Algorithms for the analysis of interaction streams

Clémence Magnien

work in collaboration with Tiphaine Viard, Matthieu Latapy, Pierluigi Crescenzi, Andrea Marino, Fabien Tarissan, Frédéric Simard, Mehdi Naima, ...

ComplexNetworks(.fr)
LIP6 (CNRS, Sorbonne Université)

clemence.magnien@lip6.fr

June 5-6, 2023

Context – Complex Networks

computer science: internet, web, peer-to-peer, usages, etc.

social sciences: collaboration, friendship, exchanges, economics, etc.

biology: brain, genes, proteins, ecosystems, etc.

linguistics: synonyms, co-occurrences, etc.

transportation: roads, air, electricity, etc.

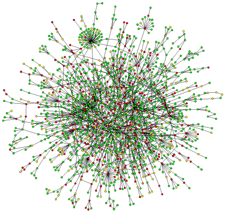
etc, etc

relation networks

Very different contexts

No formal definition

Common questions



Outline

Outline

Centrality in graphs

Goal

Capture node importance

Why?

- Interesting web page
- Understand network structure
- Network reliability
- ...

Three main notions

- Degree
- Closeness centrality
- Betweenness centrality

Centrality in graphs

Goal

Capture node importance

Why?

- Interesting web page
- Understand network structure
- Network reliability
- ...

Three main notions

- Degree
- Closeness centrality
- Betweenness centrality

Degree centrality

Degree of a node

its number of links

more links = more important

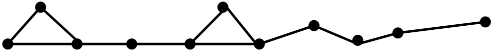
Context

- Friendships
- Marketing
- ...

Closeness centrality

Idea

closer to other nodes = more important



Formally

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

(other variants exists)

Closeness centrality – context

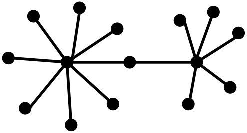
Why?

- Epidemics
- Transportation networks
- ...

Betweenness centrality

Idea

is on shortest paths between many nodes = more important



Formally

$$B(u) = \sum_{v,w \in V, v,w \neq u} \frac{\sigma(v,w,u)}{\sigma(v,w)}$$

- $\sigma(v, w)$: # of shortest paths between v and w
- $\sigma(v, w, u)$: # of shortest paths between v and w involving u

Betweenness centrality – context

Why?

- Identifiy communities/network robustness
- Influence/power in collaboration networks
- ...

How do we compute those?

Base brick

Breadth-First Search (BFS) to compute the distance

- from one source node
- to all other nodes

Complexity $O(m)$

- n: number of nodes
- m: number of links

Algorithm

noend 1 Breadth-first search algorithm

procedure BFS

  Input: Graph G, source node s.

  Q $\leftarrow$ Empty queue (FIFO)

  Add s to Q

  $d \leftarrow$ array initialized to ∞

  while Q is not empty **do**

    Remove u from Q

    for v neighbour of u **do**

      if $d[v] = \infty$ **then**

        $d[v] \leftarrow d[u] + 1$

        Add v to Q

Closeness computation

For one node

one BFS : $O(m)$

For all nodes

one BFS per node : $O(nm)$

Betweenness computation

Naive version

For one node u:

- one BFS to compute distances from u to all other vertices
- one BFS from each other node v to mark nodes on shortest paths between u and v

$\longrightarrow O(nm)$ for one node

Brandes' algorithm

$O(nm)$ for all vertices

Betweenness computation

Naive version

For one node u:

- one BFS to compute distances from u to all other vertices
- one BFS from each other node v to mark nodes on shortest paths between u and v

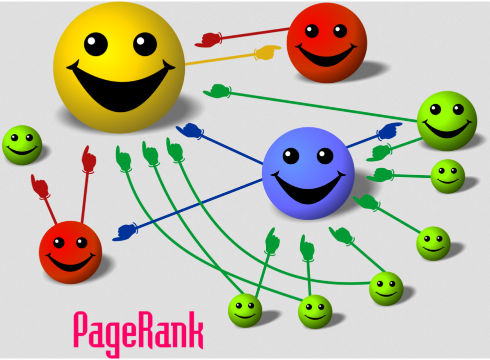
$\longrightarrow O(nm)$ for one node

Brandes' algorithm

$O(nm)$ for all vertices

What is PageRank?

- PageRank is an algorithm used by Google Search to rank websites in their search engine results.
- It counts the number and quality of links to a page.
- The underlying assumption is that an important website is likely to receive links from other important websites.



Eigenvector

Definition: given an n by n matrix M a vector A is an eigenvector of M if it has at least one non-zero value and if there exists a scalar $\lambda \in \mathbb{R}$ such that

$$M \times A = \lambda \times A$$

Definition: a top/dominant eigenvector is an eigenvector A such that its associated eigenvalue λ has maximum absolute value.

PageRank Computation

Definition: transition matrix T:

- n by n matrix
- for each directed edge (u, v), $T_{vu} = \frac{1}{d^{out}(u)}$

Definition: the PageRank vector P is given by:

$$P = (1 - \alpha) \times T \times P + \alpha \times I$$

I: vector with entries $= \frac{1}{n}$.

P is eigenvector of $\left[\frac{\alpha}{n}\right]_{n \times n} + (1 - \alpha)T$

Other spectral centrality measures

- PageRank
- Eigenvector centrality
- HITS
- Katz centrality

Based on walks in the graph

Centrality in practice

Why different notions?

- Different notions of importance

No ground truth! (in general)

Outline

1. Centrality in (static) graphs
2. Link streams formalism
   - Paths in link streams
   - Some existing definitions of temporal centrality measures
   - Algorithmic ideas
3. Finding top nodes for global closeness
   - Approach
   - Results
4. Closeness evolution
   - Observations
5. Other topics

Taking time into account

I won't talk about static graphs.

As time passes

- new web pages are created and links change
- we meet new people
- we talk to different persons
- we buy things
- ...

How to take this into account?

Relations vs. Interactions

relations (like friendship)

interactions
(like face-to-face contacts)

evolution of relations
(like new friends)

Many interesting links between the two.

Interactions important for: Recommender systems, epidemiology, anomaly detection, message passing, . . .

Relations vs. Interactions

relations (like friendship)
$\longrightarrow$ graph/networks

evolution of relations
(like new friends)
$\longrightarrow$ dynamic graphs/networks

interactions
(like face-to-face contacts)
$\longrightarrow$?

Many interesting links between the two.

Interactions important for: Recommender systems, epidemiology, anomaly detection, message passing, ...

Characteristics: speed of path vs speed of link evolution?

Framework for describing interactions?

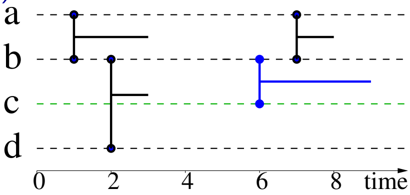
definition of link streams

Link stream $S = (T, V, E)$

T: time interval, V: node set

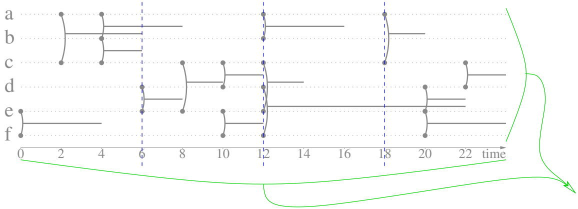
$E \subseteq T \times V \otimes V$

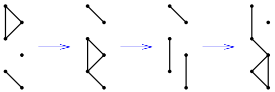
$(t, uv) \in E \;\Leftrightarrow\;$ u **and** v **interact at time** t



$$E = [1,3] \times ab \cup [8,8] \times ab \cup [2,3] \times bd \cup [6,9] \times bc$$

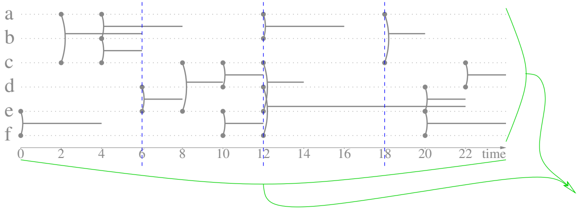
Related work: **Describe structure and dynamics?**

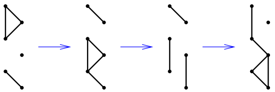




time slices
→ graph sequence

Related work: **Describe structure and dynamics?**

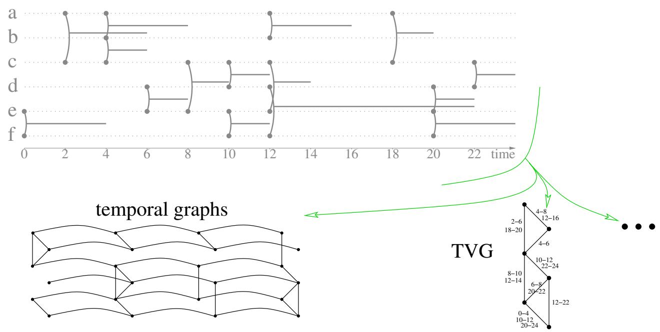




time slices
→ graph sequence

information loss
what slices?

