Temporal density of community structure

Sergey Kirgizov Éric Leclercq

MARAMI 2019, Dijon, November 8

Continuing our previous works...

- S.K. and Éric Leclercq
  Temporal density of complex networks and ego-community dynamics.
  Conference on Complex Systems (ECCS or CCS) 2016.

- I. Basaille, E. Leclercq, M. Savonnet, N. Cullot, S.K., T. Grison, E. Gavignet
  Un observatoire pour la modélisation et l'analyse des réseaux multi-relationnels. Une application à l'étude du discours politique sur Twitter.
  Document Numérique 20(1), 101, 2017.

Goal

Visualisation for exploration !

Visualisation for exploration !

A visualisation technique capable to represent a dynamic community structure in a comprehensible & æsthetic manner.

Visualisation for exploration !

A visualisation technique capable to represent a dynamic community structure in a comprehensible & æsthetic manner.

In order to approach the goal we should discuss and rethink the existing definitions of community structures and slightly modify community detection algorithms.

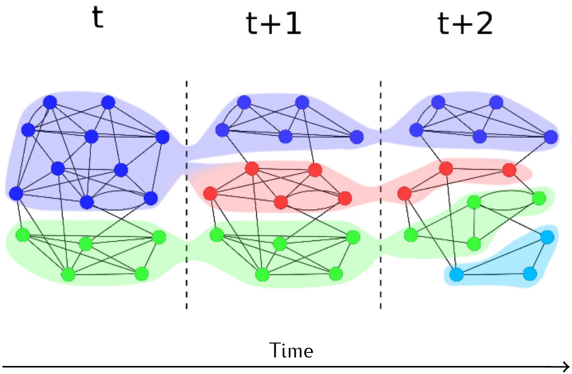
Overview

1. Community dynamics
2. Temporal density
3. Illustrations: School contacts network

Dynamic communities

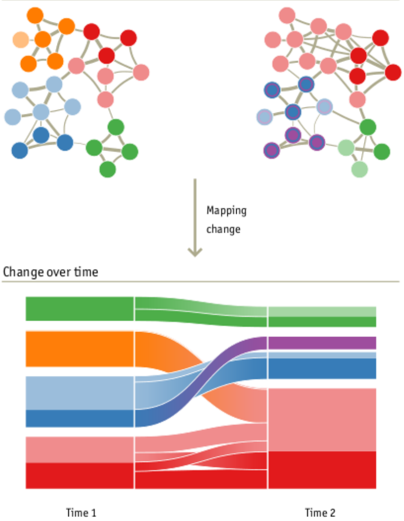
Dynamic community structures

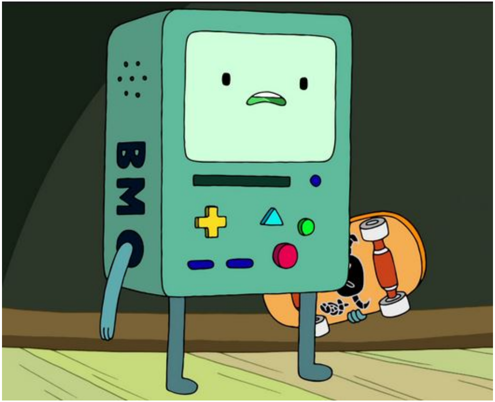
Snapshots based visualisation



Snapshot based visualisation

Mapping change in large networks
Rosvall, Bergstrom, 2010

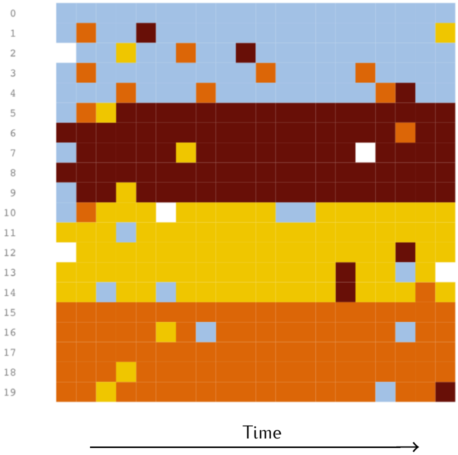




Unfortunatelly, snapshot based visualisation is not suitable for large graphs.

