An efficient algorithm for edge-connectivity degeneracy

and applications

Stratis Limnios (LIX,DaSciM Team)

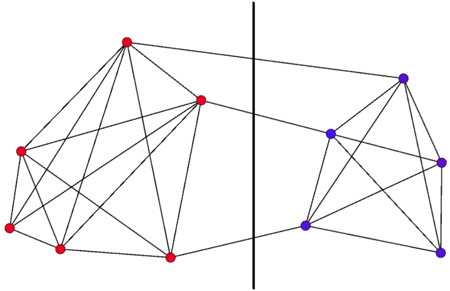
Motivation

Discovering subgraphs with dense connections and/or well **interacting nodes** is a key problem in graph mining. Hence **k-core** decomposition arose, as well as other degeneracy frameworks ((k,l)-core [GTV11], k-truss [RMV15]), to takle the following problems :

- dense subgraph discovery,
- influential spreaders discovery,
- community detection seeding,
- event detection.

Cuts versus k-core

The intuition for the proposed **edge-connectivity degeneracy** is finding subgraphs with high connectivity properties, by means of **minimal cuts**.



Useful properties and definitions for the k-edge connectivity degeneracy

The edge connectivity degeneracy framework

Why edge connectivity needs an efficient algorithm :

- finding **the minimum k-edge-connected spanning subgraph** of G (that is : select as few as possible edges in G that your selection is k-edge-connected) is **NP-hard** [GJ90].
- Several functions exist in the literature that aim to solve this problem but for most formulations those functions are also **hard to approximate.**

Dense subgraph discovery

Definition (Densest subgraph [KS09])

The problem of densest subgraph discovery is given by the following formula :

$$E^*(G) = argmax\{\epsilon(H) | H \subseteq G\}$$

with ϵ a graph density funtion (edge density for instance), and the associated degeneracy

$$\epsilon^*(G) = max\{\epsilon(H) | H \subseteq G\}$$

.

For now on $\epsilon(G) = \frac{|E(G)|}{|V(G)|}$.

The well known k-core

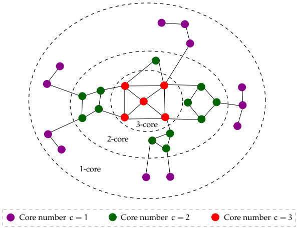
Definition (k-core)

Given the previously defined function δ, we define the k-core of a graph as :

$$\delta^*(G) = max\{\delta(H) | H \subseteq G\}$$

with δ a function yielding the minimal degree in the given subgraph.

$$\delta(H) = min\{deg(v) | v \in V(H)\}$$



Edge connectivity Definition

Definition

A graph is **d-edge connected** if it has at least two vertices and for every two vertices there are d edge disjoint paths between them.

A must have, Menger's theorem

Theorem (Menger [Men27])

Let G be a finite undirected graph and x and y two distinct vertices. Then the size of the minimum edge cut for x and y is equal to the maximum number of pairwise edge-independent paths from x to y.
Extended to subgraphs : *a maximal subgraph disconnected by no less than a k-edge cut is identical to a maximal subgraph with a minimum number k of edge-independent paths between any x, y pairs of nodes in the subgraph.*

Towards edge-connectivity cores

$\rightarrow$ As an application of **Menger's theorem** we will define an **edge-connectivity function** in means of the minimal cut in a given graph G [Men27] :

$$\lambda(S) = |\{e \in E(G) | \{e\} \cap S \neq \varnothing \wedge \{e\} \cap (V(G) \setminus S) \neq \varnothing\}|$$

Corollary (k-edge connected graph)

*A graph G is d-**edge connected** if and only if $\lambda(G) \geq d$.*

Edge-connectivity degeneracy

Definition (Edge-connectivity degeneracy [Bol13])

Let G be a graph. We define the **edge-connectivity degeneracy** of G as a follows :

$$\lambda^*(G) = \max\{\lambda(H) \mid H \subseteq G\}$$

Comments and drawbacks

As we defined the edge-connectivity degeneracy, we need to provide an algorithm that outputs at least a specific core, or at most the whole edge-connectivity decomposition.

Unfortunaty one will face the following problems :

- Computing one core can be done in $\mathcal{O}(n^3)$ (Gabow algorithm [Gab95]) without using minimal cuts.
- Moreover, computing minimal cuts in a graph can be done in $\mathcal{O}(n^2 log(n))$ with the **Karger-Stein** and provides an algorithm in $\mathcal{O}(n^2 log^3 n)$ for edge connectivity cores [KS96].

