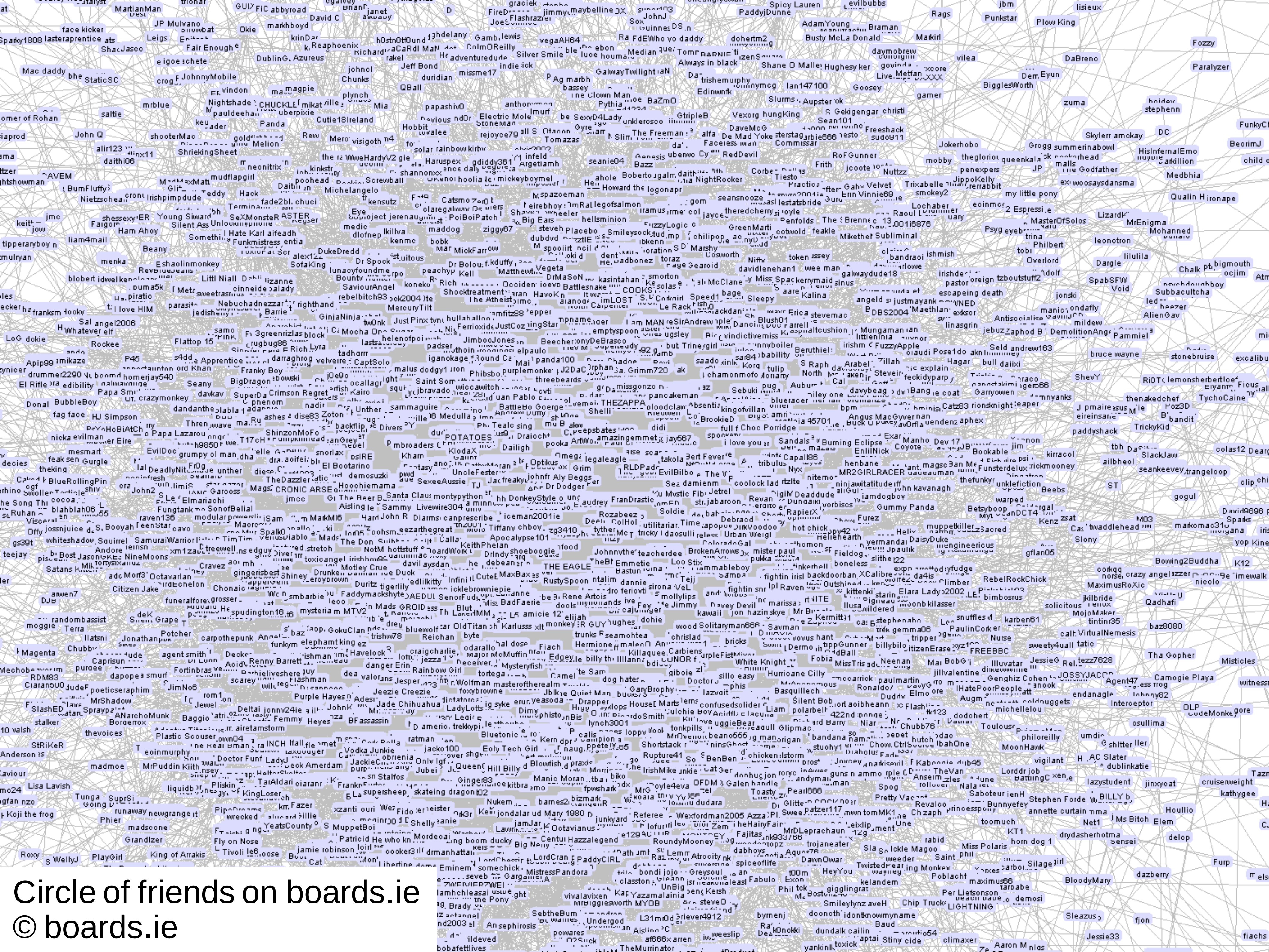
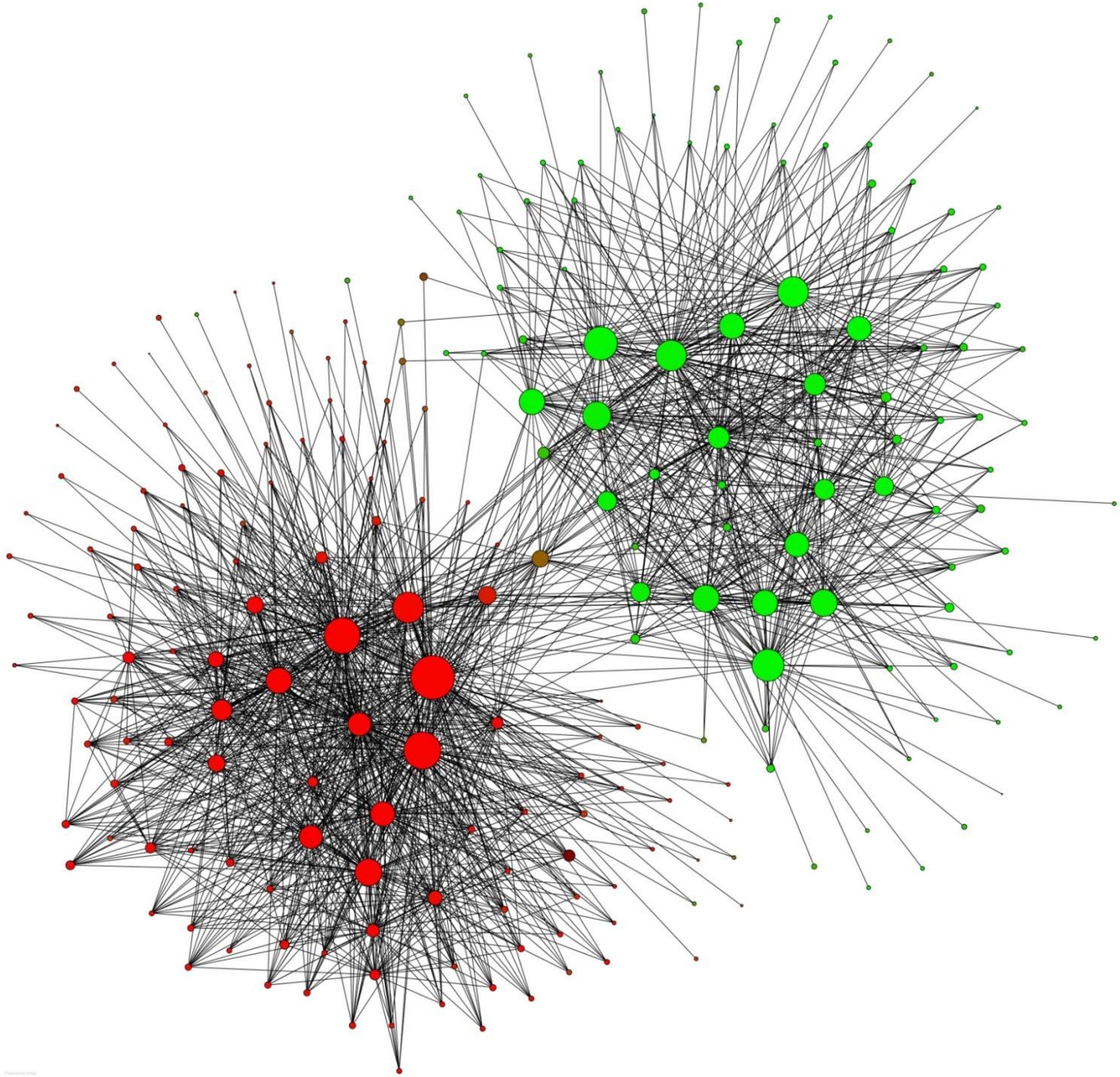
Towards evolving communities: detection and stability

Detection: V. Blondel, R. Lambiotte, E. Lefebvre
Stability: T. Aynaud

Jean-loup.guillaume@lip6.fr



Circle of friends on boards.ie
© boards.ie



What is a community?

In a network: a set of nodes which share something

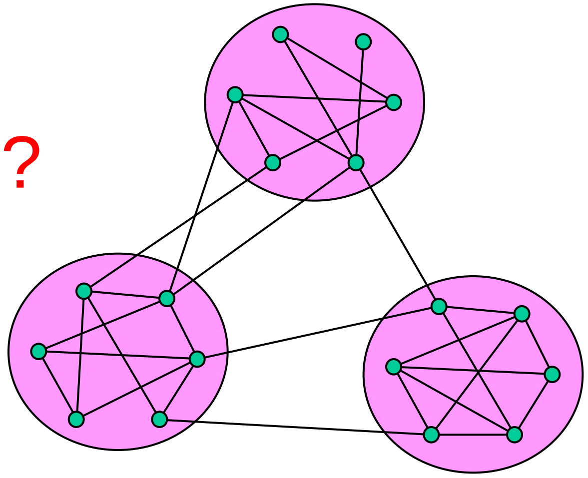
- persons with a similar interest (family members, friends)
- web pages with a similar content
- blogs on a same topic, etc.

What is a community?

In a network: a set of nodes which share something

- persons with a similar interest (family members, friends)
- web pages with a similar content
- blogs on a same topic, etc.

Relation with the structure of the network?
Densely connected groups of nodes



(Automatic) community detection

Applications

- understand the structure of these networks
- detect communities of special interest:
  - Web pages, similar files on P2P networks, …
- visualization
- improve search engines, P2P networks, routing techniques…

Challenges

- unknown number of communities of various sizes
- **scalability: web ~ billions nodes**
- overlapping communities
- **evolving communities**

