Mining Time-Dependent Communities

Qinna Wang, Eric Fleury

D-NET/INRIA, LIP, Université de Lyon
Ecole Normale Supérieure de Lyon

November 4, 2010

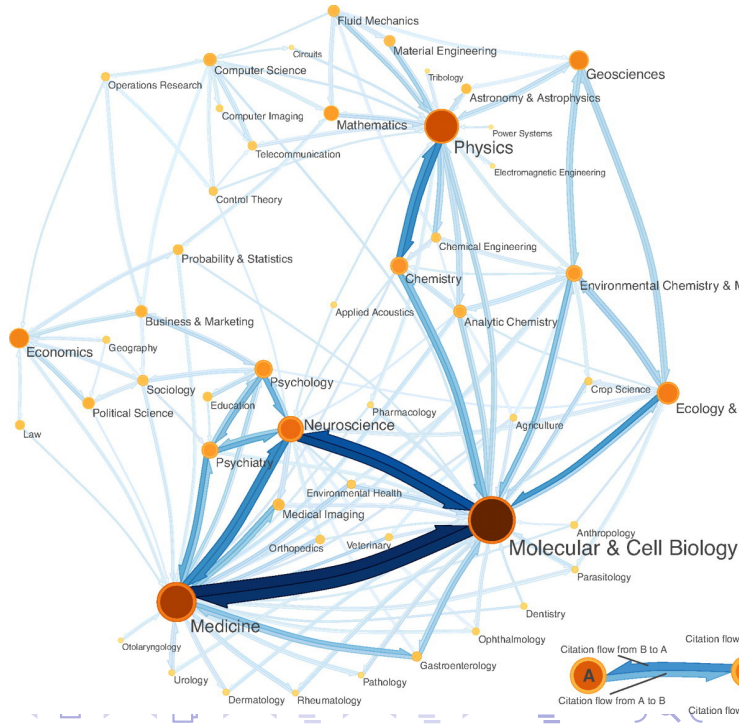
Object

Time evolution of communities is related with different structural properties and characteristics(e.x. ENRON):

- Capture critical events(economic crisis)
- Characterize community members(position of employee)
- Predict behaviors of communities(frequency of behaviors)

Dynamic communities in real networks:

- Internet based systems(e.g. Facebook, Myspace, Twitter)
- ENRON email networks
- CELLPHONE call communication networks
- etc.



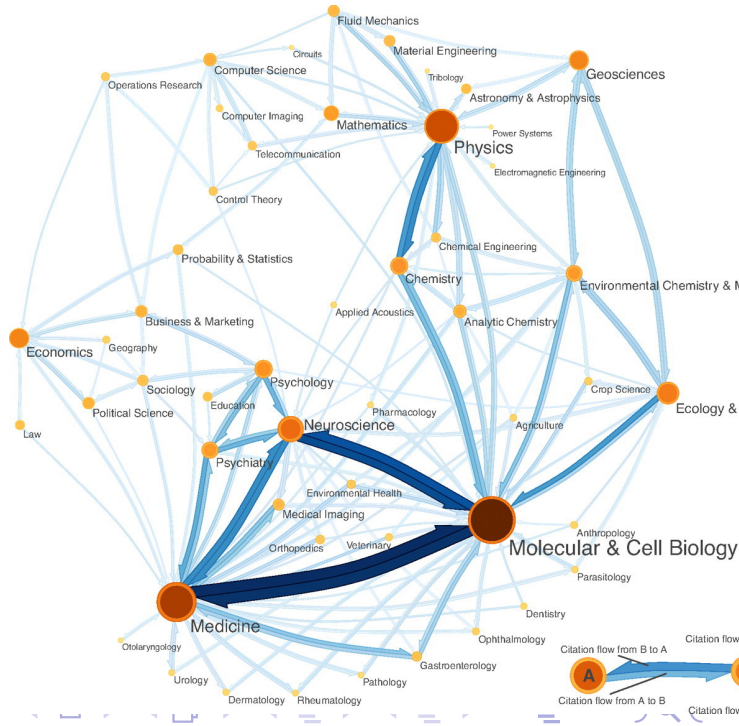
Object

Time evolution of communities is related with different structural properties and characteristics(e.x. ENRON):

- Capture critical events(economic crisis)
- Characterize community members(position of employee)
- Predict behaviors of communities(frequency of behaviors)

Dynamic communities in real networks:

- Internet based systems(e.g. Facebook, Myspace, Twitter)
- ENRON email networks
- CELLPHONE call communication networks
- etc.