Density and degeneracy inequalities

Proposition

For every undirected graph G the following inequality holds :

$$2\epsilon^*(G) \geq \delta^*(G) \geq \lambda^*(G) \geq \epsilon^*(G)$$

Corollary (2)

For every undirected graph G, we have :

$$\frac{\delta^*(G)}{2} \leq \lambda^*(G) \leq \delta^*(G)$$

Proof of proposition 1 (1/4)

Theorem (The handshaking)

For every undirected graph G, it holds that :

$$|E(G)| = \frac{1}{2} \sum_{v \in V(G)} deg_G(v)$$

Proof for the proposition (from left to right) : Let H be a subgraph of G, then thanks to the handshaking theorem :

$$\forall H \subseteq G : |E(H)| = \frac{1}{2} \sum_{v \in V(H)} deg_H(v)$$

$$|E(G)| \geq \frac{n_H}{2} min\{deg_H(v), v \in H\}$$

Proof of proposition 1 (2/4)

then :

$$2\epsilon(H) \geq \delta(H)$$

hence, since the above inequality holds for all subgraphs H of G :

$$2\epsilon^*(G) \geq \delta^*(G)$$

Suppose there exists a subgraph $H_{opt} \subseteq G$ such as $\delta^*(G) = \delta(H_{opt}) = k$, then, since k is the minimum degree in H_{opt} we can assume that the minimal cut in H_{opt} is at most k since the worst case scenario is that the minimal cut happens around the vertex with minimal degree.

Hence ensuring $\lambda^*(G) \leq \delta^*(G)$

Proof of proposition 1 (3/4)

To prove that $\lambda^*(G) \geq \epsilon^*(G)$ we will suppose that $\lambda^*(G) = k$ and that there exists a subgraph H of G where $\epsilon^*(G) = \epsilon(H) > k$. In this configuration, we can assume that the node having the minimum degree, say k', will be inferior to k. Then, cutting out this node from H we get a subgraph H' such as :

$$\begin{aligned}
\epsilon(H') &\geq \frac{|E(H)| - k'}{n_H - 1} \\
&> \frac{|E(H)| - \epsilon(H)}{n_H - 1} \\
&> \frac{|E(H)| - \epsilon(H)}{n_H - 1}
\end{aligned}$$

Proof of proposition 1 (4/4)

And note that :

$$
\begin{aligned}
\frac{|E(H)| - \epsilon(H)}{n_H - 1} - \epsilon(H) &= \frac{|E(H)| - \epsilon(H) - n_H\epsilon(H) + \epsilon(H)}{n_H - 1} \\
\frac{|E(H)| - n_H\epsilon(H)}{n_H - 1} &= 0
\end{aligned}
$$

Hence showing the contradiction as we found a densest graph in G than H. $\square$

Nested subgraphs

Theorem

Let G be a graph and let $\mathcal{C} = \{C_1, \dots, C_r\}$ be a collection of vertex disjoint connected subgraphs of G. Let also G' be the graph obtained if we contract in G all edges in the graphs in $\mathcal{C}$. If G' is d-edge connected and each graph in $\mathcal{C}$ is d-edge connected or a single vertex, then G contains a subgraph that is d-edge connected.

An efficient algorithm for the k-edge connectivity cores

The Motivation

It is obvious that the provided algorithm will not be very fast. Indeed, one has to find an original use and computational scheme in order to draw all the informations contained in such cores. Here are the different approches we will be investigating :

- How to compute efficiently these **specific cores**.
- Which core holds the most relevant informations.
- How to use these cores for applications.

Karger-Stein fast mincut algorithm (1/2)

CONTRACT [KS96] :

Data: A graph $G = (V, E)$, $t \in \mathbb{N}$
Result: minimum cut of G
while $|V(G)| > t$ **do**
  Pick a random edge e of the G;
  $G \leftarrow G$;
  return G
end

Algorithm 1: Contraction algorithm $\mathcal{O}(n)$

Karger-Stein fast mincut algorithm (2/2)

FASTCUT [KS96] :

Data: A graph $G = (V, E)$
Result: minimum cut of G
$n \leftarrow |V(G)|$;
if *n<6* **then**
    compute minimum cut of G via brute force and return it;
else
    t$\leftarrow 1 + \frac{n}{\sqrt{2}}$;
    $H_1 \leftarrow CONTRACT(G, t)$;
    $H_2 \leftarrow CONTRACT(G, t)$;
    $X_1 \leftarrow FASTCUT(H_1)$;
    $X_2 \leftarrow FASTCUT(H_2)$;
    return minimum cut of X_1 and X_2
end

Algorithm 2: Fastcut algorithm $\mathcal{O}(n^2 log(n))$

Let $k = \delta^*(G)$. We know that $k/2 \leq \lambda^*(G) \leq k$. For every $z \in [k/2, k]$ (starting from k) we find the z-edge-connectivity core as follows :

Initialisation : Set `found` = FALSE.

