Fast unfolding of community hierarchies in large networks

V.D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre

Based on E. Lefebvre master's thesis
Paper available at: arXiv:0803.0476v1

Email: jean-loup.guillaume@lip6.fr

We propose

a modularity optimization algorithm which:

- gives excellent results for modularity;
- directly produces a hierarchy structure;
- is incredibly simple (local greedy approach);
- can work on external memory.

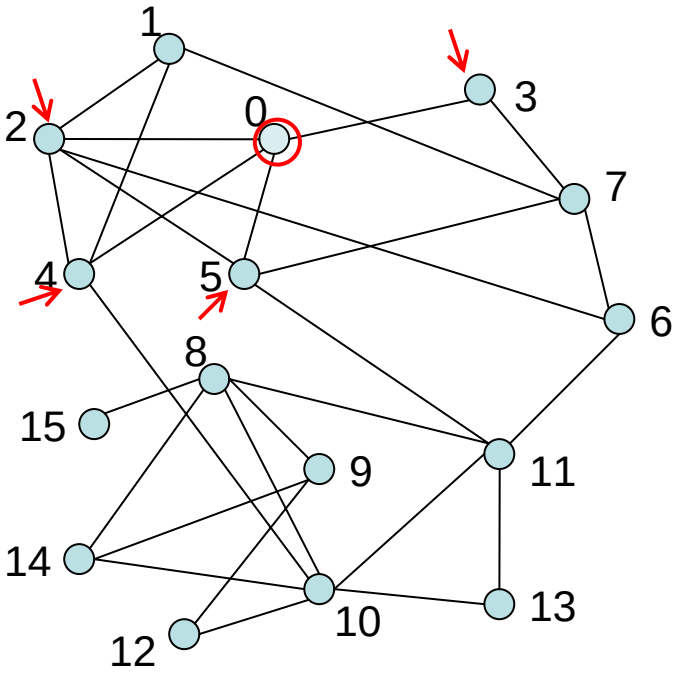
Can deal with millions nodes / billions links

e.g. 118M nodes/1B links in 152mn

Outline

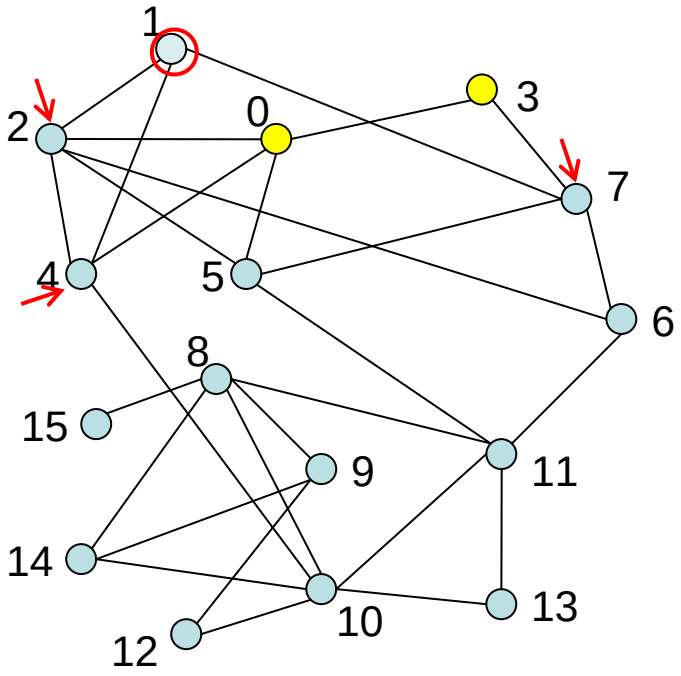
- The algorithm
- Experimental results
- Case study:
  - Belgian phone call network