Outline

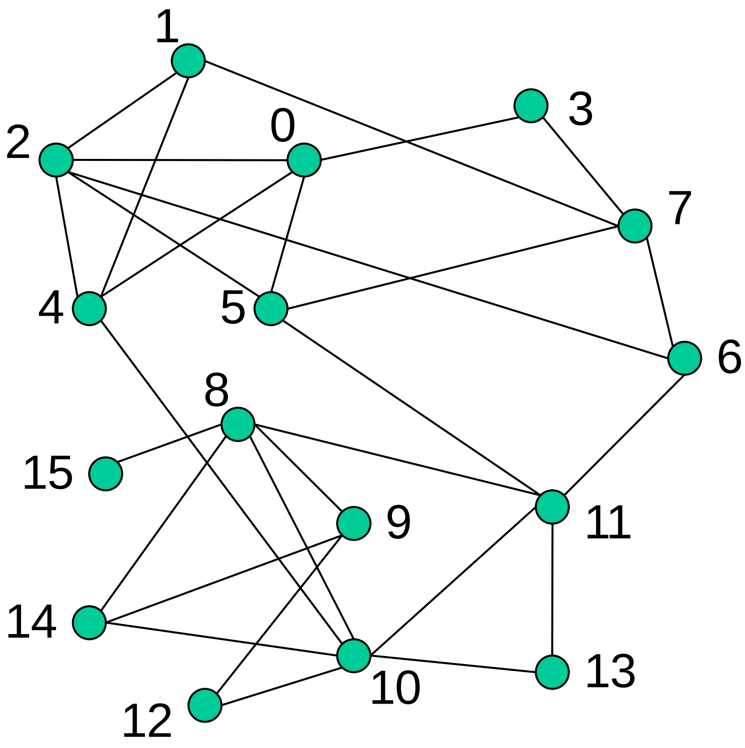
Static community detection

- Louvain algorithm
- experimental results

Dynamic communities

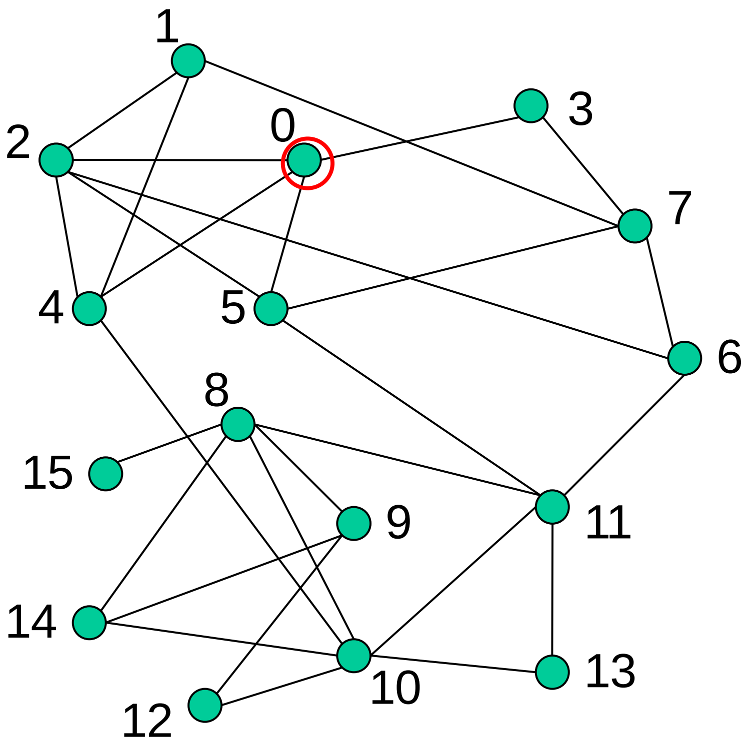
- past studies
- stability issues

An example

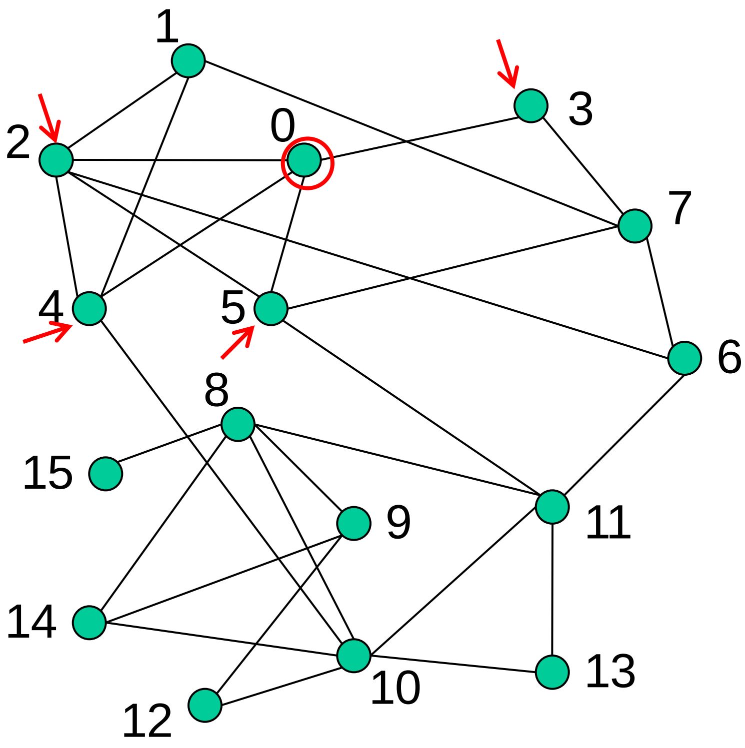


Each node belongs to an atomic community

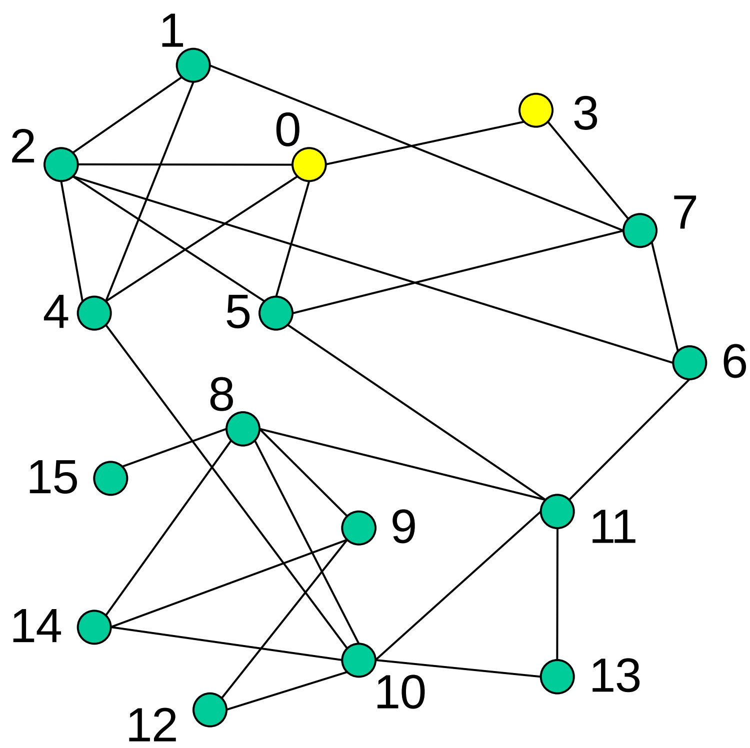
An example



An example

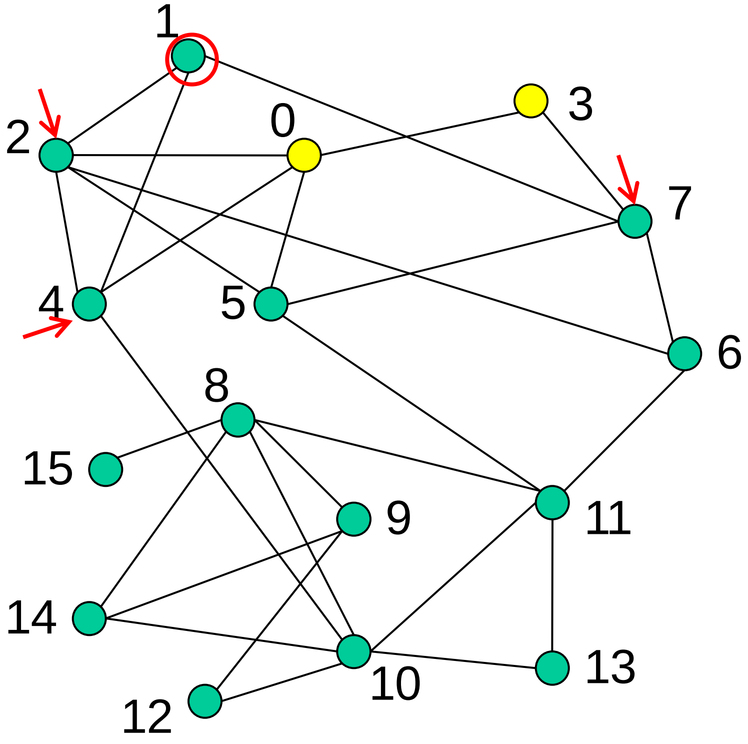


An example



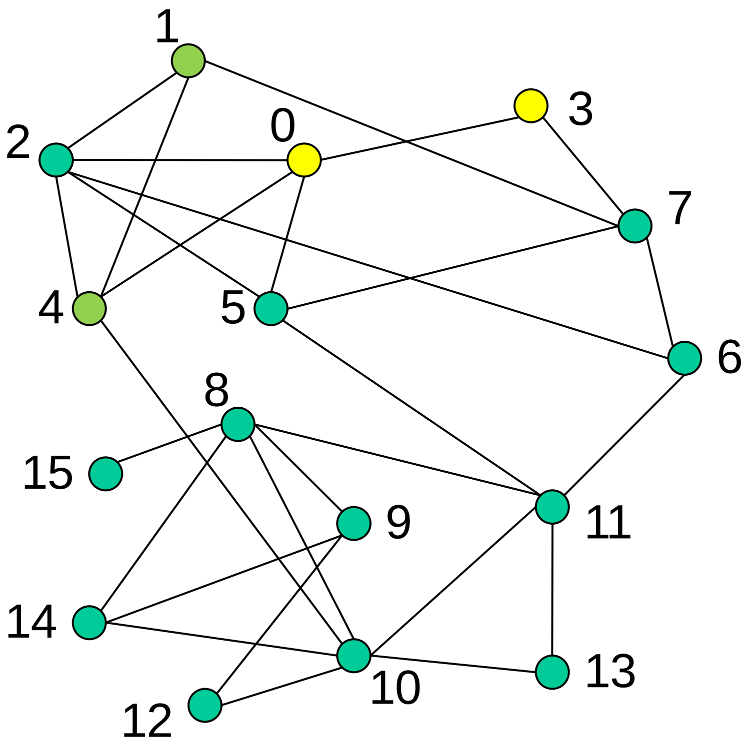
Pass 1 – Iteration 1
insert 0 in c[3]

An example



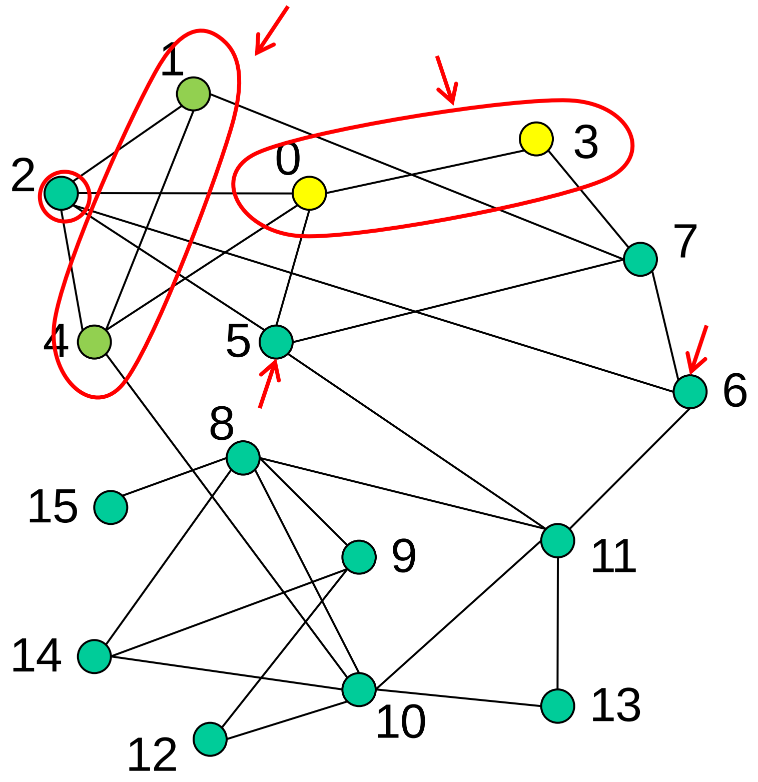
Pass 1 – Iteration 1
insert 0 in c[3]

An example



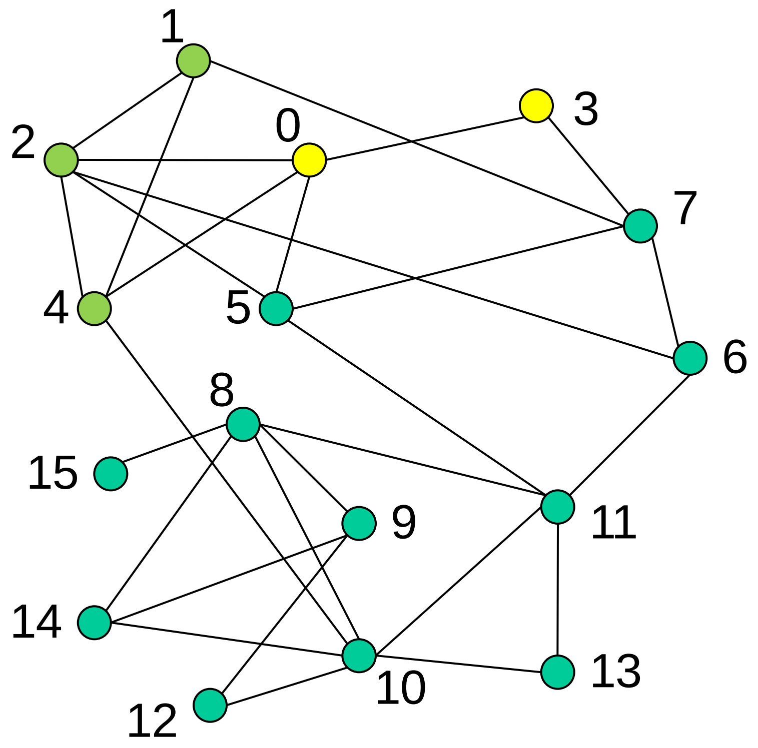
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]

An example



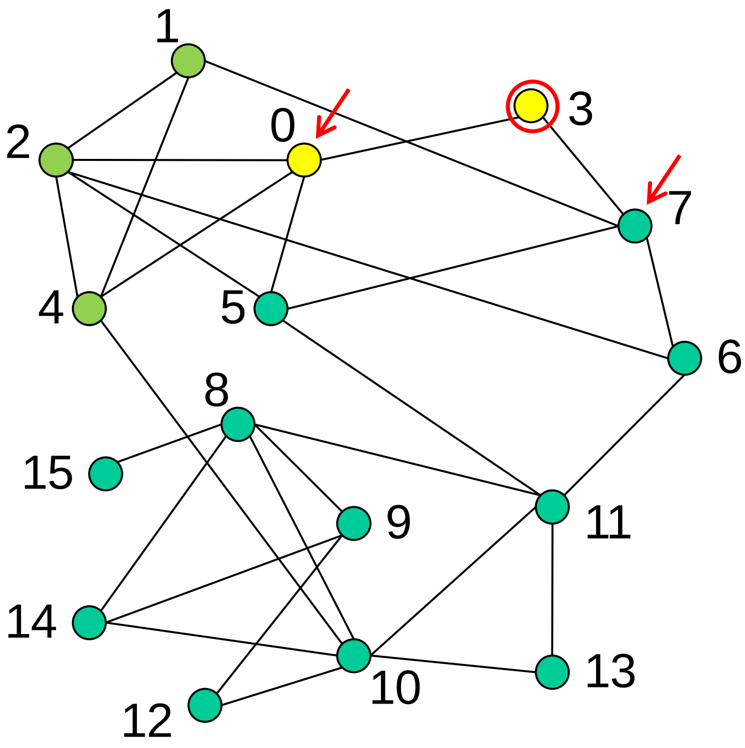
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]

An example



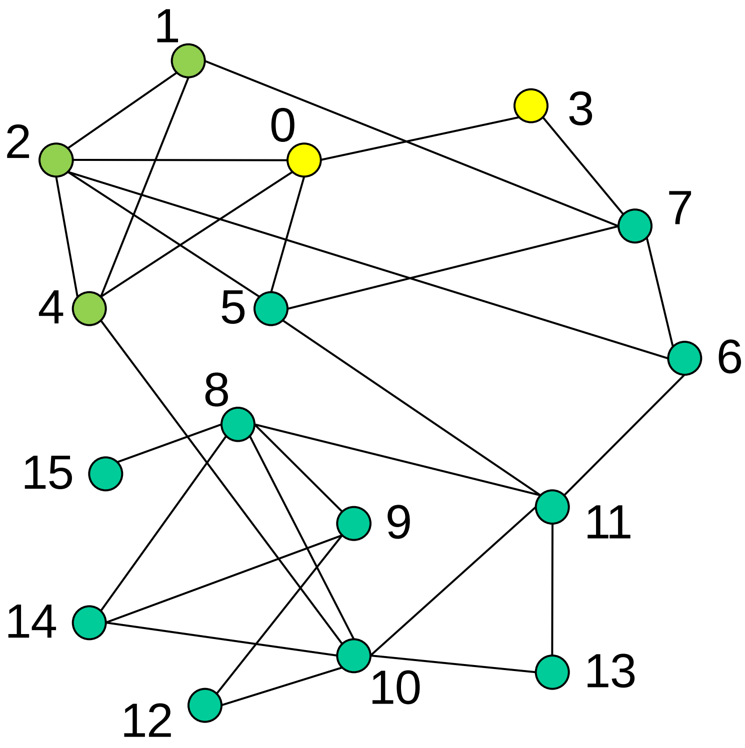
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]
insert 2 in c[1,4]

An example



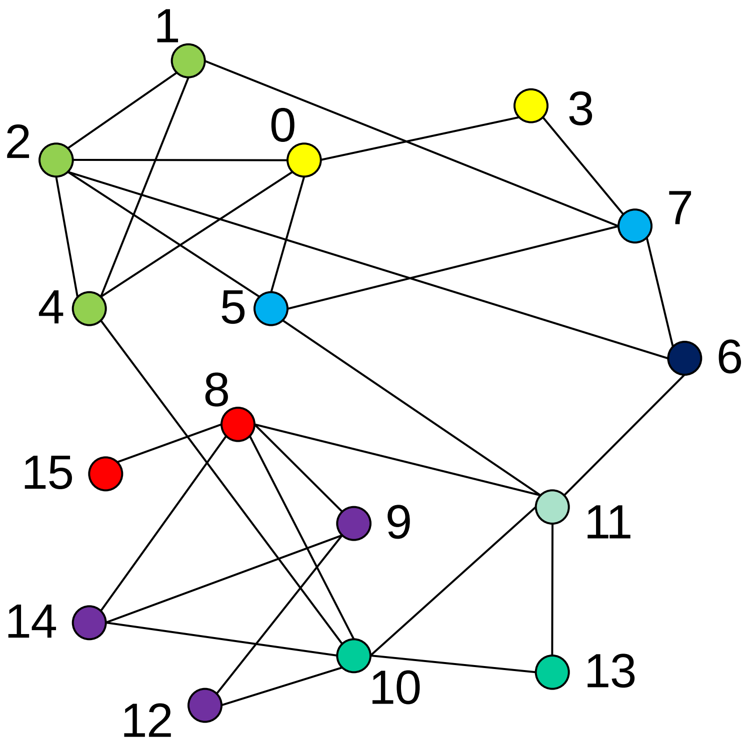
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]
insert 2 in c[1,4]

An example



Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]
insert 2 in c[1,4]
insert 3 in c[0]

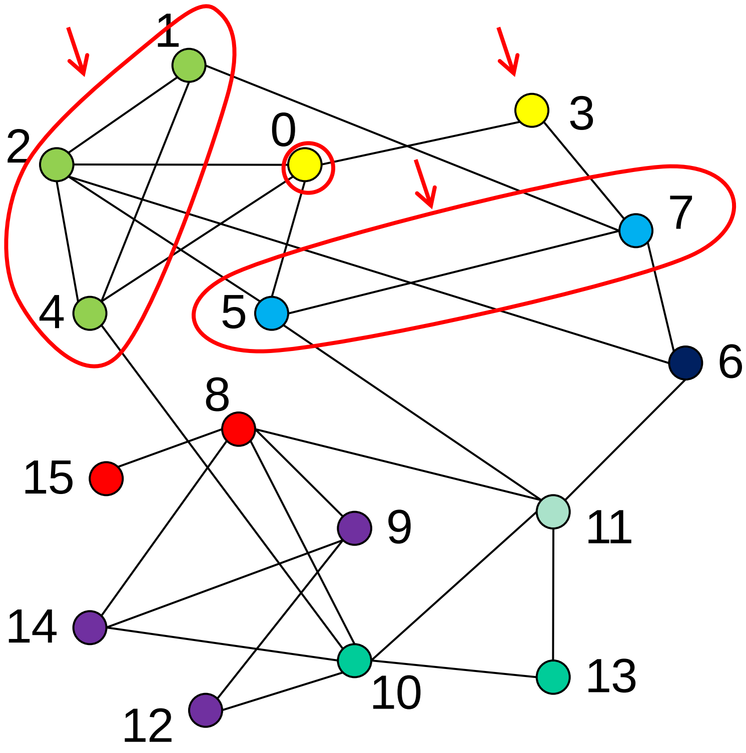
An example



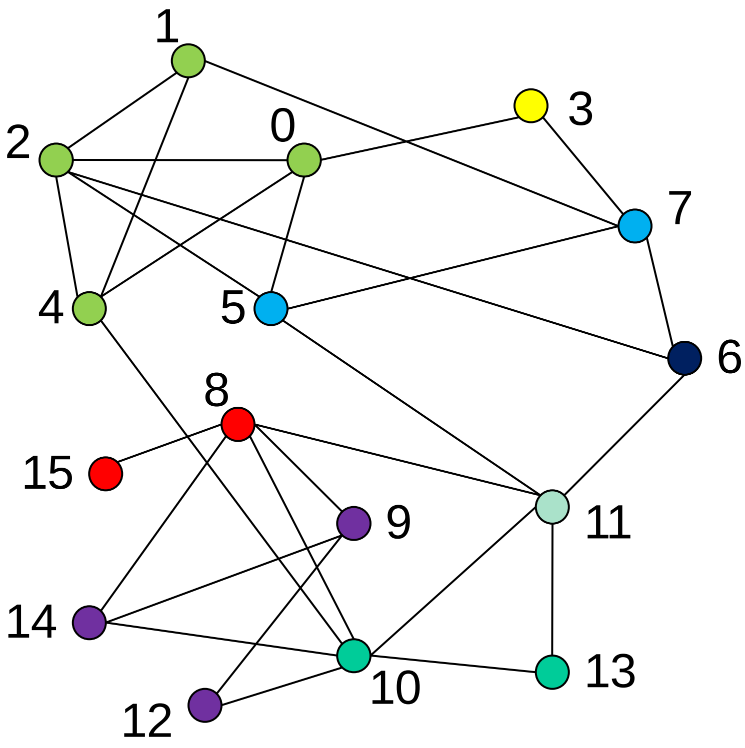
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]
insert 2 in c[1,4]
insert 3 in c[0]
insert 4 in c[1]
insert 5 in c[7]
insert 6 in c[11]
insert 7 in c[5]
insert 8 in c[15]
insert 9 in c[12]
insert 10 in c[13]
insert 11 in c[10,13]
insert 12 in c[9]
insert 13 in c[10,11]
insert 14 in c[9,12]
insert 15 in c[8]