A step towards the goal...

“Intrinsically Dynamic Network Communities”
Computer Networks 2011
Bivas Mitra, Lionel Tabourier, Camille Roth



Lines corresponds to nodes, colors to different communities.

Many real networks changes smoothly...

We need another method.

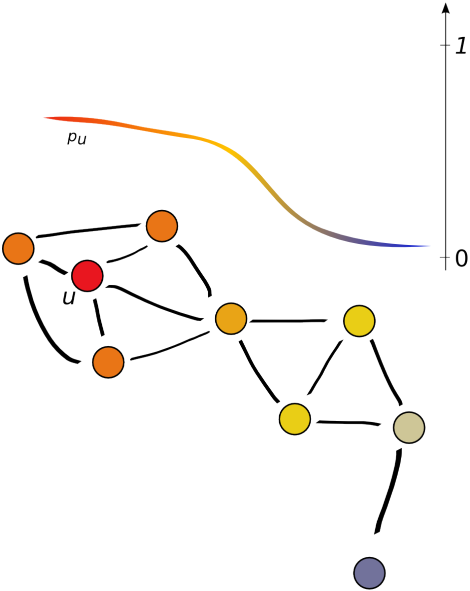
Let us approach the problem of visualisation by considering ego-community structures

Ego-community structure

Node u is a centre of the ego-community

Let $p_u(v)$ be equal to the probability for a node v be in the community of node u.

Red color means high probability, blue means small.



Algorithms

Many algorithms can provide such community structure

1. Eigenvector centrality
2. **Personalised pagerank**
3. Shi–Malik's normalised relaxed mincut
4. Pons–Latapy's Walktrap
5. Danisch–Guillaume–Le Grand's Carryover opinion
6. Heat propagation based methods
7. Kleinberg's HITS

   etc, etc

Personalised pagerank can find good communities (even if they overlap) in real-world networks (DBLP, Youtube, Amazon)

Community membership identification from small seed sets

Kloumann et Kleinberg, SIGKDD, 2014

Overlapping Community Detection Using Neighborhood-Inflated Seed Expansion

Whang, Gleich, Dhillon, 2015

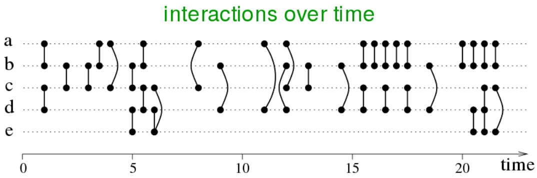
By pagerank we find *instantaneous* ego-communities

What about their dynamics?

Discrete dynamic graph models

Dynamic graphs

- Snapshots, Hopcroft et al., 2004, Leskovec et al., 2005
- Time-varying graphs, Casteigts et al., 2012, Wehmuth et al., 2013
- Link Streams, Viard, Latapy, and Magnien, 2016 **illustrated below**



Changes in these models are discrete.

Typical dynamic dataset: stream of links

A node interacts with another node at time t.

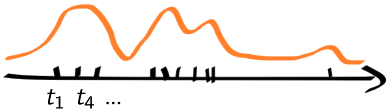
a	b	t_1
c	b	t_2
d	c	t_3
a	b	t_4
d	b	t_5
	...	

