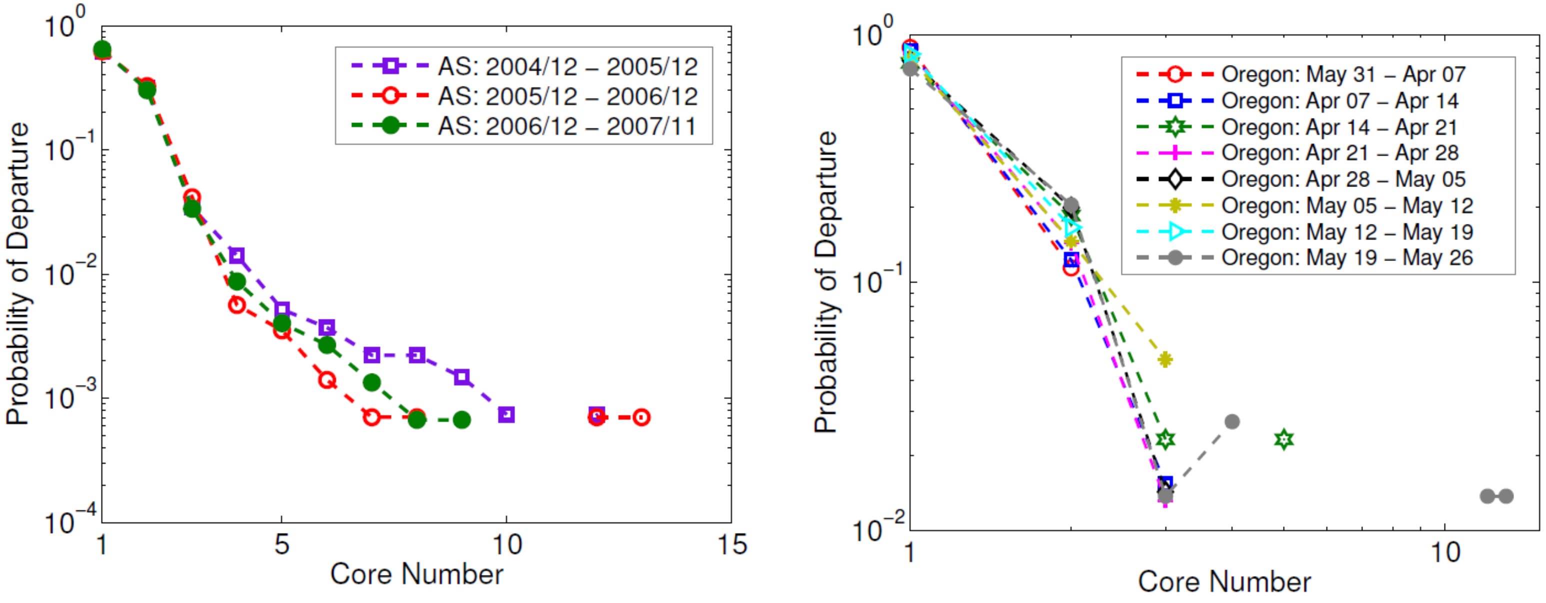
Application – Network Modeling and Analysis

Message networks: analysing the large instant-messaging network (Microsoft Messenger system) [WWW 2008]



Application – Network Modeling and Analysis

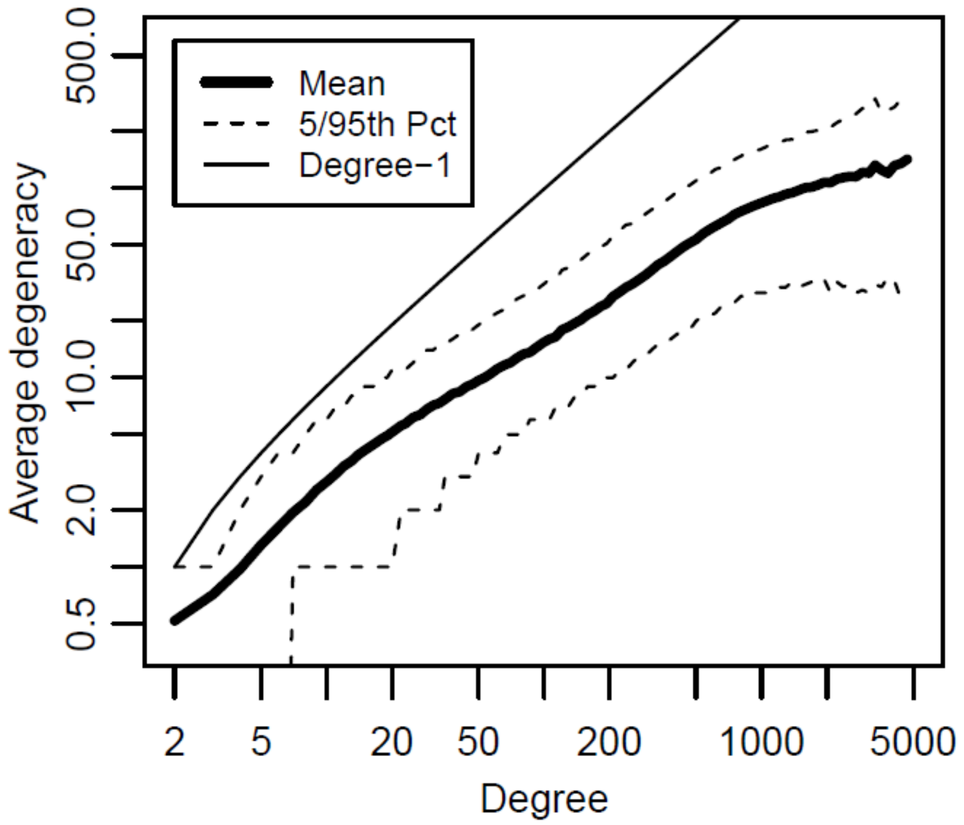
Social networks: modeling engagement dynamics in social graphs [CIKM 2013] [Social Networks 1983]



(a) CAIDA (b) OREGON

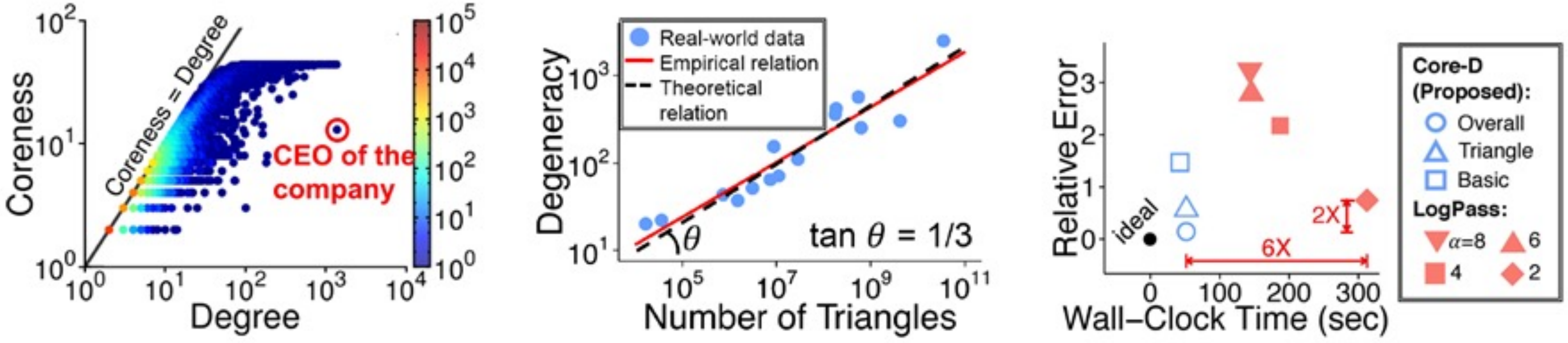
Application – Network Modeling and Analysis

Social networks: the anatomy of the Facebook social graph [Corr 2011]



Application – Network Modeling and Analysis

Complex networks: pattern and anomaly analysis using k-core analysis [ICDM 2016] [KAIS 2018]



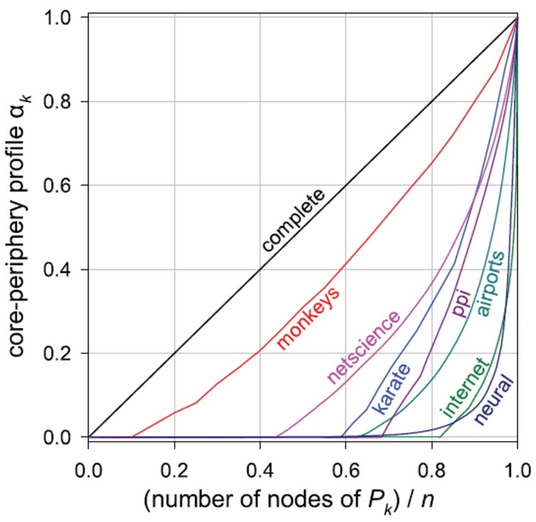
(a) P1: MIRROR PATTERN
A1: Anomaly Detection

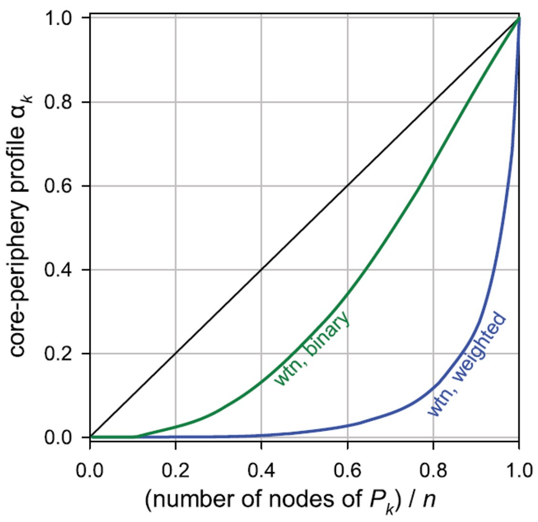
(b) P2: CORE-TRIANGLE PATTERN

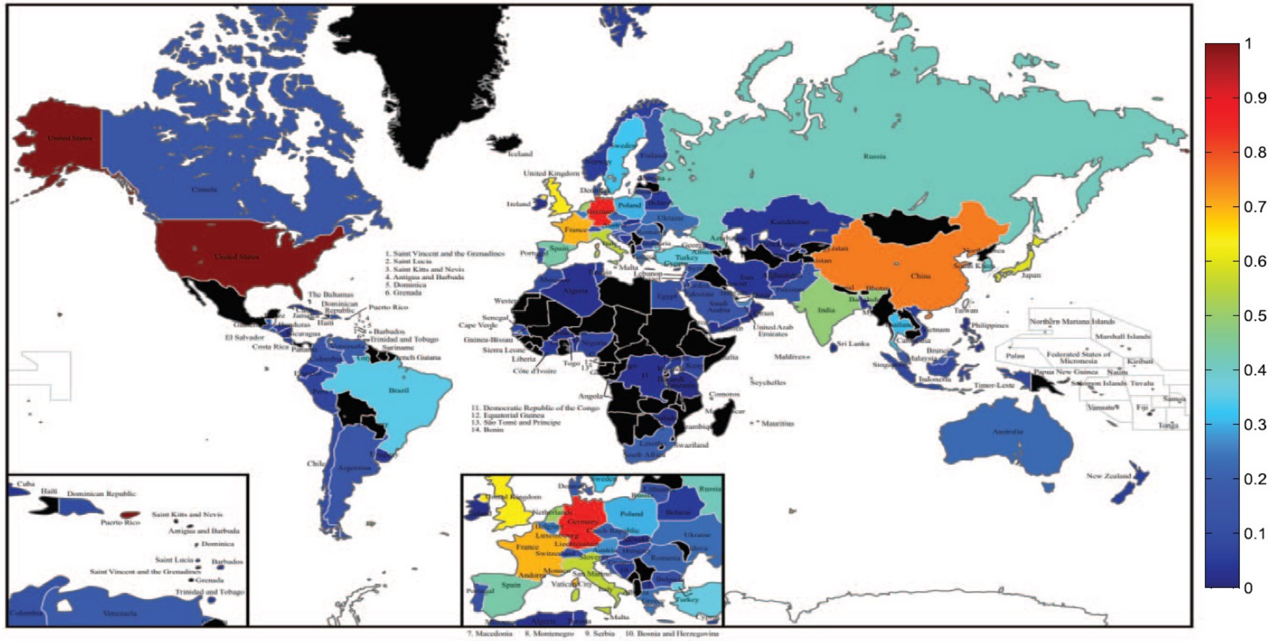
(c) A2: CORE-D Algorithm

Application – Network Modeling and Analysis

Complex networks: pcore-periphery network structure [Scientific reports 2013]