An example

Pass 1 – Iteration 2

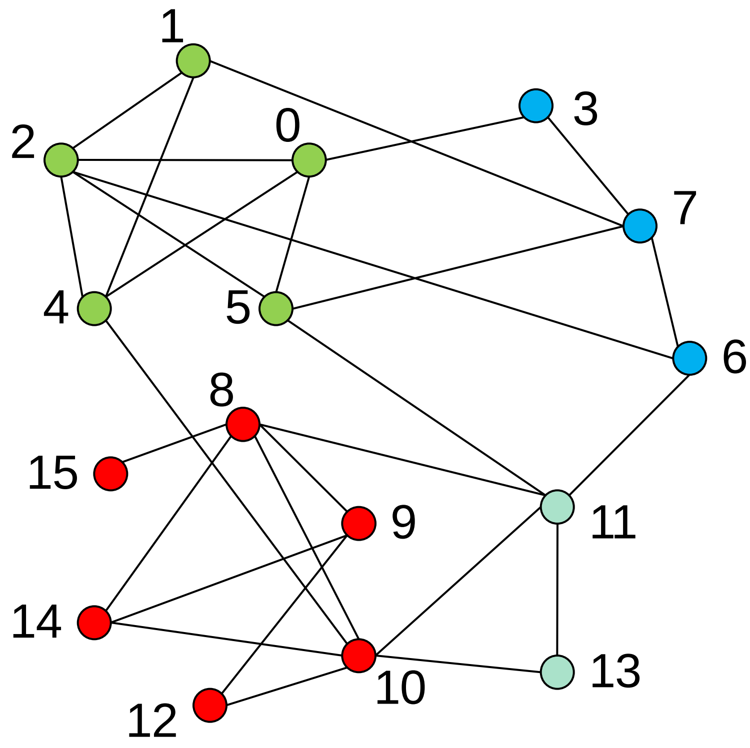


An example



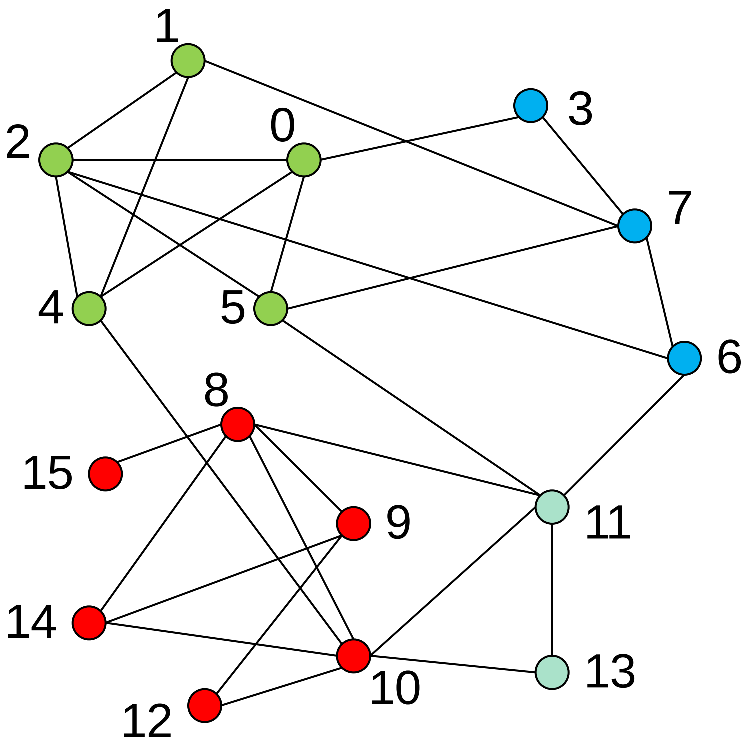
Pass 1 – Iteration 2
insert 0 in c[4]
…

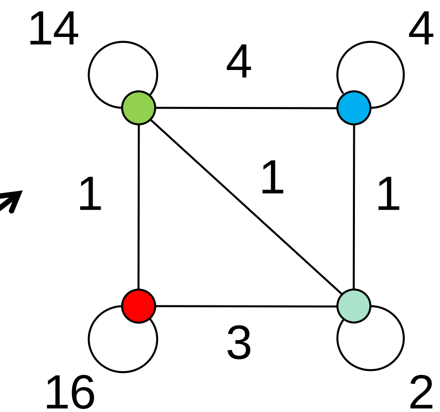
An example



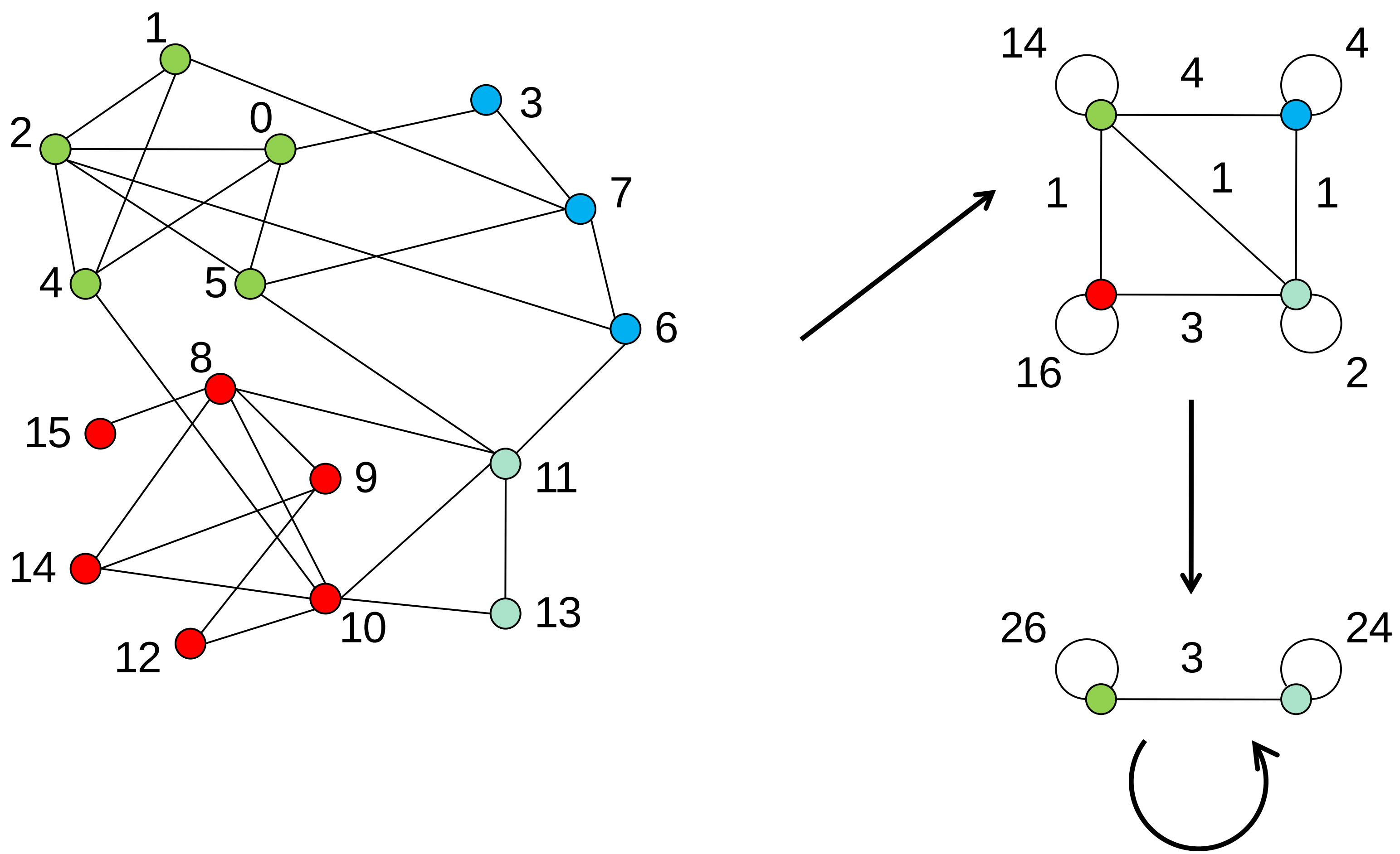
After 4 iterations, no more changes

An example





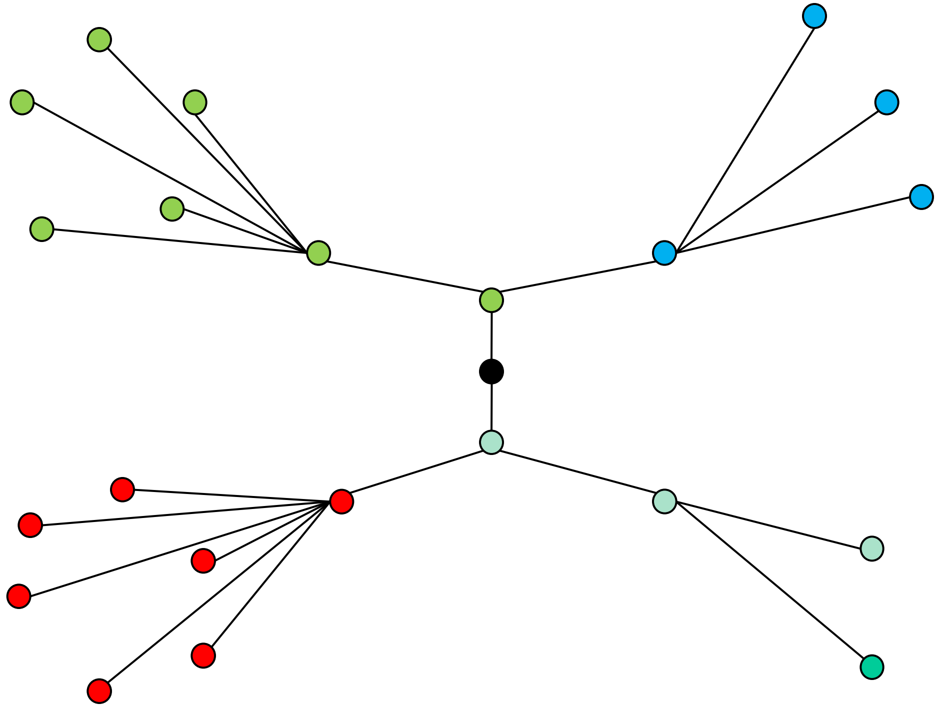
An example



An example

Gives a tree (not a binary one):

- each level is meaningful
- intermediary levels are less subject to resolution problems



Modularity

The most widely accepted measure of quality:

$$Q = \frac{1}{2m}\sum_C \left[e_C - \frac{a_C^2}{2m} \right]$$

Links with an extremity in C

Links inside C

Contribution of an isolated node is:

$$Q(i) = -\left(\frac{k_i}{2m} \right)^2$$

Degree of i

Moving a node

An isolated node 'i' can be moved to C with a gain:

$$\Delta Q(C,i) = \left[\frac{e_C + k_{i,C}}{2m} - \left(\frac{a_C + k_i}{2m} \right)^2 \right] - \left[\frac{e_C}{2m} - \left(\frac{a_C}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

Links from i to C

Only related to i and C

Complexity linear with degree of i

Experimental results (time)

	Karate	Arxiv	Internet	Web nd.edu	Belgian Phone Calls	Web UK-2005	Web Webbase01
	n=34/m=77	9k/24k	70k/351k	325k/1M	2.5M/6.3M	39M / 783M	118M/1B
Newman Girvan Clauset Moore	0s	3.6s	799s	5034s			
Pons Latapy	0s	3.3s	575s	6666s			
Wakita Tsurumi (expected)	0s	0s	8s	52s	1279s	(3days)	
Our approach	**0s**	**0s**	**<1s**	**<1s**	**47s**	**252s**	**469s**
	3 passes	5 passes	5 passes	5 passes	5 passes	4 passes	5 passes

Experimental results (Q)

	Karate	Arxiv	Internet	Web nd.edu	Belgian Phone Calls	Web UK-2005	Web Webbase01
	n=34/m=77	9k/24k	70k/351k	325k/1M	2.5M/6.3M	39M / 783M	118M/1B
Newman Girvan Clauset Moore	0s 0.38	3.6s 0.772	799s 0.692	5034s 0.927			
Pons Latapy	0s 0.42	3.3s 0.757	575s 0.729	6666s 0.895			
Wakita Tsurumi (expected)	0s	0s	8s	52s	1279s	(3days)	
Our approach	0s **0.42**	0s **0.813**	<1s **0.781**	<1s **0.935**	47s **0.769**	252s **0.979**	469s **0.984**
	3 passes	5 passes	5 passes	5 passes	5 passes	4 passes	5 passes

Evolving communities

Louvain is fine for static networks, but most are evolving

- new pages/sites appear on the web, people begin new relationship, posts are created on blogs, etc.

Basic idea

- succession of static networks + efficient techniques for static networks = compute communities at each time step

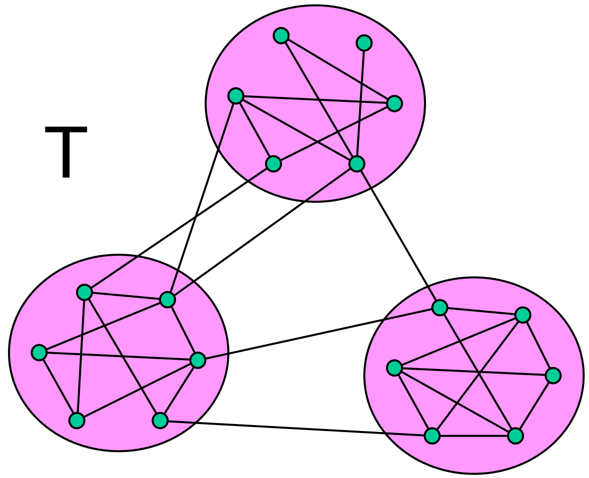
Evolving communities

Louvain is fine for static networks, but most are evolving

new pages/sites appear on the web, people begin new relationship, posts are created on blogs, etc.

Basic idea

succession of static networks + efficient techniques for static networks = compute communities at each time step



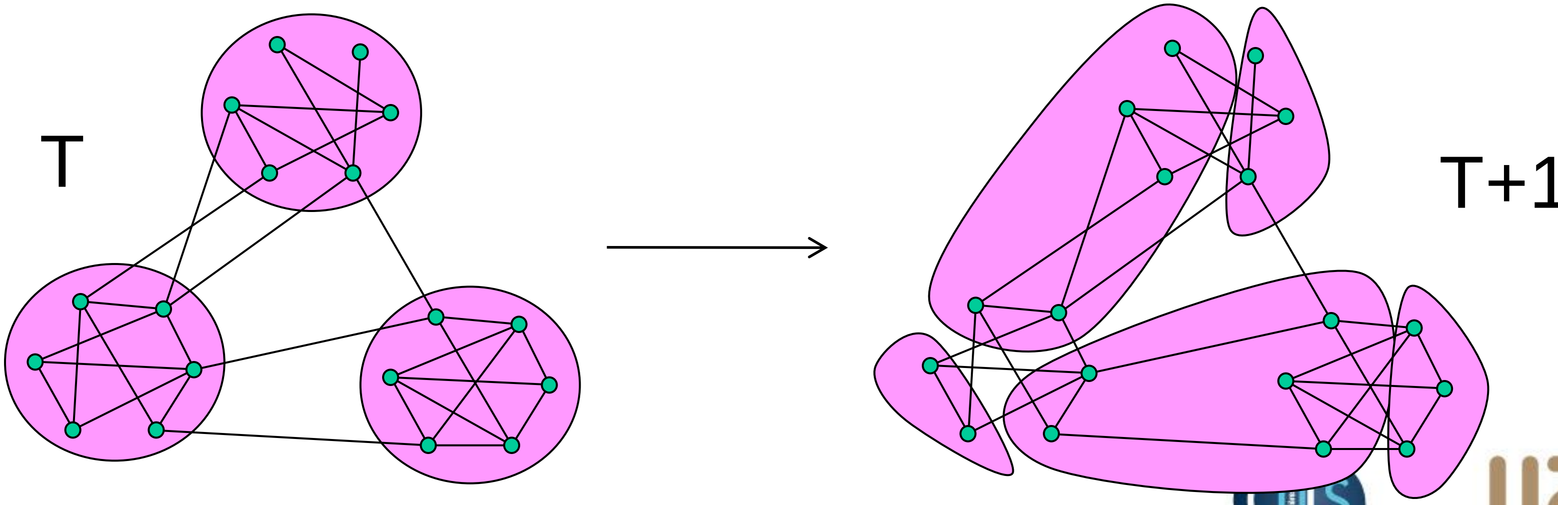
Evolving communities

Louvain is fine for static networks, but most are evolving

- new pages/sites appear on the web, people begin new relationship, posts are created on blogs, etc.

Basic idea

- succession of static networks + efficient techniques for static networks = compute communities at each time step



Evolving communities

Louvain is fine for static networks, but most are evolving

- new pages/sites appear on the web, people begin new relationship, posts are created on blogs, etc.