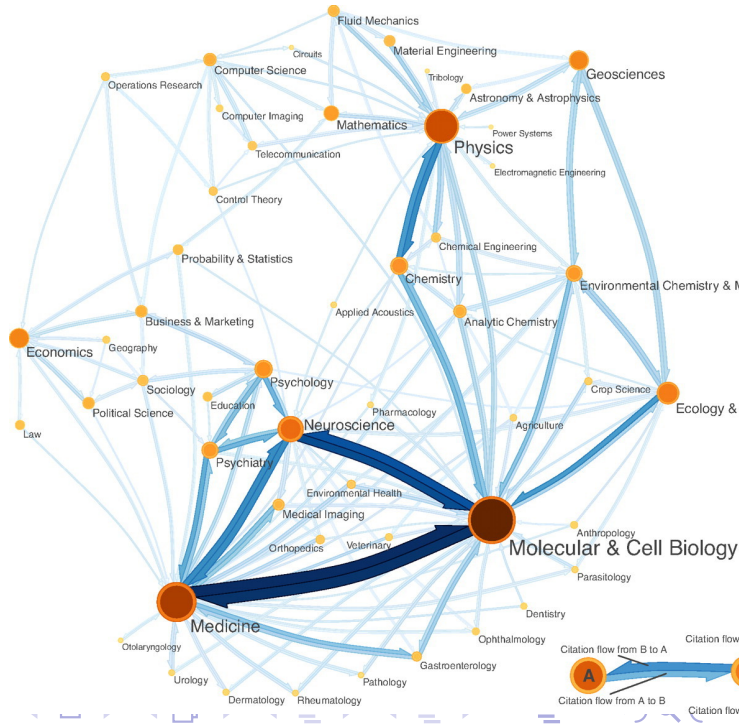
Object

Time evolution of communities is related with different structural properties and characteristics(e.x. ENRON):

- Capture critical events(economic crisis)
- Characterize community members(position of employee)
- Predict behaviors of communities(frequency of behaviors)

Dynamic communities in real networks:

- Internet based systems(e.g. Facebook, Myspace, Twitter)
- ENRON email networks
- CELLPHONE call communication networks
- etc.



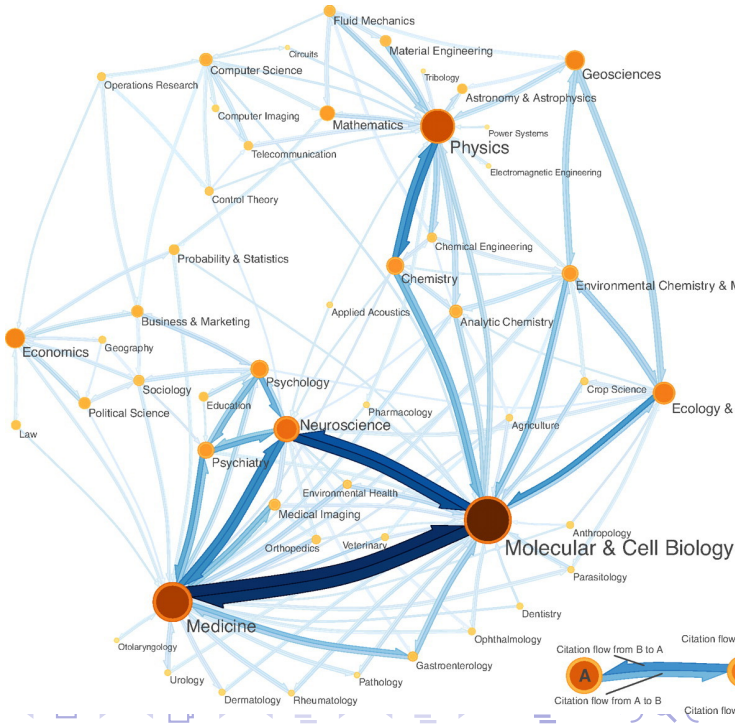
Object

Time evolution of communities is related with different structural properties and characteristics(e.x. ENRON):

- Capture critical events(economic crisis)
- Characterize community members(position of employee)
- Predict behaviors of communities(frequency of behaviors)

Dynamic communities in real networks:

- Internet based systems(e.g. Facebook, Myspace, Twitter)
- ENRON email networks
- CELLPHONE call communication networks
- etc.