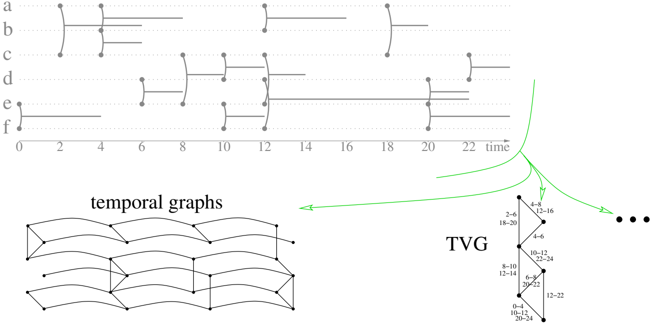
Related work: **Describe structure and dynamics**



lossless but graph-oriented

+ other more specific contributions

Related work: **Describe structure and dynamics**



lossless but graph-oriented

+ other more specific contributions

what we propose

deal with the stream directly

Approach

very careful definition of the most basic concepts

$\hookrightarrow$ **building blocks for higher-level concepts**

+ ensure consistency with graph theory

+ ensure classical relations are preserved

Note on different models

Link streams, time varying graphs (TVG), temporal networks, ...

Formally

Same power of representation

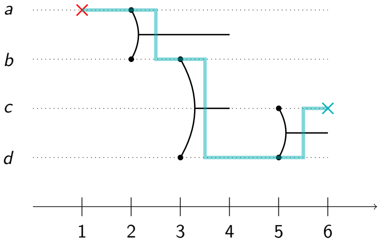
Intuition

Two possible cases

- evolving graph
- sequence of temporal links

Outline

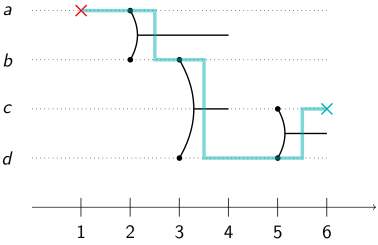
Paths in link stream



Path from (α, u) to (ω, v): $(t_0, u_0, v_0), (t_1, u_1, v_1), \ldots (t_k, u_k, v_k)$:

- $u_0 = u, v_k = v$
- $t_0 \geq \alpha, t_1 \leq \omega$
- $(t_1, u_i v_i) \in E$
- $t_i \leq t_{i+1}$

Paths in link stream



Path from (α, u) to (ω, v): $(t_0, u_0, v_0), (t_1, u_1, v_1), \ldots (t_k, u_k, v_k)$:

Path characteristics

- k: path length
- t_k: arrival time
- $t_k - t_0$: path duration

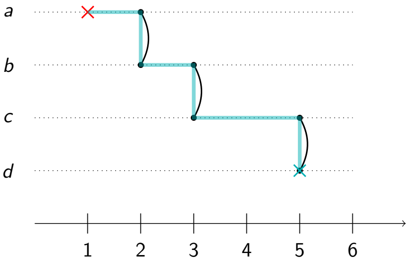
Some remarks

I will consider in general

- instantaneous links only
- no two links at the same time

Everything can be extended to the general case

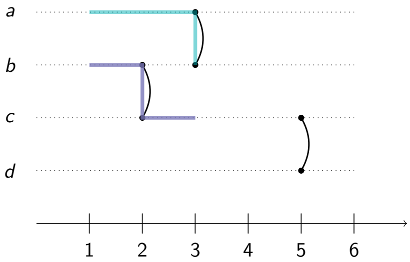
Path examples



Paths are not symmetrical

No path from d to a

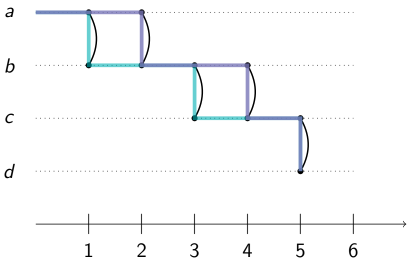
Path examples



Paths are not transitive

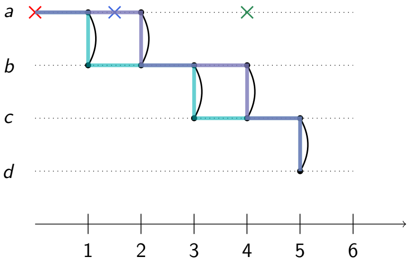
No path from a to c

Path examples



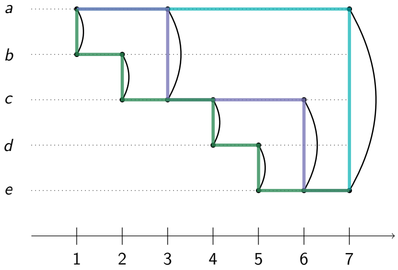
Different paths exist between two nodes

Path examples



Path existence depends on starting time

Shortest paths

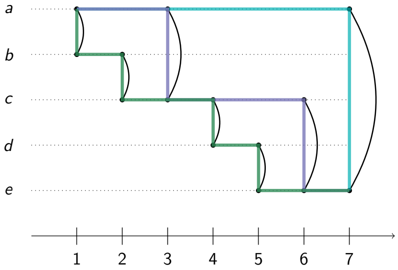


Different paths from $(1, a)$ to $(7, e)$

Length 4 – Length 2 – Length 1: shortest path

We are not always willing to wait for the shortest path!

Shortest paths

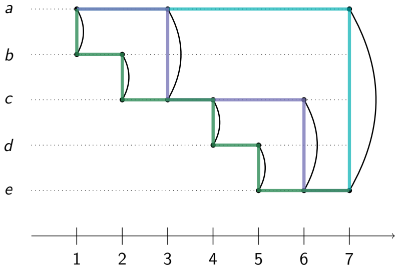


Different paths from $(1, a)$ to $(7, e)$

Length 4 – Length 2 – Length 1: shortest path

We are not always willing to wait for the shortest path!

Shortest paths

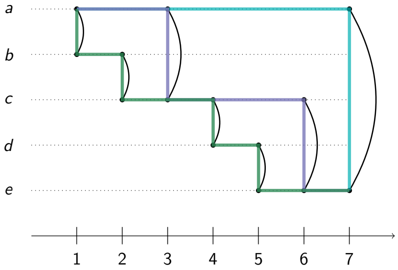


Different paths from $(1, a)$ to $(7, e)$

Length 4 – Length 2 – Length 1: shortest path

We are not always willing to wait for the shortest path!

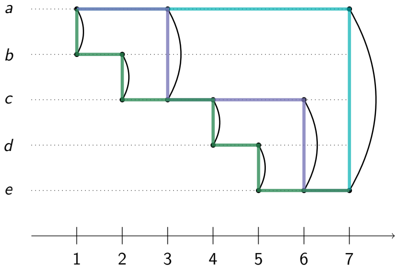
Foremost path



Different paths from $(1, a)$ to $(7, e)$

Arrival time 6 – Arrival time 7 – Arrival time 5: foremost path

Fastest path



Different paths from $(1, a)$ to $(7, e)$

Duration 4 – Duration 3 – Duration 1: fastest path

No best definition

Relevance for different questions

Foremost paths: quick information/virus spread
Shortest: more robust paths?
Fastest: identify key instants?

We will focus on foremost path

Foremost path

Time to reach

$T_t(u, v) = t_a - t$, with t_a:

- earliest arrival time of a path from u starting at time t to v

Foremost path

Path realizing the time to reach

Outline

From path to centrality

How do we go from temporal paths to centrality definitions?
How to take into account path starting and ending times?

Incomplete review of exising definitions

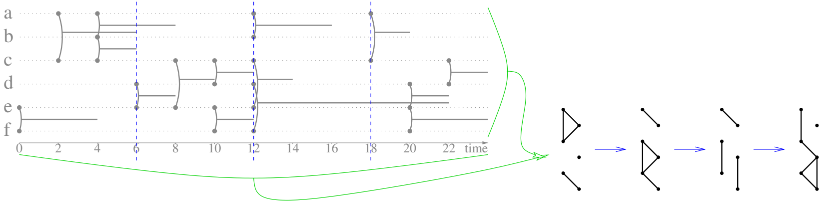
From path to centrality

How do we go from temporal paths to centrality definitions?
How to take into account path starting and ending times?