Basic idea

- succession of static networks + efficient techniques for static networks = compute communities at each time step

Outline

Static community detection

- Louvain algorithm
- experimental results

Dynamic communities

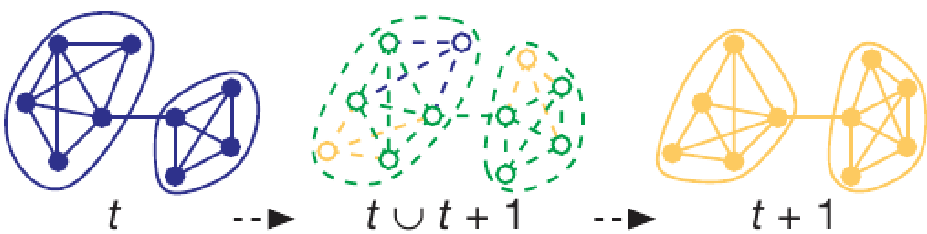
- past studies
- stability issues

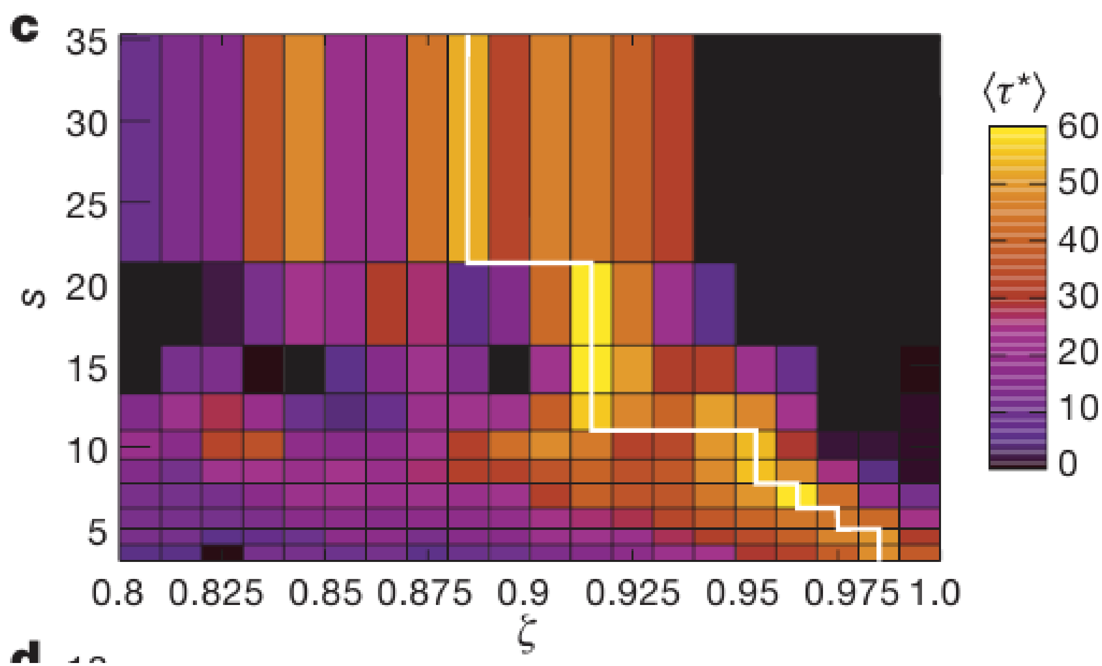
Past studies

Find a definition of community which facilitates the matching

overlapping cliques

Palla, Barabasi and Vicsek, Nature 2007





Past studies

Find a definition of community which allows to track them

Only focus on very stable communities

- make random perturbations of the graph (remove 5% of nodes)
- search for communities not affected by the perturbation
- these communities should be less affected by real modifications

Hopcroft, Khan, Kulis and Selman, PNAS 2004

Past studies

Find a definition of community which allows to track them

Only focus on very stable communities

Temporal network

- build a network with temporal links between static networks
- use any fast algorithm for static networks
- gives temporal communities

Jdidia, Robardet and Fleury, ICDIM 2007

Past studies

Find a definition of community which allows to track them

Only focus on very stable communities

Temporal network

Modify the quality function

- Quality = function($Q_{snapshot}$, $Q_{dynamic}$)
- $Q_{dynamic}$: ensure stability or good partitioning at different time steps

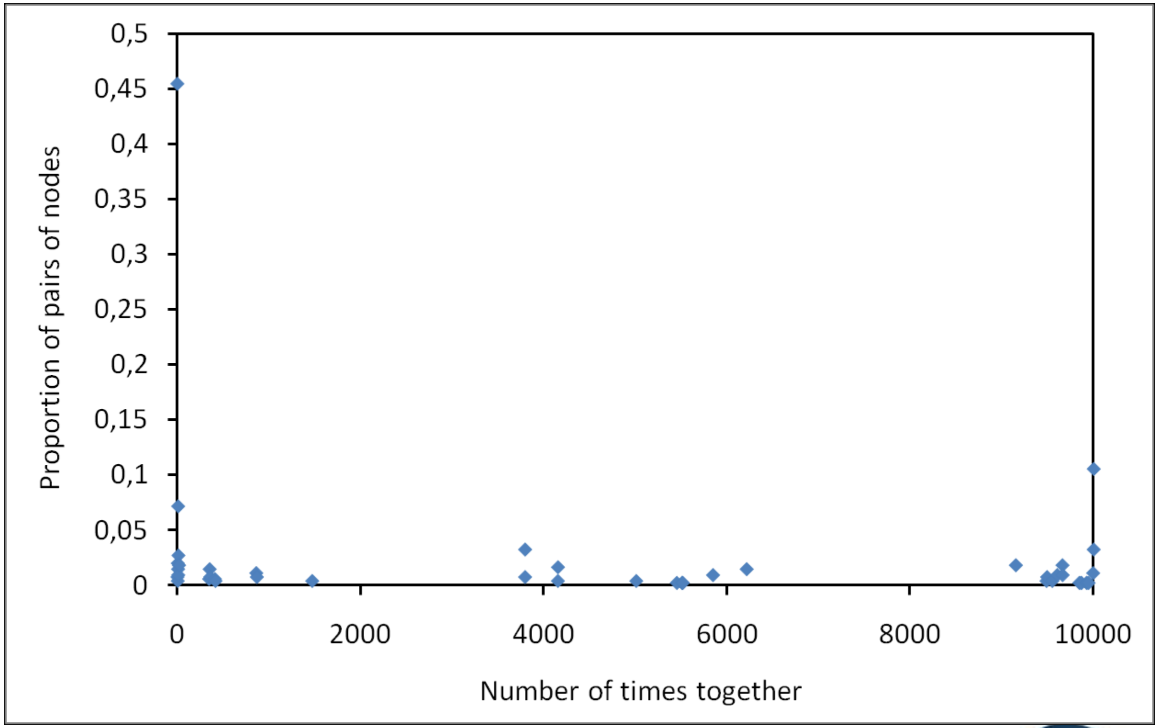
D. Chakrabarti, R. Kumar, and A. Tomkins, SIGKDD 2006.

Y. Lin et al, Transactions on Knowledge Discovery from Data

Our approach

Non deterministic algorithms are not stable at all

- 10000 experiments on the same graph
- Look at pairs of nodes

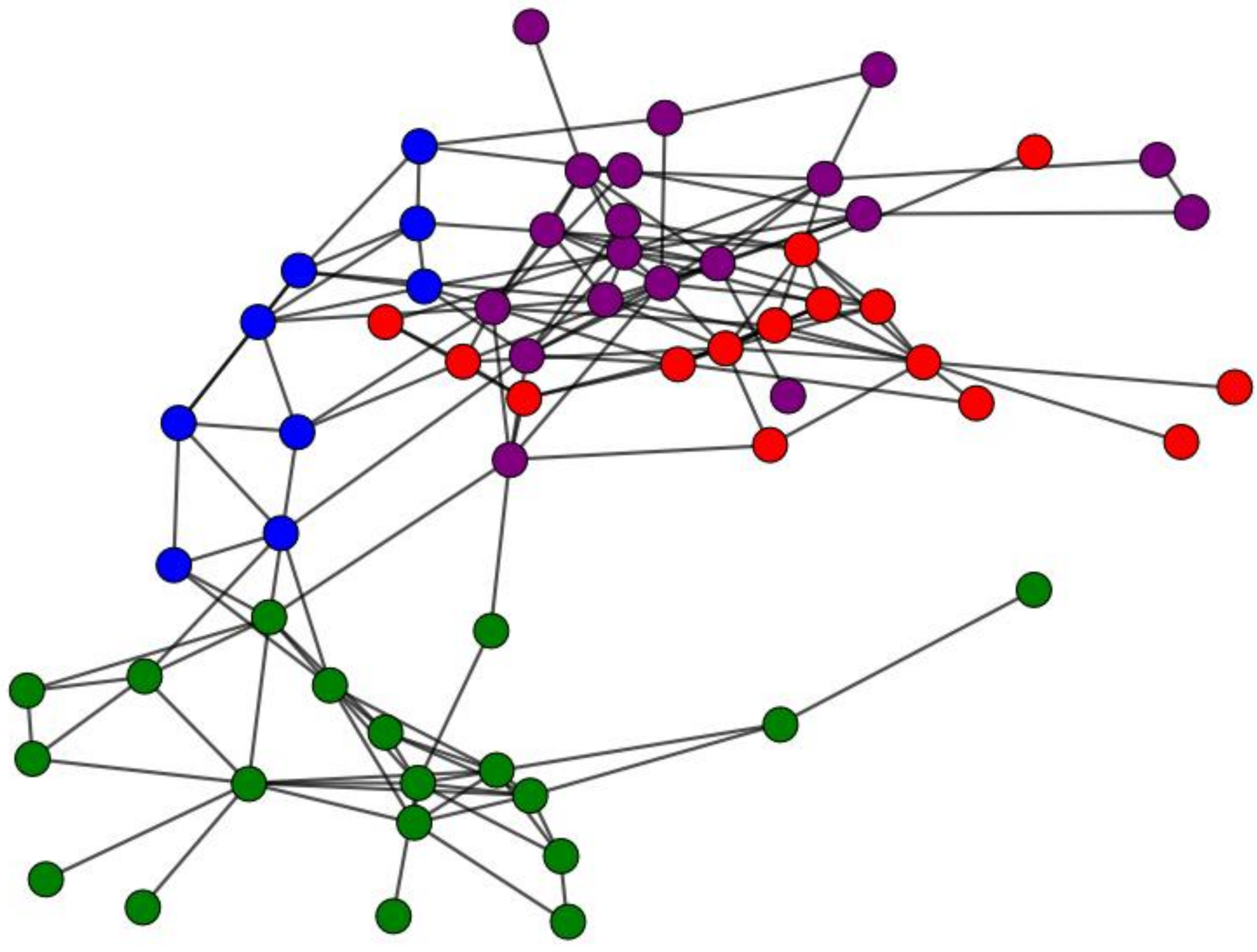


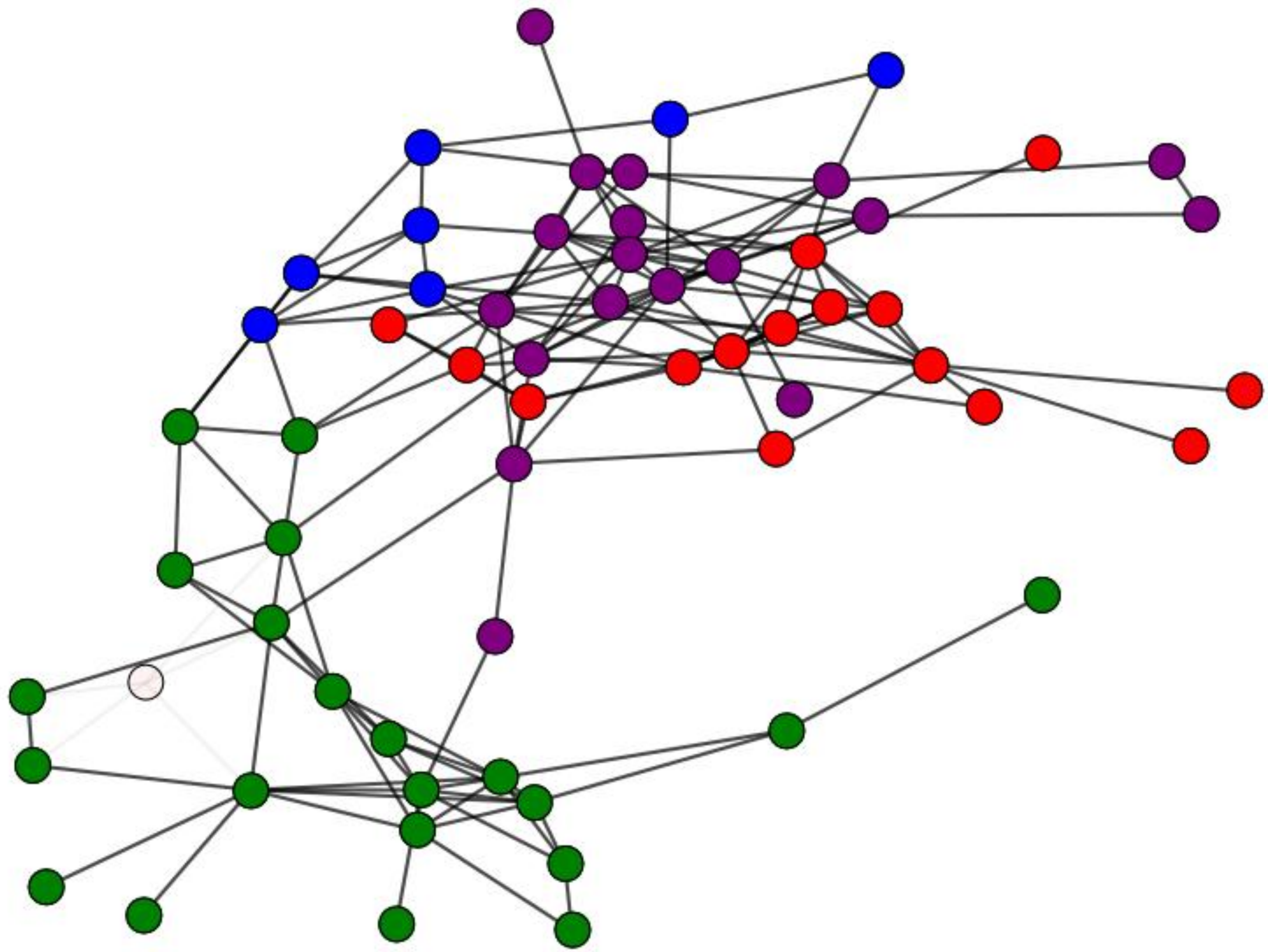
Our approach

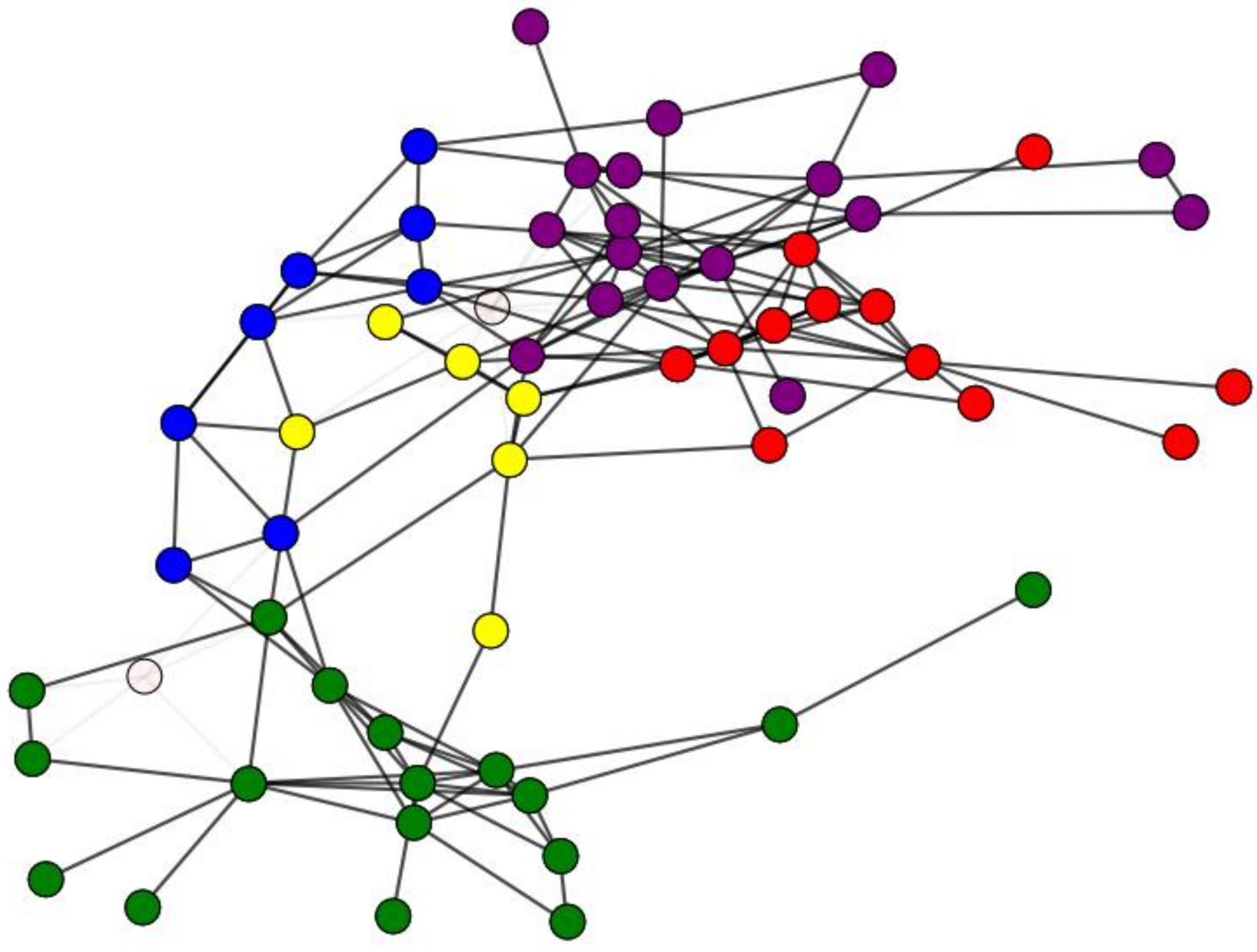
Non deterministic algorithms are not stable at all