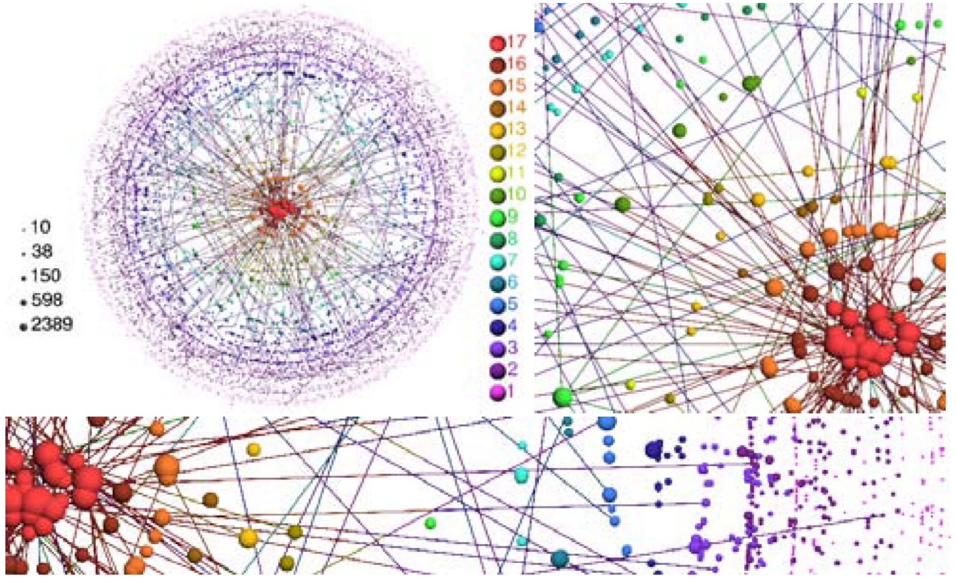


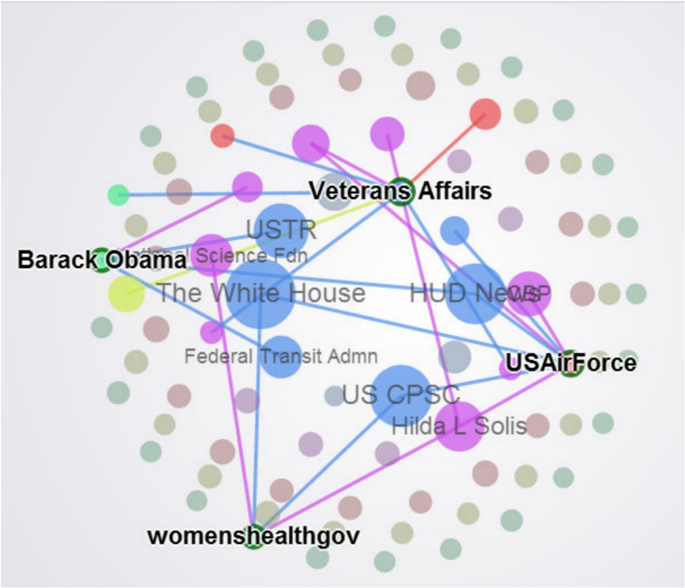


world trade network

Application – Network Visualization

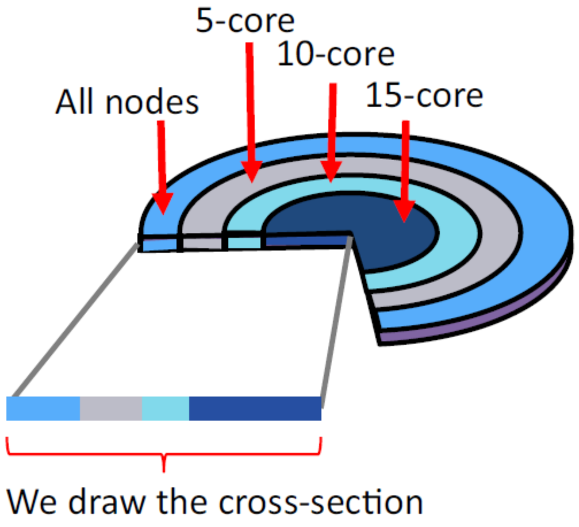
Large scale networks fingerprinting and visualization using hierarchical decomposition [NIPS 2006] [VLDB 2012] [ICDE 2012] [KDD 2012]

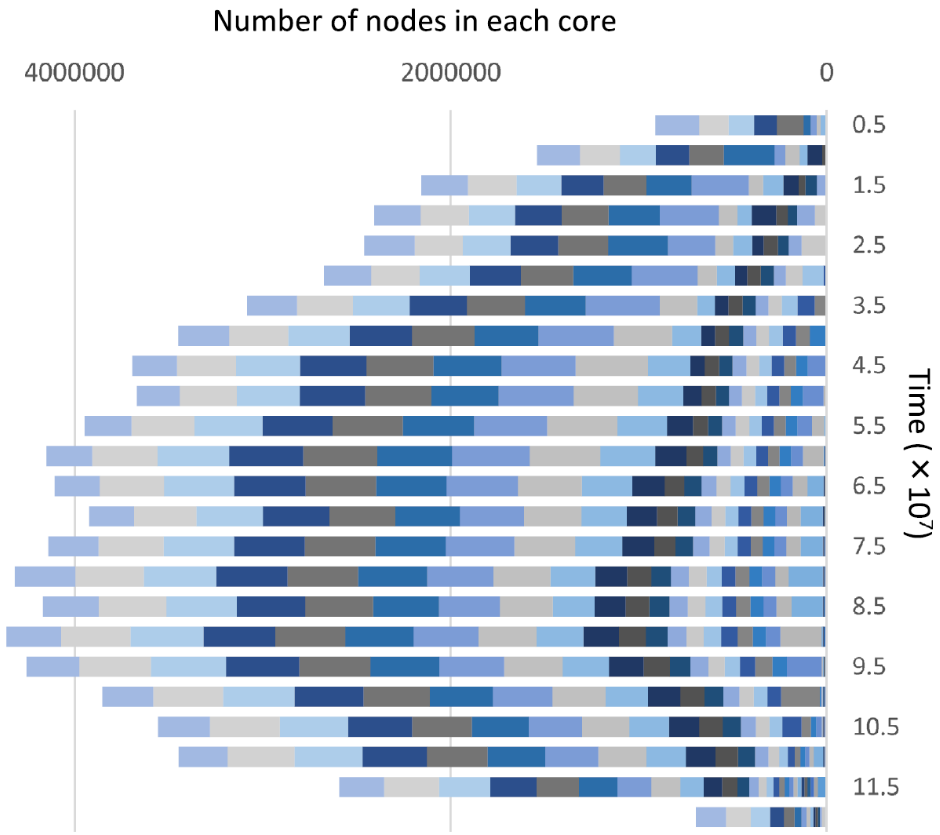




Application – Reasoning the collapse of a social network

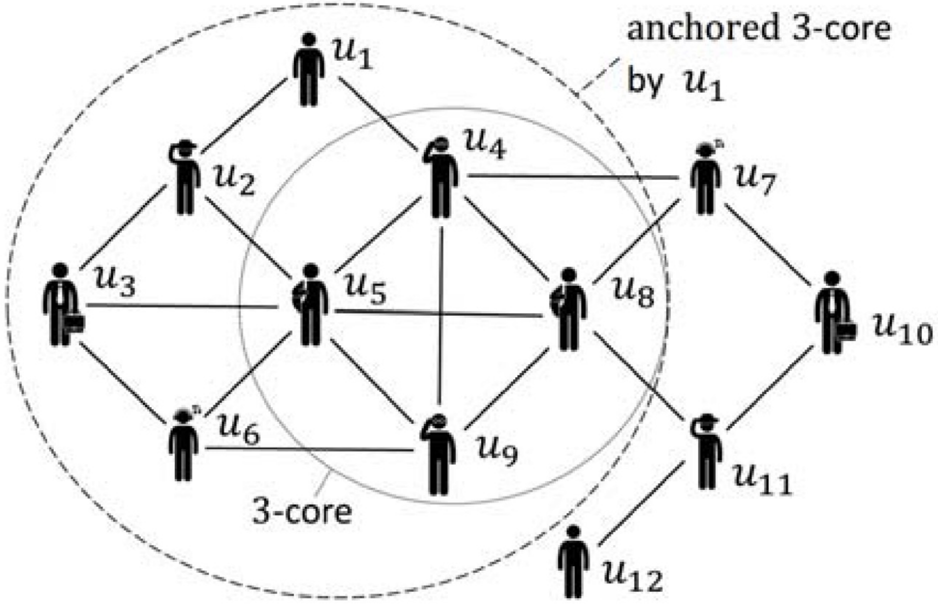
Friendster network: revealing the mechanism of collapse [SNAM 2017] [COSN 2013]

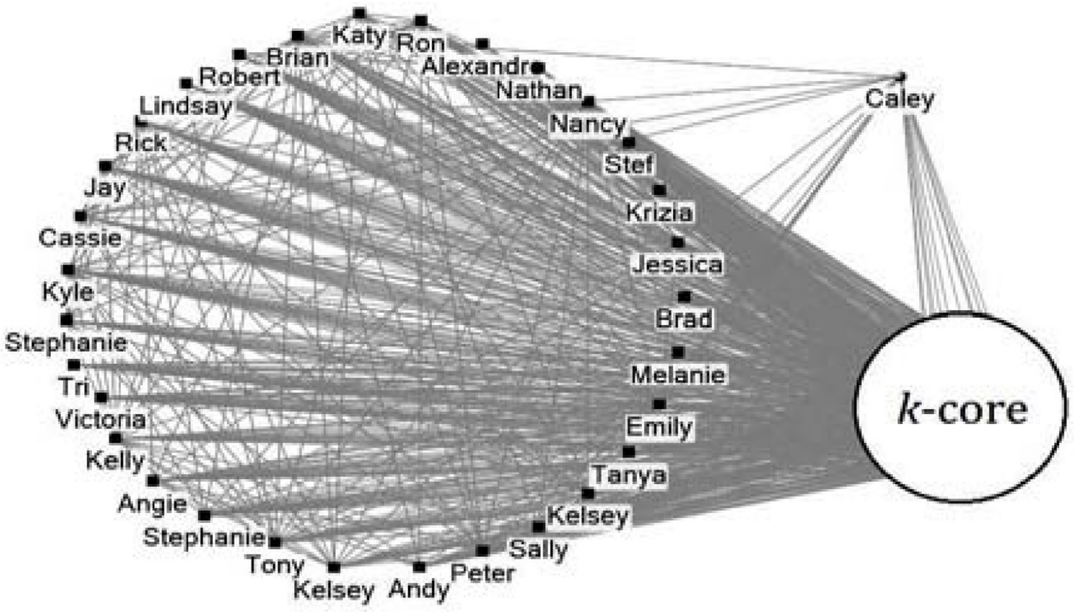




Application – Prevent Network Unraveling

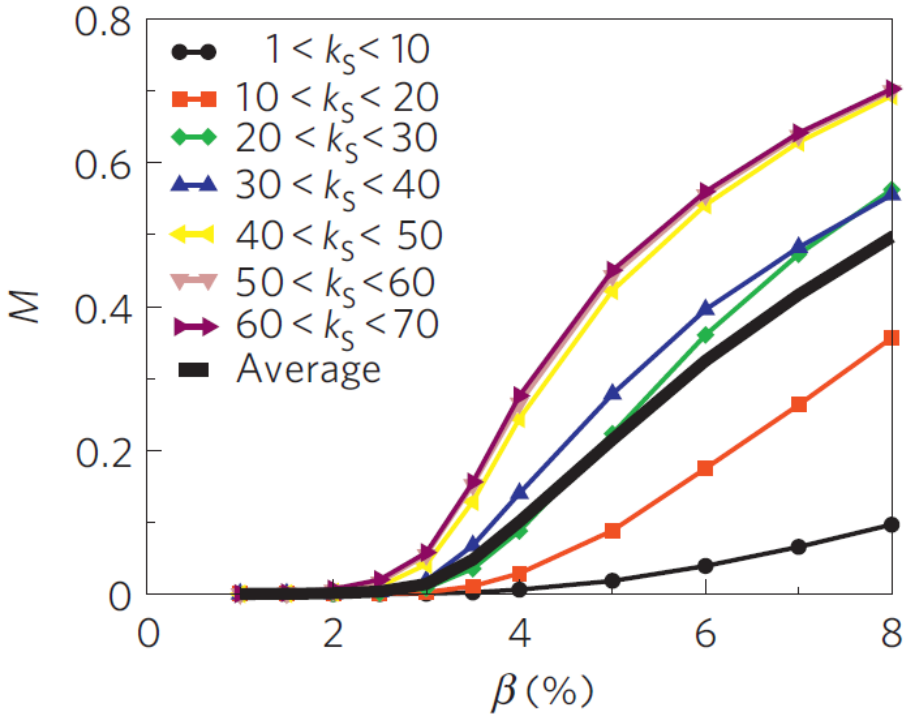
The Anchored Core and Truss Problems [SIAM J Discrete Math 2015]. The algorithms [VLDB 2017] [ICDE 2018].