Incomplete review of exising definitions

First approach: snapshots

[Uddin et al, 2013]



Advantages and drawbacks

Works for any static centrality metrics
Takes data evolution into account
Information loss
Which snapshot size?
Does not consider all paths

Temporal paths, first approach

[Nicosia et al, 2013]

- choose a notion of optimal path/centrality
- compute centrality/for each node
- for paths starting at time 0

Advantages and drawbacks

uses temporal paths

considers only paths that occur early in the dataset

Temporal paths, first approach

[Nicosia et al, 2013]

- choose a notion of optimal path/centrality
- compute centrality/for each node
- for paths starting at time 0

Advantages and drawbacks

uses temporal paths
considers only paths that occur early in the dataset

Temporal paths, second approach

Compute centrality values
For all starting times

Aggregate into a single value per node

Temporal paths, second approach

Compute centrality values
For all starting times

Aggregate into a single value per node

Spectral centrality measures

Possible to extend spectral measures.

Need adjacency/transition matrices!

- combine adjacency matrices at different time steps
- block matrix: rows/columns correspond to temporal vertices

Temporal closeness

Time to reach

$T_t(u,v) = t_a - t$, with t_a:

- earliest arrival time of a path from u starting at time t to v

Temporal closeness at time t

$$C_t(u) = \sum_{v \neq u} \frac{1}{T_t(u,v)+1}$$

Why $+1$?

Temporal closeness

Time to reach

$T_t(u, v) = t_a - t$, with t_a:

- earliest arrival time of a path from u starting at time t to v

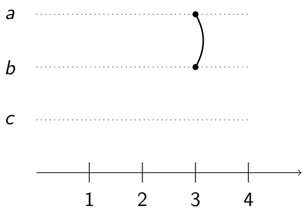
Temporal closeness at time t

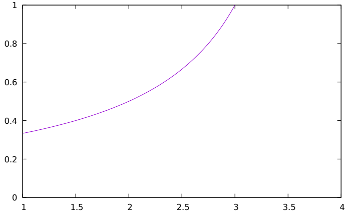
$$C_t(u) = \sum_{v \neq u} \frac{1}{T_t(u, v) + 1}$$

Why $+1$?

Example

Closeness of a

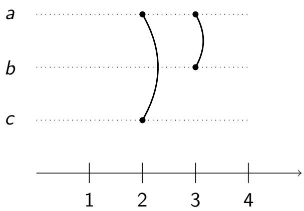


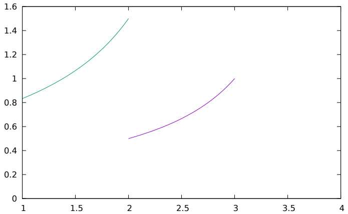


$$\frac{1}{(4-x)}$$

Example

Closeness of *a*





Aggregate in a single value

$C_t(u)$: one value for each $t \longrightarrow$ what is an important node?
Several possibilities:

- maximum value
- average value
- time during which it has a high rank
- ...

$$C(u) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} C_t(u) dt$$

Aggregate in a single value

$C_t(u)$: one value for each $t \longrightarrow$ what is an important node?
Several possibilities:

- maximum value
- average value
- time during which it has a high rank
- ...

$$C(u) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} C_t(u)dt$$

Outline

1. Centrality in (static) graphs
2. Link streams formalism
   - Paths in link streams
   - Some existing definitions of temporal centrality measures
   - Algorithmic ideas
3. Finding top nodes for global closeness
   - Approach
   - Results
4. Closeness evolution
   - Observations
5. Other topics

Two basic algorithmic ideas – 1: going forward in time

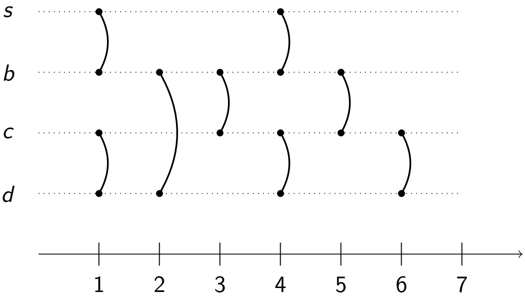
Compute earliest arrival times from:

- a single source node s
- to all other nodes
- for all starting times

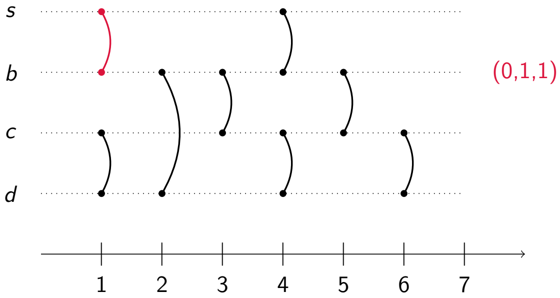
By considering each link once

Complexity $O(m)$

Two basic algorithmic ideas – 1: going forward in time



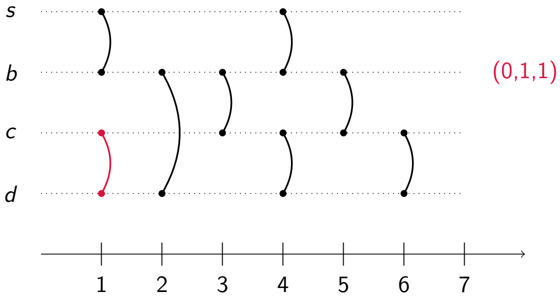
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

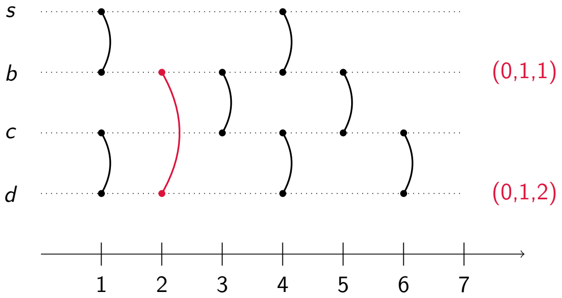
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

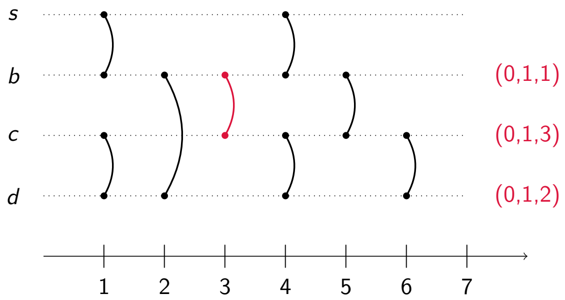
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

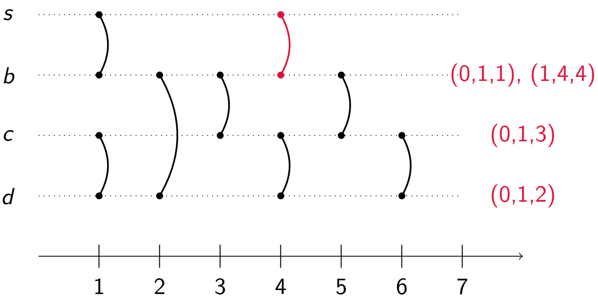
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

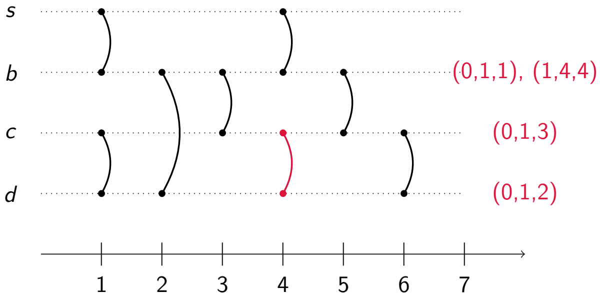
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

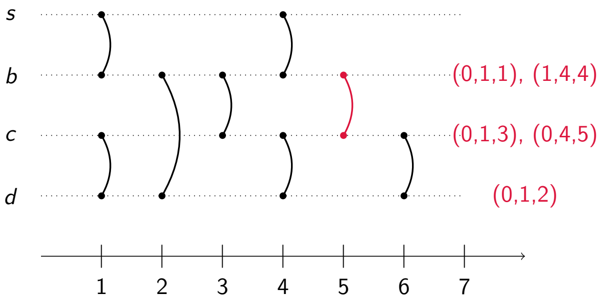
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