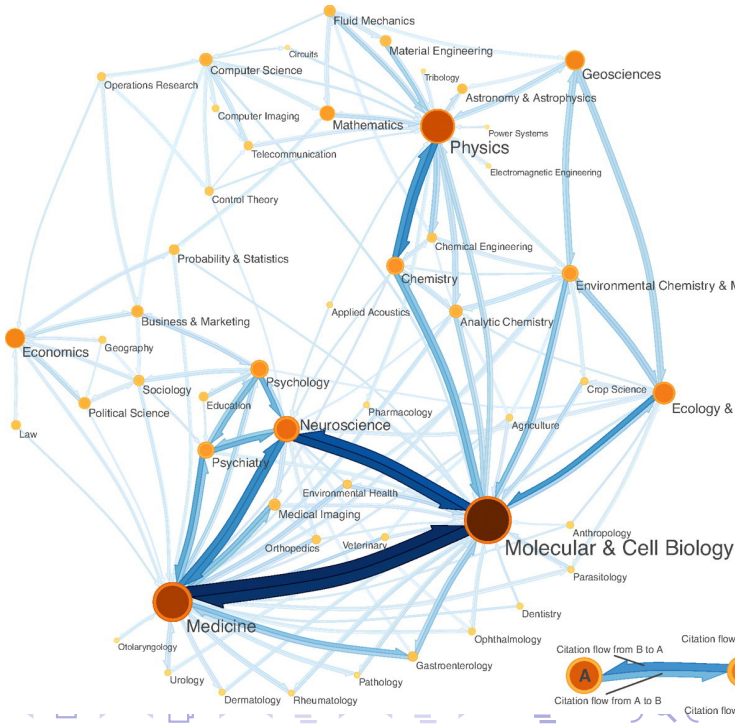
Object

Time evolution of communities is related with different structural properties and characteristics(e.x. ENRON):

- Capture critical events(economic crisis)
- Characterize community members(position of employee)
- Predict behaviors of communities(frequency of behaviors)

Dynamic communities in real networks:

- Internet based systems(e.g. Facebook, Myspace, Twitter)
- ENRON email networks
- CELLPHONE call communication networks
- etc.



Open questions

Traditional graph partition often fails:

- Overlapping: individuals may be shared between communities
- Robust in topology evolution: the community evolution should follow its histories.

Problem:

Are there new graph techniques to handle time-dependent overlapping communities?

- CPM: adjacent k-cliques as communities
- LFM: fitness optimization
- Link partition: consider the problem of community structure as the link partition
- GraphScope
- Multi-slice modularity

Open questions

Traditional graph partition often fails:

- Overlapping: individuals may be shared between communities
- Robust in topology evolution: the community evolution should follow its histories.

Problem:

Are there new graph techniques to handle time-dependent overlapping communities?

- CPM: adjacent k-cliques as communities
- LFM: fitness optimization
- Link partition: consider the problem of community structure as the link partition
- GraphScope
- Multi-slice modularity

Open questions

Traditional graph partition often fails:

- Overlapping: individuals may be shared between communities
- Robust in topology evolution: the community evolution should follow its histories.

Problem:

Are there new graph techniques to handle time-dependent overlapping communities?

- CPM: adjacent k-cliques as communities
- LFM: fitness optimization
- Link partition: consider the problem of community structure as the link partition
- GraphScope
- Multi-slice modularity

Core nodes in static networks

Cores , also called seeds, are embedded into communities and used to denote communities.

Identifying cores is a key metric to identify communities.

Different choices:

- Maximum cliques: [Lee2009]
- A random node: LFM algorithm
- Strong clusters: a set of nodes keep stable memberships.
  - Overlapping size:

$$R_{C_{j;b}}(C_{i;a}) = \frac{|C_{i;a} \cap C_{j;b}|}{|C_{i;a} \cup C_{j;b}|} . \quad (1)$$

  - Among the founded partitions, select the partition $\mathcal{P}_{max}$;
  - For $C_i \in \mathcal{P}_{max}$, select the strong cluster C' which has the largest overlapping size $R_{C'}(C_i)$ to its core.

Core nodes in static networks

Cores , also called seeds, are embedded into communities and used to denote communities.

Identifying cores is a key metric to identify communities.

Different choices:

- Maximum cliques: [Lee2009]
- A random node: LFM algorithm
- Strong clusters: a set of nodes keep stable memberships.
  - Overlapping size:

$$R_{C_{j;b}}(C_{i;a}) = \frac{|C_{i;a} \cap C_{j;b}|}{|C_{i;a} \cup C_{j;b}|} . \quad (1)$$

  - Among the founded partitions, select the partition $\mathcal{P}_{max}$;
  - For $C_i \in \mathcal{P}_{max}$, select the strong cluster C' which has the largest overlapping size $R_{C'}(C_i)$ to its core.

Core nodes in static networks

Cores , also called seeds, are embedded into communities and used to denote communities.

Identifying cores is a key metric to identify communities.