Step 1 : Let A_k be the k-core of G.

Step 2 : We run the Karger-Stein algorithm on A_k in order to find an edge cut of size $< z$. If no such set is found, then we **know** that $\lambda^*(A_k) \geq z \Rightarrow \lambda^*(G) \geq z$. If such a cut is found for some $S \subseteq V(A_k)$ we set `found` = TRUE, $A_k^{(1)} = A_k[S]$ and $A_k^{(2)} = A_k \setminus S$. We recursively run the Karger-Stein algorithm on $A_k^{(1)}$ and $A_k^{(2)}$. When this recursion finishes we obtain a partition of A_k in to sets $\mathcal{S} = \{S_1, \ldots, S_q\}$ where each S_i either is a singleton or induces a z-edge-connected subgraph in A_k.

K-edge connectivity finding algorithm (2/2)

Step 3 : If $\mathcal{S}$ is not trivial, we know that $\lambda^*(A_k) \geq z \Rightarrow \lambda^*(G) \geq z$. We then construct G' by contracting all non-trivial S_i's. We denote by G' the resulting graph. Notice that, by Proposition 2, $\lambda^*(G') \geq z \Leftrightarrow \lambda^*(G) \geq z$. We then set $G := G'$, `found` $:=$ TRUE and we go to step 1.

Step 4 : If $\mathcal{S}$ is trivial and `found` $=$ TRUE we use the essential singletons in $\mathcal{S}$ to build the z-edge core partition and stop.

Step 5 : If $\mathcal{S}$ is trivial and `found` $=$ FALSE we set $k := k - 1$ and we go to step 1.

Because of Corollary 2, k will never become less than $\delta^*(G)/2$, therefore the above algorithm will terminate.

To compute the recursive decomposition of A_k we used **Breadth first Search** (BFS) algorithm as each level of the tree contains the decompositions of the previous level.

Complexity : We remind that the best existing algorithm for this task has a complexity of $\mathcal{O}(n^2 log^3 n)$. Here we provided an algorithm that computes the k-edge connectivity core more efficiently, i.e. considering the number of nodes in the $(k/2)^{th}$-core is $n_{k/2}$, the complexity of the proposed algorithm is at most $\mathcal{O}(\frac{k}{2} n_{k/2}^2 log^2(n_{k/2}))$.

Core properties

and illustrations for applications

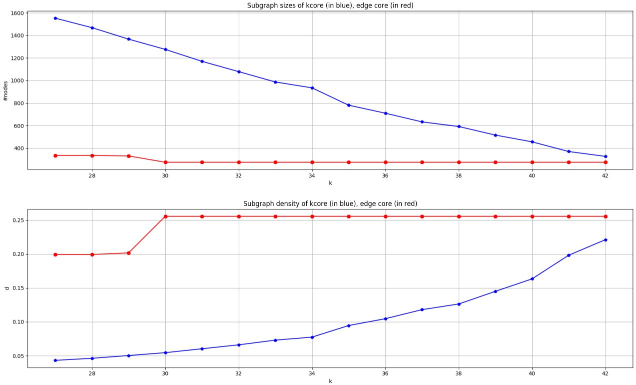
Datasets used for the experiments

We used the following datasets from Stanford's library SNAP, those datasets are social networks, such as collaboration networks :

Network Name	Nodes	Edges	k_{max}	$\lvert\mathcal{C}\rvert$
Email-Enron	$33,696$	$180,811$	43	275
DBLP (weighted)	$1,088,681$	$4,512,205$	258	4