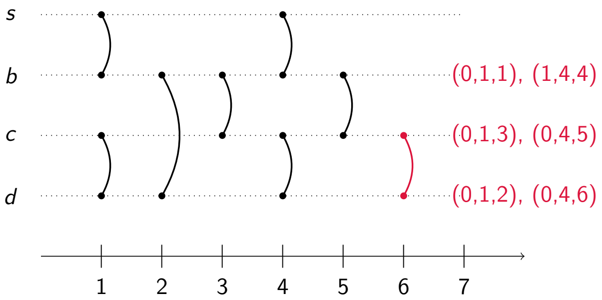
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

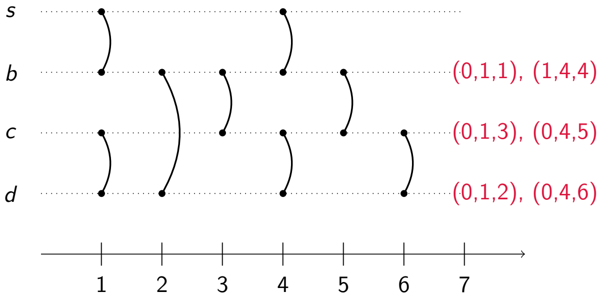
Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

Two basic algorithmic ideas – 1: going forward in time



Triple (t_1, t_2, ta)

For t, $t_1 < t \leq t_2$, earliest arrival time is ta

Algorithm (assumes edges are directed)

procedure EARLIEST ARRIVAL TIMES

  for $x \leftarrow 1$ to $|V|$ **do** $\tau[x] = [(t_\alpha - 2, t_\alpha - 1, \infty)]$

  while there are other edges to be read **do**

    let $e \leftarrow (x, y, t)$ be the next edge

    $\tau[s] \leftarrow (t-1, t, t-1)$

    $(l_x, r_x, a_x) \leftarrow$ last elem. of $\tau[x]$

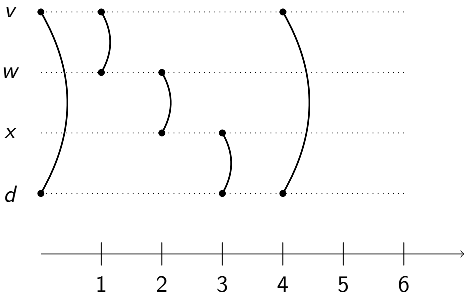
    $(l_y, r_y, a_y) \leftarrow$ last elem. of $\tau[y]$

    if $r_x > r_y$ **then**

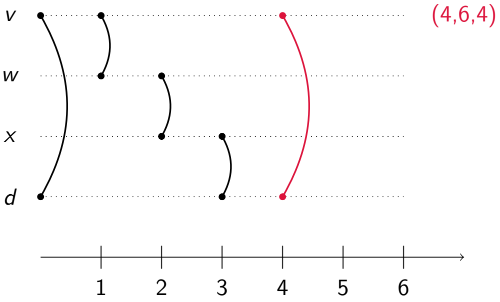
      append (r_y, r_x, t) to $\tau[y]$

  Return τ

Two basic algorithmic ideas – 2: going backwards



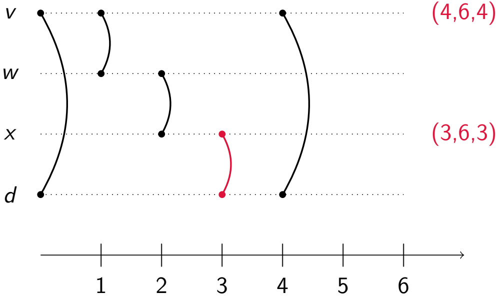
Two basic algorithmic ideas – 2: going backwards



Triple (t_1, t_2, t_s),

For arrival time t, $t_1 < t \leq t_2$, latest strating time is t_s

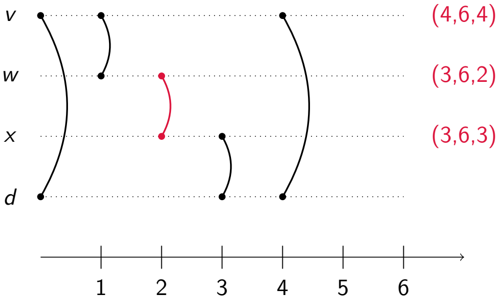
Two basic algorithmic ideas – 2: going backwards



Triple (t_1, t_2, t_s),

For arrival time t, $t_1 < t \leq t_2$, latest strating time is t_s

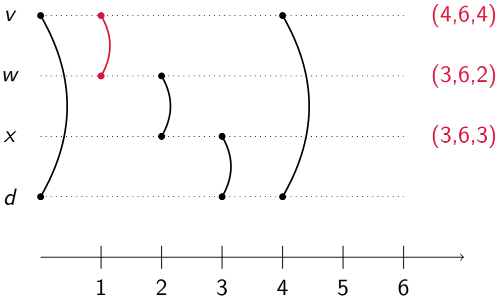
Two basic algorithmic ideas – 2: going backwards



Triple (t_1, t_2, t_s),

For arrival time t, $t_1 < t \leq t_2$, latest strating time is t_s

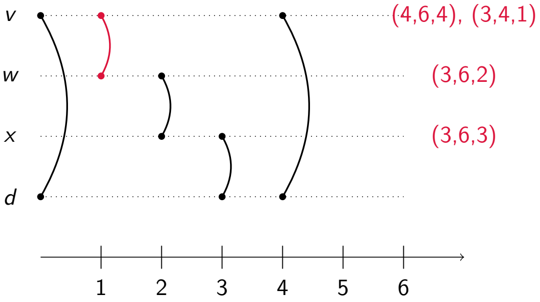
Two basic algorithmic ideas – 2: going backwards



Triple (t_1, t_2, t_s),

For arrival time t, $t_1 < t \leq t_2$, latest strating time is t_s

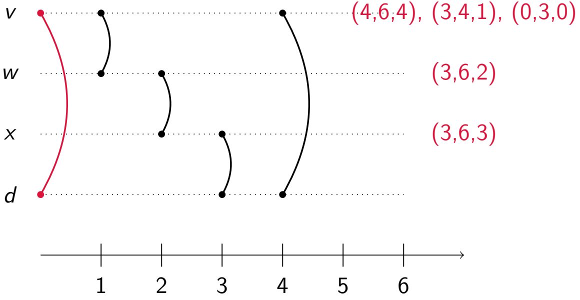
Two basic algorithmic ideas – 2: going backwards



Triple (t_1, t_2, t_s),

For arrival time t, $t_1 < t \leq t_2$, latest strating time is t_s

Two basic algorithmic ideas – 2: going backwards



Triple $(t_1, t_2, t_s), (t_3, t_4, t_{s2})$

For arrival time t, $t_1 < t \leq t_2$, latest strating time is t_s
For starting time t', $t_{s2} < t' \leq t_s 2$, earliest arrival time is t_1

Backward Algorithm (assumes edges are directed)

procedure LATEST STARTING TIMES
    for $x \leftarrow 1$ to $|V|$ **do** $\tau[x] = [(\omega+1, \omega+2, \infty)]$
    while there are other edges to be read **do**
        let $e \leftarrow (x, y, t)$ be the next edge
        $\tau[d] \leftarrow (t, t+1, t+1)$
        $(l_x, r_x, s_x) \leftarrow$ last elem. of $\tau[x]$
        $(l_y, r_y, s_y) \leftarrow$ last elem. of $\tau[y]$
        if $l_y < l_x$ **then**
            append (l_y, l_x, t) to $\tau[x]$
    Return τ

Data Structure

Link Stream storage:

list of temporal links, ordered by (increasing or decreasing) time

No need to store stream in memory

Outline

Outline

1. Centrality in (static) graphs
2. Link streams formalism
   - Paths in link streams
   - Some existing definitions of temporal centrality measures
   - Algorithmic ideas
3. Finding top nodes for global closeness
   - Approach
   - Results
4. Closeness evolution
   - Observations
5. Other topics