Application – Discovering Influential Nodes

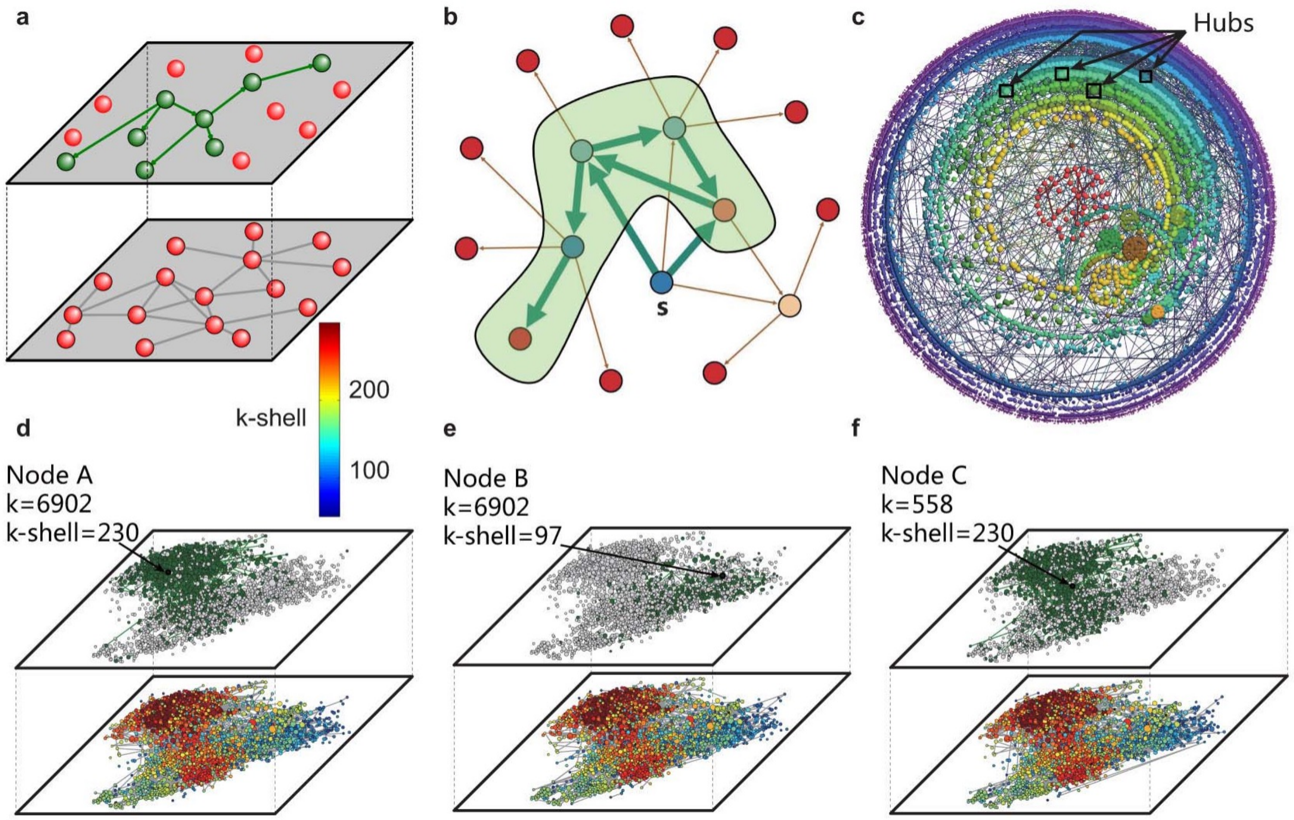
The most effective spreaders are located in the core of the network, fairly independent of their degree. [Nature Physics 2010]



Application – Discovering Influential Nodes

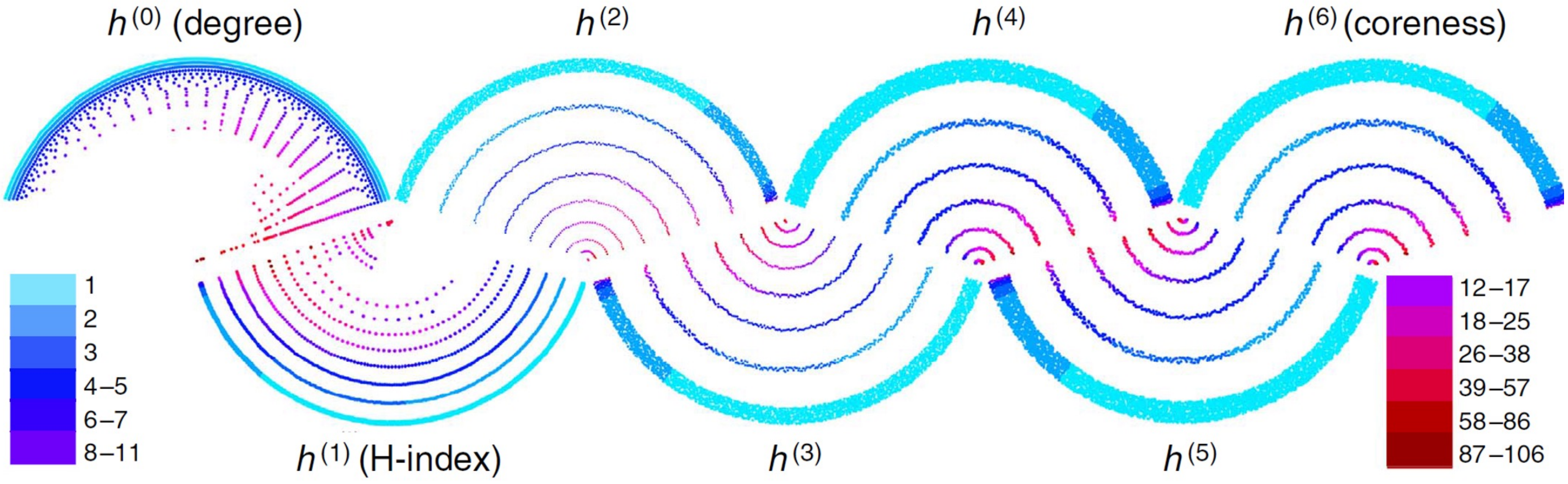
The widely-used degree and PageRank fail in ranking users' influence.

The best spreaders are consistently located in the k-core across dissimilar social platforms. [Scientific Reports 2014]



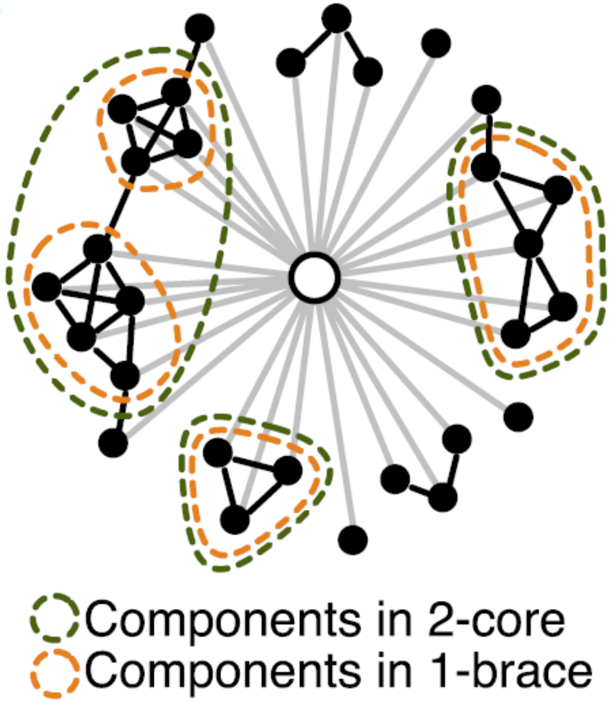
Application – Evaluating Node Influence

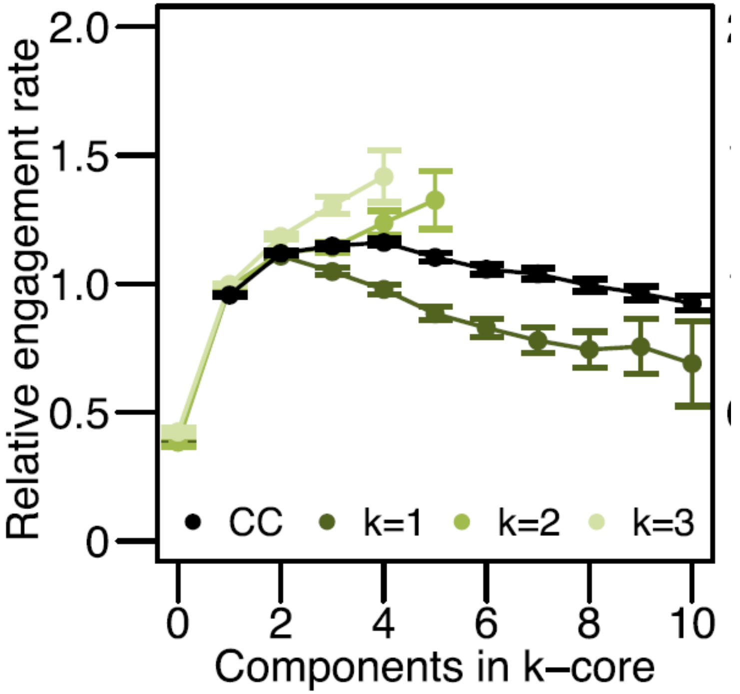
The H-index of a network node and its relation to degree and coreness [Nature Com 2016]



Application – Social Contagion

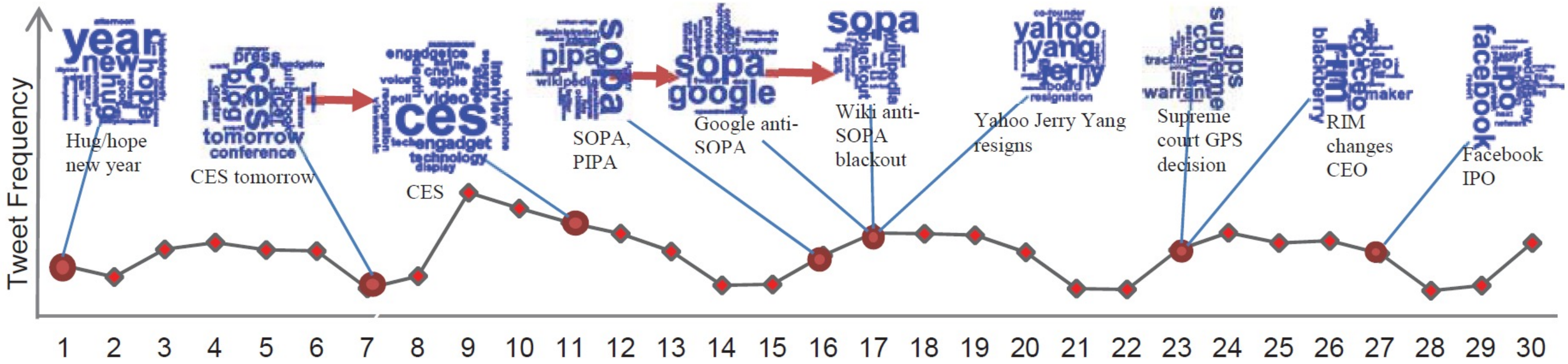
Evaluating social contagion based structural diversity [PNAS 2012]





Application – Community Discovery

<u>Online social streams</u>: a context-aware story-teller [CIKM 2014]