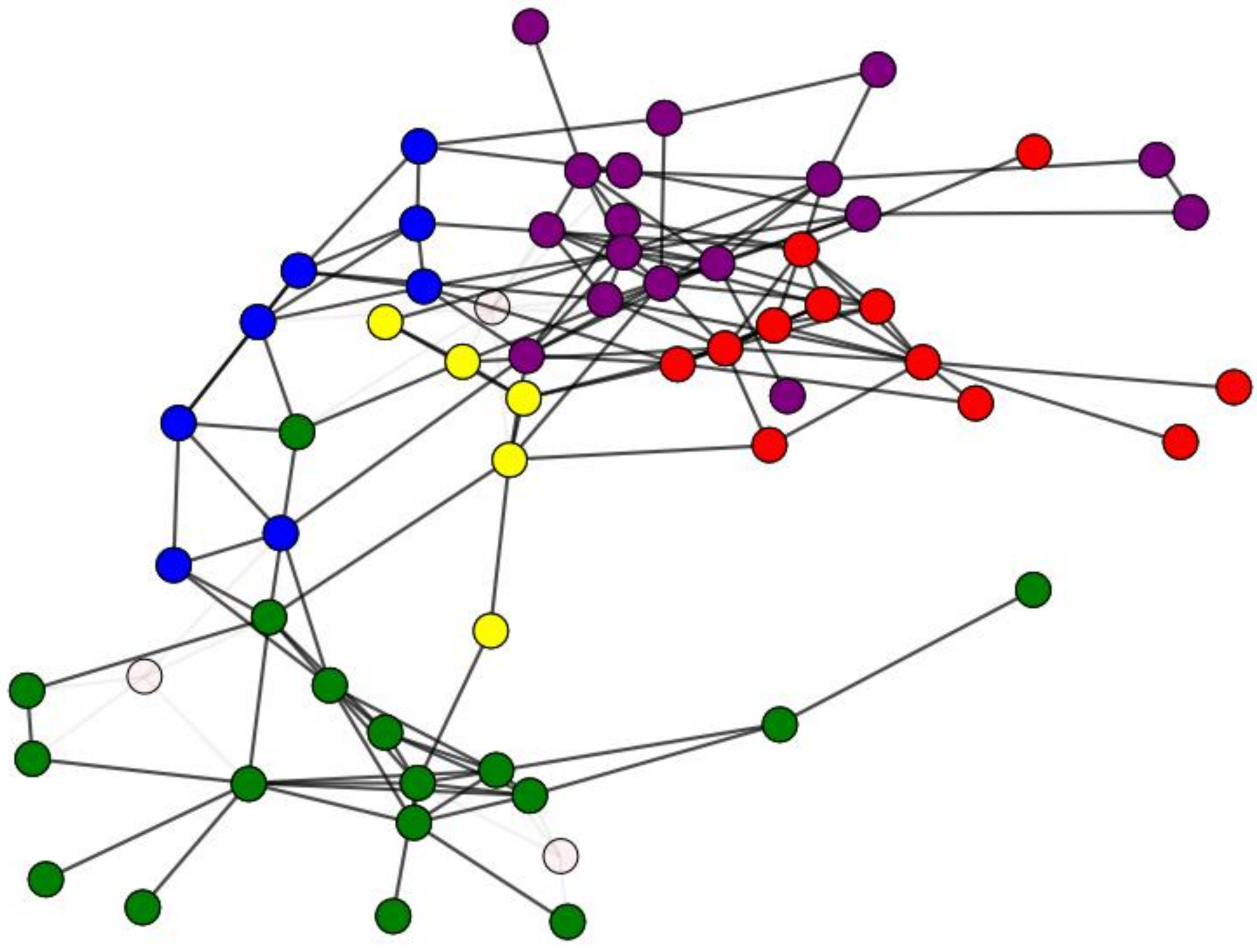
Use simple, well known, community detection algorithms
- Newman's fast: greedy optimization based on modularity
- Walktrap: greedy optimization based on random walks
- Louvain algorithm

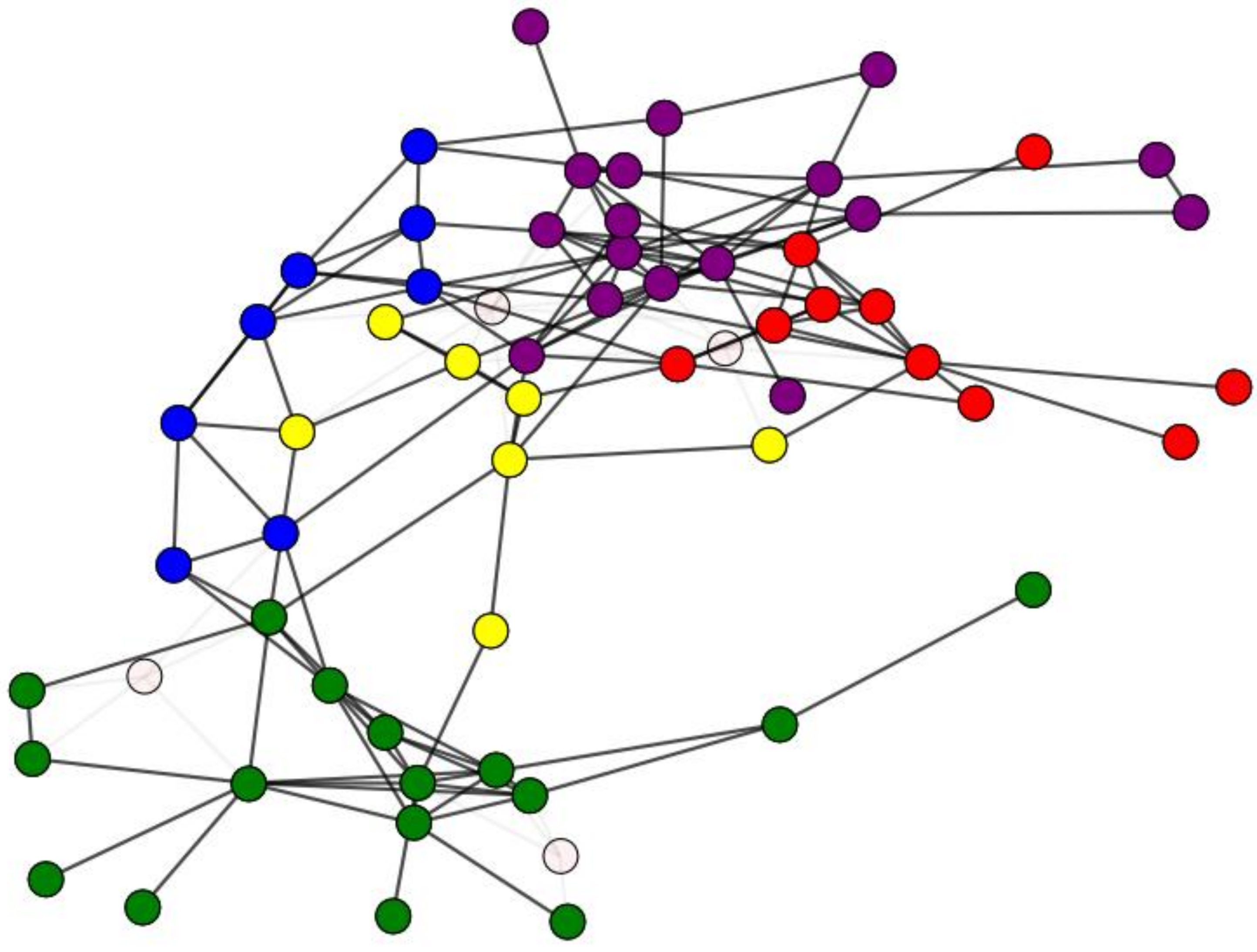
Simulate a basic evolution
- removal of random nodes from a network one by one
- computation of communities after each removal

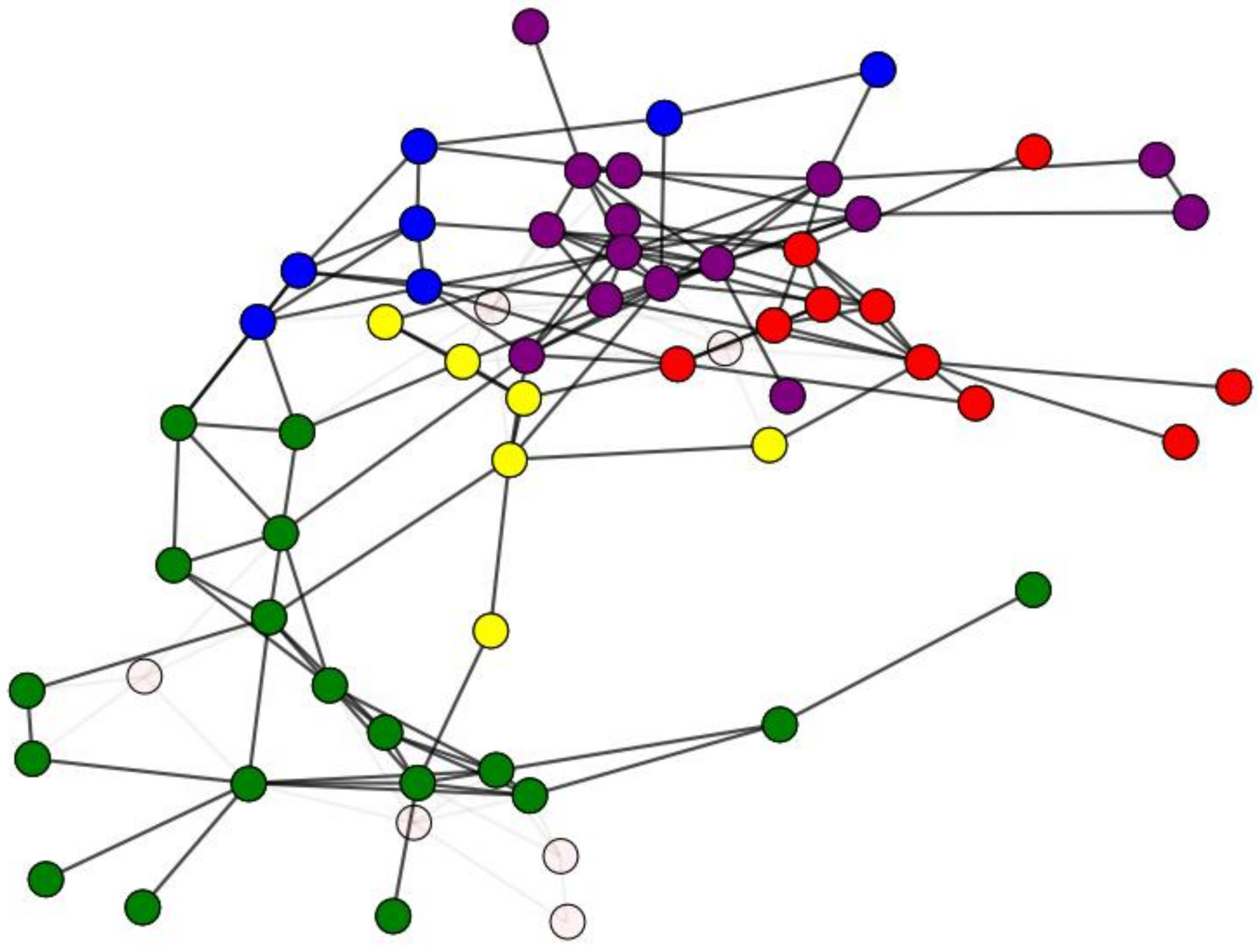


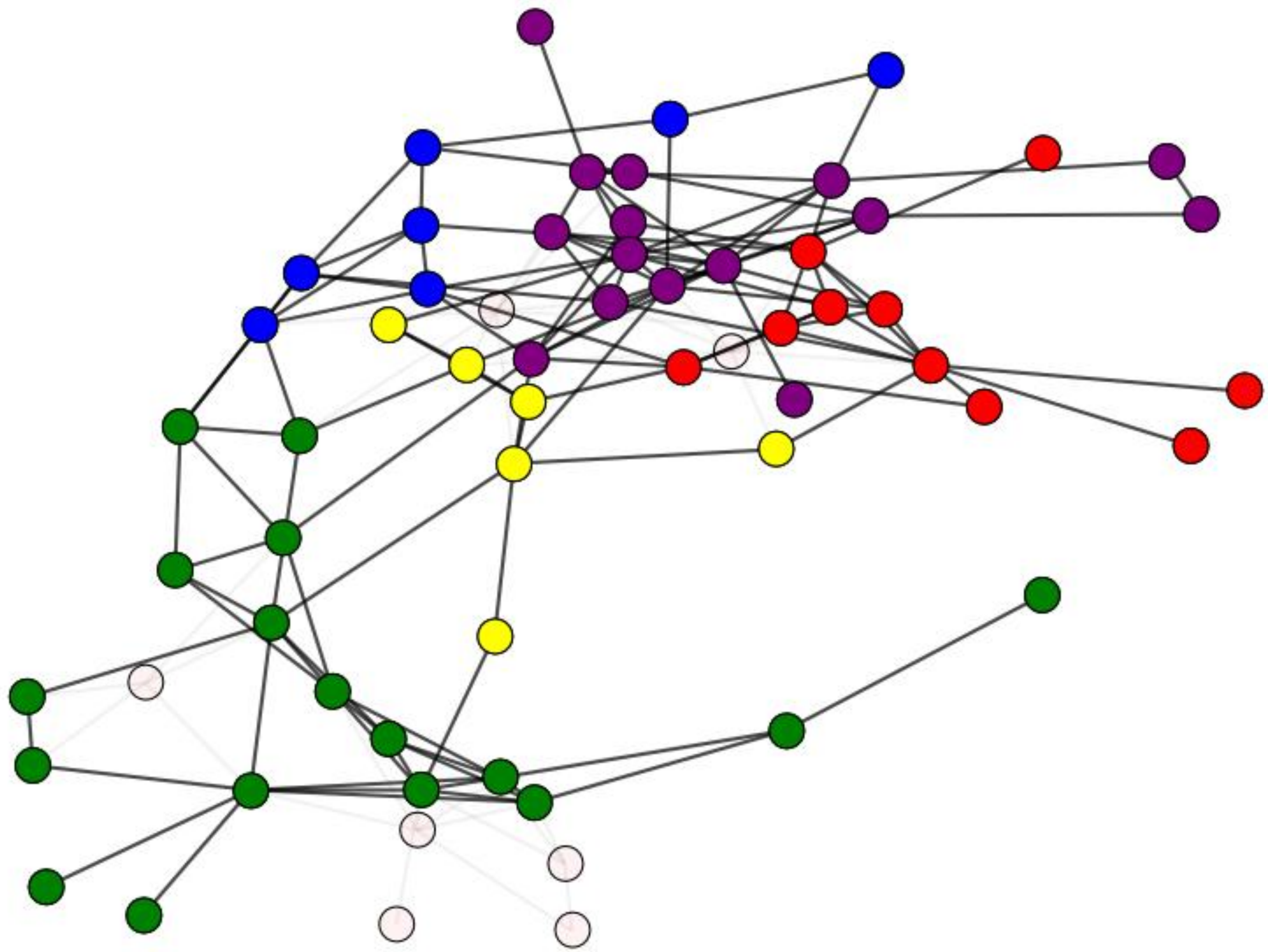










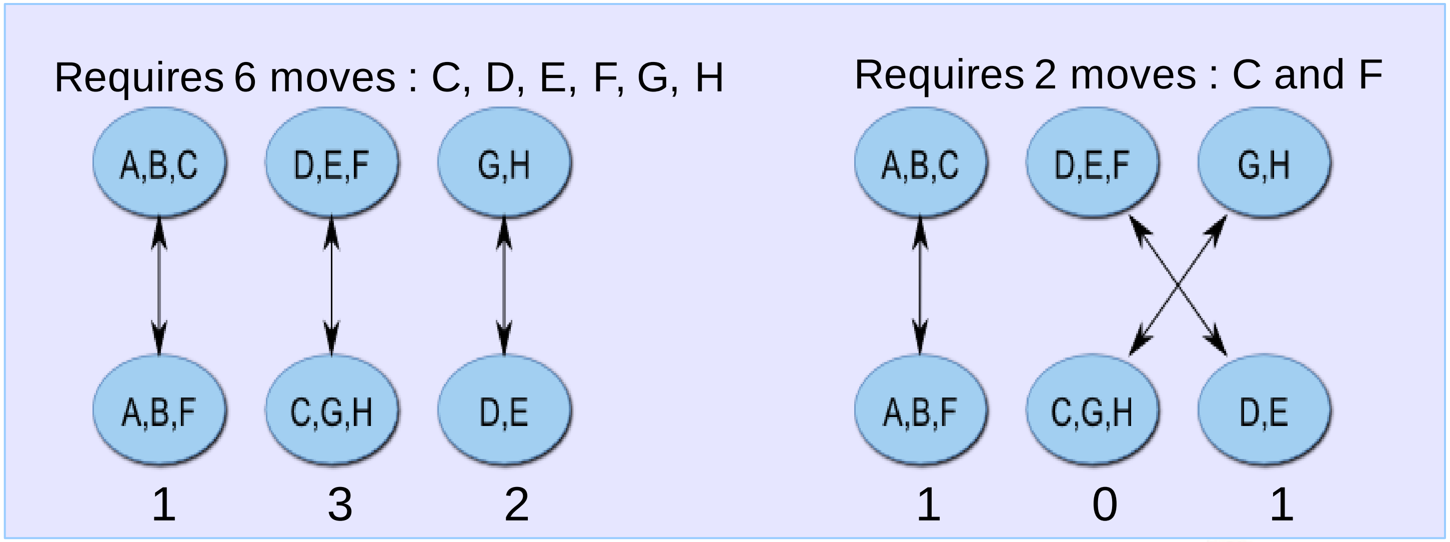


Evaluation – two parameters

Quality of the decomposition at each time step (modularity)