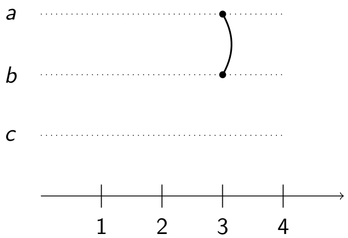
Global closeness defition

Overall closeness over time

$$C(u) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} C_t(u) dt$$

α, ω: fist and last time in the stream

Example



$$C(a) = \int_1^3 \frac{1}{4-x}dx = \log 3$$

TODO Histoire de la gestion du dernier triplet

Algorithm - Closeness (assumes edges are directed)

procedure CLOSENESS
    for $x \leftarrow 1$ to $|V|$ **do** $\tau[x] = (t_\alpha - 2, t_\alpha - 1, \infty)$
    $C \leftarrow 0$
    for edges (x, y, t), t increasing **do**
        $\tau[s] \leftarrow (t-1, t, t-1)$
        $(l_x, r_x, a_x) \leftarrow \tau[x]$
        $(l_y, r_y, a_y) \leftarrow \tau[y]$
        if $r_x > r_y$ **then**
            $C \leftarrow C + \ln \frac{a_y - \max(\alpha, l_y) + 1}{a_y - \max(\alpha, r_y) + 1}$
            $\tau[y] \leftarrow (r_y, r_x, t)$
    for $x \leftarrow 1$ to $|V|$ **do** $(l_x, r_x, a_x) \leftarrow \tau[x]$
        $C \leftarrow C + \log \frac{a_x - \max(\alpha, l_x) + 1}{a_x - \max(\alpha, r_x) + 1}$
    Return $\frac{C}{(\omega - \alpha)}$

Complexity

Above algorithm

$O(m)$

Must be run for each node

Global complexity $O(nm)$

Untractable in many cases

Other approach – sampling

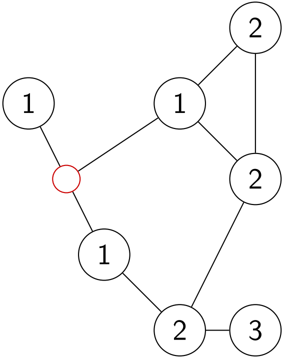
General idea: sample paths randomly

- Which paths?
- Which complexity?

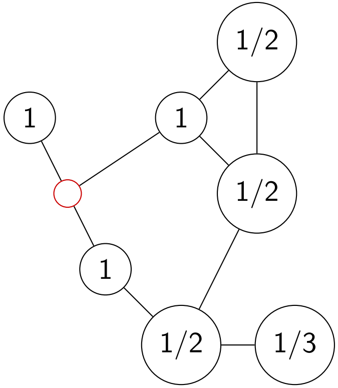
Back to graphs

1 BFS: Distances from one node to all others

$\longrightarrow$ this node's closeness



Compute the contribution of one node to the others' closeness



Nothing changed?

We must compute the contribution

- from all nodes
- to all nodes

Global complexity is still $O(nm)$

Must we?

Nothing changed?

We must compute the contribution

- from all nodes
- to all nodes

Global complexity is still $O(nm)$

Must we?

Sampling

[Cohen et al 2014]

Reduce computation time

compute the contribution of some nodes

Given u, v

$$C(u,v) = \frac{1}{d(u,v)}$$

$$C(v) = \sum_{u \neq v} C(u,v)$$

Given $X \subseteq V$

$$\frac{n}{|X|} \sum_{u \in X} C(u,v)$$

Back to linkstreams

Closeness of u

$$C(u) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} C_t(u) dt$$

$$C(u) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} \sum_{d \neq u} \frac{1}{T_t(u,d) + 1}$$

Contribution of d to the closeness of u

$$C(u,d) = \frac{1}{\omega - \alpha} \int_{\alpha}^{\omega} \frac{1}{T_t(u,d) + 1} dt$$

Estimation

Random node sample $X = \{x_1, \ldots, x_h\}$

$$C^X(u) = \frac{n}{h} \sum_{i=1}^{h} C(u, x_i)$$

$h << n$

Contribution – algorithm

procedure CONTRIBUTION
    for $x \leftarrow 1$ to $|V|$ **do** $\tau[x] = (\omega+1, \omega+2, \infty)$; $S[x] \leftarrow \infty$
    $C \leftarrow 0$
    for edges (x, y, t), t decreasing **do**
        $\tau[d] \leftarrow (t, t+1, t+1)$; $S[d] \leftarrow t$
        $(l_x, r_x, s_x) \leftarrow \tau[x]$
        $(l_y, r_y, s_y) \leftarrow \tau[y]$
        if $l_y < l_x$ **then**
            $C \leftarrow C + \ln \frac{\min(\omega, l_x) - t + 1}{\min(\omega, l_x) - s_x + 1}$
            $\tau[x] \leftarrow (l_y, l_x, t)$; $S[x] \leftarrow s_x$
    for $x \leftarrow 1$ to $|V|$ **do** $(l_x, r_x, s_x) \leftarrow \tau[x]$
        $C \leftarrow C + \log \frac{\min(\omega, l_x) - s_x + 1}{\min(\omega, l_x) - S[x] + 1} + \log \frac{l_x - \alpha + 1}{l_x - s_x + 1}$
    Return $\frac{C}{(\omega - \alpha)}$

Theoretical result

Theorem

If $h = |X| = \Theta(\log n/\epsilon^2)$, then $\forall u$,
$|C(u) - C^X(u)| \leq \epsilon$ with high probability

Choice of h to get a good estimate?

$\Theta(\log n/\epsilon^2)$ in practice?
Choice of ϵ?

Our approach

1. Choose a sample of h nodes
2. Compute the approximated closeness of all nodes
3. Sort by decreasing approximated closeness and select the $K > k$ top nodes
4. Compute their exact closeness
5. Rank them and select the k top nodes

In practice $h = K = 1024$ works for finding the top 100-nodes

Complexity

$$O(2048 \times m)$$

Outline

Datasets

Link streams from different contexts

- Internet topology
- Movie actors
- Public transportation
- Facebook
- Twitter
- Linux mailing list

$$20,000 \leq n \leq 3.5 \cdot 10^6$$
$$80,000 \leq m \leq 16 \cdot 10^6$$

Methodology

Compute exact closeness values

For all datasets except Twitter

to evaluate the quality of results

Run approximate method with $h = 32, 64, 128, 256, 512, 1024$

50 times each

Running times

Name	*Nodes*	*Edges*	EXACT	APX-1024
FANT	34 464	87 331	1 815	33
MELB	19 493	1 098 227	6 258	380
TOPO	34 761	154 842	1 649	47
FBWA	46 952	876 993	12 184	264
COME	162 303	666 568	29 601	203
LINU	63 400	1 096 400	19 313	317
ALL	527 535	3 152 994	484 906	941
TWIT	3 511 241	16 438 790	*97 553 304	28 449

*estimated $\sim$ 3 years

Accuracy

Compare approximate result to exact

Two ways to evaluate accuracy:

- absolute error
- relative error

Advantages and drawbacks

Absolute error

Mean absolute error

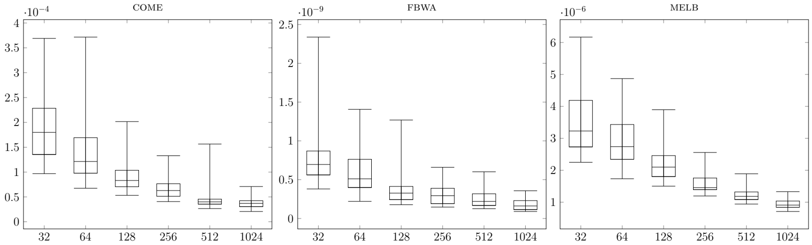
$$\sum_{v} |C^X(u) - C(u)|/n$$

One value per experiments $\longrightarrow$ 50 values for each value of h

We plot

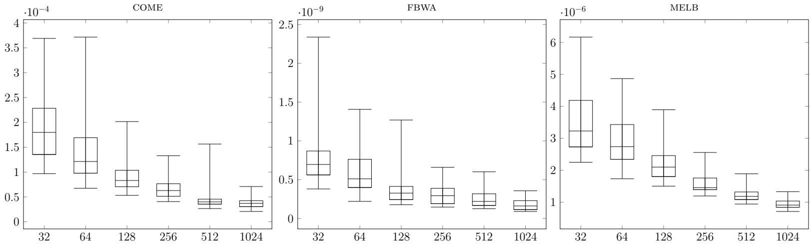
- median
- 25% and 75% quartiles
- minimum and maximum

Absolute error, examples



Average closeness: $6.1 \cdot 10^{-4}, 5.4 \cdot 10^{-9}, 2.4 \cdot 10^{-5}$
Average error and variability decrease with h.

Absolute error, examples



Average closeness: $6.1 \cdot 10^{-4}, 5.4 \cdot 10^{-9}, 2.4 \cdot 10^{-5}$
Average error and variability decrease with h.

