

# Learning from Networks

## Graph Analytics: Node-Level

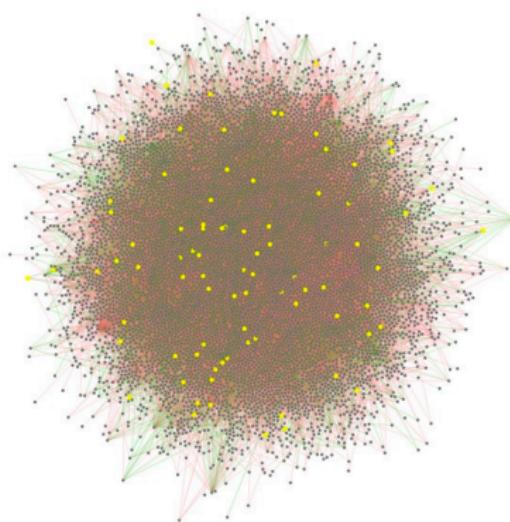
Fabio Vandin

October 8<sup>th</sup>, 2024

## Motivation

In several applications: interested in computing some scores of importance for nodes based on the network structure

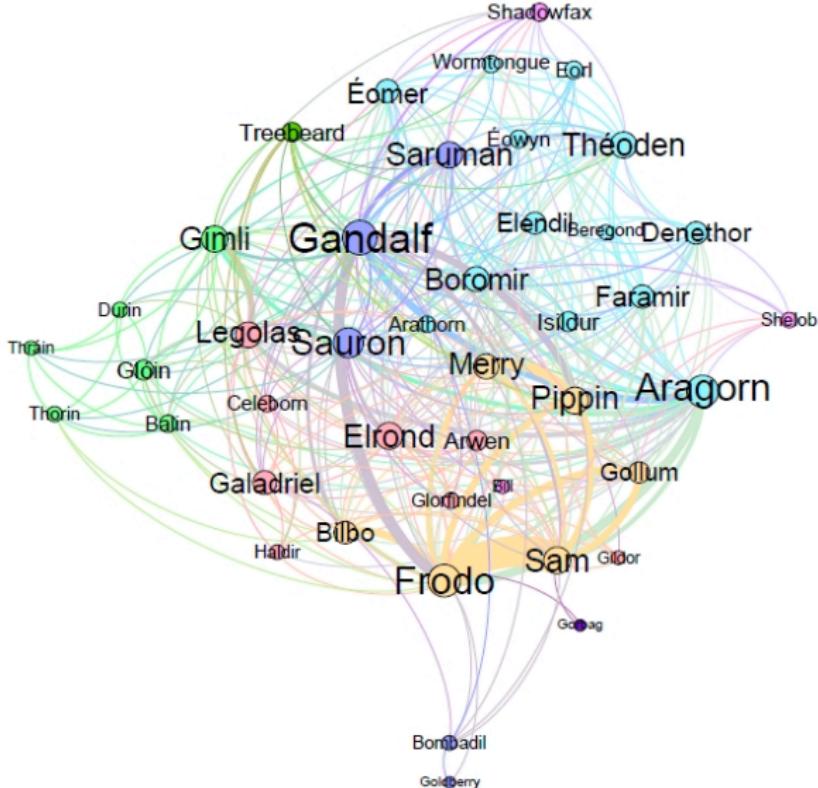
### Example: Protein-Protein Interaction Network



What proteins are most important in the network?

Useful for analyzing mutations in diseases, drug development,...

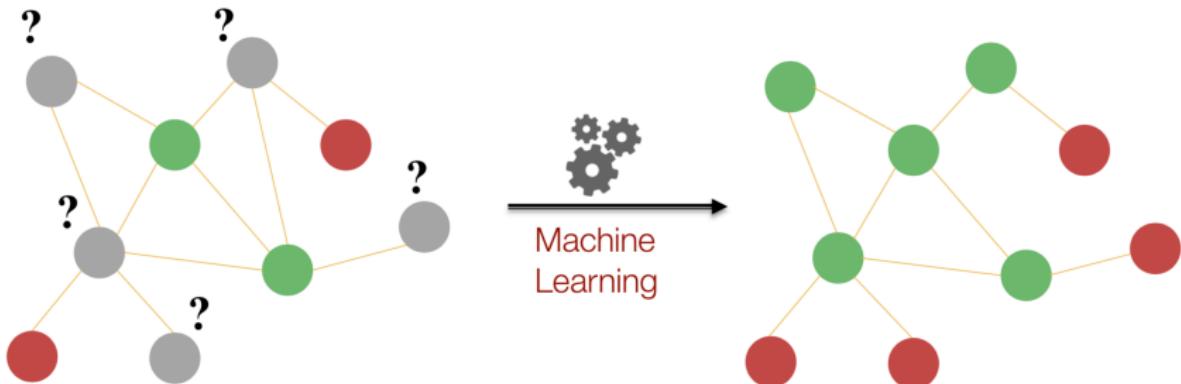
## Motivation (continue)



<http://www.morethanbooks.eu/graph-network-of-the-lord-of-the-rings/>

## Motivation: A Different View

If you have a machine learning task defined at the node level, you need *features* for nodes.



Features:

- *external*: qualities/measures for each node, not related to the network
- *topological*: features related to the network structure

The scores of importance are often good features for nodes.