Stability of the algorithm

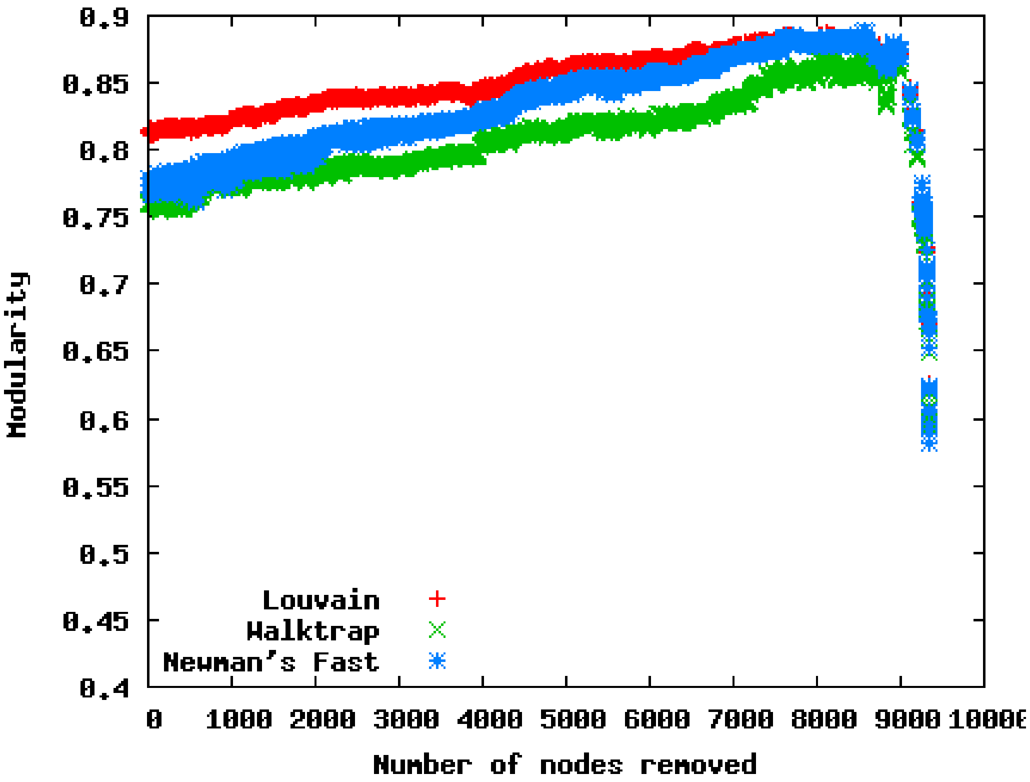
- minimal number of transformation to change one part in the other
- easier to interpret than mutual information

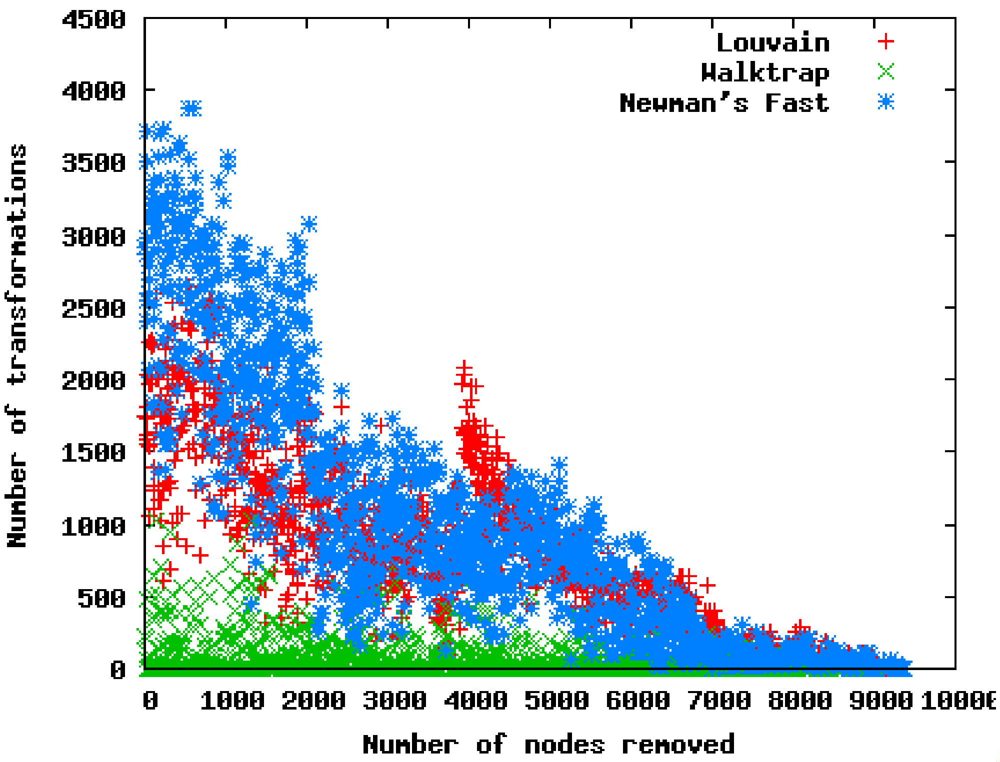


Quality / Stability

Louvain and Newman's are really unstable (9K nodes)

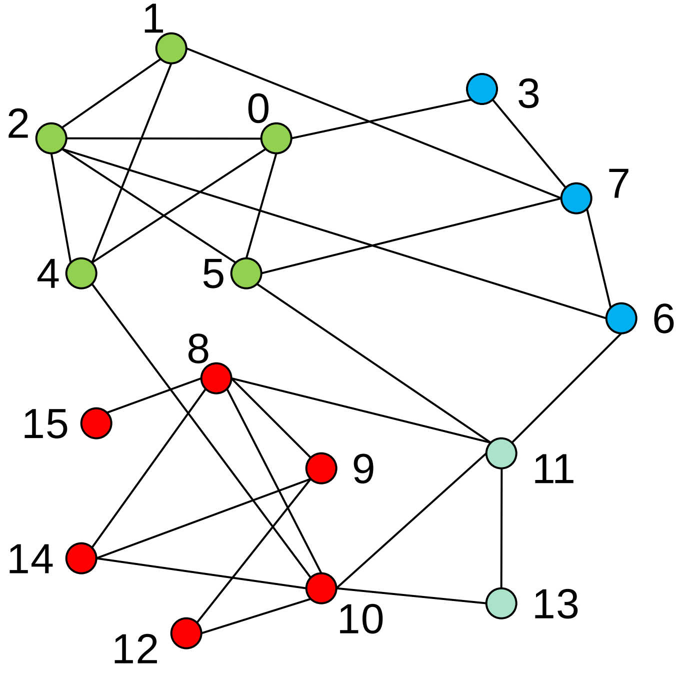
Walktrap is more stable but with lower quality





Stabilizing Louvain

Before modification: classical computation

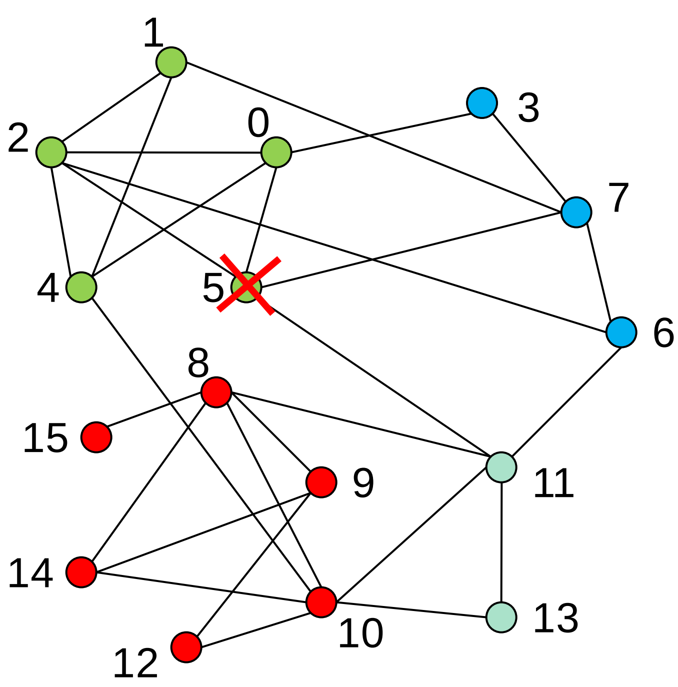


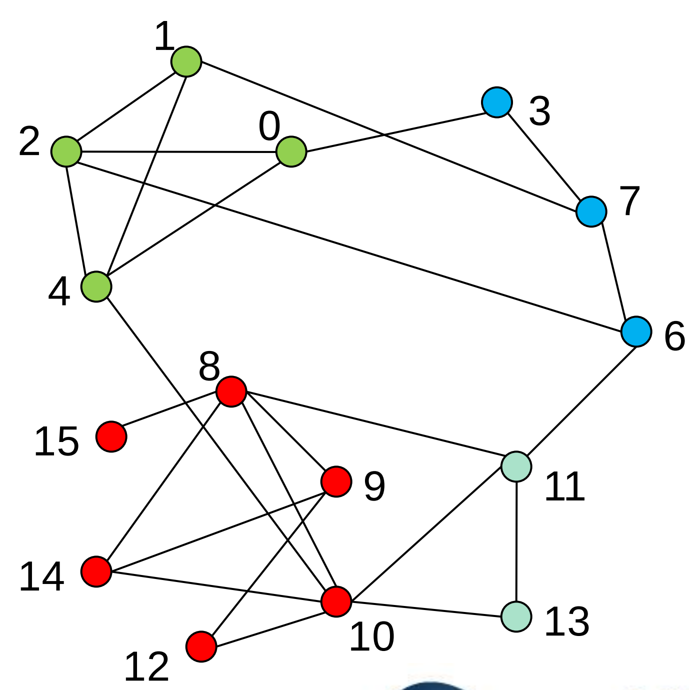
Stabilizing Louvain

Before modification: classical computation

After modification: initialize with the previous communities

launch the algorithm again

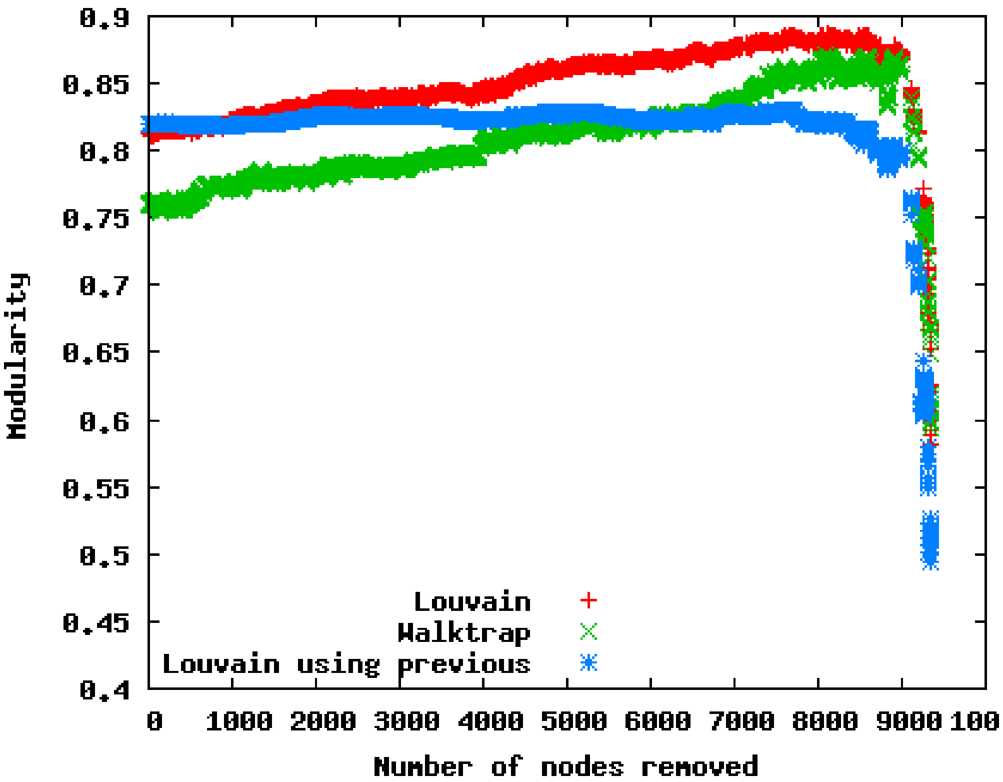


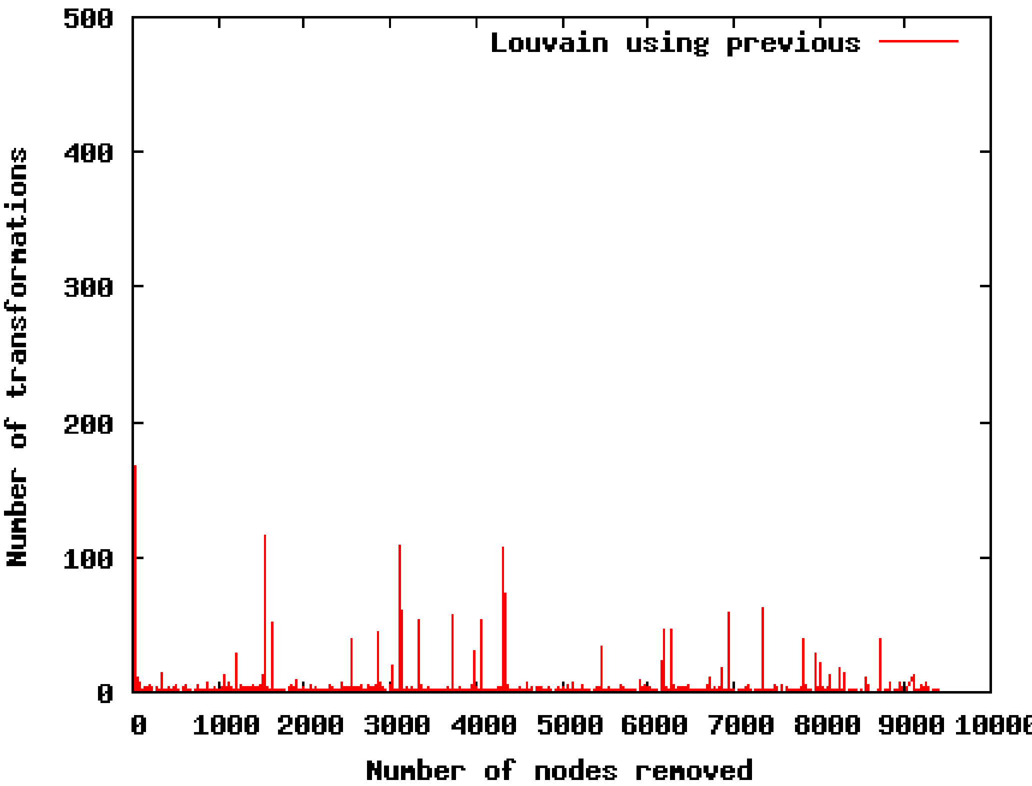


Quality / Stability

Tradeoff:

- loss of quality but still better than walktrap for a long period
- much more stable, with only few pikes