Relative error

For a node u and sample X

$$|C^X(u) - C(u)|/C(u)$$

Rank nodes according to closeness

small rank = high closeness

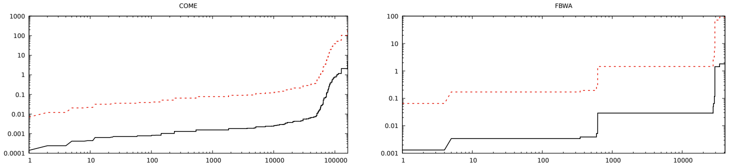
We study relative error with respect to rank

For each rank i

we are interested in the maximum error observed for all ranks $\leq i$.

Relative error, examples

Maximum and average value over 50 experiments, $h = 1024$



- Relative error can be quite high
- Is low for top nodes

Note discussion sur pourquoi les nœuds de faible closeness sont mal estimés, sur le fait qu'une erreur relative > 1 veut dire une closeness sur estimée, n'arrive que pour le max

Quality of ranking

How do we compare rankings?

Kendall tau-coefficient

$$C(u_1), \dots, C(u_i), \dots, C(u_j), \dots, C(u_n)$$
$$C^X(u_1), \dots, C^X(u_i), \dots, C^X(u_j), \dots, C^X(u_n)$$

nodes u_i and u_j are concordant if

$$C(u_i) < C(u_j) and C^X(u_i) < C^X(u_j)$$

$$\tau = \frac{\#\text{concordant pairs} - \#\text{discordant pairs}}{\#\text{pairs}}$$

We use a variant that puts more weight on top nodes

Quality of ranking

How do we compare rankings?

Kendall tau-coefficient

$$C(u_1), \ldots, C(u_i), \ldots, C(u_j), \ldots, C(u_n)$$
$$C^X(u_1), \ldots, C^X(u_i), \ldots, C^X(u_j), \ldots, C^X(u_n)$$

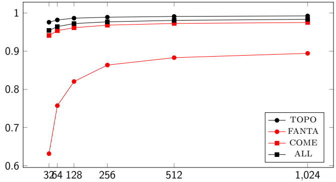
nodes u_i and u_j are concordant if

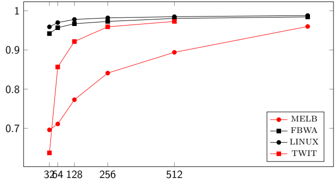
$$C(u_i) < C(u_j) and C^X(u_i) < C^X(u_j)$$

$$\tau = \frac{\#\text{concordant pairs} - \#\text{discordant pairs}}{\#\text{pairs}}$$

We use a variant that puts more weight on top nodes

Quality of ranking, results





Twitter: used ranking for $h = 1024$ as reference
Larger than 0.9 for $h = 1024$ in all cases

Quality for top-k

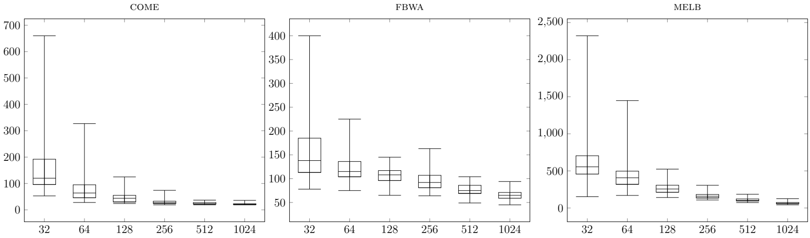
We care only about the first nodes in the order

How to evaluate the quality?

$\gamma(k)$: maximum approximated rank among the top-k exact nodes
Note : Example au tableau

Quality in practice

Results for top-20



In practice,

- use $h = 1024$
- compute approximate ranking,
- select the top h nodes and compute their exact closeness

always works for finding the top 100 nodes

Open questions

- Why do some datasets behave worse than others?
- what if edge have travelling time?

Outline

How does closeness evolve with time?

Is it enough to study the global closeness?

Are important nodes always important?
Are they important most of the time?
Do unimportant nodes become important?
...

Need to compute the closeness

- of all nodes
- for all starting times

$O(nmT)$????

Here

We assume networks not too large

We can store a $n \times n$ matrix

Foremost paths and time to reach

It is possible to compute the time to reach:

- for all pairs of nodes
- for all starting times

in a single reading of the input
$\mathcal{O}(n^2)$ space

[Kossinets, Kleinberg, Watts, 2008]

How would you do it?

Foremost paths and time to reach

It is possible to compute the time to reach:

- for all pairs of nodes
- for all starting times

in a single reading of the input
$\mathcal{O}(n^2)$ space

[Kossinets, Kleinberg, Watts, 2008]

How would you do it?

Algorithm - Principle

Perform the backward in time algorithm

for all destination nodes at once!

In memory data

$n \times n$ evolving matrix
$[i][j]$: earliest arrival time from i to j

Why not the forward algorithm?

Algorithm - Principle

Perform the backward in time algorithm

for all destination nodes at once!

In memory data

$n \times n$ evolving matrix
$[i][j]$: earliest arrival time from i to j

Why not the forward algorithm?

Algorithm - Principle

Perform the backward in time algorithm

for all destination nodes at once!

In memory data

$n \times n$ evolving matrix
$[i][j]$: earliest arrival time from i to j

Why not the forward algorithm?

Algorithm

In memory data

$n \times n$ evolving matrix
$[i][j]$: earliest arrival time from i to j

Idea

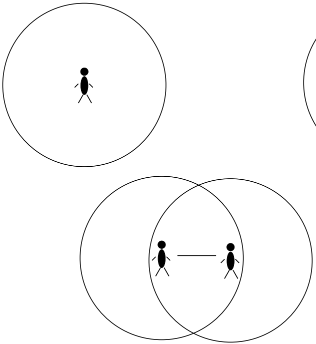
Deal with time from end to beginning

- Suppose we computed all t.t.r. for starting times $> t$
- Link (u, v) at time t
  - u et v can reach each other at time t
  - for each node $x \neq u, v$
    - dux: t.t.r. from u to x, dvx: t.t.r. from v to x
    - if $T_t(u, x) < d_t(T, x)$ then u should go through v to reach x earlier
    - and conversely

Datasets (1) – Rollernet experiment

[Tournoux *et al*, 2009]

Proximity networks between individuals
Captured *via* sensors



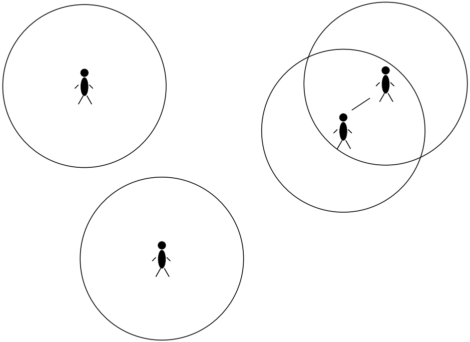
Rollerblade tour

- 62 nodes
- $\sim$ 3h

Datasets (1) – Rollernet experiment

[Tournoux *et al*, 2009]

Proximity networks between individuals
Captured *via* sensors



Rollerblade tour

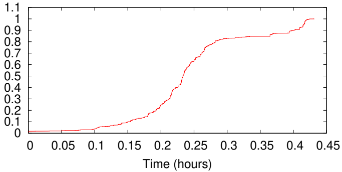
- 62 nodes
- $\sim$ 3h

Datasets (2) – Enron email

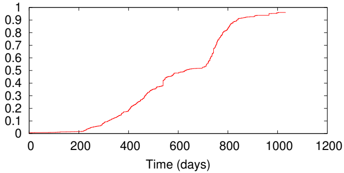
All emails between Enron employees

- 151 employees
- more than three years

Fraction of reachable pairs



Rollernet



Enron

Compare to linkstream duration

- Rollernet: 3 hours
- Enron : 3 years

Very different dynamics

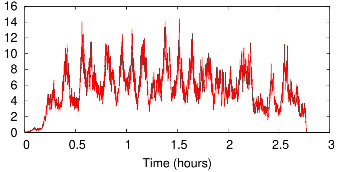
Outline

Temporal efficiency

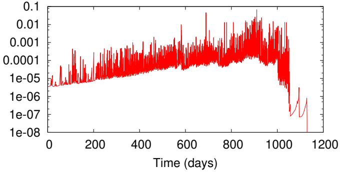
Efficiency: average closeness

$$E_t(G) = \frac{1}{n} \sum_{v \in V} C_t(v)$$

Temporal efficiency - observations



Rollernet

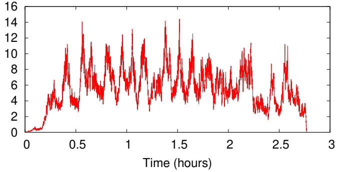


Enron

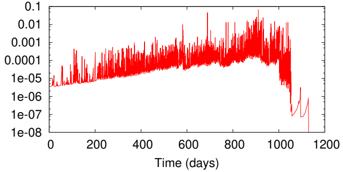
- Fluctuates a lot
- Global increase over time for Enron

What about individual nodes?