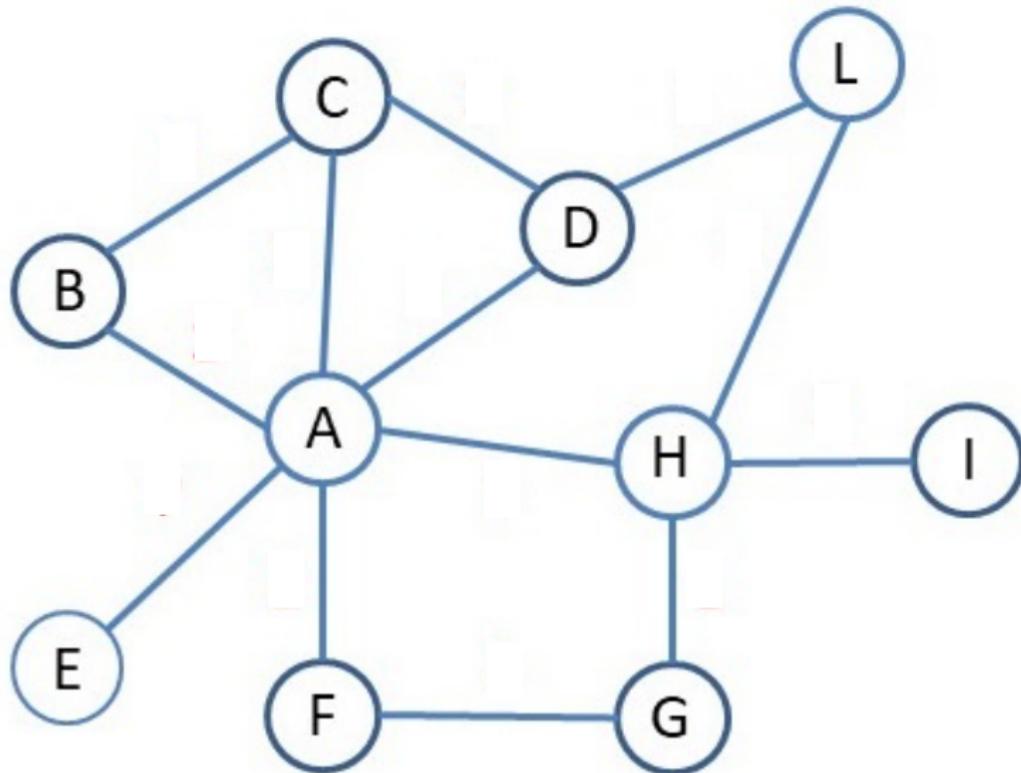
## A First Score: Node Degree

What is the easiest node score we can derive from a network?

The node degree:  $\text{degree}(u)$

Complexity to compute the node degree for all  $v \in V$ ?  $\Theta(|V|)$

## Example



## Node Centralities

Several scores have been proposed to measure the *centrality* of a node in a network.

The most important and commonly used ones are:

- *closeness centrality*
- *betweenness centrality*

# Closeness Centrality

**Intuition:** a node is central if it is fairly close to the other nodes in the network.

Let  $\underline{G = (V, E)}$  be a connected, undirected graph (weighted or unweighted)

## Definition

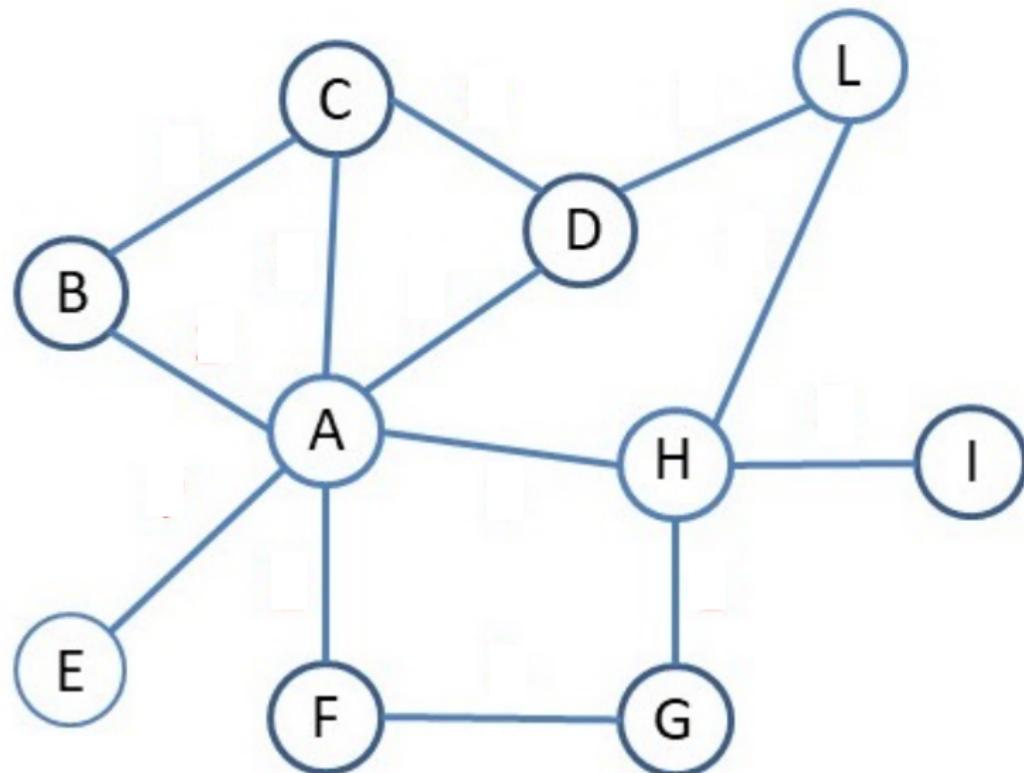
Given a node  $v$ , the closeness centrality  $c(v)$  of  $v$  is:

$$c(v) = \frac{n - 1}{\sum_{u \neq v, u \in V} d(v, u)}$$

**Note:** since  $d(v, v) = 0$  for all  $v \in V$ :

$$c(v) = \frac{n - 1}{\sum_{u \in V} d(v, u)}$$

## Closeness Centrality: Example Unweighted Graph



## Distance: Weighted vs Unweighted Graphs

Given  $u, v \in V$ , let  $d(u, v)$  the *distance* between  $u$  and  $v$ .