Application – Community Discovery

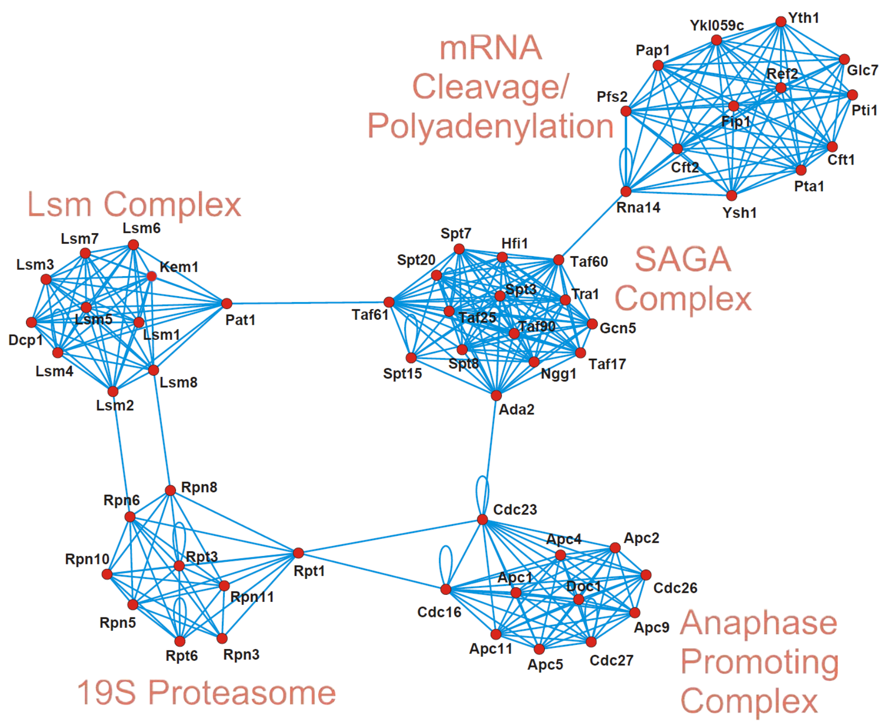
Social networks: persistent community search [ICDE 2018], spatial community search [ICDE 2018], attributed community detection [VLDB 2017] [VLDB 2017] [VLDB 2018] , influential community search [VLDB 2015]

Blogosphere in US

Communities in Gowalla

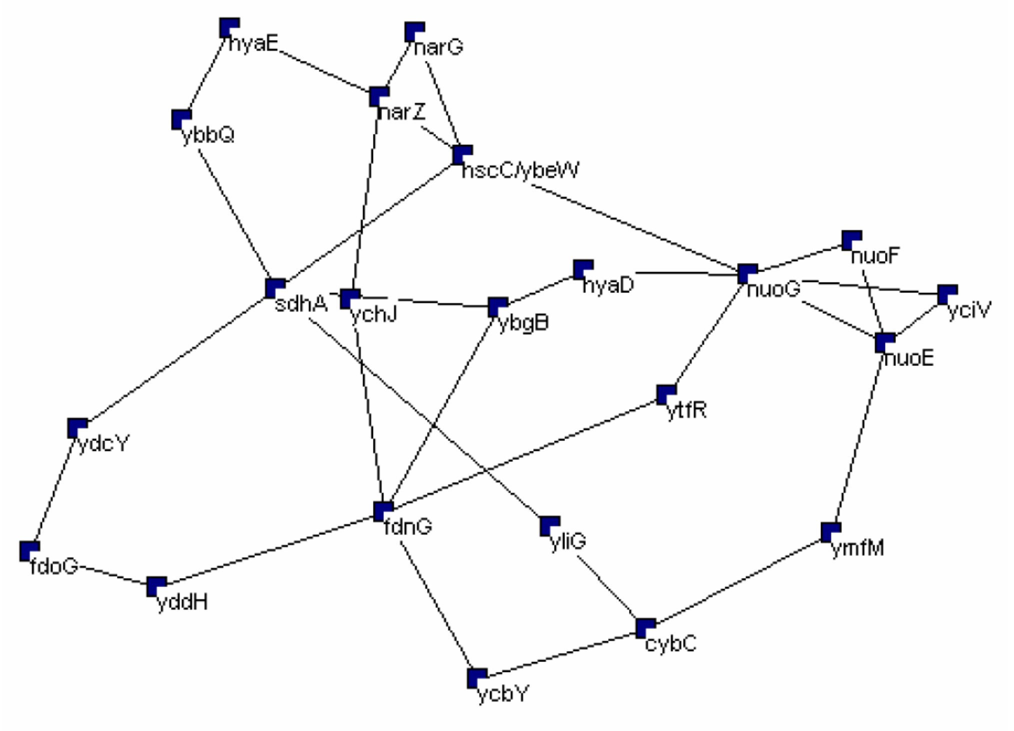
Application – Community Discovery

Protein interaction networks: finding molecular complexes [BMC Bio 2003]



Application – Protein Function Prediction

Protein interaction networks: prediction of protein functions based on k-cores of the networks and amino ccid sequences [Genome Info 2003]



More Applications

- **Graph clustering**

  Giatsidis, Christos, et al. "Corecluster: A degeneracy based graph clustering framework." AAAI. 2014.

- **Graph similarity**

  Nikolentzos, Giannis, et al. "A Degeneracy Framework for Graph Similarity." *IJCAI*. 2018.

- **Community evaluation**

  Giatsidis, Christos, Dimitrios M. Thilikos, and Michalis Vazirgiannis. "Evaluating cooperation in communities with the k-core structure." ASONAM, 2011.

- **Influence maximization**

  Elsharkawy, Sarah, et al. "Effectiveness of the k-core nodes as seeds for influence maximisation in dynamic cascades." *International Journal of Computers* 2 (2017).

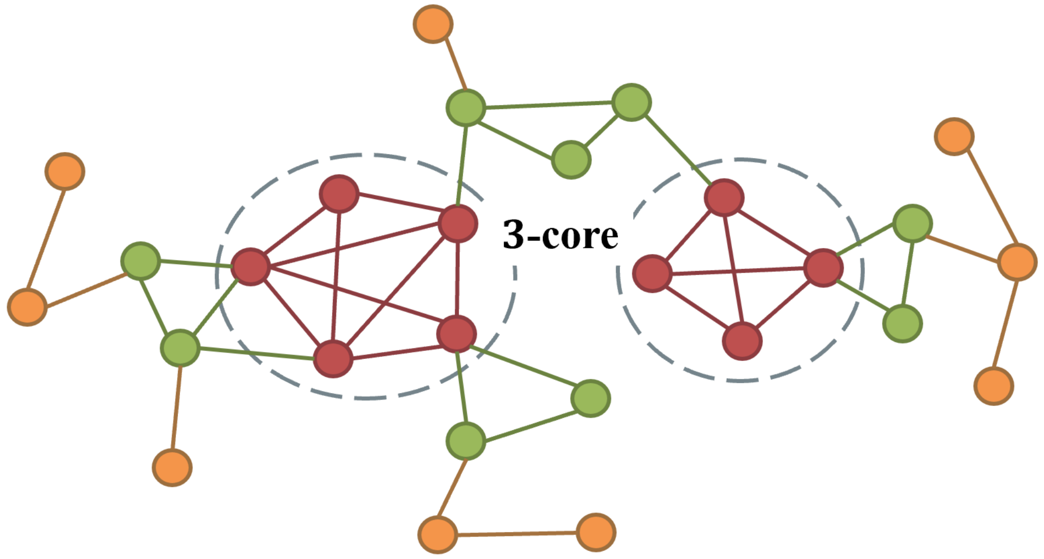
- **Graph generating**

  Baur, Michael, et al. "Generating graphs with predefined k-core structure." *Proceedings of the European Conference of Complex Systems*. 2007.

K-CORE

K-Core Model

K-core is a maximal subgraph in which each vertex has at least k neighbors in the subgraph.



*Erdős, Paul, and András Hajnal. "**On chromatic number of graphs and set-systems.**" Acta Mathematica Hungarica 17.1-2 (1966): 61-99.*

Computing the k-Core

Iteratively remove every vertex whose degree is less than k.

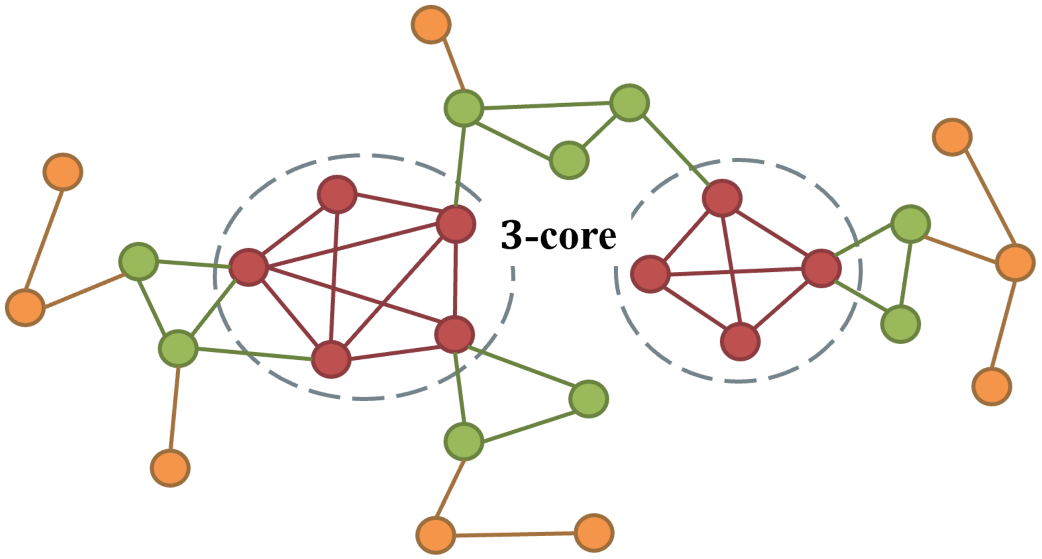
$O(m+n)$

Algorithm : ComputeCore(G, k)

Input : G : a graph, k : degree constraint
Output : $C_k(G)$

while exists $v \in G$ with $deg(v, G) < k$ **do**
  $G \leftarrow G \setminus \{v \cup E(v, G)\}$;

return G



Computing the k-Core

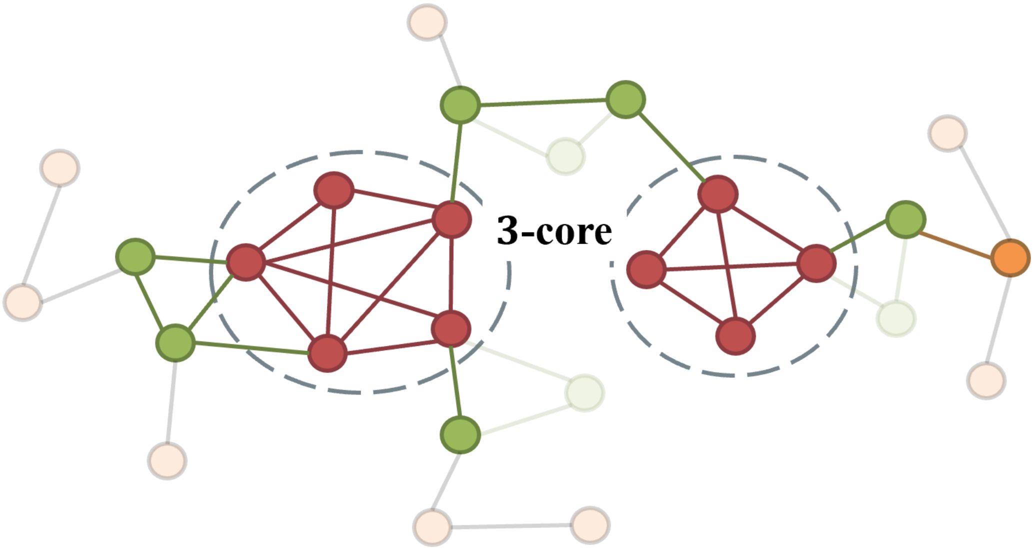
Iteratively remove every vertex whose degree is less than k.

$O(m+n)$

Algorithm : ComputeCore(G, k)

Input : G : a graph, k : degree constraint
Output : $C_k(G)$

while exists $v \in G$ with $deg(v, G) < k$ **do**
  $G \leftarrow G \setminus \{v \cup E(v, G)\}$;
return G



Computing the k-Core

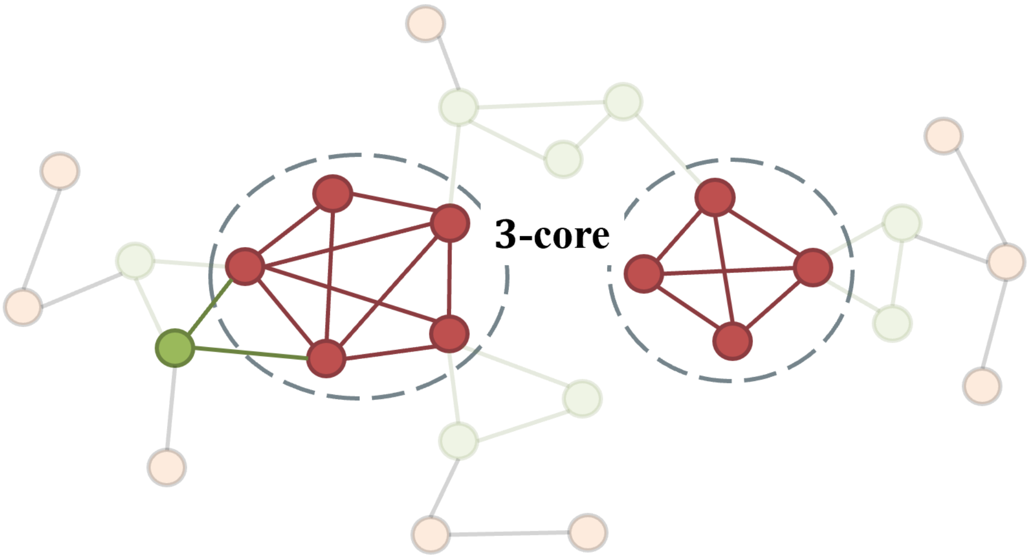
Iteratively remove every vertex whose degree is less than k.