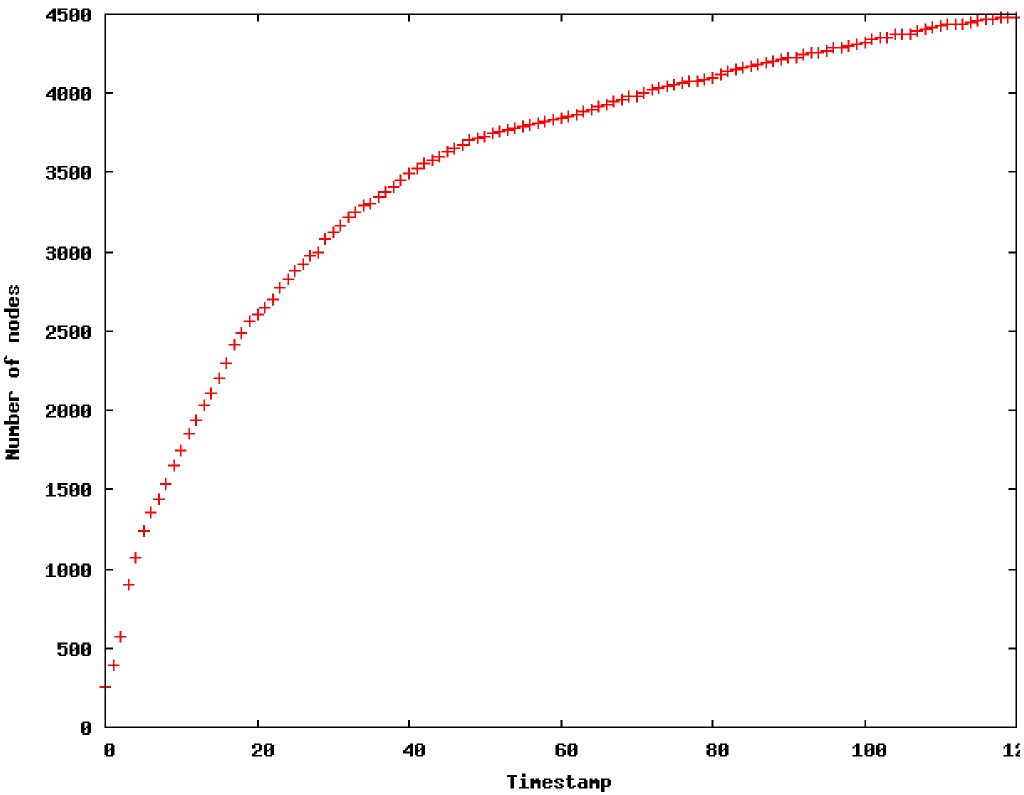


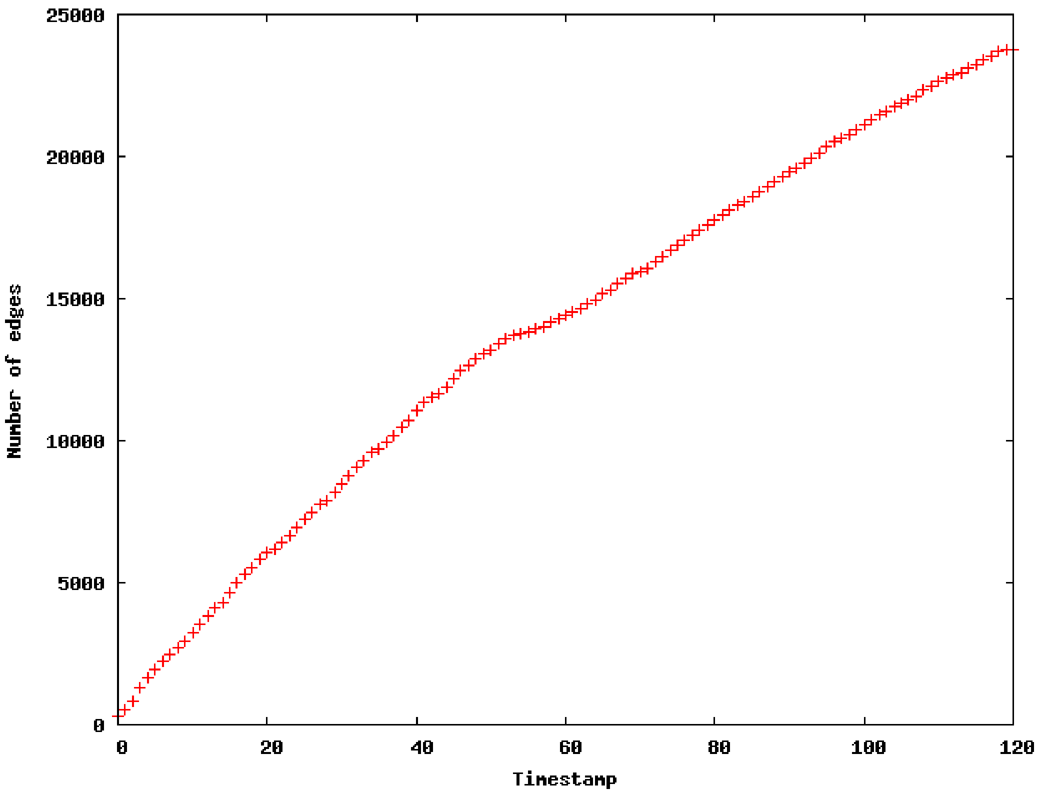


A real network

Blog network:

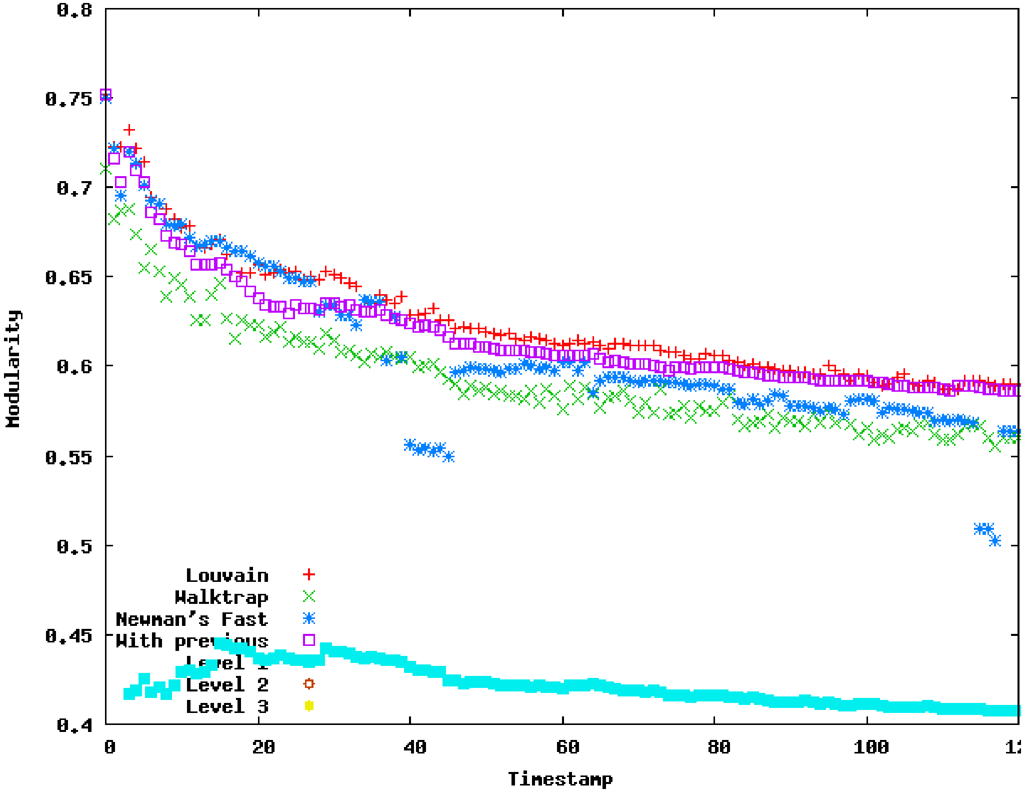
- hyperlinks between a set of blogs, measured everyday during 4 months
- real (more complex) evolution, but only growing

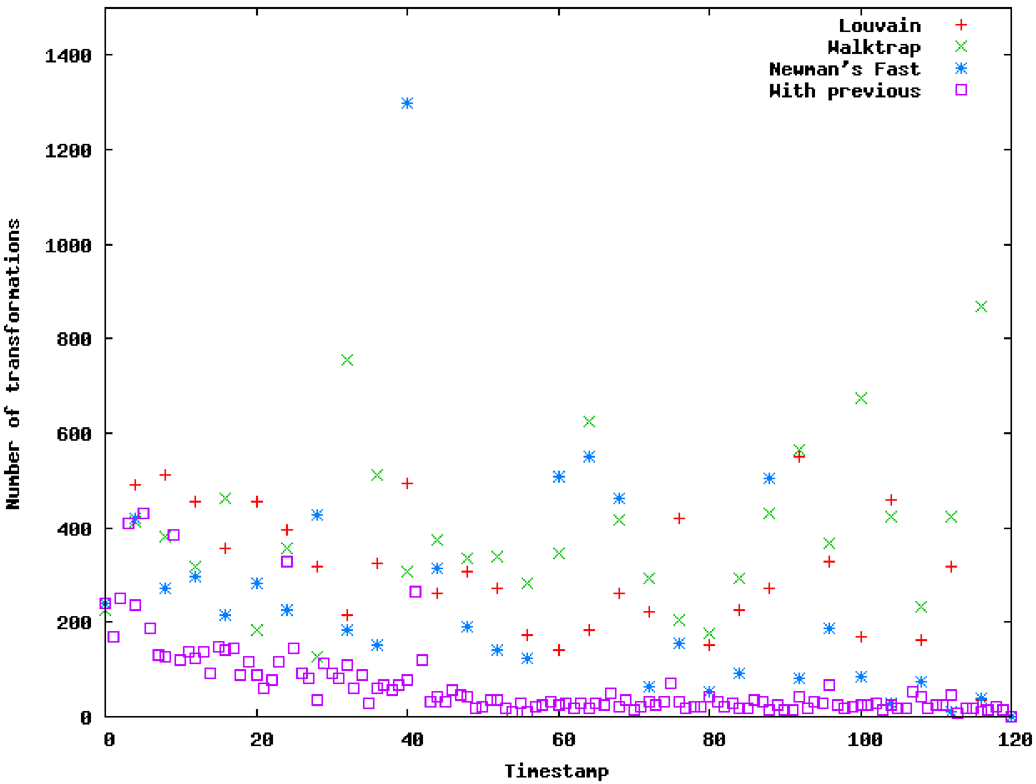




A real network (2)

- Results are very similar
  - still events but more stable compared to the number of modifications
  - Louvain stabilized performs well

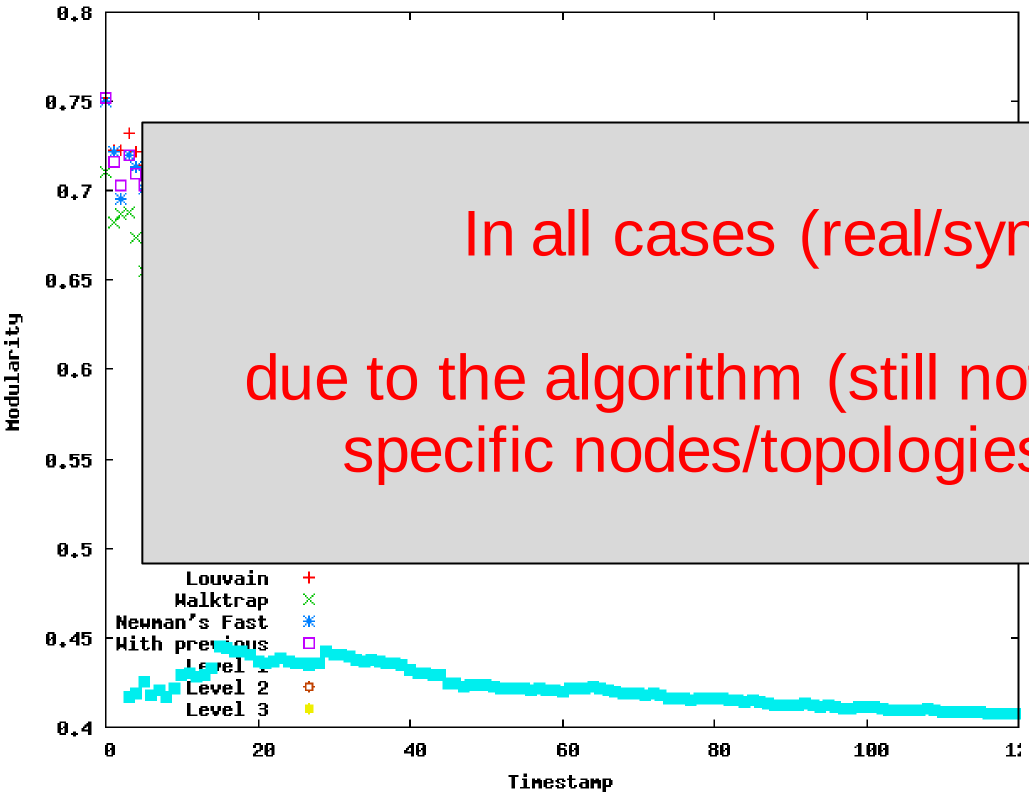


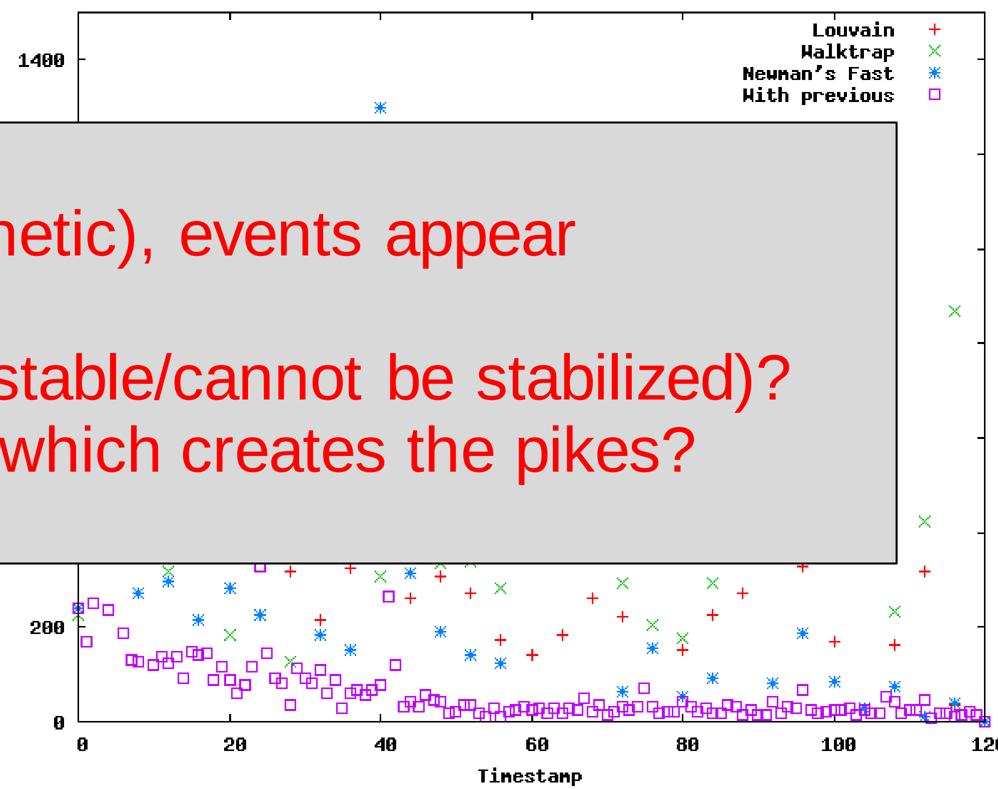


A real network (2)

Results are very similar

- still events but more stable compared to the number of modifications
- Louvain stabilized performs well





In all cases (real/synthetic), events appear

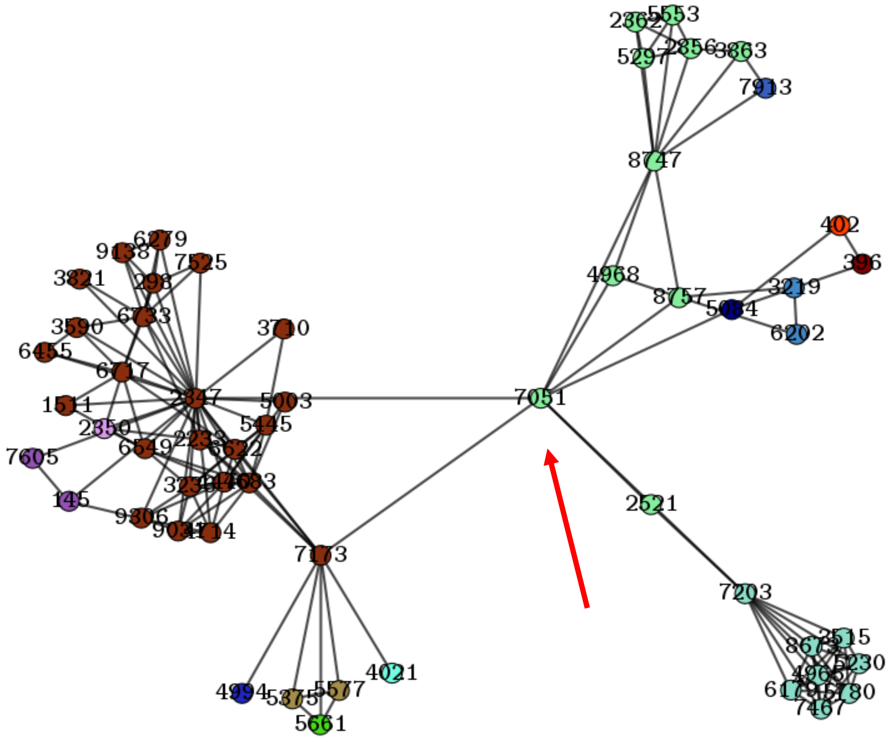
due to the algorithm (still not stable/cannot be stabilized)?