$G$  *unweighted*:  $d(u, v)$  is the number of edges in a *shortest path* between  $u$  and  $v$

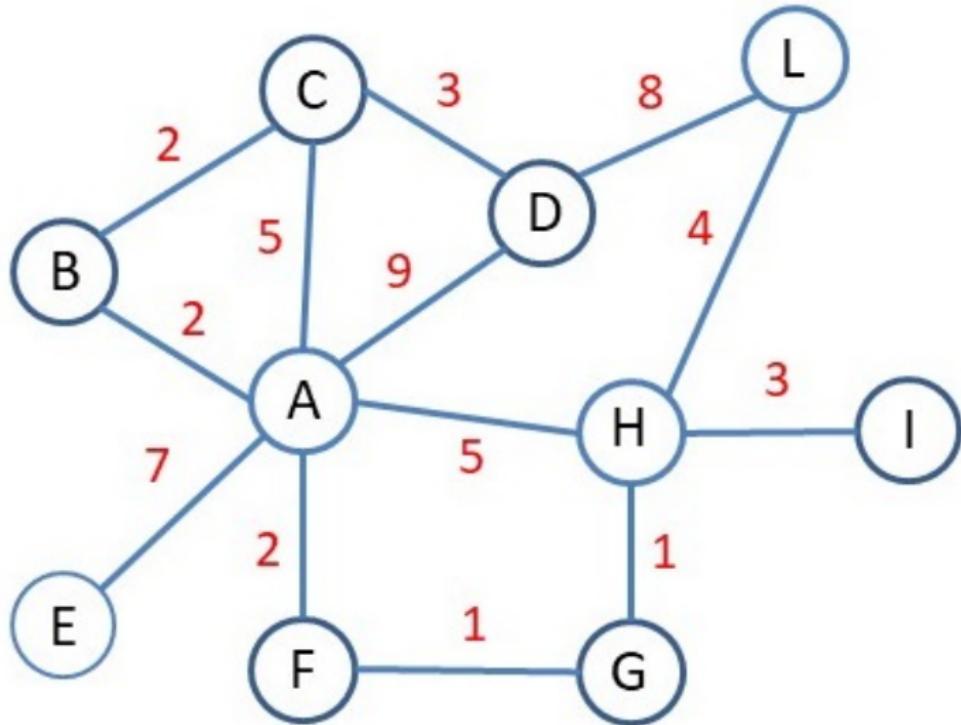
$G$  *weighted*:

- edge weight function  $w : E \rightarrow \mathbb{R}^+$  ( $w(e) \in \mathbb{R}^+$ )
- length of path  $[u_1, u_2, \dots, u_k]$  is:

$$\sum_{i=1}^{k-1} w(u_i, u_{i+1})$$

- $d(u, v)$  is the minimum length of a path from  $u$  to  $v$
- a path of length  $d(u, v)$  between  $u$  and  $v$  is a shortest path

## Closeness Centrality: Example Weighted Graph



## Computing Closeness Centrality: One Node

Given  $v \in V$ , how can we compute its closeness centrality  $c(v)$ ?

We need the values  $d(v, u) \forall u \in V \dots$

### Definition

Given a simple, undirected graph  $G = (V, E)$  and a vertex  $v \in V$ , the Single-Source Shortest Paths (SSSP) problem requires to find all the distances between  $v$  and the other vertices in  $V$  (and the relative shortest paths).

If  $G$  is unweighted: SSSP can be solved by a simple modification of  $\text{BFS}(G, v)!$

## Computing Closeness Centrality: One Node (continue)

Let  $\text{distBFS}(G, v)$  be  $\text{BFS}(G, v)$  with the following changes:

- every node  $u \in V$  has a field  $u.\text{distance}$
- initially:  $u.\text{distance}$  has value  $0, \forall u \in V$
- when node  $w$  is visited and then inserted in list  $L_{i+1}$ , we set  $w.\text{distance}$  to  $i + 1$ . That is, the instruction

“visit  $w$ ”

is substituted with

“ $w.\text{distance} \leftarrow i + 1$ ”

-complexità  $\Theta(|E|)$  (grafo connesso)

### Proposition

When  $\text{distBFS}(G, v)$  terminates,  $u.\text{distance} = d(v, u)$ .

## Computing Closeness Centrality: One Node (continue)

**Algorithm** ClosenessCentrality( $G, v$ )

**Input:** unweighted graph  $G = (V, E)$  with  $|V| = n$  and  $|E| = m; v \in V$

**Output:** closeness centrality  $c_v$  of  $v$

$\text{distBFS}(G, v); \Theta(m)$

$sum_v \leftarrow 0;$

**forall**  $u \in V$  **do**

$sum_v \leftarrow sum_v + u.distance;$

**return**  $(n - 1) / sum_v;$

$\} \Theta(n)$

Complexity?  $\Theta(n+m) = \Theta(m)$  dato che  $G$  connesso  $\rightarrow m \geq n-1$

## Computing Closeness Centralities

What about computing the node centralities of all nodes in  $G$ ?

**Algorithm** ClosenessCentralities( $G$ )

**Input:** unweighted graph  $G = (V, E)$  with  $|V| = n$  and  $|E| = m$

**Output:** closeness centrality  $c(v)$  for all  $v \in V$

**forall**  $v \in V$  **do**

$\lfloor c(v) \leftarrow \text{ClosenessCentrality}(G, v);$

**return** values  $c(v);$

**Complexity?**  $\Theta(n \cdot m) \Rightarrow$  dipende da relazione  $n, m$

## Computing Closeness Centrality: Weighted Graph

Let  $G$  be edge weighted: given  $v \in V$ , how can we compute its closeness centrality  $c(v)$ ?

We need the values  $d(v, u) \forall u \in V \dots$

Dijkstra algorithm: solves the SSSP problem using a priority queue

Complexity of Dijkstra algorithm: depends on how the priority queue is implemented

- priority queue implemented with a heap:  
 $\Theta(\min\{n^2, (n + m) \log n\})$
- priority queue implemented with a Fibonacci heap (list of heaps):  $\Theta(m + n \log n)$

## Computing Closeness Centralities: Weighted Graph

Let ClosenessCentralityW( $G, v$ ) be the algorithm, based on Dijkstra algorithm, to compute the closeness centrality for node  $v$ .