$O(m+n)$

Algorithm : ComputeCore(G, k)

Input : G : a graph, k : degree constraint
Output : $C_k(G)$

```
while exists v ∈ G with deg(v, G) < k do
    G ← G \ {v ∪ E(v, G)};
return G
```



Computing the k-Core

Iteratively remove every vertex whose degree is less than k.

$O(m+n)$

Algorithm : ComputeCore(G, k)

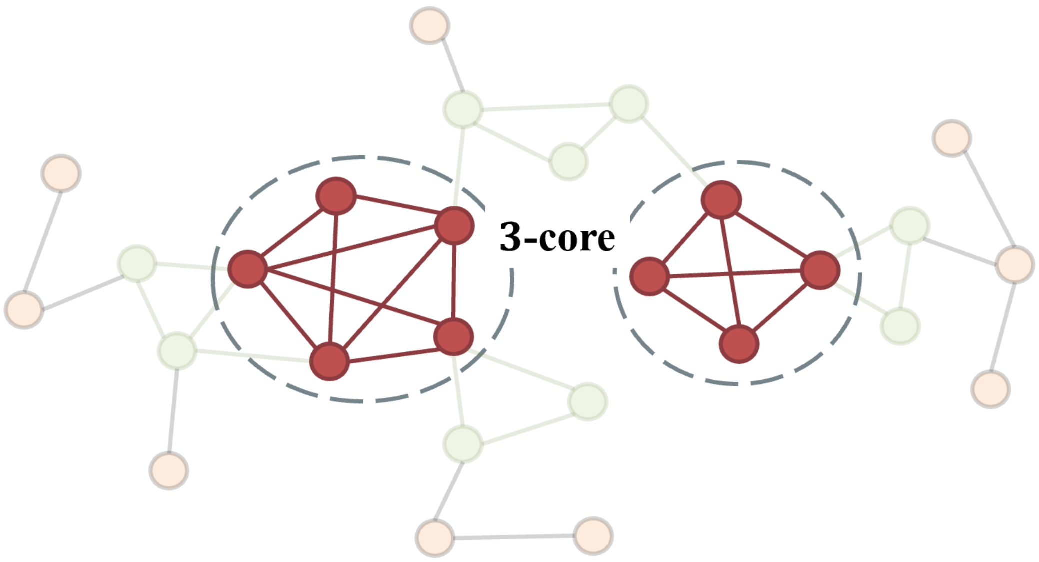
Input : G : a graph, k : degree constraint

Output : $C_k(G)$

while exists $v \in G$ **with** $deg(v, G) < k$ **do**

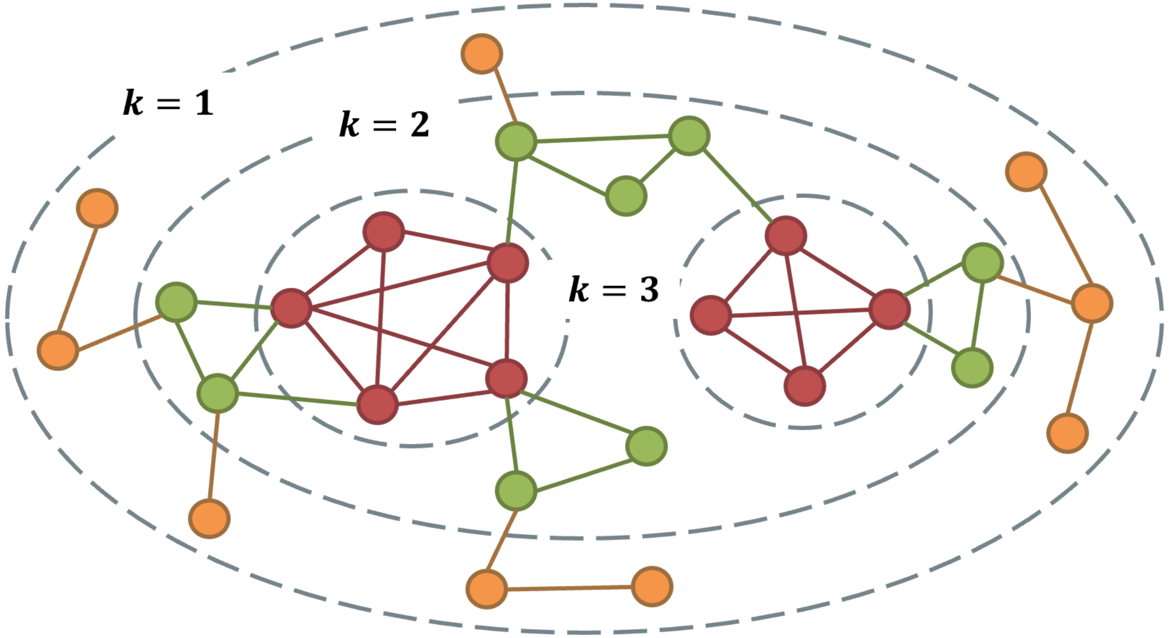
  $G \leftarrow G \setminus \{v \cup E(v, G)\}$;

return G



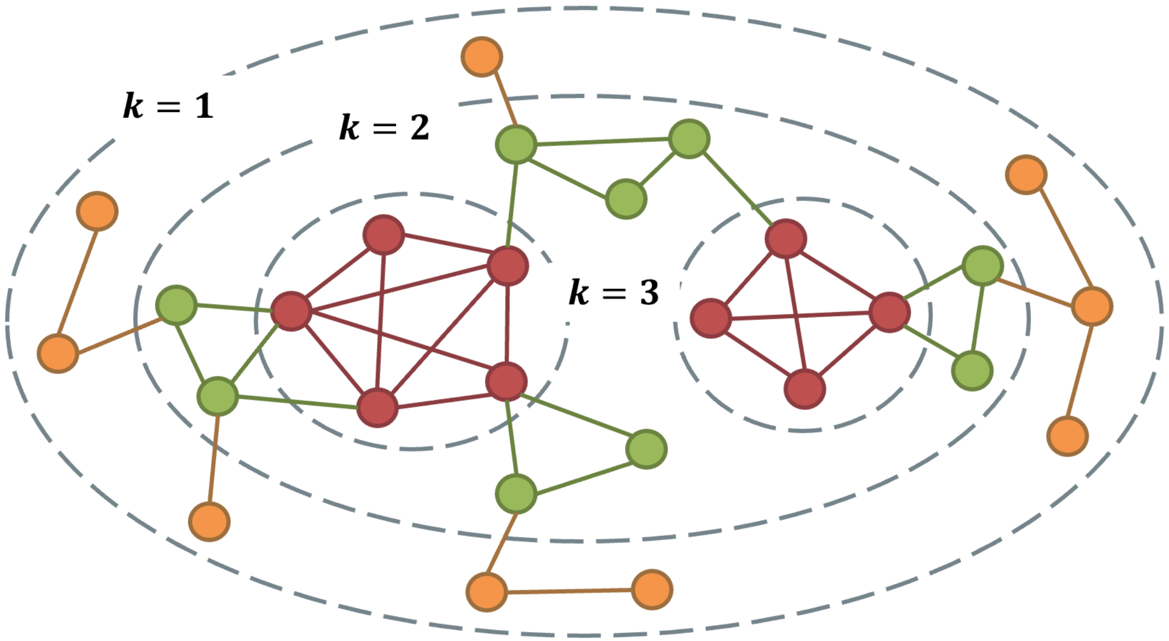
Core Number of a Vertex v

$k(v)$ = the largest k such that the k-core contains v



Core Decomposition

Compute the core number of every vertex.



Core Decomposition

The following algorithms are optional, as long as you know how to compute the core number of a given graph.

Global-view: peel low-degree vertices iteratively from the whole graph.

Local-view 1: update the upper bound of core number for each vertex until converge

Core Decomposition: Global-view (Peeling)

Iteratively delete the vertex with the lowest degree.   O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph

Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

$d(u) \leftarrow deg(u, G)$ for every $u \in V$;

order the vertices in V in increasing order of their degrees;

for each $u \in V$ in the order **do**

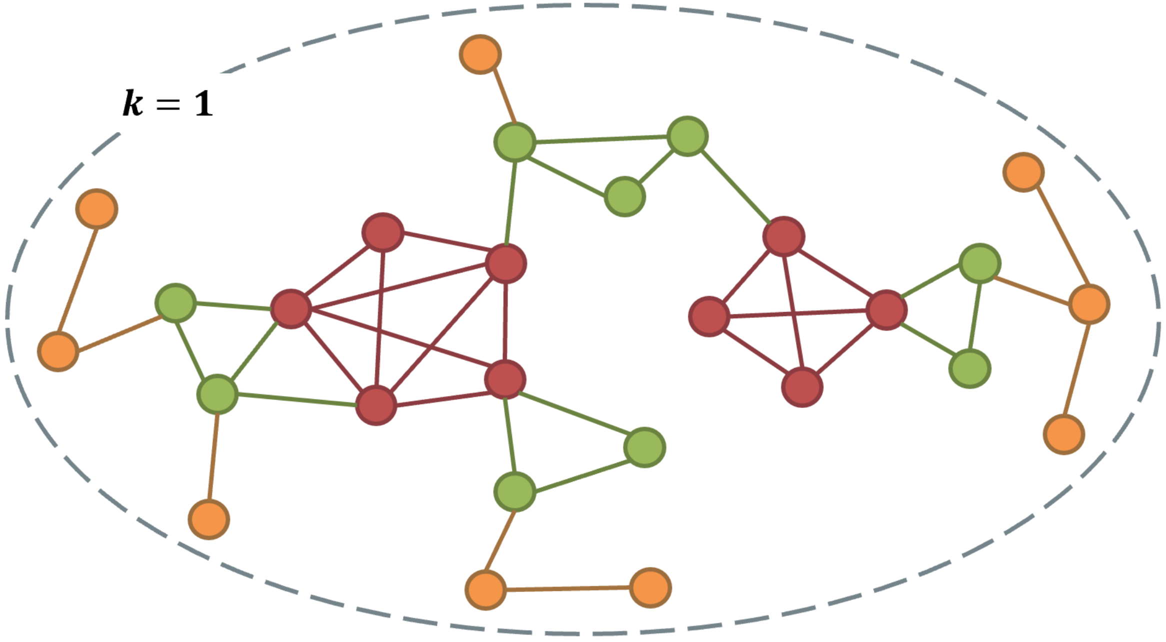
  $cn(u) \leftarrow d(u)$;

  for each $v \in N(u)$ with $d(v) > d(u)$ **do**

    $d(v) \leftarrow d(v) - 1$;

    reorder V accordingly;

return $cn(u)$ of every $u \in V$



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

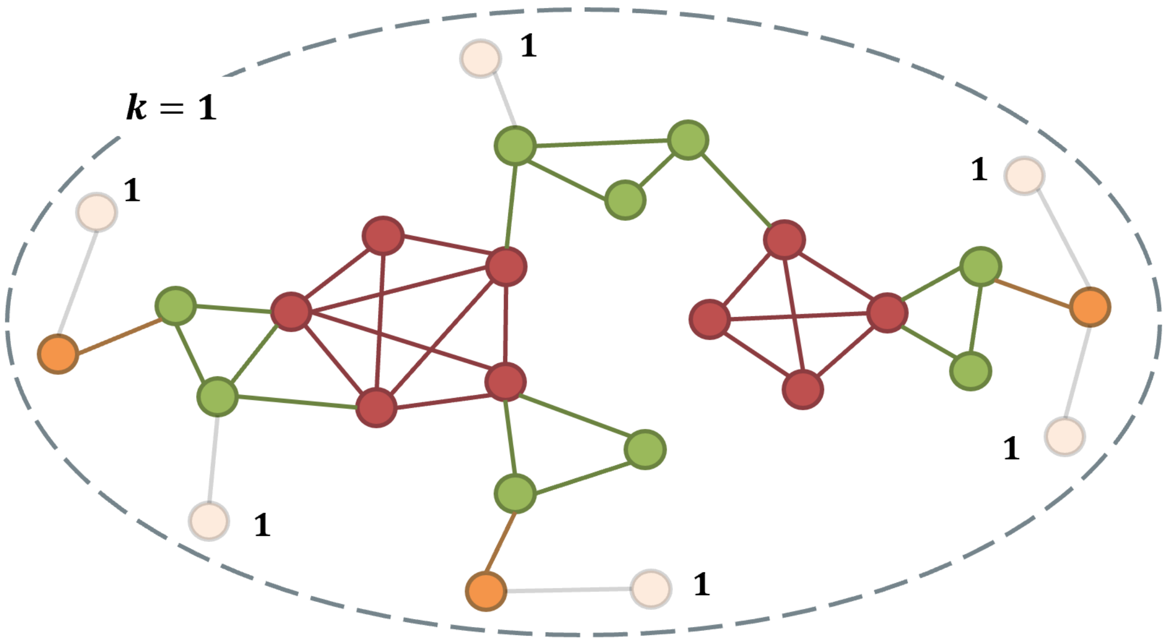
Core Decomposition: Global-view (Peeling)