Smooth the discrete input data

Link stream between a and b

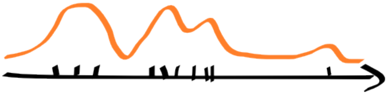
a	b	t_1
c	b	t_2
d	c	t_3
a	b	t_4
d	b	t_5
	...	

transforms to



(by Parzen–Rosenblatt method)

Slice 'n' pagerank

Time-density between *a* and *b* 

Time-density between *c* and *b* 

Time-density between *a* and *c* 

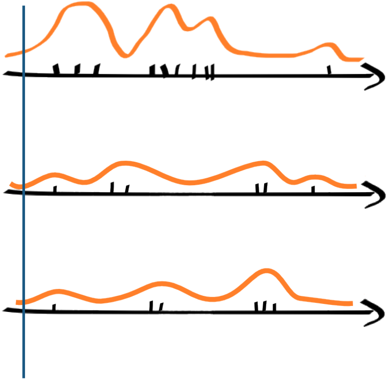
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between a and b

Time-density between c and b

Time-density between a and c



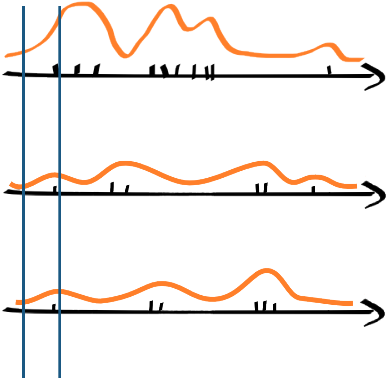
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between *a* and *b*

Time-density between *c* and *b*

Time-density between *a* and *c*



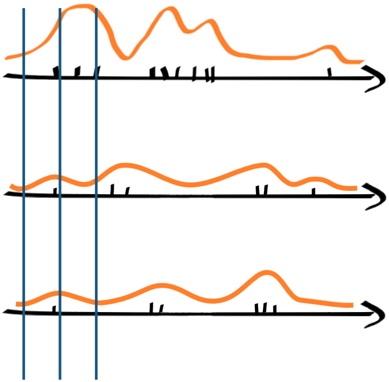
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between a and b

Time-density between c and b

Time-density between a and c



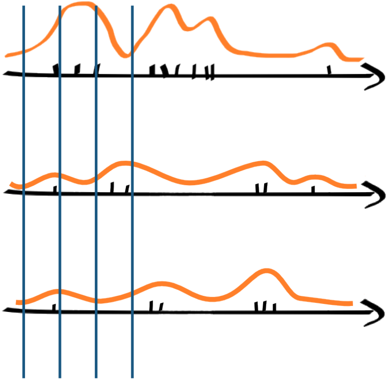
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between a and b

Time-density between c and b

Time-density between a and c



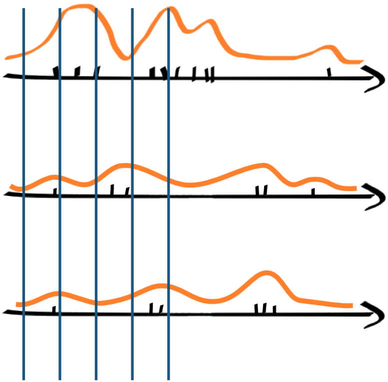
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between a and b

Time-density between c and b

Time-density between a and c



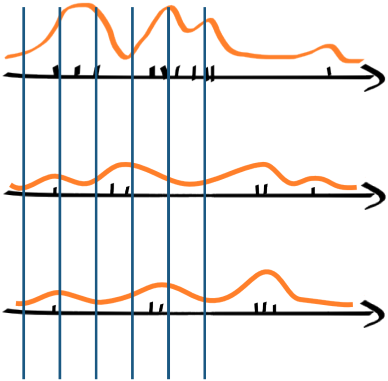
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between *a* and *b*

Time-density between *c* and *b*

Time-density between *a* and *c*



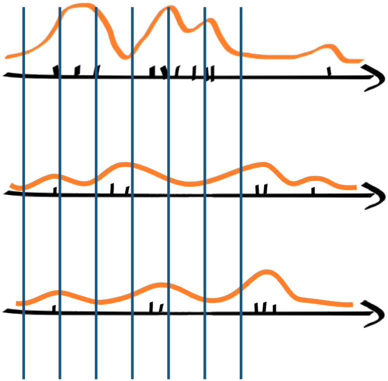
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between *a* and *b*