Analogously to before, the node centralities of all nodes in  $G$  are computed as follows. *on Fibonacci heap:  $\Theta(m + n \log n)$*

**Algorithm** ClosenessCentralities( $G$ )

**Input:** weighted graph  $G = (V, E, w)$  with  $|V| = n$ ,  $|E| = m$ ,  
 $w : E \rightarrow \mathbb{R}^+$  *connected*

**Output:** closeness centrality  $c(v)$  for all  $v \in V$

**forall**  $v \in V$  **do**

$c(v) \leftarrow \text{ClosenessCentralityW}(G, v);$

**return** values  $c(v);$

**Complexity?**  $\Theta(n(m + n \log n))$  if the version of Dijkstra algorithm based on Fibonacci heaps is used.

Can we do better?

# Computing Closeness Centralities: Weighted Graph (continue)

We need the values  $d(v, u) \forall u, v \in V$ .

## Definition

Given a simple graph  $G = (V, E)$ , the All-Pairs Shortest Paths (APSP) problem requires to find all the distances  $d(u, v)$  for all pairs  $u, v \in V$  (and the relative shortest paths).

Floyd-Warshall algorithm: solves the APSP!

mejorio perché non  
sempre  $n^3$

Complexity?  $\Theta(n^3) \Rightarrow$  rispetto a  $\Theta(n(m+n \log n))$ , meglio se  $m \ll n^2$

Johnson's algorithm: solves the APSP problem. Time?  $\Theta(n^2 \log n + nm)$

**Note:** while these algorithms do not improve on Dijkstra algorithm, they can be used when negative weights are present (no negative-weight cycles are allowed).

# Computing Closeness Centralities for Large Graphs

Up to know: we can compute the closeness centralities for all nodes in a network  $G = (V, E)$ ,  $|V| = n$ ,  $|E| = m$ , in time

- $G$  unweighted:  $\Theta(nm)$
- $G$  weighted:  $\Theta(n(m + n \log n))$

These complexities are impractical for large networks!

What about computing *approximations* of closeness centralities instead of the exact values?

# Approximating Closeness Centralities

$$c(1) = d(1,1) + d(1,2) + \dots + d(1,n)$$

$$c(2) = d(2,1) + d(2,2) + \dots + d(2,n)$$

$$c(3) = d(3,1) + d(3,2) + \dots + d(3,n)$$

↓

calcolo  $c(1)$ , poi posso ricavare  
 $d(1,2)$  in  $c(2)$

↓

anziché n BFS, calcolo k BFS

come scelgo k?

in calcolo  $c_v$ , uso solo  
k distanze  $d(v, u_1), \dots, d(v, u_k)$ ,  
invece di tutte  $(n-1)$   
distanza → stima alla fine varrà



$$\hat{c}(v) = \frac{n-1}{\frac{n}{k} \sum_{i=1}^k d(v, u_i)}$$

## Approximating Closeness Centralities (continue)

## Eppstein-Wang Algorithm

D. Eppstein, J. Wang, (2006). *Fast approximation of centrality*. Journal of Graph Algorithms and Applications, 5(5), 39.

Same algorithm for both weighted and unweighted graphs, only difference: how the SSSP problem is solved.

## Eppstein-Wang Algorithm (continue)

**Algorithm** ApproximateClosenessCentralities( $G, k$ )

**Input:** weighted/unweighted graph  $G = (V, E)$  with  $|V| = n$ ,  
 $|E| = m$ ;  $k \in \mathbb{N}$  connex

**Output:** approximation  $\hat{c}(v)$  of  $c(v)$  for all  $v \in V$

$sum_v \leftarrow 0$  for all  $v \in V$ ;

**for**  $i \leftarrow 1$  to  $k$  **do**

$v_i \leftarrow$  random vertex chosen uniformly at random from  $V$ ;

    solve SSSP problem with source  $v_i$ ;

**forall**  $v \in V$  **do**

$sum_v \leftarrow sum_v + d(v_i, v)$ ;

**forall**  $v \in V$  **do**

$\hat{c}(v) \leftarrow 1 / \left( \frac{n \cdot sum_v}{k(n-1)} \right)$ ;