Iteratively delete the vertex with the lowest degree.   O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph
Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

```
d(u) ← deg(u, G) for every u ∈ V;
order the vertices in V in increasing order of their degrees;
for each u ∈ V in the order do
    cn(u) ← d(u);
    for each v ∈ N(u) with d(v) > d(u) do
        d(v) ← d(v) − 1;
        reorder V accordingly;
return cn(u) of every u ∈ V
```



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

Core Decomposition: Global-view (Peeling)

Iteratively delete the vertex with the lowest degree. O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph

Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

$d(u) \leftarrow deg(u, G)$ for every $u \in V$;

order the vertices in V in increasing order of their degrees;

for each $u \in V$ in the order **do**

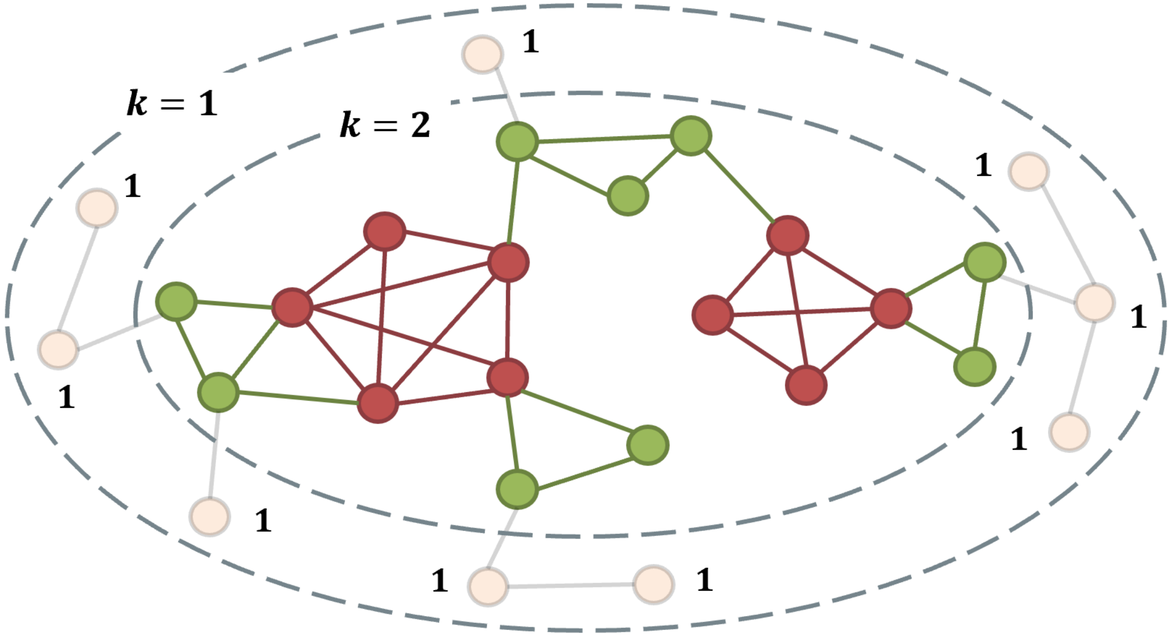
    $cn(u) \leftarrow d(u)$;

    for each $v \in N(u)$ with $d(v) > d(u)$ **do**

        $d(v) \leftarrow d(v) - 1$;

        reorder V accordingly;

return $cn(u)$ of every $u \in V$



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

Core Decomposition: Global-view (Peeling)

Iteratively delete the vertex with the lowest degree. O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph

Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

$d(u) \leftarrow deg(u, G)$ for every $u \in V$;

order the vertices in V in increasing order of their degrees;

for each $u \in V$ in the order **do**

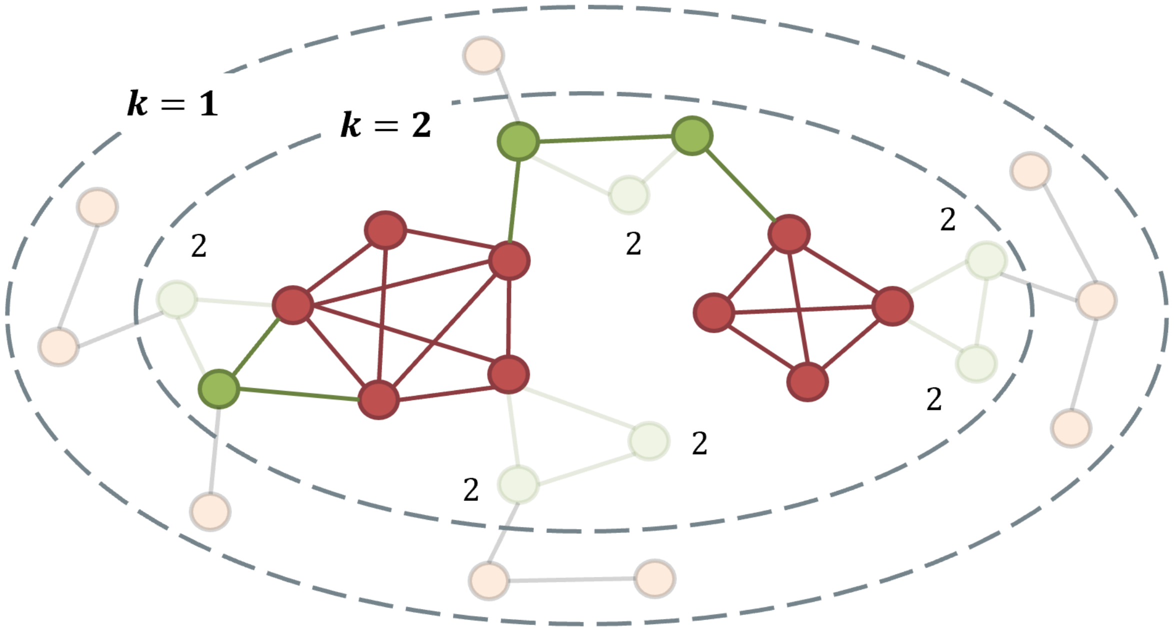
  $cn(u) \leftarrow d(u)$;

  for each $v \in N(u)$ with $d(v) > d(u)$ **do**

    $d(v) \leftarrow d(v) - 1$;

    reorder V accordingly;

return $cn(u)$ of every $u \in V$



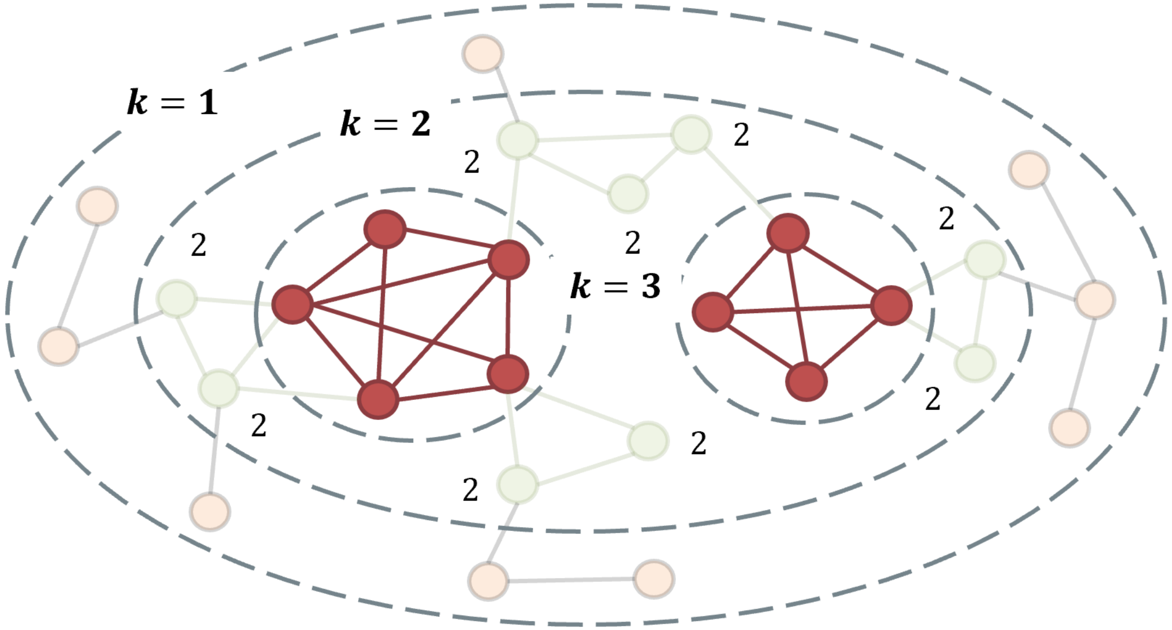
Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

Core Decomposition: Global-view (Peeling)

Iteratively delete the vertex with the lowest degree. O(m+n)

Algorithm : CoreDecomposition

```
Input  : G = (V, E) : a graph
Output : {cn(u) | u ∈ V}: core number of every vertex in G
d(u) ← deg(u, G) for every u ∈ V;
order the vertices in V in increasing order of their degrees;
for each u ∈ V in the order do
    cn(u) ← d(u);
    for each v ∈ N(u) with d(v) > d(u) do
        d(v) ← d(v) − 1;
        reorder V accordingly;
return cn(u) of every u ∈ V
```



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

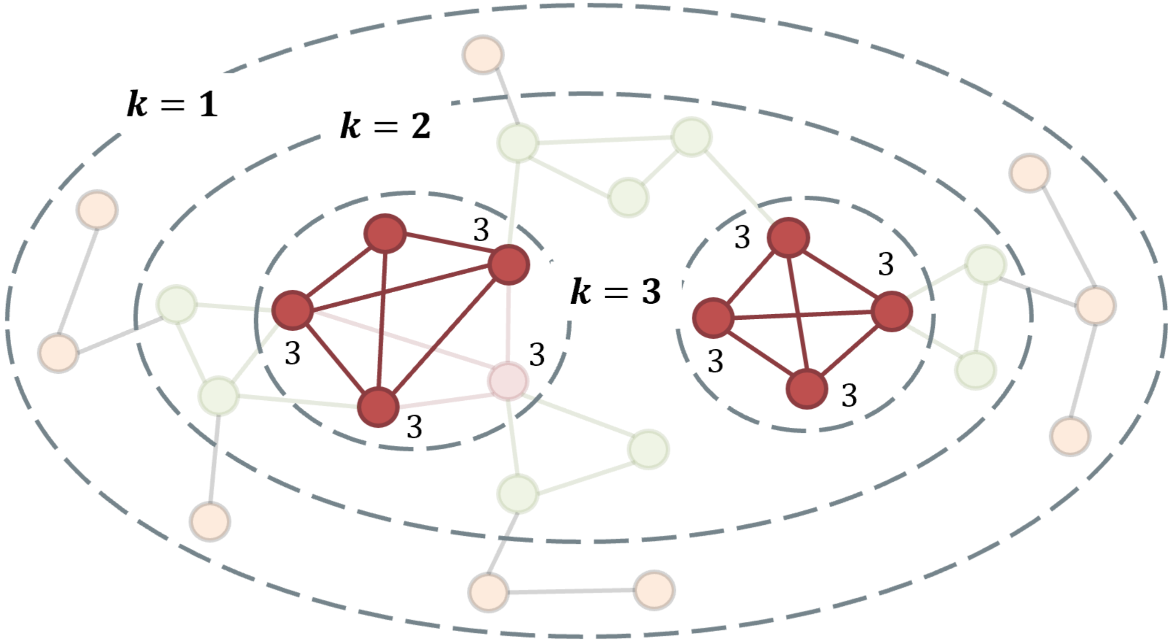
Core Decomposition: Global-view (Peeling)

Iteratively delete the vertex with the lowest degree. O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph
Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