Temporal efficiency - observations



Rollernet

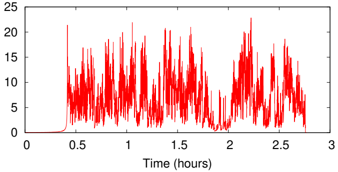


Enron

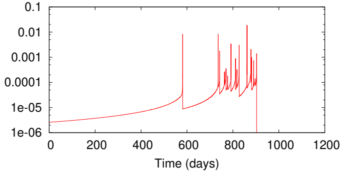
- Fluctuates a lot
- Global increase over time for Enron

What about individual nodes?

Temporal closeness of random nodes



Rollernet

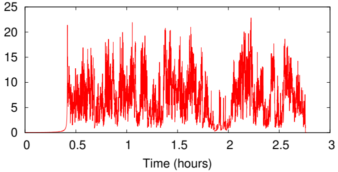


Enron

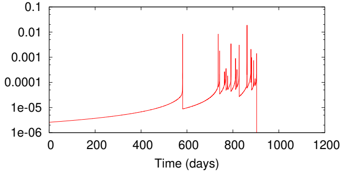
Problems

- Fluctuates a lot
- Difficult to interpret
- Comparison with other nodes ?

Temporal closeness of random nodes



Rollernet



Enron

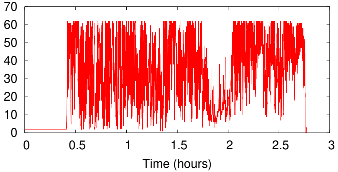
Problems

- Fluctuates a lot
- Difficult to interpret
- Comparison with other nodes ?

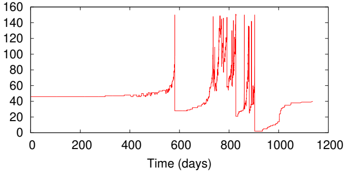
Global comparison

- Compute closeness for all nodes at all times
- Compute rank of node at each time

Rank evolution



Rollernet



Enron

small rank = small closeness

Observations

- Artefact in Enron: arbitrary rank in case of ties
- correlation between closeness and rank, but not perfect
- Very different behaviors

Global statistics

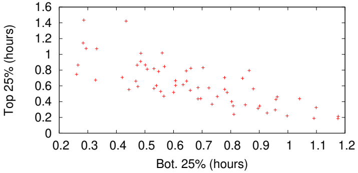
Idea

An important node will often have a high rank

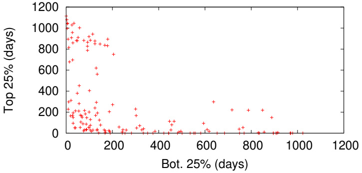
For each node

Compute time spent with rank in top/bottom 25%

Global statistics



Rollernet



Enron

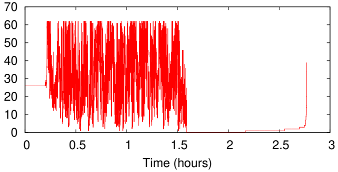
Note that $x + y <$ total duration

Observations

- Differences between nodes
- Rollernet: no node important or unimportant for more than half dataset duration
- Enron: some overall important or unimportant nodes

Extremes - Rollernet

Rank of nodes with longest time in top 25% ranks - bottom 25% ranks

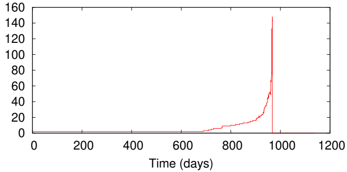


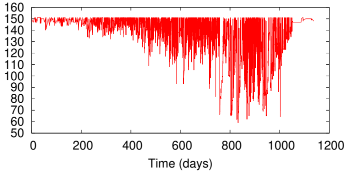
Observations

- Most important (numerically) $\neq$ globally important
- No notion of global importance?

Extremes - Enron

Rank of nodes with longest time in top 25% ranks - bottom 25% ranks





Observations

- Bottom node has only two links, yet reaches high rank
- Top node has consistently high rank for half the dataset
- Globally important $\neq$ always important

Delta centrality

Impact on a given node on efficiency

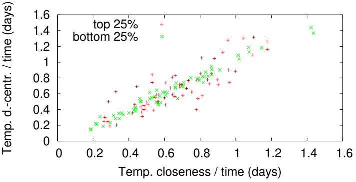
$$\Delta(u) = \frac{E_t(G) - E_t(G \backslash u)}{E_t(G)}$$

$\longrightarrow$ for each node, one value per time instant

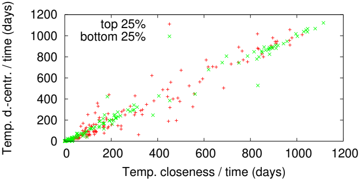
To evaluate the global impact of a node

- Rank for all times
- Compute time spent in top/bot 25%

Global statistics



Rollernet



Enron

Observations

- High correlation between high/low closeness and high/low Delta-centrality
- Some nodes have a relatively higher delta-centrality than closeness

$\longrightarrow$ high impact on paths

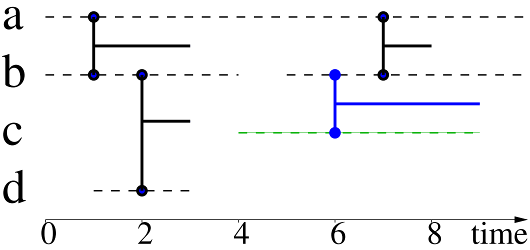
Conclusion

Results

- Importance does vary with time
- Notion of global importance not always meaningful
- Different observations for different datasets

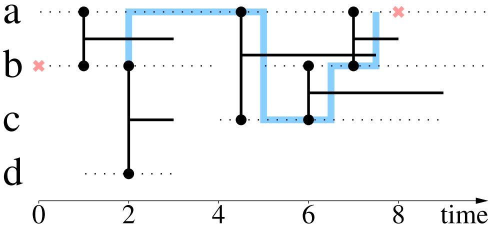
Outline

More general case – dynamics on nodes



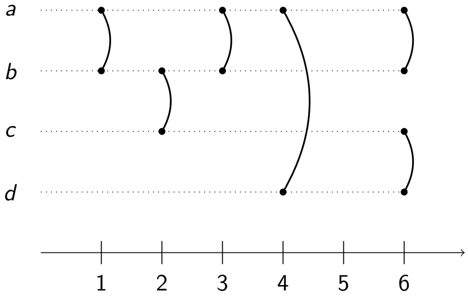
Node b is present during $[0, 4] \cup [5, 10]$

Impact on paths

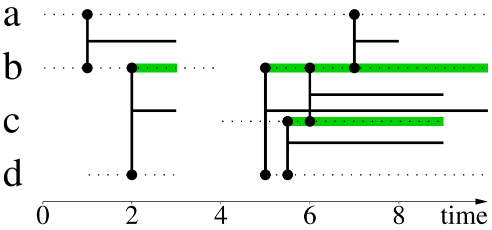


Degree centrality?

Neighborhood of a node?