**return** values  $\hat{c}(v)$ ;

## Eppstein-Wang Algorithm: Example

$$K=3$$

$$\text{it. 1: } v_1 = G$$

$$\text{it. 2: } v_2 = G$$

$$\text{it. 3: } v_3 = C$$

$$3+3+1=7$$

B

C

D

L

A

H

I

E

F

G

$$1+1+2=4$$

$$0+0+3=3$$

$$4+4+0=8$$

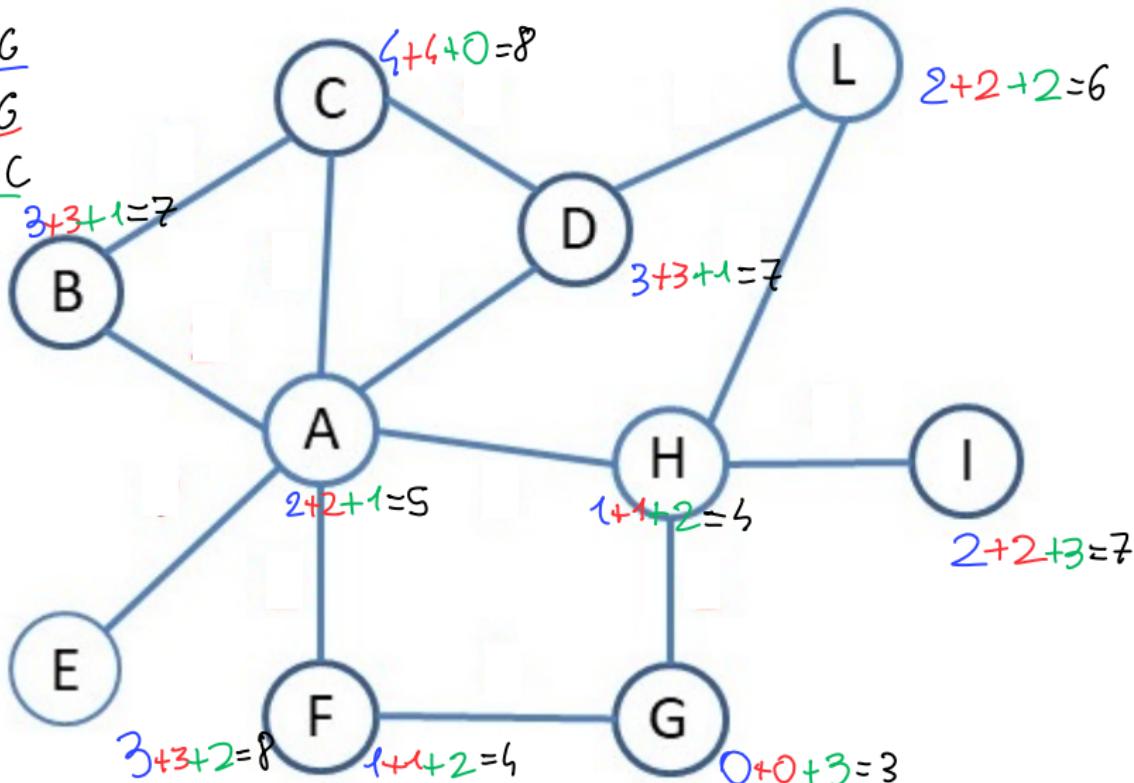
$$2+2+2=6$$

$$2+2+1=5$$

$$1+1+2=4$$

$$2+2+3=7$$

$$3+3+2=8$$



## Eppstein-Wang Algorithm: Analysis

Ex: se  $c(v) = 0.49$  e  $\hat{c}(v) = 0.5$ ,  $\left| \frac{1}{\hat{c}(v)} - \frac{1}{c(v)} \right| \approx 0.04$

$$\text{if } \approx 0.04 \quad \text{if } \approx 0.01, \quad \text{if } \approx 15$$

For simplicity, we are going to consider the inverse centrality estimator  $\frac{1}{\hat{c}(v)}$  and the inverse centrality  $\frac{1}{c(v)}$ .

### Proposition

$$\mathbb{E} \left[ \frac{1}{\hat{c}(v)} \right] = \frac{1}{c(v)}.$$

Dim: per vertice  $v$ ,  $X_i$ : r.v. con valore  $d(v, v_i) \cdot \frac{n}{n-1}$ ,  $v_i$  = nodo scelto  $\alpha$  it.  $i$

$$\frac{1}{\hat{c}(v)} = \frac{1}{K} \cdot \sum_{i=1}^K X_i \Rightarrow \mathbb{E} \left[ \frac{1}{\hat{c}(v)} \right] = \frac{1}{K} \cdot \sum_{i=1}^K \mathbb{E}[X_i]$$

In ogni it.  $i$  scegliamo  $v_i$  random uniformemente, ogni vertice ha prob.  $\frac{1}{n}$  di essere scelto  $\Rightarrow \forall i \in [1, K], \mathbb{E}[X_i] = \sum_{j=1}^n \frac{1}{n} \cdot d(v, v_j) \cdot \frac{n}{n-1} =$

$$= \sum_{u \in V} \frac{d(v, u)}{n-1} = \frac{1}{n-1} \left( \sum_{u \in V} d(v, u) \right)$$

$$\begin{aligned}\mathbb{E}\left[\frac{1}{c(v)}\right] &= \frac{1}{k} \cdot \sum_{i=1}^k \frac{1}{n-1} \left( \sum_{u \in V} d(v, u) \right) = \frac{1}{k} \cdot \frac{1}{n-1} \cdot \sum_{u \in V} d(v, u) = \\ &= \frac{1}{n-1} \sum_{u \in V} d(v, u) = \frac{1}{c(v)}\end{aligned}$$

## Eppstein-Wang Algorithm: Analysis (continue)

We are now going to prove that the estimates  $\frac{1}{\hat{c}(v)}$  are close to their expectations  $\frac{1}{c(v)}$  if the number  $k$  of iterations in the algorithm is large enough.

We use (without proving it) the following *concentration* result.

### Hoeffding's inequality

Let  $X_1, X_2, \dots, X_k$  are independent random variables, with  $a_i \leq X_i \leq b_i$  for all  $i \in \{1, \dots, k\}$  and let  $\mu = \mathbb{E} \left[ \frac{\sum_{i=1}^k X_i}{k} \right]$ .

Then for  $\varepsilon > 0$ :

$$\mathbb{P} \left[ \left| \frac{\sum_{i=1}^k X_i}{k} - \mu \right| \geq \varepsilon \right] \leq 2e^{-2k^2\varepsilon^2 / \sum_{i=1}^k (b_i - a_i)^2}$$

## Eppstein-Wang Algorithm: Analysis (continue)

Let  $\Delta(G) = \max_{u,v \in V} d(u, v)$  be the *diameter* of a graph  $G$ .

### Proposition

Let  $\varepsilon > 0$  and  $\delta \in (0, 1)$  be constants. If  $k \geq \frac{1}{2\varepsilon^2} \left( \log \frac{2n}{\delta} \right) \left( \frac{n}{n-1} \right)^2$  then the inverse centrality estimator  $1/\hat{c}(v)$  from ApproximateClosenessCentralities( $G, k$ ) is within an additive factor  $\varepsilon\Delta(G)$  of  $1/c(v)$  for all vertices  $v \in V$  with probability  $\geq 1 - \delta$ .

**Note:** in several real-world networks  $\Delta(G) \in O(\log n)$  - small world phenomenon.

Dim: prendo vertice arbitrario  $v \in V$

Fissa  $X_i$  r.v. con valore  $d(v, v_i) \cdot \frac{n}{n-1}$ .  $v_i$ : vertice preso in it.  $i$  di adj.