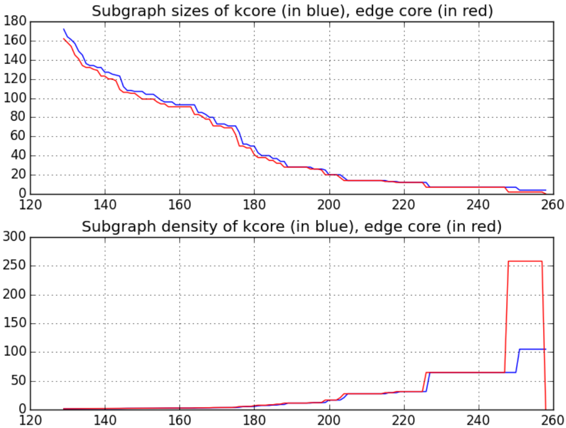
Comparing edge-connectivity cores and degree cores (first case)

We noticed while experimenting two tendances in the edge connectivity core production :



— Core decompositions for the Email-Enron dataset

Comparing edge-connectivity cores and degree cores (second case)



– Core decompositions for the DBLP $(1999 - 2016)$ dataset

Further study

As we notice, the edge-connectivity cores are either **very similar to the deepest k-core very fast**, either **very similar to the corresponding k-core**. This, especially for the first case, can problematic as the time to compute the whole edge connectivity cores decomposition is very slow, even with our algorithm.

Thankfully, since the complexity of our approche is depending on the size of the $(k/2)^{th}$-core, the algorithm can work well **even on very large datasets.**

Illustration of the first behaviour (EmailEnron)

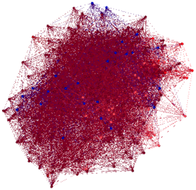


Figure – Enron 42-core (blue) and 22-edge-comnectivity core (red)

Illustration of the first behaviour (EmailEnron)