Different choices:

- Maximum cliques: [Lee2009]
- A random node: LFM algorithm
- Strong clusters: a set of nodes keep stable memberships.
  - Overlapping size:

$$R_{C_{j;b}}(C_{i;a}) = \frac{|C_{i;a} \cap C_{j;b}|}{|C_{i;a} \cup C_{j;b}|} . \quad (1)$$

  - Among the founded partitions, select the partition $\mathcal{P}_{max}$;
  - For $C_i \in \mathcal{P}_{max}$, select the strong cluster C' which has the largest overlapping size $R_{C'}(C_i)$ to its core.

A matrix $\mathbf{P} = [p_{ij}]_{n \times n}$

How to uncover strong clusters? A matrix $\mathbf{P} = [p_{ij}]_{n \times n}$, whose element p_{ij} presents the probability of (i, j) belonging to the same community.

Assumption

The probability p_{ij} implies membership:

1. *$p_{ij} \geq \beta$ represents (i, j) holding a stable membership ($\beta = 100\%$, typically);*
2. *if $(i, j) \in C$ and $(i, k) \in C$, then $(j, k) \in C$.*

- Results of Louvain algorithm depend on the ordering of clustering nodes;
- Compute $\mathbf{P}$ by repeating Louvain algorithm until $\|P_{ij}^{k+1} - P_{ij}^{k}\| < \varepsilon$.

Community detection on static networks

Fitness function:

$$f_C = \frac{\sum_{i \in C} k_i^{int}}{(\sum_{i \in C} k_i^{int} + \sum_{i \in C} k_i^{out})^{\alpha}} , \quad (2)$$

Optimization technique:

1. Select one core C_s to a community C,
2. For each node i which is adjacent to a community C, calculate its node fitness, i.e. $F_i = F_{C \cup i} - F_{C \setminus i}$.
3. Select the node i which brings the largest positive F_i.
4. If the node i exists, add it to C and repeat step 2; otherwise, stop and return the final community C.

Fitness optimization is suggested as a local optimization strategy for overlapping community detection.

Overview of static community detection

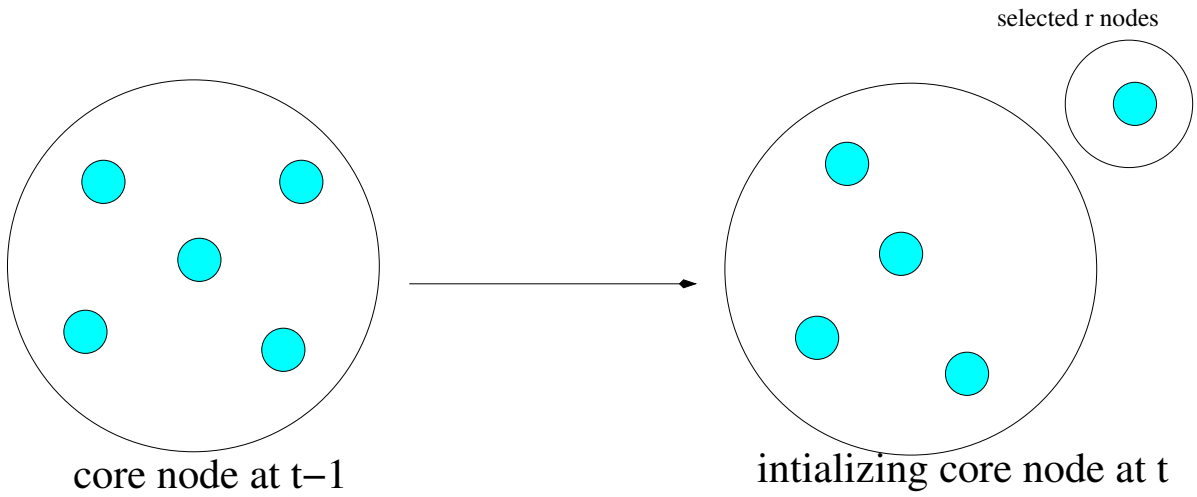
- Our overlapping community detection can be summarized as:
  1. Find strong clusters C' by repeating Louvain algorithm until the convergence of $\mathbf{P}$.
  2. Select one strong clusters C' which has the largest overlapping size with the community $C_i \in \mathcal{P}_{max}$ to the core for the expansion until no addition of peripheral nodes would improve its fitness.
  3. Go back and continue step 2 until all cores have been expanded.
- Complexity: $\mathcal{O}(K|E| + M|E|)$

Open question

Can we design community structure detection for dynamic networks?