Time-density between *c* and *b*

Time-density between *a* and *c*



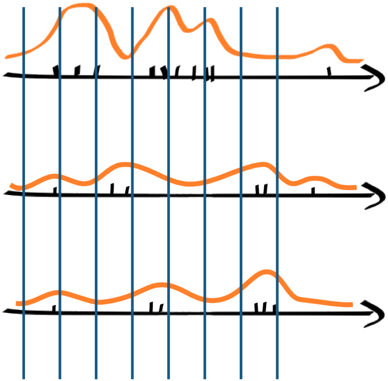
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between *a* and *b*

Time-density between *c* and *b*

Time-density between *a* and *c*



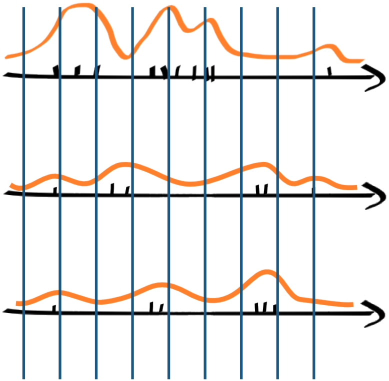
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between *a* and *b*

Time-density between *c* and *b*

Time-density between *a* and *c*



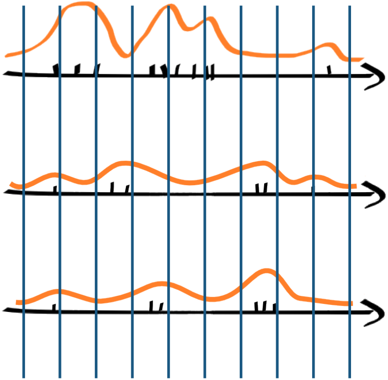
1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Slice 'n' pagerank

Time-density between *a* and *b*

Time-density between *c* and *b*

Time-density between *a* and *c*

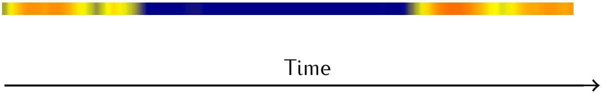


1. Slice smoothed data
2. Construct a weighted graph for every slice
3. Compute personalised pagerank vector for every slice
4. Collect and plot the results !

Represent temporal community membership probability as color row.

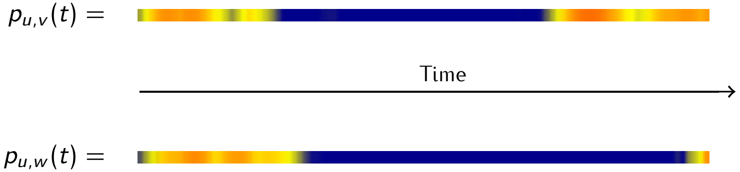
Red — high probability
Yellow — average probability
Dark Blue — zero.

$p_{u,v}(t) =$



Represent temporal community membership probability as color row.

Red — high probability
Yellow — average probability
Dark Blue — zero.



Every graph node x have one color row $p_{u,x}(t)$. It shows us the membership degree for the node x in the community of node u at the time t.

Why color changes smoothly ?

Visualisation of ego-community dynamics

Adjacency matrix $A(t)$ and community structure both depends on time.

Let $p_{u,v}(t)$ be a scaled probability for a node v be in the ego-community of node u at time t.

Proposition

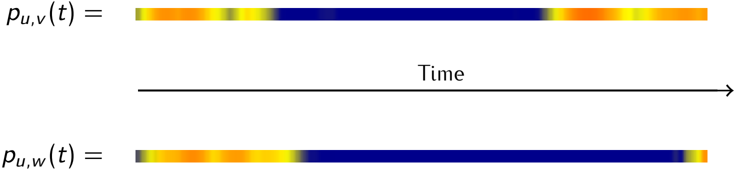
When adjacency matrix $A(t)$ is smooth then $p_{u,v}(t)$ is also smooth.

Schema of the proof

Pagerank is a eigenvector of certain irreducible and aperiodic matrix that depends smoothly on adjacency matrix.
Use Theorem 3.2 from **A Note on Perturbations of Stochastic Matrices** by Huppert et Willems, 2000

Represent temporal community membership probability as color row.