An example



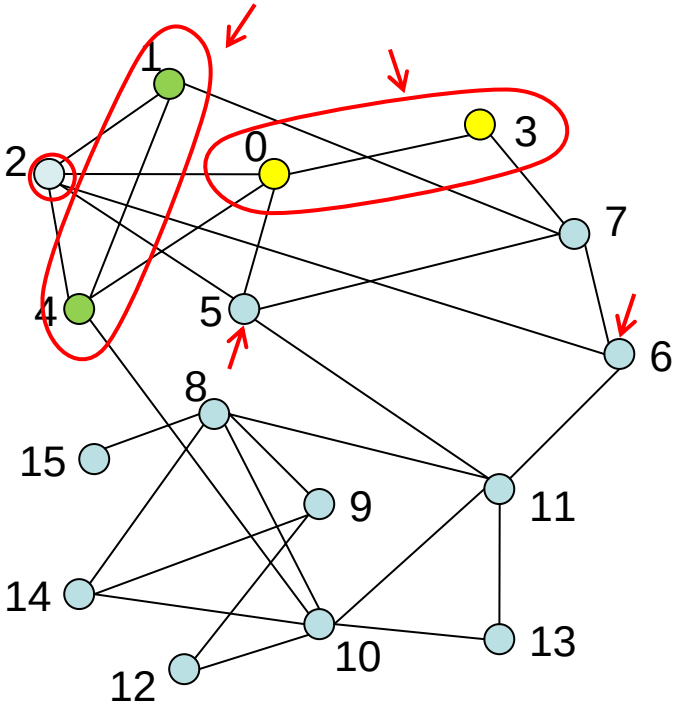
Pass 1 – Iteration 1
Each node belongs to an atomic community

An example



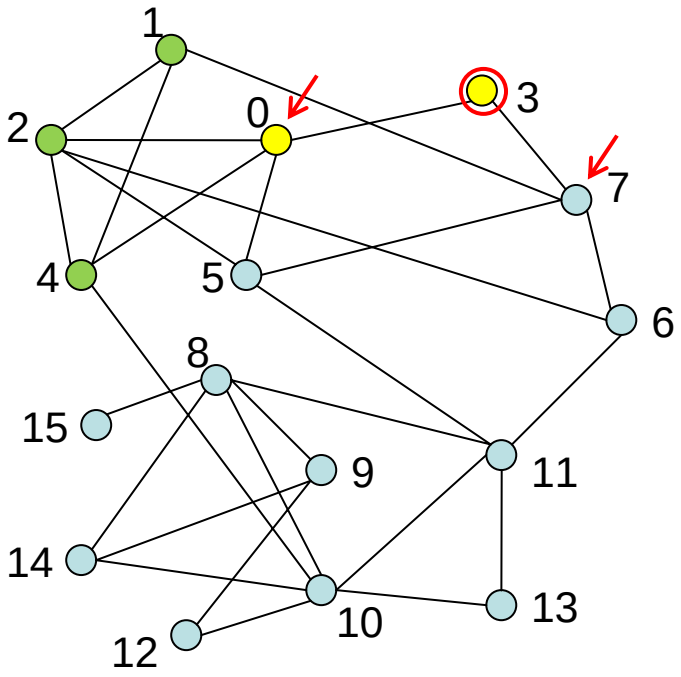
Pass 1 – Iteration 1
insert 0 in c[3]

An example



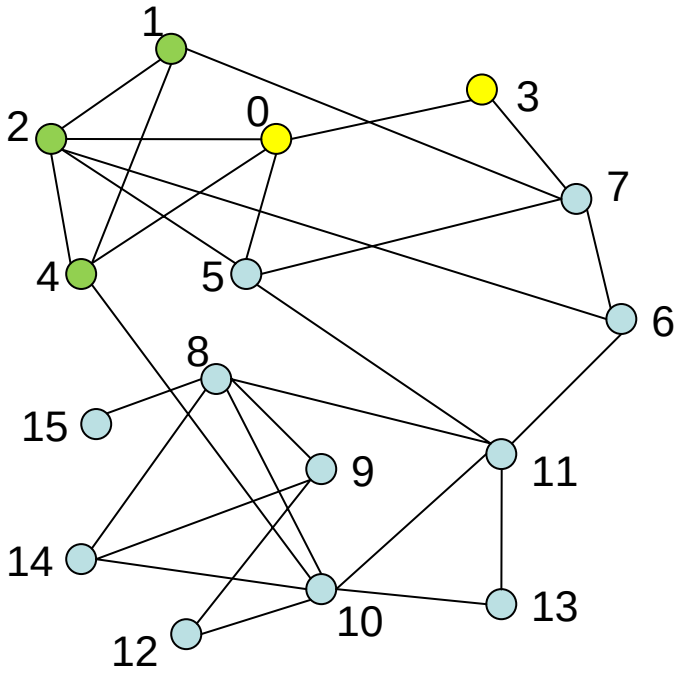
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]