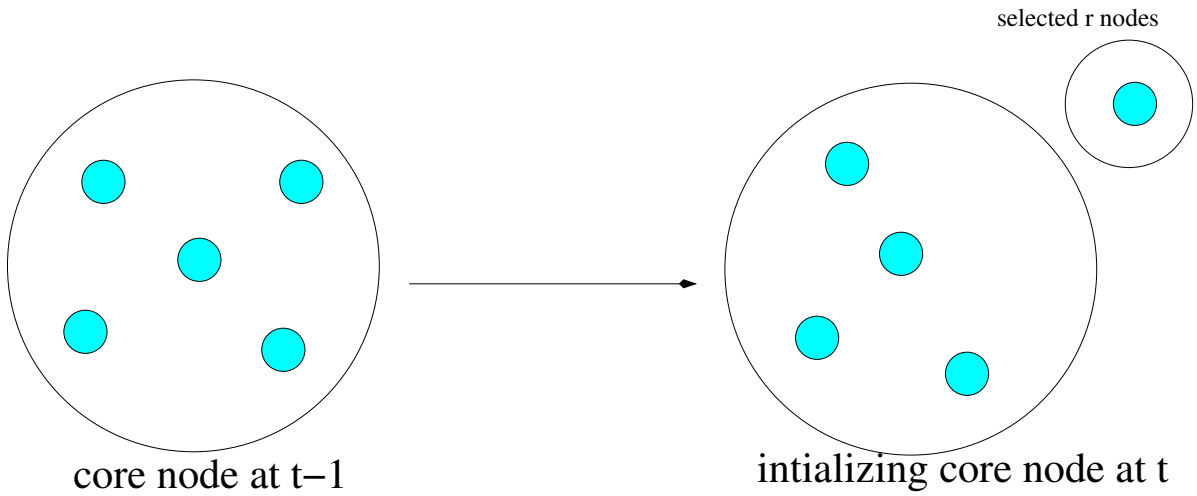
Our method

- Initializing each core $C_{si;t-1} \in \mathcal{C}_{s;t-1}$ into a sub-community before community detection on $\mathcal{G}_t$
  $\mathcal{C}_t$ is naturally related with its history.
- For a given core $C_{si;t-1}$, select the proportionality $r \in [0, 1]$ of the core nodes $i \in \mathcal{C}_{s;t-1}$ and each selected node is considered as a single sub-community when running Louvain algorithm.
  Our modification makes our results be affected with the parameter r.



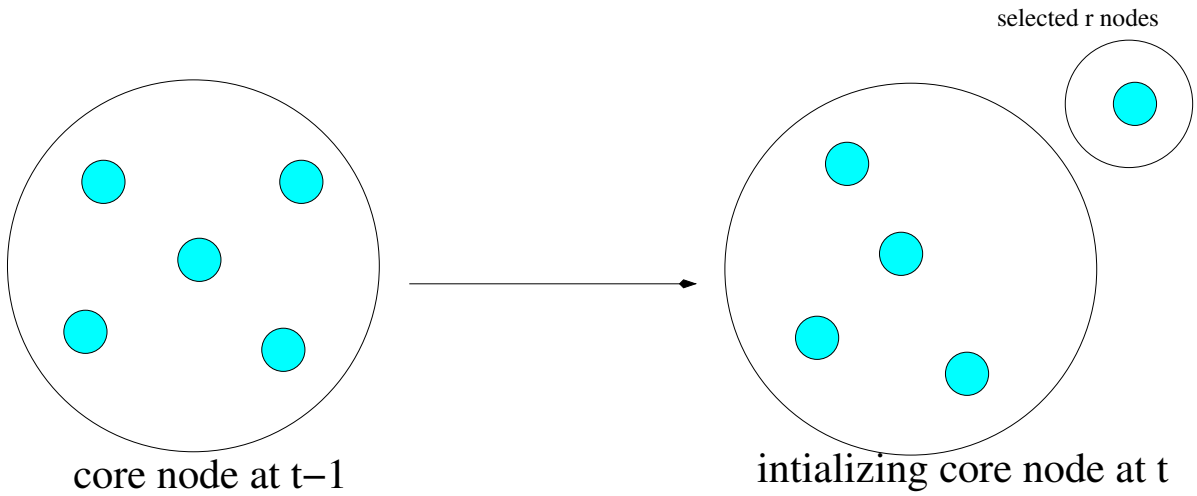
Our method

- Initializing each core $C_{si;t-1} \in \mathcal{C}_{s;t-1}$ into a sub-community before community detection on $\mathcal{G}_t$
  $\mathcal{C}_t$ is naturally related with its history.
- For a given core $C_{si;t-1}$, select the proportionality $r \in [0, 1]$ of the core nodes $i \in C_{s;t-1}$ and each selected node is considered as a single sub-community when running Louvain algorithm.
  Our modification makes our results be affected with the parameter r.