$d(u) \leftarrow deg(u, G)$ for every $u \in V$;
order the vertices in V in increasing order of their degrees;
for each $u \in V$ in the order **do**
    $cn(u) \leftarrow d(u)$;
    for each $v \in N(u)$ with $d(v) > d(u)$ **do**
        $d(v) \leftarrow d(v) - 1$;
        reorder V accordingly;
return $cn(u)$ of every $u \in V$



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

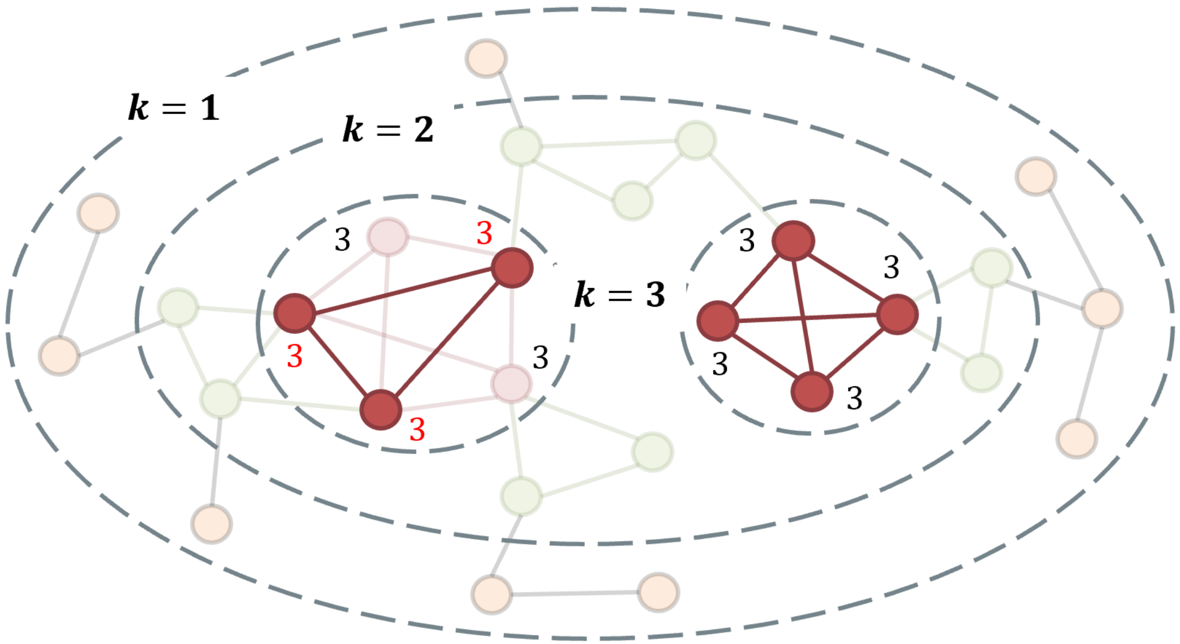
Core Decomposition: Global-view (Peeling)

Iteratively delete the vertex with the lowest degree. O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph
Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

$d(u) \leftarrow deg(u, G)$ for every $u \in V$;
order the vertices in V in increasing order of their degrees;
for each $u \in V$ in the order **do**
  $cn(u) \leftarrow d(u)$;
  for each $v \in N(u)$ with $d(v) > d(u)$ **do**
    $d(v) \leftarrow d(v) - 1$;
    reorder V accordingly;
return $cn(u)$ of every $u \in V$



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

Core Decomposition: Global-view (Peeling)

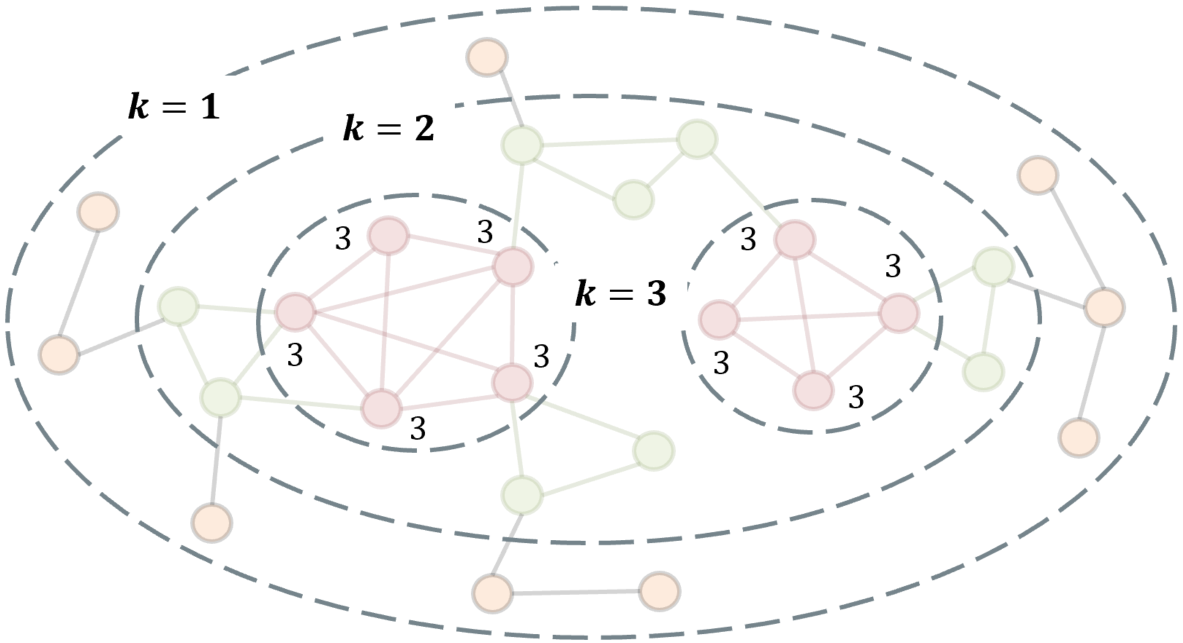
Iteratively delete the vertex with the lowest degree. O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph

Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

```
d(u) ← deg(u, G) for every u ∈ V;
order the vertices in V in increasing order of their degrees;
for each u ∈ V in the order do
    cn(u) ← d(u);
    for each v ∈ N(u) with d(v) > d(u) do
        d(v) ← d(v) − 1;
        reorder V accordingly;
return cn(u) of every u ∈ V
```



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

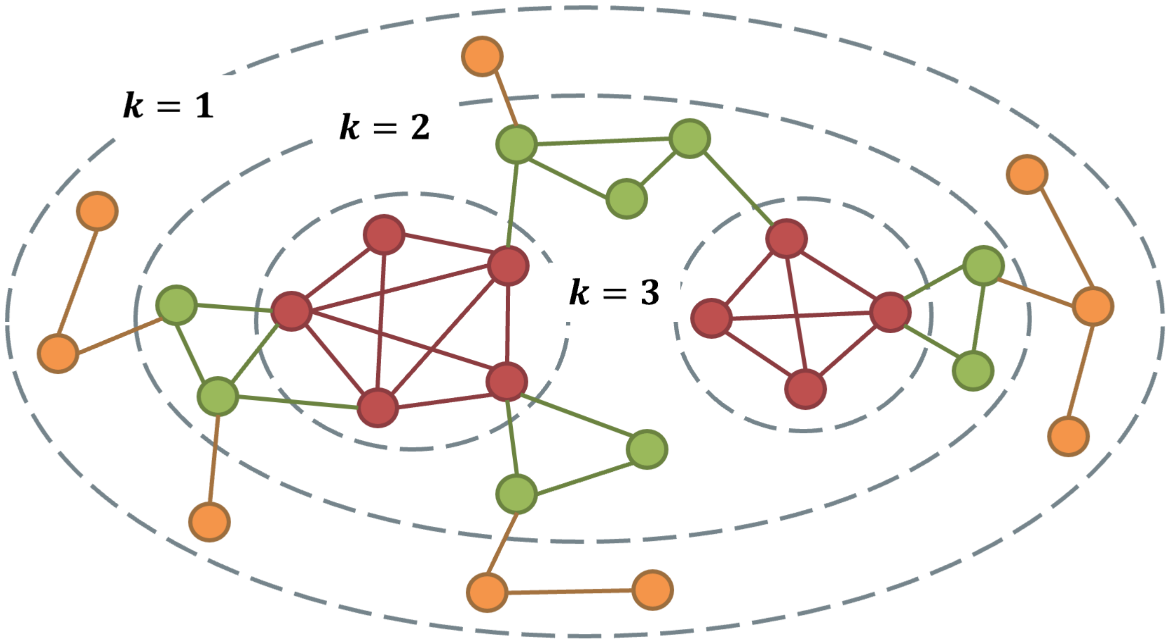
Core Decomposition: Global-view (Peeling)

Using bin-sort -> O(m+n)

Algorithm : CoreDecomposition

Input : $G = (V, E)$: a graph
Output : $\{cn(u) \mid u \in V\}$: core number of every vertex in G

$d(u) \leftarrow deg(u, G)$ for every $u \in V$;
order the vertices in V in increasing order of their degrees;
for each $u \in V$ in the order **do**
    $cn(u) \leftarrow d(u)$;
    for each $v \in N(u)$ with $d(v) > d(u)$ **do**
        $d(v) \leftarrow d(v) - 1$;
        reorder V accordingly;
return $cn(u)$ of every $u \in V$



Batagelj, Vladimir, and Matjaz Zaversnik. ***"An O (m) algorithm for cores decomposition of networks."*** *arXiv preprint cs/0310049 (2003).*

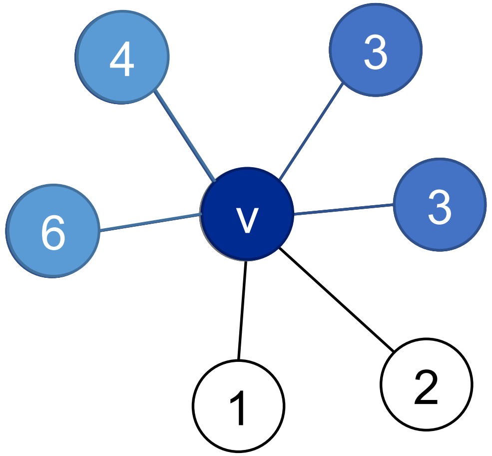
Core Decomposition: Local-view (Converging)

Locality Theorem:

Given a vertex and its core number k:

There exists at least k neighbors with core number k;

There does not exist k+1 neighbors with core number k+1.

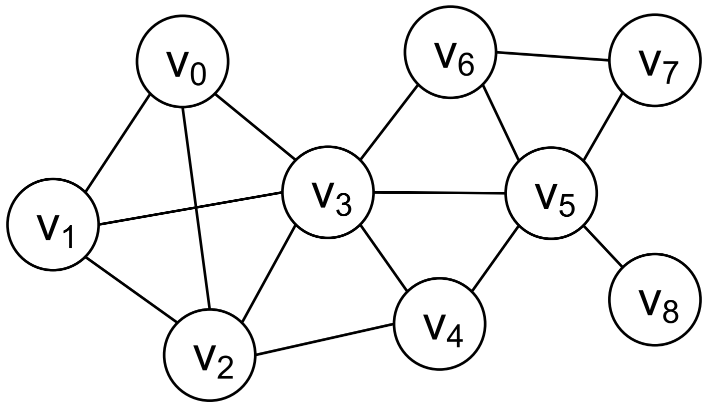


Core(V) = 3 | 4 neighbors with core number at least 3

Core(V) = 4 | Only 2 neighbors with core number at least 4

Montresor, Alberto, Francesco De Pellegrini, and Daniele Miorandi. ***"Distributed k-core decomposition."*** *IEEE Transactions on parallel and distributed systems 24.2 (2013): 288-300.*

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

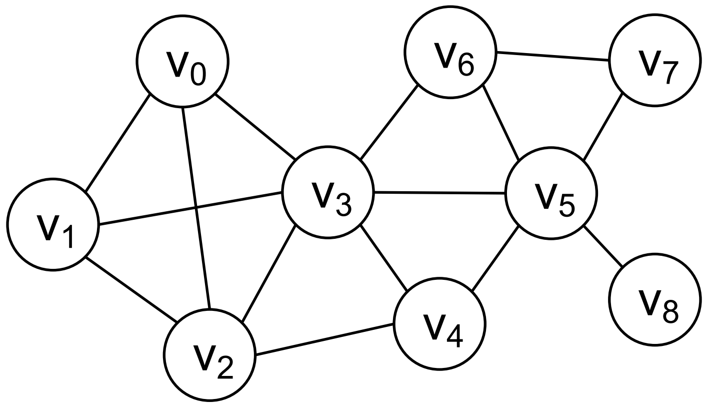
Initialize the core number by degree

Iteration *1*

isContinue ***False***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	4	6	3	5	3	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