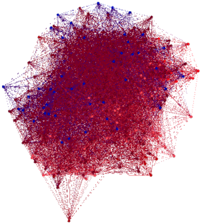


Figure – Enron 41-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

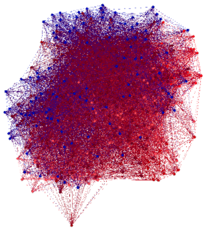


Figure – Enron 40-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

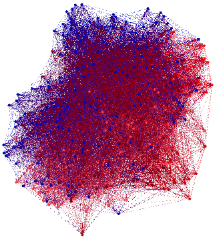


Figure – Enron 39-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

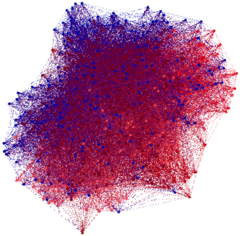


Figure – Enron 38-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

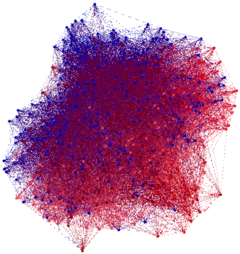


Figure – Enron 37-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

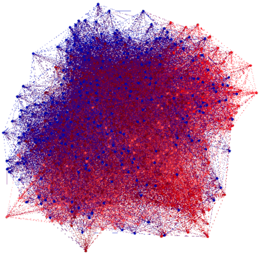


Figure – Enron 36-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

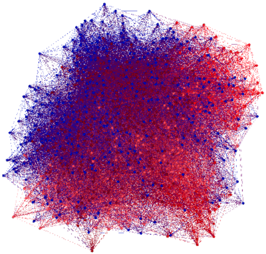


Figure – Enron 35-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

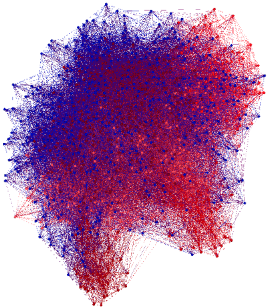


Figure – Enron 34-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

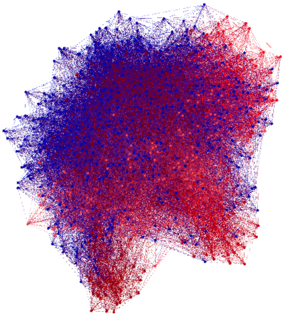


Figure – Enron 33-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

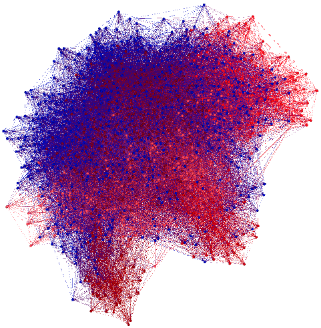


Figure – Enron 32-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

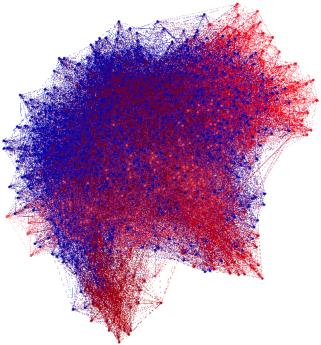


Figure – Enron 31-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

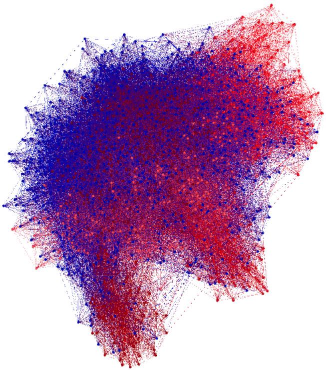


Figure – Enron 30-core (blue) and 22-edge-connectivity core (red)

Illustration of the first behaviour (EmailEnron)

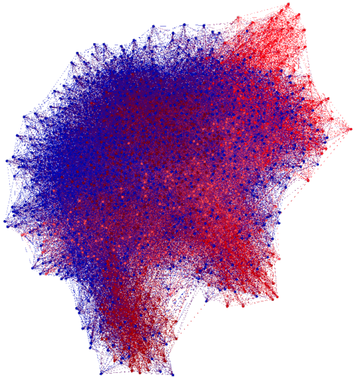


Figure – Enron 28-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

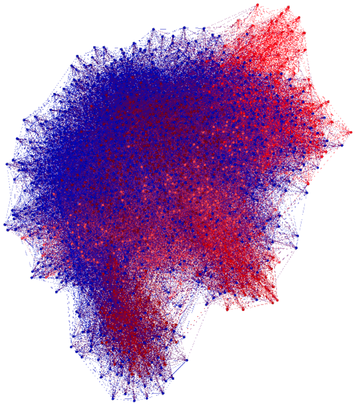


Figure – Enron 27-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

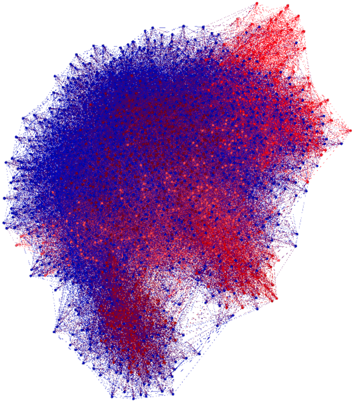


Figure – Enron 26-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

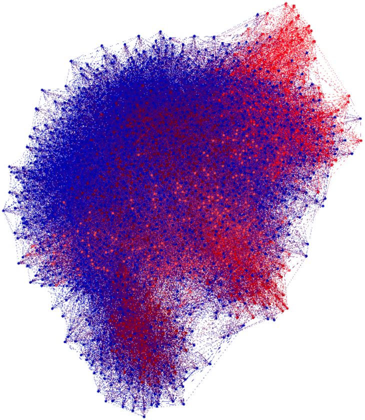


Figure – Enron 24-core (blue) and 22-edge-conectivity core (red)

Illustration of the first behaviour (EmailEnron)

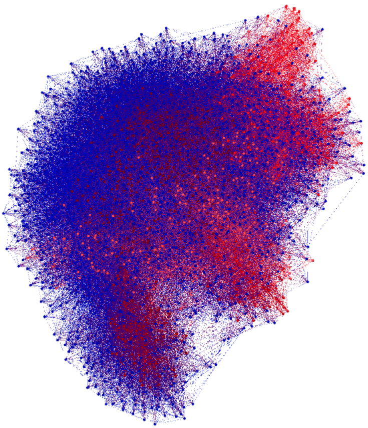
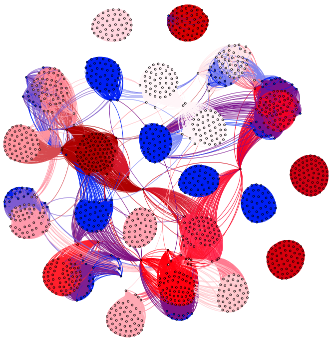


Figure – Enron 22-core (blue) and 22-edge-connectivity core (red)

Illustration of the second behaviour (DBLP)



– edge core decomposition for the DBLP dataset

Getting good spreaders

Since the returned subgraphs are **highly connected**, we assume that if one node of one of these subgraphs is infected then the infection will spread very fast to the rest of this subgraph.