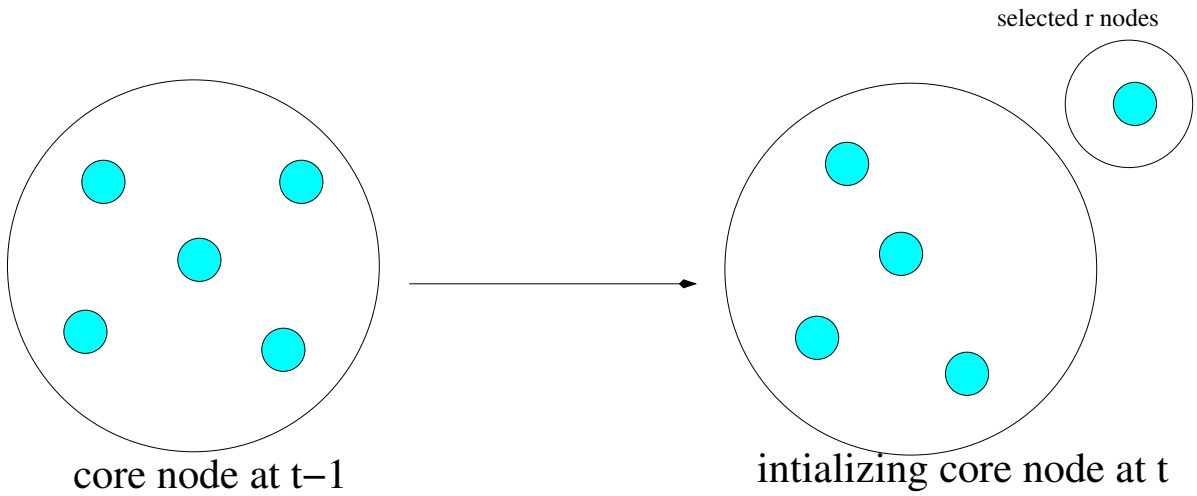
Our method

- Initializing each core $C_{si;t-1} \in \mathcal{C}_{s;t-1}$ into a sub-community before community detection on $\mathcal{G}_t$
  $\mathcal{C}_t$ is naturally related with its history.
- For a given core $C_{si;t-1}$, select the proportionality $r \in [0, 1]$ of the core nodes $i \in \mathcal{C}_{s;t-1}$ and each selected node is considered as a single sub-community when running Louvain algorithm.
  Our modification makes our results be affected with the parameter r.



Our method

- Initializing each core $C_{si;t-1} \in \mathcal{C}_{s;t-1}$ into a sub-community before community detection on $\mathcal{G}_t$
  $\mathcal{C}_t$ is naturally related with its history.
- For a given core $C_{si;t-1}$, select the proportionality $r \in [0, 1]$ of the core nodes $i \in \mathcal{C}_{s;t-1}$ and each selected node is considered as a single sub-community when running Louvain algorithm.
  Our modification makes our results be affected with the parameter r.



Community track

How to track community evolution?

Definition

The front F_i

- F_i is used to denote and identify a D_i;
- F_i is the first core of D_i.
- when $C_{i;t}$ is found and it fails to match any dynamic community, a new D_i is created whose front is $C_{si;t}$, where $C_{si;t}$ is the core of $C_{i;t}$.

- Select $\mathcal{F} = \{F_1, \ldots, F_k\}$ to denote $\mathcal{D} = \{D_1, \ldots, D_k\}$;
- The matching between $C_{i;t}$ and D_j is through $R_{C_{si;t}}(F_j)$ with $\theta \in [0, 1]$:
  - if $R_{C_{si;t}}(F_j) \geq \theta$, $C_{i;t}$ is considered as the observation of D_j at time t;
  - if several fronts fit, $C_{i;t}$ is matched in descending ordering of their overlapping size;
  - otherwise, a new D_{k+1} is formed and denoted by the front $F_{k+1} \equiv C_{si;t}$.

Greene et al. Benchmark

Greene et al. Benchmark: A version of LFR benchmark graphs with community evolution, in which the change of community structure is controlling by different community events:

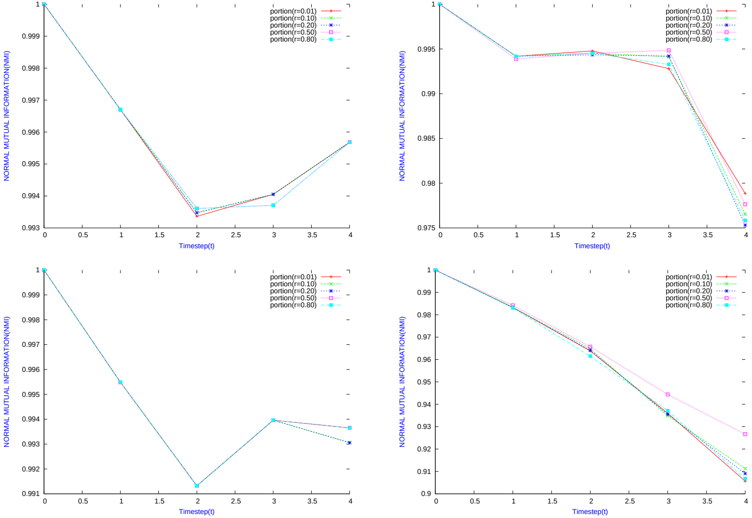
1. Intermittent communities, in which 10% of existing communities are unobserved for their concealment at each time step;
2. Expansion and contraction, in which 40 randomly selected communities expand or contract their 25% size at each time step;
3. Birth and death, in which 40 new communities are constructed to replace 40 existing communities;
4. Merging and splitting, in which 40 communities are randomly selected to be split, while 40 cases of the merging of two random communities happen.

Comparing two community structure

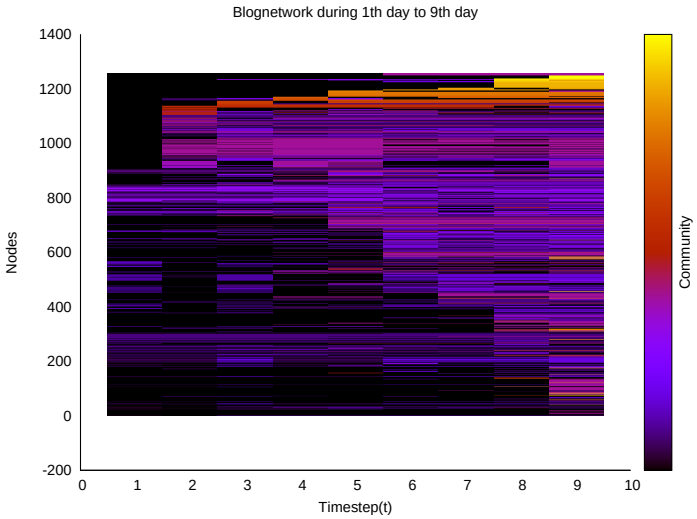
Normal mutual information $N(\mathbf{X}|\mathbf{Y})$ measures the similarity between $(\mathbf{X}, \mathbf{Y})$; NMI is 1 if the community structure is identical; the higher NMI is, the more similar, $(\mathbf{X}, \mathbf{Y})$ is.

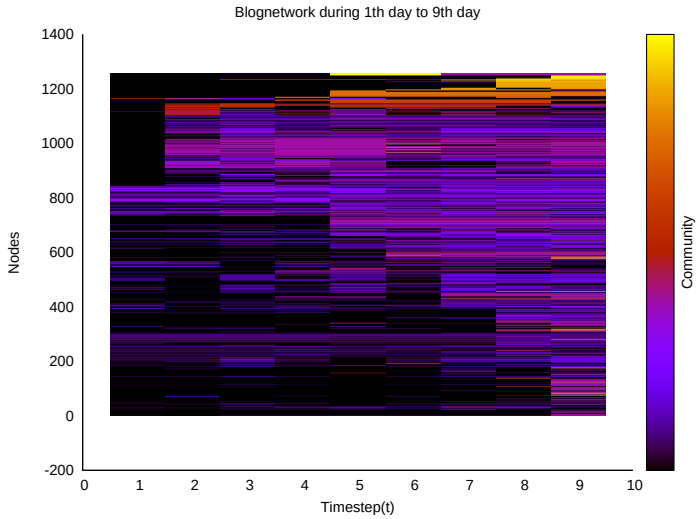
1. For each $I \in \{1, \ldots, C''\}, k \in \{1, \ldots, C'\}$, compute $H(X_k|Y_I)$;
2. Compute $H(X_k|\mathbf{Y}) = min_{I \in \{1,\ldots,C''\}} H(X_k|Y_I)$;
3. Compute $H(\mathbf{X}|\mathbf{Y})_{norm} = \frac{1}{|C'|} \sum_k \frac{H(X_k|\mathbf{Y})}{H(X_k)}$
4. $N(\mathbf{X}|\mathbf{Y}) = 1 - \frac{1}{2}[H(\mathbf{X}|\mathbf{Y})_{norm} + H(\mathbf{Y}|\mathbf{X})_{norm}]$
   $0 \leq N(\mathbf{X}|\mathbf{Y}) \leq 1$