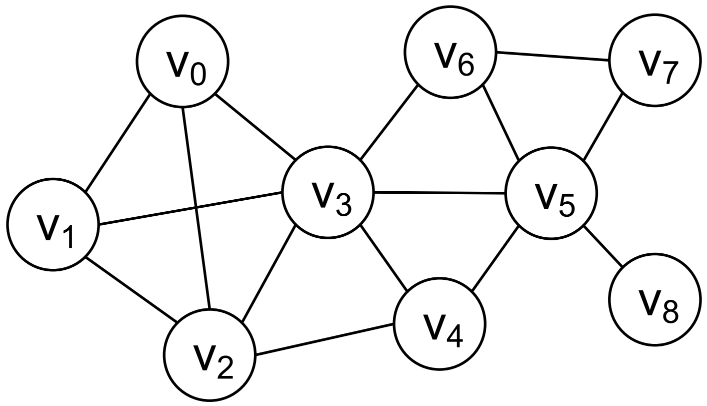
Initialize the core number by degree

Iteration *1*

isContinue ***False***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	4	6	3	5	3	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

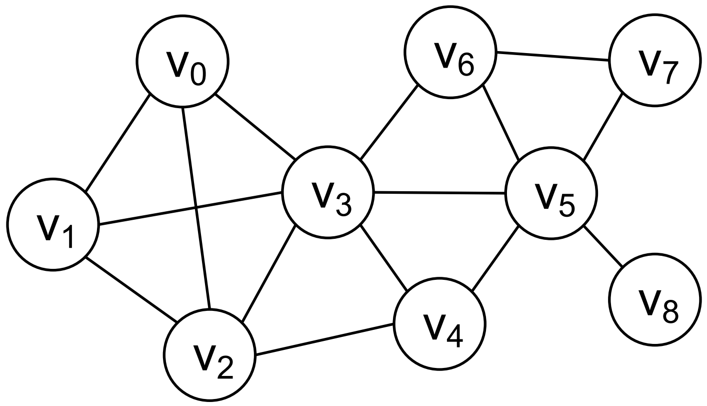
Initialize the core number by degree

Iteration *1*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	6	3	5	3	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

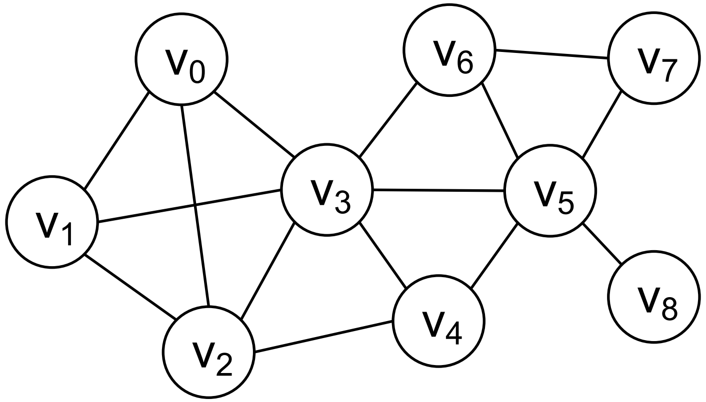
Initialize the core number by degree

Iteration *1*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	5	3	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

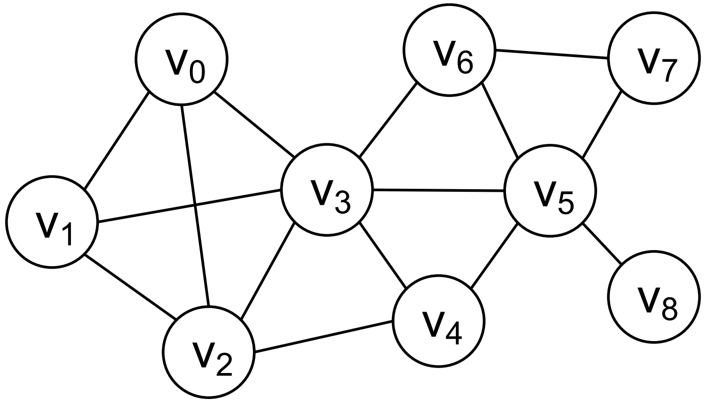
Initialize the core number by degree

Iteration *1*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	5	3	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

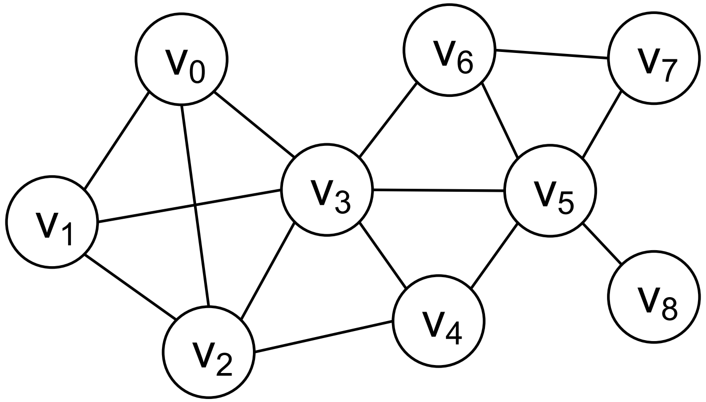
Initialize the core number by degree

Iteration *1*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	3	3	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

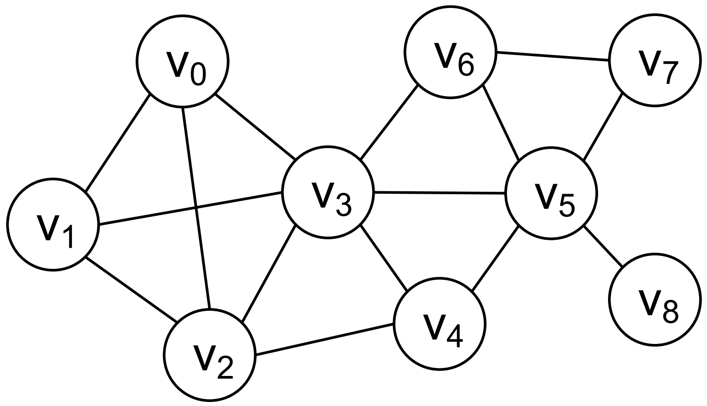
Initialize the core number by degree

Iteration ***1***

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	3	2	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

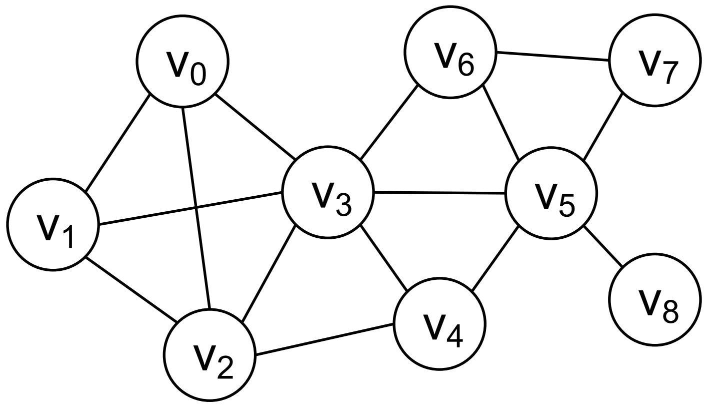
Initialize the core number by degree

Iteration *1*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	3	2	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

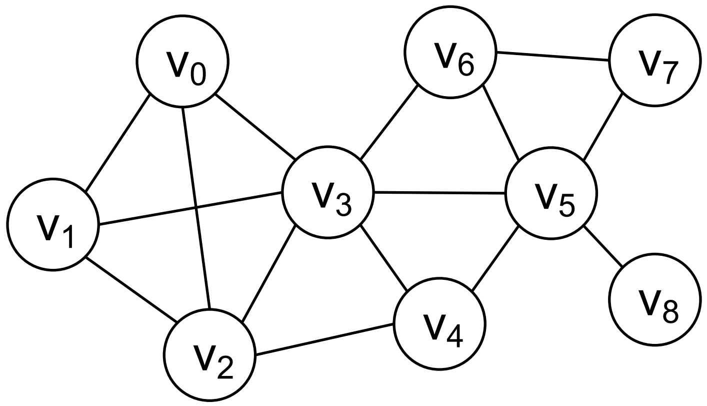
Initialize the core number by degree

Iteration *1*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	3	2	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

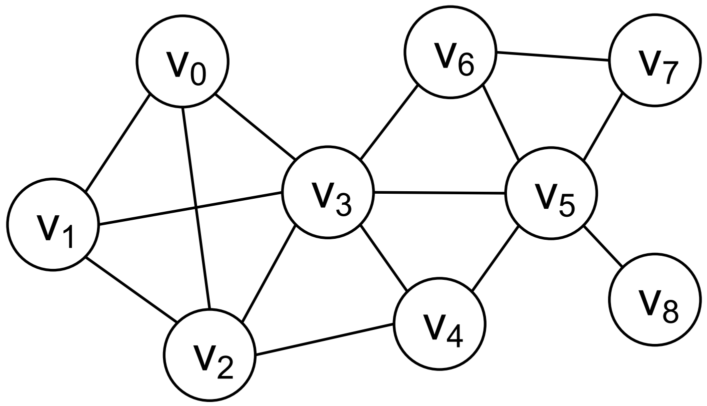
Initialize the core number by degree

Iteration *1*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	3	2	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

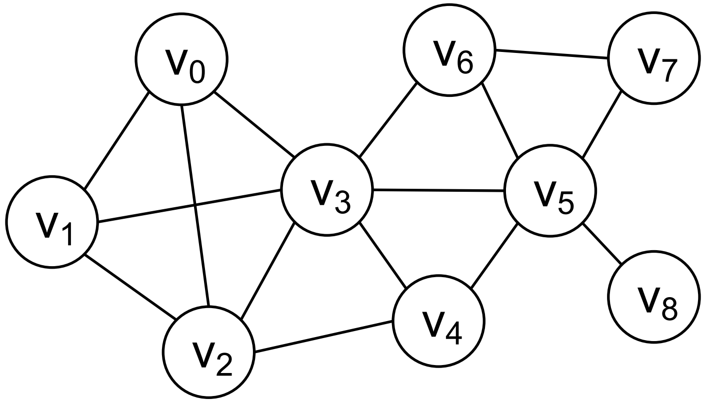
Initialize the core number by degree

Iteration *2*

isContinue ***False***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	3	2	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

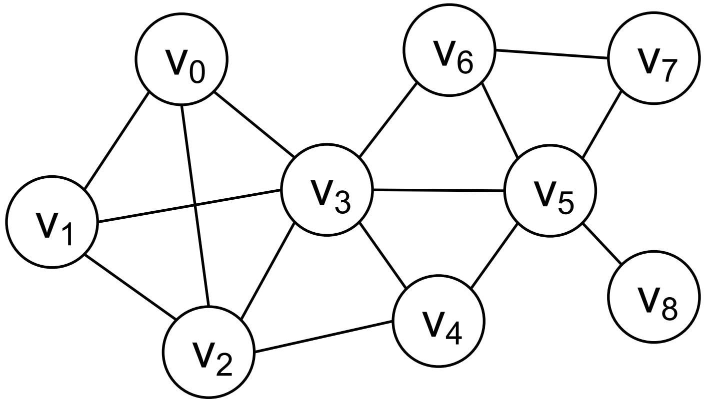
Initialize the core number by degree

Iteration *2*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	3	2	2	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

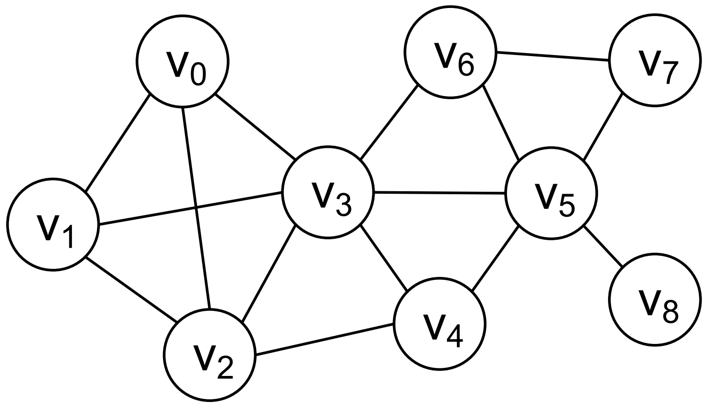
Initialize the core number by degree

Iteration *3*

isContinue ***True***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	2	2	2	2	1

Core Decomposition: Local-view (Converging)



Given a vertex and its core number k:
There exists at least k neighbors with core number k;
There does not exist k+1 neighbors with core number k+1.

Initialize the core number by degree

Iteration ***4***

isContinue ***False***

ID	0	1	2	3	4	5	6	7	8
Core	3	3	3	3	2	2	2	2	1