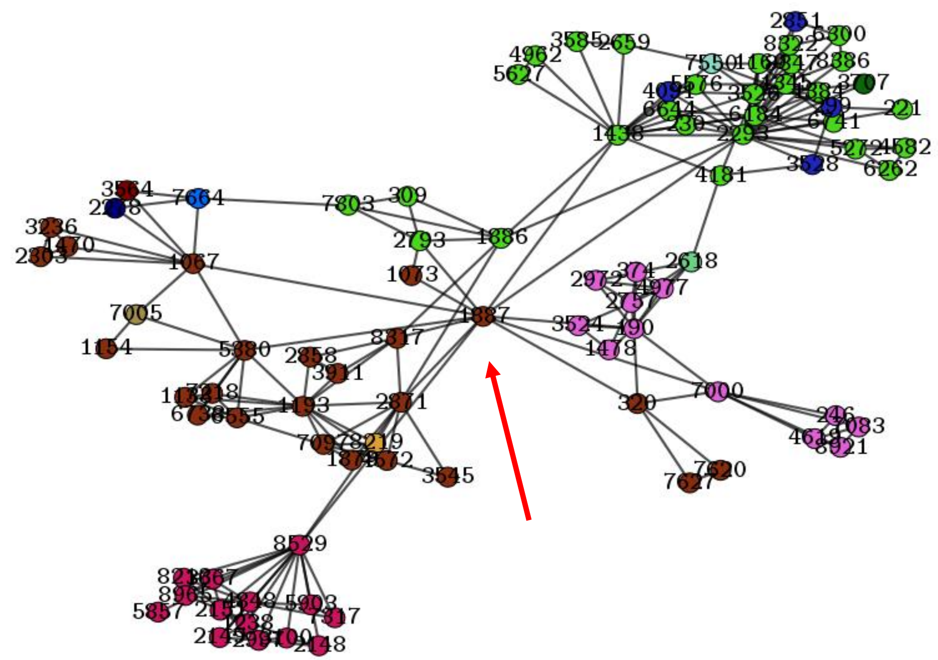
specific nodes/topologies which creates the pikes?

Understanding events

For each node of the network

- compute communities before and after removal (stabilized Louvain)
- number of modifications between before and after
- find topological properties which could explain the changes

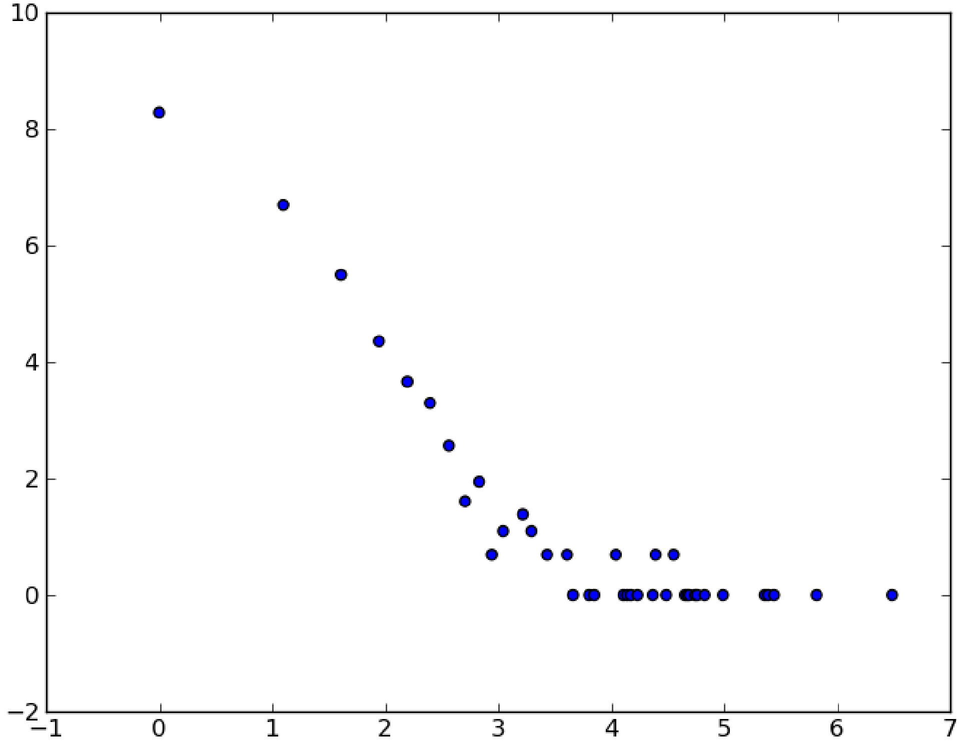




Number of modifications

Distribution is very heterogeneous

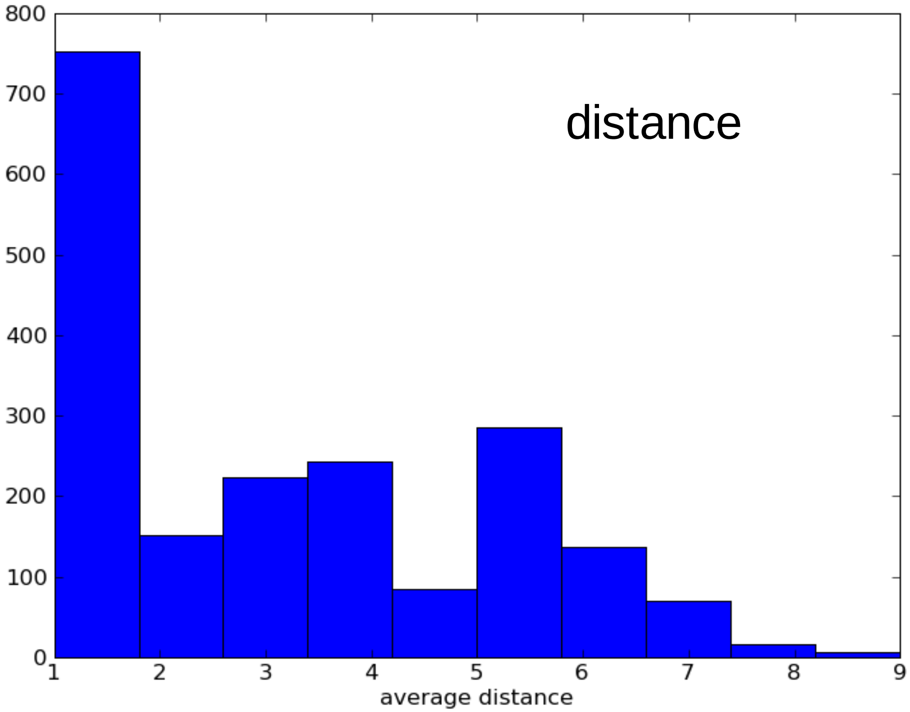
- log-log with exponent 2 (sorry for that)

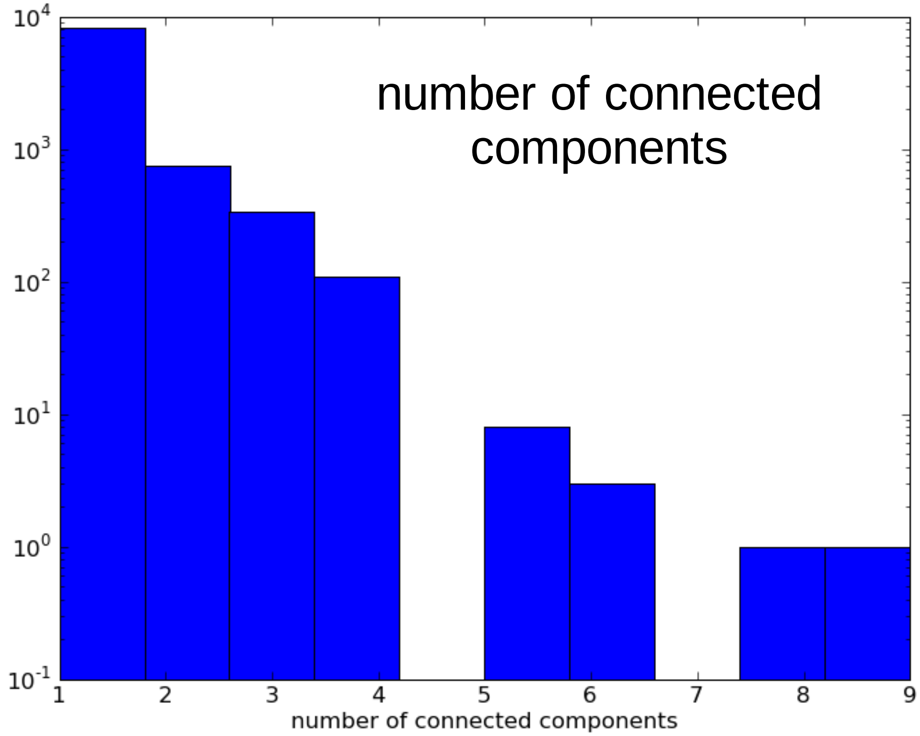


Which nodes are moving?

Close to the removed node in general

Few connected component involved

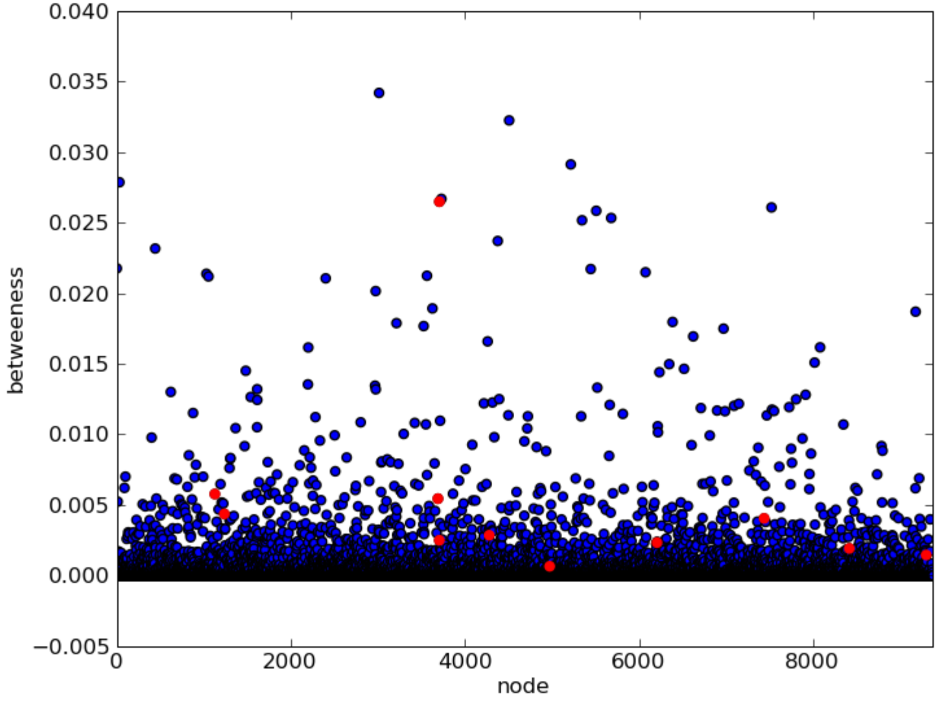


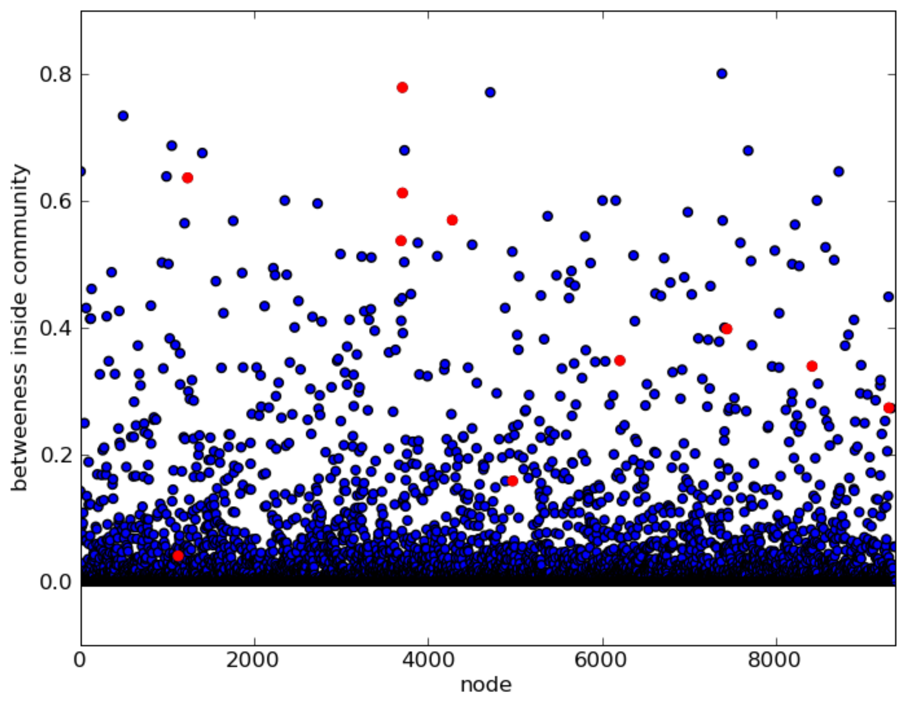


Betweeness centrality

Left: global, right: local

Local parameters are “more” relevant

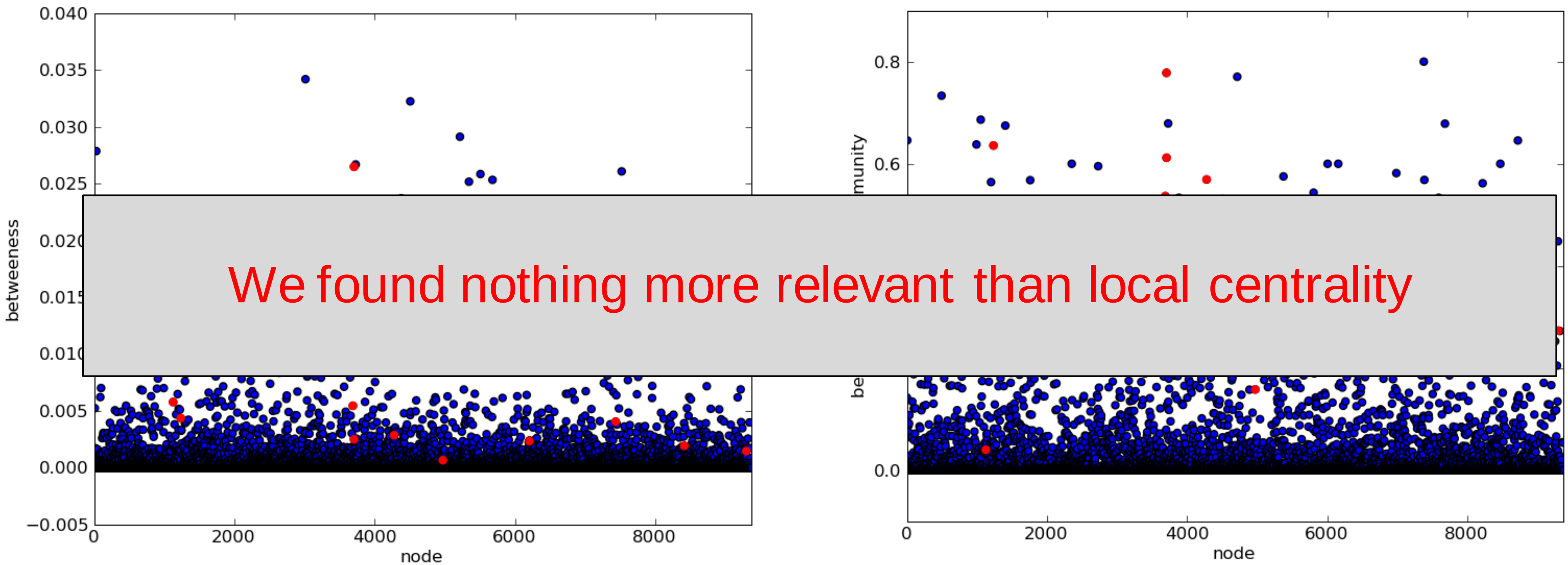




Betweeness centrality

Left: global, right: local

Local parameters are “more” relevant



We found nothing more relevant than local centrality

Conclusion / perspectives

Classic algorithms cannot be used directly to compute dynamic communities

- Louvain stabilized is much more stable than other algorithms
- are there too much constraints?
- loss of quality: restart if the quality decreases too much?

Locality is important

- moving nodes are near removed nodes
- impact is generally higher with local properties than global ones

Study more complex evolutions / real networks / algorithms