Hence we had the following idea :

$\rightarrow$ **we keep the contracted subgraphs** at each step of the algorithm, previously in dark red (for the first ones found) to lighter red (for the last ones encountered) and **aim to extract the separating edges**. Finally we keep the nodes at the end of these edges with the **higher degrees**.

Results and Discussion

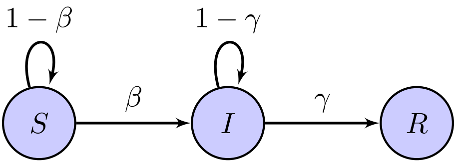
Evaluation

We will evaluate the spreaders we found with the **Susceptible-Infected-Recovered** (SIR) model on the previously presented datasets :

Network Name	**Nodes**	**Edges**	k_{max}	T_{max}	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{T}\rvert$	β
Email-Enron	$33,696$	$180,811$	43	22	275	45	0.01
Wiki-Vote	$7,066$	$100,736$	53	23	336	50	0.009

Susceptible-Infected-Recovered (SIR) model

1. Set candidate node as infected (I state)
2. An infected node can infect its susceptible neighbors with probability β
3. An infected node can recover (stop being active) with probability γ
4. Count the total number of infected individuals (avg. over multiple runs)

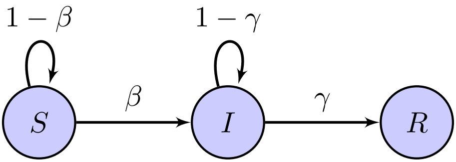


State diagram of the SIR model

[Anderson et al., Oxford university press '92]

Susceptible-Infected-Recovered (SIR) model

1. Set β close to the epidemic threshold $\tau = \frac{1}{\lambda_1}$
   - λ_1 being the largest eigenvalue of $\mathbf{A}$ (adjacency matrix)
2. Set $\gamma = 0.8$



State diagram of the SIR model

[Anderson et al., Oxford university press '92]

Stepwise evalution of spreading performance

Time Step

	Method	2	4	6	8	10	*Final step*	*Max step*
Email-Enron	**edge core**	21.47	115.13	350.13	428.23	250.93	2,647.74	28
	truss	8.44	46.66	204.08	418.77	355.84	2,596.52	33
	core	4.78	31.97	152.55	367.28	364.13	2,465.60	37
	top degree	6.89	34.13	155.48	360.89	357.08	2,471.67	36
Wiki-Vote	**edge core**	5.15	12.15	24.72	40.96	52.74	626.09	34
	truss	2.92	6.92	15.27	28.73	42.46	560.66	52
	core	1.92	4.78	10.65	20.66	32.40	466.01	57
	top degree	2.43	5.46	12.05	23.05	35.55	502.88	62

Discutions and future work

As we saw, the cores produced by the edge-connectivity have a lot of very interesting properties, such as **the ability to produce very good spreaders**.

Nevertheless, another application of this work can focus on **influence maximisation problems**. For instance one can find a sufficiently dense/big core and try a greedy algorithm in order **to find the most influencial nodes** in those subgraphs (lets say one for each disconnected subgraph) with the intuition that it will **influence very quickly** the rest of the subgraph, hence maximizing influence localy.

Thank

You !

Béla Bollobás, *Modern graph theory*, vol. 184, Springer Science & Business Media, 2013.

Harold N Gabow, *A matroid approach to finding edge connectivity and packing arborescences*, Journal of Computer and System Sciences **50** (1995), no. 2, 259–273.

Michael R. Garey and David S. Johnson, *Computers and intractability ; a guide to the theory of np-completeness*, W. H. Freeman & Co., New York, NY, USA, 1990.

Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis, *D-cores : Measuring collaboration of directed graphs based on degeneracy*, Data Mining (ICDM), 2011 IEEE 11th International Conference on, IEEE, 2011, pp. 201–210.

David R. Karger and Clifford Stein, *A new approach to the minimum cut problem*, J. ACM **43** (1996), no. 4, 601–640.

Samir Khuller and Barna Saha, *On finding dense subgraphs*, Automata, Languages and Programming (2009), 597–608.

Karl Menger, *Zur allgemeinen kurventheorie*, Fundamenta Mathematicae **10** (1927), no. 1, 96–115.

Maria-Evgenia G. Rossi, Fragkiskos D. Malliaros, and Michalis Vazirgiannis, *Spread it good, spread it fast : Identification of influential nodes in social networks*, Proceedings of the 24th International Conference on World Wide Web (New York, NY, USA), WWW '15 Companion, ACM, 2015, pp. 101–102.