Red — high probability
Yellow — average probability
Dark Blue — zero.



Every graph node x have one color row $p_{u,x}(t)$. It shows us the membership degree for the node x in the community of node u at the time t.

Sort color rows by greedy stacking

Let u be the pre-selected center of community structure.

1. Add all nodes, except u to the set D.
2. Add a $p_{u,u}(t)$ to the stack.
3. Find a node $x \in D$, such that the distance between current stack's top (or bottom) and $p_{u,x}(t)$ is minimal.
4. Add $p_{u,x}(t)$ to the stack.
5. Remove x from D.
6. If D is not empty, go to 3.
7. Draw the stack!

Illustrations

DATASET: Primary school temporal network data

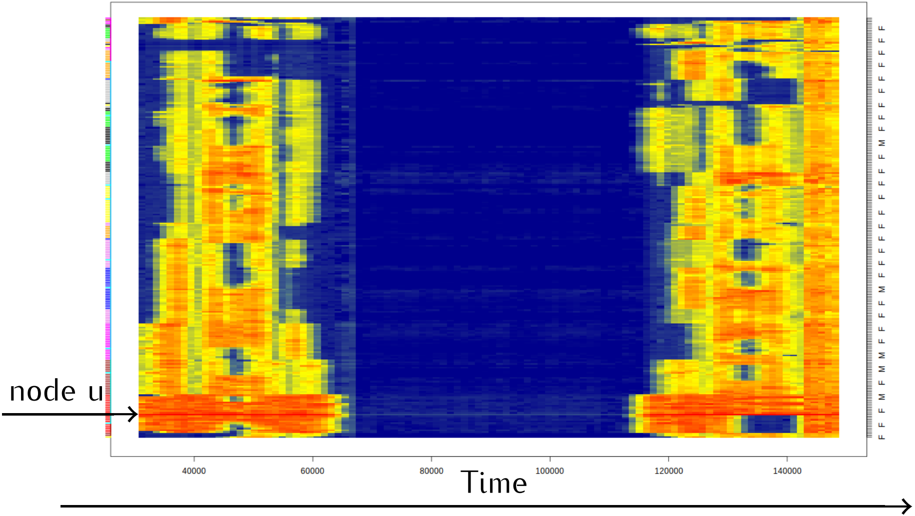
Release data: Sep 30, 2015

This data set contains the temporal network of contacts between the children and teachers used in the study published in BMC Infectious Diseases 2014, 14:695. The file contains a tab-separated list representing the active contacts during 20-second intervals of the data collection. Each line has the form "t i j Ci Cj", where i and j are the anonymous IDs of the persons in contact, Ci and Cj are their classes, and the interval during which this contact was active is [t – 20s, t]. If multiple contacts are active in a given interval, you will see multiple lines starting with the same value of t. Time is measured in seconds.

Terms and conditions

242 nodes, 125 773 links

Rows correspond to students. Columns are instantaneous ego-centred community structures around a selected student. The left x-axis denotes the classes of participants. Right x-axis is about sex. Red color means “in the community of u with high probability at a certain time”, blue — probability is near zero.



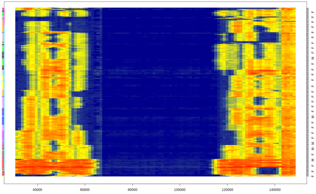
Conclusion

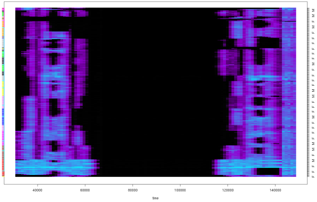
Temporal density approach for

Community evolution visualisation

Future

- Find more applications
- Extend to global communities structures
- On-line data processing
- Better and faster color row stacking
- Play with different colors ?!





Thanks for your attention!

Sergey Kirgizov, `http://kirgizov.link`

Éric Leclercq, `http://eric-leclercq.fr`

`https://github.com/kerzol/ego-evolution`