Ottieniamo:  $\frac{1}{\hat{C}(v)} = \frac{1}{K} \cdot \sum_{i=1}^K X_i \Rightarrow \mathbb{E}\left[\frac{1}{\hat{C}(v)}\right] = \frac{1}{C(v)}$

$\forall i \in [1, K]: 0 \leq X_i \leq \Delta(G) \cdot \frac{n}{n-1}$

Ziehtfolgend: ε in definizione ε in prop.  
 $\uparrow$   $\uparrow$

$$\mu = \frac{1}{C(v)}, a_i = 0, b_i = \Delta(G) \cdot \frac{n}{n-1}, E_H = \epsilon \Delta(G) \Rightarrow$$

$$\Rightarrow \Pr\left[\left|\frac{1}{\hat{C}(v)} - \frac{1}{C(v)}\right| > \epsilon \Delta(G)\right] \leq 2e^{-2K^2(\epsilon \Delta(G))^2 / \sum_{i=1}^K \left(\Delta(G) \cdot \frac{n}{n-1}\right)^2} =$$

$$= 2e^{-2K^2 \epsilon^2 \Delta^2(G) / K \cdot (\Delta(G))^2 \cdot \left(\frac{n}{n-1}\right)^2} \stackrel{\text{up}}{\leq} 2e^{-2 \cdot \frac{1}{2K^2} \left(\log \frac{2n}{\delta}\right) \left(\frac{n}{n-1}\right)^2 \cdot \epsilon^2 / \left(\frac{n}{n-1}\right)^2} =$$

$$= \frac{1}{2n} \cdot \frac{\delta}{n} = \frac{\delta}{n^2} \Rightarrow \Pr\left[\left|\frac{1}{\hat{C}(v)} - \frac{1}{C(v)}\right| > \epsilon \Delta(G)\right] \stackrel{\text{up}}{\leq} \frac{\delta}{n^2}$$

Prendo evento  $\star = \exists \text{ almeno un vertice } v : \left| \frac{1}{c(v)} - \frac{1}{\bar{c}(v)} \right| > \epsilon \Delta(G)$   $\Rightarrow$

$$\Rightarrow \star = \bigcup_{u \in V} E_u \Rightarrow \Pr[\star] \leq \sum_{u \in V} \Pr[E_u] = n \cdot \frac{\delta}{n} = \delta \Rightarrow$$

$$\Rightarrow \Pr[\bar{\star}] \geq 1 - \delta \quad \blacksquare$$



## Eppstein-Wang Algorithm: Analysis (continue)

### Corollary

If  $k \in \Theta\left(\frac{\log n}{\varepsilon^2}\right)$  for a constant  $\varepsilon > 0$ , then the inverse centrality estimator  $1/\hat{c}(v)$  from

`ApproximateClosenessCentralities( $G, k$ )` is within an additive factor  $\varepsilon\Delta(G)$  of  $1/c(v)$  for all vertices  $v \in V$  with high probability, i.e. with probability  $\geq 1 - 1/n$ .

## Eppstein-Wang Algorithm: Complexity

The solution of  $k$  SSSP problems are needed.  $\Rightarrow$  dominator tempo

- $G$  unweighted:  $\Theta(km)$
- $G$  weighted:  $\Theta(k(m + n \log n))$

Fixing  $k \in \Theta\left(\frac{\log n}{\varepsilon^2}\right)$  as in the Corollary, the complexity is:

- $G$  unweighted:  $\Theta\left(\frac{m \log n}{\varepsilon^2}\right)$
- $G$  weighted:  $\Theta\left(\frac{\log n}{\varepsilon^2}(m + n \log n)\right)$

## Closeness Centrality: More Advanced Algorithms

While Eppstein-Wang algorithm provides rigorous guarantees and useful results, the error of the estimates may be large when the distribution of distances in  $G$  is skewed (e.g., the diameter  $\Delta(G)$  is large).

More advanced algorithm to estimate closeness centralities:

S. Chechik, E. Cohen, and H. Kaplan. *Average distance queries through weighted samples in graphs and metric spaces: high scalability with tight statistical guarantees.* APPROX/RANDOM 2015.

## Closeness Centrality: Chechik-Cohen-Kaplan algorithm

**Basic idea:** use sampling to choose a sample  $S$  of  $k$  nodes for which the SSSP problem is solved and use the corresponding distances to estimate centralities for all nodes (as in Eppstein-Wang algorithm).

**Main difference with Eppstein-Wang:**  $S$  is not picked by choosing nodes uniformly at random

- for each vertex  $v \in V$ , compute a probability  $p_v$ , then include  $v$  in  $S$  with probability  $p_v$  independently of all other events (*Poisson sampling*)
- $p_v$ 's computation:
  - first draw a small sample  $S_0$  (uniformly at random) and compute  $W_s = \sum_{v \in V} d(s, v)$  for all  $s \in S_0$
  - $p_v = \max \left\{ \frac{1}{n}, \max_{s \in S_0} \frac{d(s, v)}{W_s} \right\}$

Provides better guarantees than Eppstein-Wang (e.g., no dependence on the diameter  $\Delta(G)$  of  $G$ ).

## Closeness Centrality: Example

Bergamini et al., 2019. *Computing top- $k$  Closeness Centrality Faster in Unweighted Graphs*, ACM Transactions on Knowledge Discovery from Data (TKDD).

Analysis of the Internet Movie DataBase (IMDB) graph:

- nodes are actors;
- edges: two actors are connected if they played together in a movie (TV-series are ignored)

Considered:

- snapshots of the actor graph, taken every 5 years from 1940 to 2010, and 2014.
- some genres excluded: awards shows, documentaries, game-shows, news, realities, and talk-shows

## Closeness Centrality: Example (continue)

Most central actors in the IMDB graph with respect to the closeness centrality measure.



Semels ('40)



Corrado ('45)



Flowers ('50-'80)



Welles ('85-'90)



Lee ('95-'00)



Hitler ('05-'10)



Madsen ('14)

Top 10  
2014

- |                      |
|----------------------|
| Madsen, Michael (I)  |
| Trejo, Danny         |
| Hitler, Adolf        |
| Roberts, Eric (I)    |
| De Niro, Robert      |
| Dafoe, Willem        |
| Jackson, Samuel L.   |
| Keitel, Harvey (I)   |
| Carradine, David     |
| Lee, Christopher (I) |

## Closeness Centrality for Disconnected Graphs?

Various extension's, we consider: *Lin's index* [N. Lin. Foundations of social research. McGraw-Hill, 1976.]