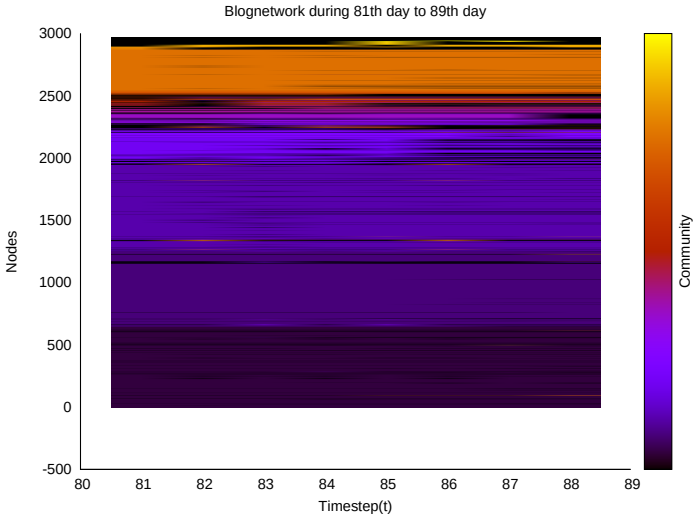
Results

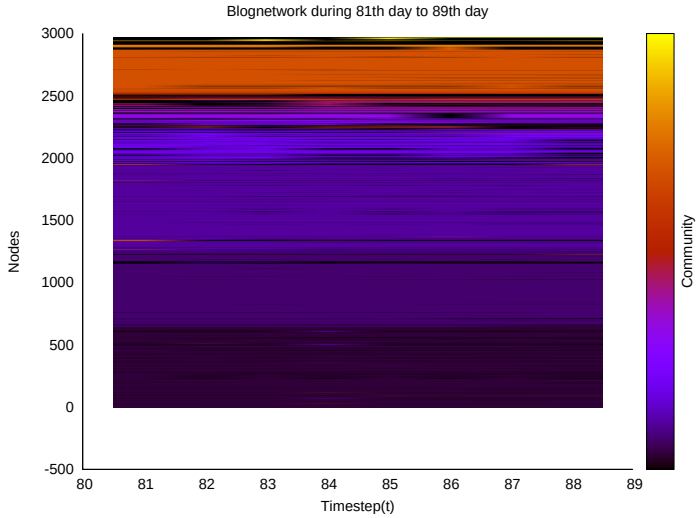

portion(r=0.01)
portion(r=0.10)
portion(r=0.20)
portion(r=0.50)
portion(r=0.80)
NORMAL MUTUAL INFORMATION(NMI)
Timestep(t)
0 0.5 1 1.5 2 2.5 3 3.5 4
1 0.999 0.998 0.997 0.996 0.995 0.994 0.993
1 0.995 0.99 0.985 0.98 0.975
1 0.999 0.998 0.997 0.996 0.995 0.994 0.993 0.992 0.991
1 0.99 0.98 0.97 0.96 0.95 0.94 0.93 0.92 0.91 0.9

Tests on blog networks









Results

- The evolution of core nodes in $\{\mathcal{G}_1, \ldots, \mathcal{G}_9\}$ is less stable than $\{\mathcal{G}_{81}, \ldots, \mathcal{G}_{89}\}$: more nodes are added into the core set, more parts of core nodes change their memberships, and more emergence of new dynamic communities over time.
- For different values $r = 0.01, r = 0.50$, we observe the few difference: the similar number of dynamic communities, the similar core node assignment, the similar modularity value and the similar tendency of modularity. It seems that the outputs of our method is little influenced by the proportionality r.

Summary

- A new framework to resolve the problem of dynamic community detection, which regularizes communities by initializing sub-communities corresponding the cores and the proportionality r.
- Perfect experimental performances on synthetic graphs.
- Future work: on real networks.
- We expect our method be a powerful tool in mining features and properties of complex networks.

Summary

- A new framework to resolve the problem of dynamic community detection, which regularizes communities by initializing sub-communities corresponding the cores and the proportionality r.
- Perfect experimental performances on synthetic graphs.
- Future work: on real networks.
- We expect our method be a powerful tool in mining features and properties of complex networks.

Thank you!

Questions???

Benchmark graphs

LFR Benchmark graphs: A version of planted l-partition model.

- n nodes are assigned into l groups;
- Each node has p_{in} of being connected to nodes of its group;
- Each node has p_{out} of being connected to nodes of different groups;
- All nodes have the same mixing parameter $\mu := p_{in}/(p_{in} + p_{out})$.

A graph is built with a set of parameters: the mixing parameter, the average degree, the maximum degree.

- Each node is assigned a degree from a power-low distribution;
- Each community is given a size from a power-low distribution;
- Nodes are randomly assigned into communities; if the community is complete, a random node is kicked out; the procedure continues until all nodes are assigned to communities