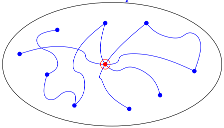
Neighborhood



Neighborhood of d

d's degree: 9.5

Betweenness centrality in graphs



$$\boxed{B(v) = \sum_{u \in V, w \in V} \underbrace{\frac{\sigma(u,w,v)}{\sigma(u,w)}}_{\substack{\text{fraction of all sp} \\ u \longrightarrow w \\ \text{involving } v}}}$$

Betweenness centrality in link streams

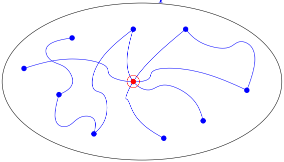
graphs :

nodes,

shortest paths,

all u and v

shortest paths



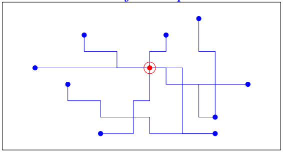
linkstreams :

temporal nodes,

shortest fastest paths (sfp),

all (t, u), (t', v)

shortest fastest paths



$$\boxed{B(t,v) = \sum_{u \in V, w \in V} \int_{i \in T, j \in T} \underbrace{\frac{\sigma((i,u),(j,w),(t,v))}{\sigma((i,u),(j,w))}}_{\text{fraction of all sfp}} \mathrm{d}i\,\mathrm{d}j}$$

Betweenness centrality in link streams

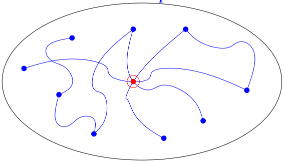
graphs :

nodes,

shortest paths,

all u and v

shortest paths



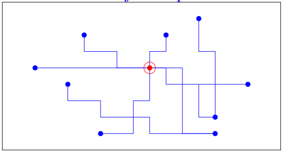
linkstreams :

temporal nodes,

shortest fastest paths (sfp),

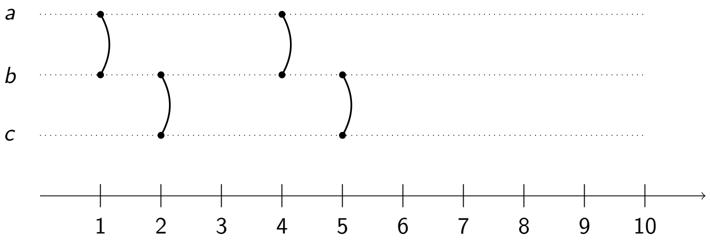
all (t, u), (t', v)

shortest fastest paths



$$\boxed{B(t,v) = \sum_{u \in V, w \in V} \int_{i \in T, j \in T} \underbrace{\frac{\sigma((i,u),(j,w),(t,v))}{\sigma((i,u),(j,w))}}_{\text{fraction of all sfp}} \,\mathrm{d}i\,\mathrm{d}j}$$

Betweenness Centrality in link streams



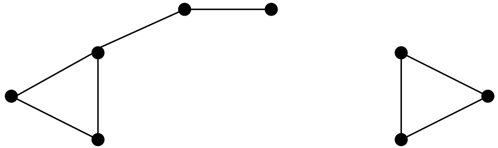
Connectivity and strong connectivity

In static networks

Connected component

Set of nodes such that there exists a path

- between any pair of nodes



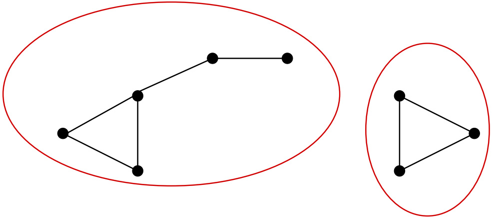
Connectivity and strong connectivity

In static networks

Connected component

Set of nodes such that there exists a path

- between any pair of nodes



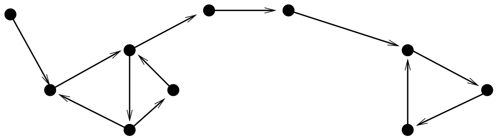
Connectivity and strong connectivity

In static directed networks

Strongly connected component

Set of nodes such that there exists a path

- from any node
- to any other node



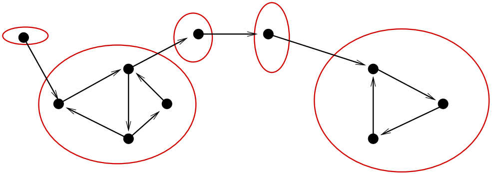
Connectivity and strong connectivity

In static directed networks

Strongly connected component

Set of nodes such that there exists a path

- from any node
- to any other node



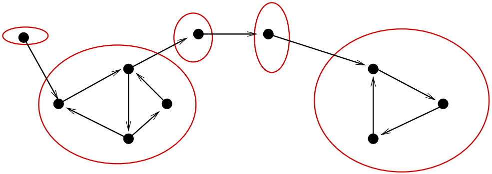
Connectivity and strong connectivity

In static directed networks

Strongly connected component

Set of nodes such that there exists a path

- from any node
- to any other node



(Strongly) connected component form a partition of nodes

Previous definitions

A link stream is strongly connected if:

There exists a temporal path

- from any node
- to any other node

Previous definitions

A link stream is strongly connected if:

There exists a temporal path

- from any node
- to any other node

Covers vastly different cases:

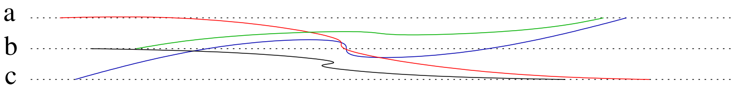
Previous definitions

A link stream is strongly connected if:

There exists a temporal path

- from any node
- to any other node

Covers vastly different cases:



Previous definitions

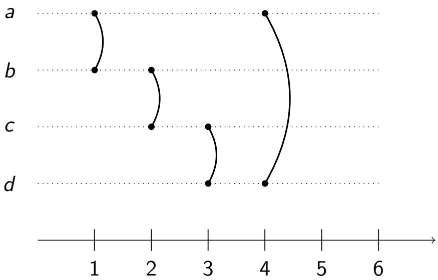
A link stream is strongly connected if:

There exists a temporal path

- from any node
- to any other node

Covers vastly different cases:

Connectivity, problem 1

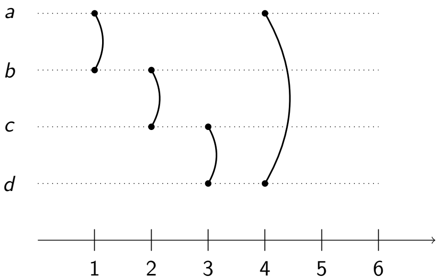


There are paths

- from a to b and back
- from a to c and back
- from c to b and back

they require d! (and no path from d to b)

Connectivity, problem 1

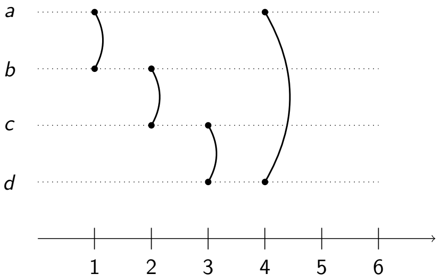


There are paths

- from a to b and back
- from a to c and back
- from c to b and back

they require d! (and no path from d to b)

Connectivity, problem 2

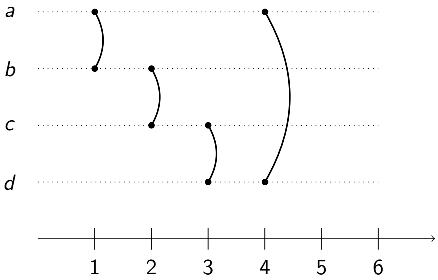


$\{a, b, c\}$ and $\{a, c, d\}$ are connected components

Components overlap!

Huge number of components in practice

Connectivity, problem 2



$\{a, b, c\}$ and $\{a, c, d\}$ are connected components

Components overlap!

Huge number of components in practice

What is a community?

Set of nodes that share something:

- Affiliation (friends, colleagues, club, ...)
- Similar interests (tagging systems, ...)
- Similar contents (movies, books, products, web pages, ...)
- ...

What is the connexion with the network structure?

What is a community?

Set of nodes that share something:

- Affiliation (friends, colleagues, club, ...)
- Similar interests (tagging systems, ...)
- Similar contents (movies, books, products, web pages, ...)
- ...

What is the connexion with the network structure?

What is a community?

Set of nodes that share something:

- Affiliation (friends, colleagues, club, ...)
- Similar interests (tagging systems, ...)
- Similar contents (movies, books, products, web pages, ...)
- ...

What is the connexion with the network structure?

More densely connected inside than outside

Community detection

Goal: Identify communities automatically

Applications:

- Understand the structure of a network
- Help visualization
- Detect specific communities (web pages, proteins, ...)
- Improve information retrieval (search engines, recommendation, ...)

Challenges

- Unknown number of communities
- Unknown sizes of communities
- Scalability

Challenges

- Unknown number of communities
- Unknown sizes of communities
- Scalability

Many definitions of a community

Using betweenness centrality for community detection

[Girvan and Newman, 2002]

- **Step 1:** All nodes are in the same community (initialization)
- **Step 2:** Compute the betweenness of each link
- **Step 3:** Delete the highest betweenness link
- **Step 4:** Iterate from step 1

Use definition of betweenness centrality in link streams?

I hope this lecture

- Made you grasp the challenges of dealing with temporal paths
- Showed you that digging into the data can be easy
- Showed that it is not easy to find relevant indicators

. . . and gave you some hints!