Let  $G = (V, E)$  be an undirected graph. Let  $C_v$  be the connected component of  $v$ , i.e., of the set of vertices reachable from  $v$ . Let  $n_v = |C_v|$

### Definition

*Lin's index* for  $v \in V$  is

$$c(v) = \frac{(n_v - 1)^2}{(n - 1) \sum_{u \in C_v} d(v, u)}$$

Note that

$$c(v) = \frac{(n_v - 1)^2}{(n - 1) \sum_{u \in C_v} d(v, u)} = \frac{n_v - 1}{\sum_{u \in C_v} d(v, u)} \frac{n_v - 1}{n - 1}$$

# Betweenness Centrality

**Intuition:** a node is central if it appears on several shortest paths in the network.

Let  $G = (V, E)$  be an undirected graph (weighted or unweighted) with  $|V| = n$

Let  $\sigma_{s,t}$  be the number of shortest paths from node  $s$  to node  $t$

Let  $\sigma_{s,t}(v)$  be the number of shortest paths from node  $s$  to node  $t$  that pass through node  $v$ .

## Definition

Given a node  $v$ , the betweenness centrality  $b(v)$  of  $v$  is:

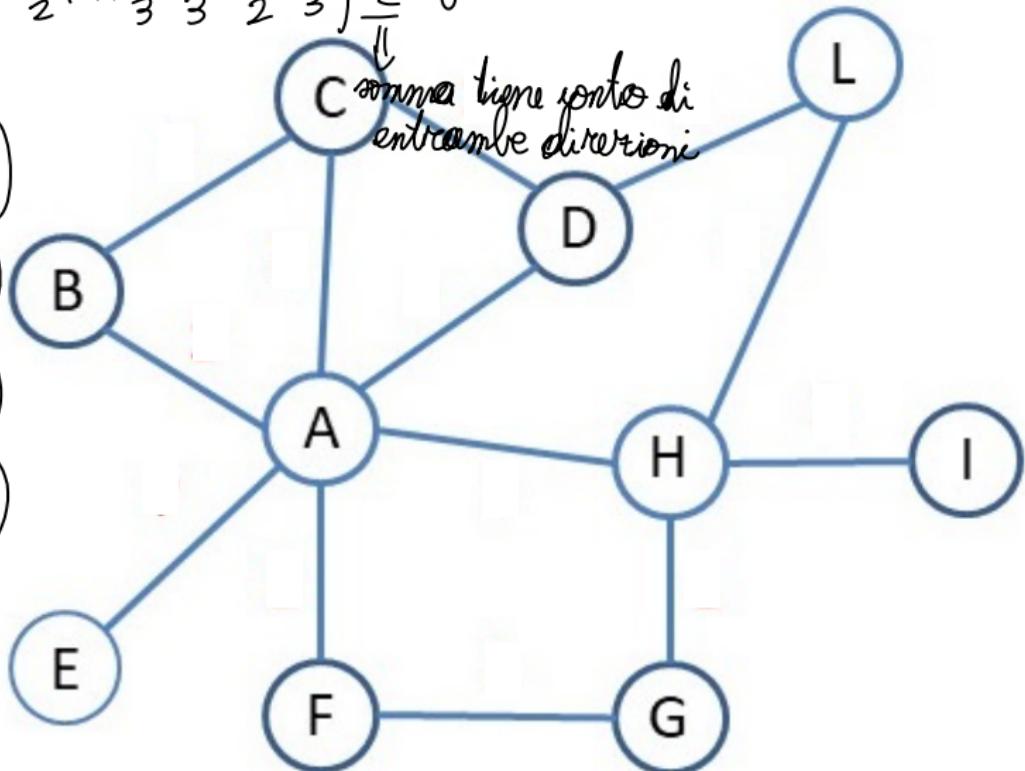
$$b(v) = \sum_{s,t \in V : s \neq v \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

## Betweenness Centrality: Example

$$b(D) = \left(0 + \frac{1}{2} + 1 + \frac{1}{3} + \frac{2}{3} + \frac{1}{2} + \frac{1}{3}\right) \cdot 2 = 6$$

$\downarrow$   
somma lunghezze di  
entrambe direzioni

$$\begin{aligned} \sigma_{A,H} &= 1 \\ \sigma_{A,H}(D) &= 0 \\ \sigma_{A,L} &= 2 \\ \sigma_{A,L}(D) &= 1 \\ \sigma_{C,L} &= 1 \\ \sigma_{C,L}(D) &= 1 \\ \sigma_{B,L} &= 3 \\ \sigma_{B,L}(D) &= 2 \\ \sigma_{E,L} &= 2 \\ \sigma_{E,L}(D) &= 1 \\ \sigma_{F,L} &= 3 \\ \sigma_{F,L}(D) &= 1 \end{aligned}$$



# Betweenness Centrality (continue)

**Note:** from the previous definition  $b(v)$  can be  $> 1$ .

## Definition

Given a node  $v$ , the **normalized betweenness centrality**  $b(v)$  of  $v$  is:

$$b(v) = \frac{1}{n(n-1)} \sum_{s,t \in V : s \neq v \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \Rightarrow b(v) \in [0, 1]$$

numero esatto di termini:  $n(n-1)(n-2)$   
con  $\frac{1}{n(n-1)}$ , + semplice e non cambia ordine di nodi secondo  $b(v)$

**Note:** given  $G$ ,  $n = |V|$  is fixed  $\Rightarrow$  the two definitions are equivalent.

# Computing the Betweenness Centrality

To compute  $b(v)$ , we need to compute for each  $s, t \in V$  ( $s \neq v \neq t$ ):

- $\underline{\sigma_{s,t}(v)}$
- $\underline{\sigma_{s,t}}$

**How?** [Brandes's algorithm [Brandes (2001). *A faster algorithm for betweenness centrality.* Journal of Mathematical Sociology.]

**Idea:** use a augmented BFS, which works also for weighted graphs

Complexity:  $\Rightarrow$  quando raggiungo nodo, CROSS EDGES mi danno percorsi alternativi  $\Rightarrow$  posso calcolare valori

- $G$  is unweighted:  $O(nm)$  alternativi  $\Rightarrow$  posso calcolare valori
- $G$  is weighted:  $O(n(m + n \log n))$  meccanismo singole o back-propagation

Unfeasable for large graphs!

# Approximating Betweenness Centralities

**Algorithm** ApproximateBetweennessCentralities( $G, k$ )

**Input:** weighted/unweighted graph  $G = (V, E)$  with  $|V| = n$ ,  
 $|E| = m; k \in \mathbb{N}$

**Output:** approximation  $\hat{b}(v)$  of  $b(v)$  for all  $v \in V$

$\hat{b}(v) \leftarrow 0$  for all  $v \in V$ ;

**for**  $i \leftarrow 1$  to  $k$  **do**

$(s, t) \leftarrow$  uniform vertex pair from  $V$  with  $s \neq t$ ;

$\mathcal{P}_{s,t} \leftarrow$  all shortest paths from  $s$  to  $t$ ;

$\pi_i \leftarrow$  shortest path chosen uniformly at random from  $\mathcal{P}_{s,t}$ ;

**forall**  $v \in \pi_i$  **do**

**if**  $v \neq s$  and  $v \neq t$  **then**  $\hat{b}(v) \leftarrow \hat{b}(v) + \frac{1}{k}$ ;

**return** values  $\hat{b}(v)$ ;

## Approximating Betweenness Centralities (continue)

### Note:

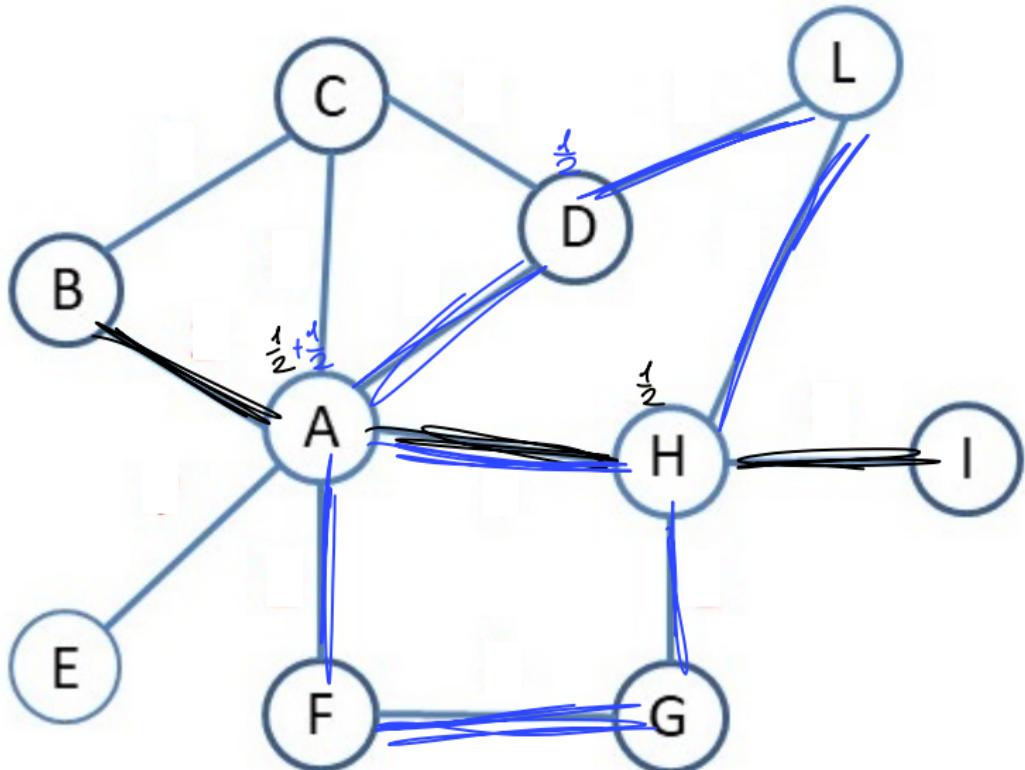
- $\mathcal{P}_{s,t}$  can be computed by a modified SSSP algorithm (BFS for unweighted graphs, Dijkstra's algorithm for weighted graphs) that keeps track of all shortest paths
- the SSSP algorithm starts from  $s$  and can stop as soon as all shortest paths to  $t$  are found

## Approximating the Betweenness Centrality: Example

$k=2$

$i=1$   
 $P_{2,t} = P_{B,1}$

$i=2$   
 $P_{2,t} = P_{F,L}$



# Approximating the Betweenness Centrality: Analysis

## Proposition

For any  $v \in V$ :

$$\mathbb{E} [\hat{b}(v)] = b(v)$$



## Approximating the Betweenness Centrality: Analysis (continue)

How many samples are needed to have  $\hat{b}(v)$  close to  $b(v)$ ?

*vertex diameter*  $VD(G)$  of  $G$ : maximum number of vertices among all shortest paths in  $G$ . ( $VD(G) - 1$  is equal to the *diameter* of  $G$  if  $G$  is unweighted.)

Riondato, Karnaropoulos (2016)

If  $k \geq \frac{2}{\varepsilon^2} (\lfloor \log_2(VD(G) - 2) \rfloor + \ln(1/\delta))$ , then

$$\Pr[\exists v \in V \text{ s.t. } |\hat{b}(v) - b(v)| > \varepsilon] < \delta.$$

**Proof:** based on the VC dimension of shortest paths.

Other improvements: Riondato and Upfal (2018), Borassi and Natale (2019), Pellegrina and Vandin (2023).

## Other Centrality Measures

There are several other centrality measures often implemented in graph analytics libraries:

- PageRank centrality
- harmonic centrality
- ...

Some useful libraries:

- <https://graph-tool.skewed.de/>
- <https://networkx.org/>
- <https://networkit.github.io/>
- <https://snap.stanford.edu/>

# Comparison on Zachary's Karate Club Data

See jupyter notebook.