An example



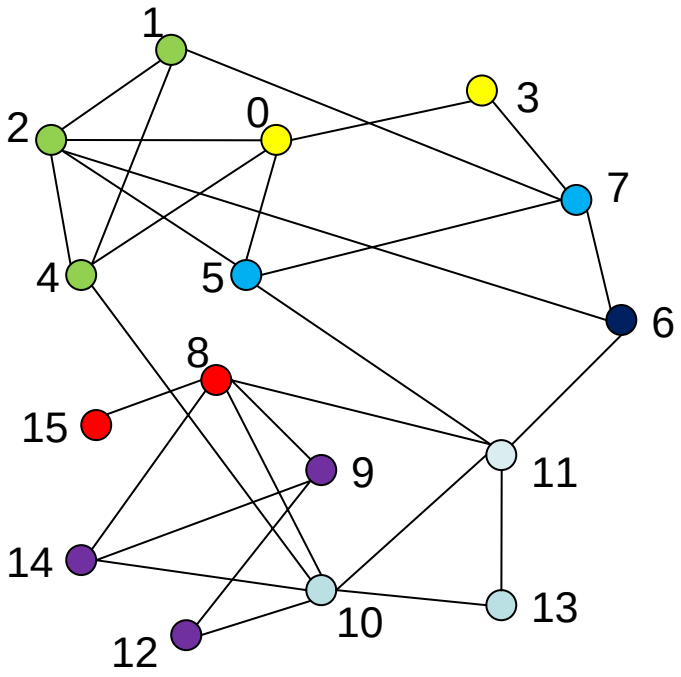
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]
insert 2 in c[1,4]

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]

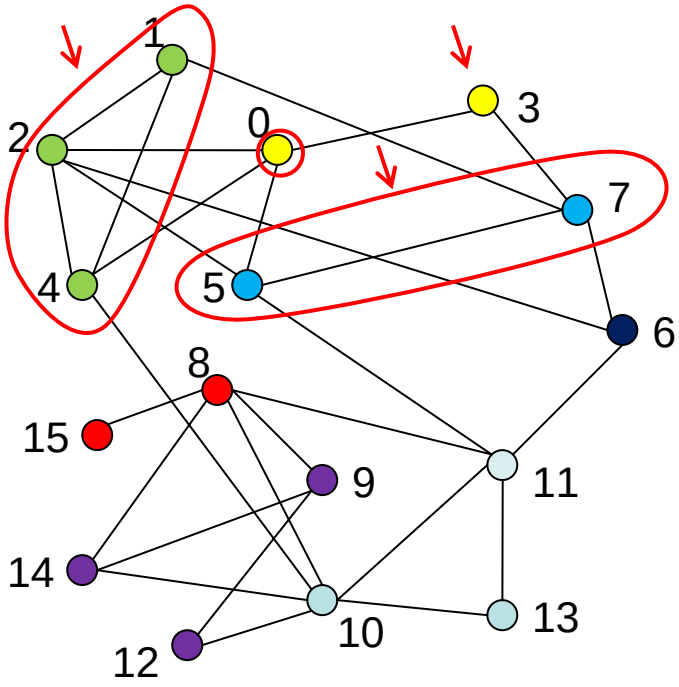
An example



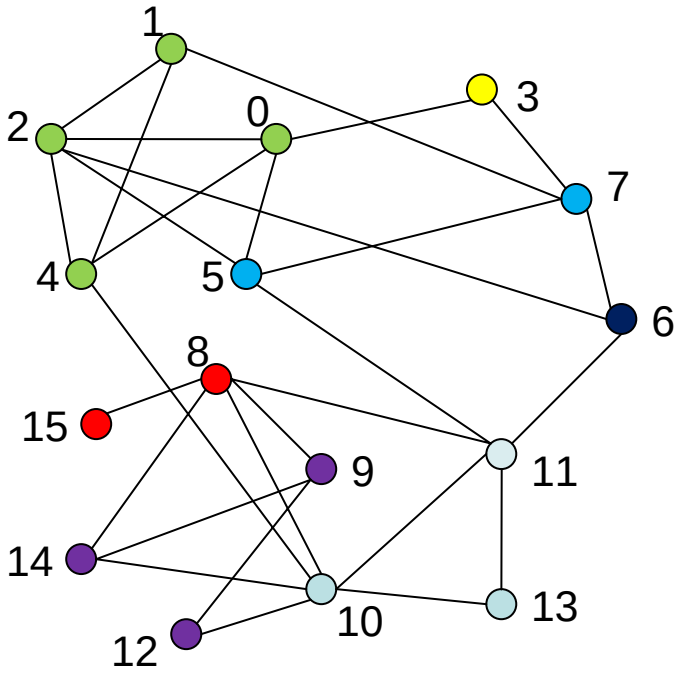
Pass 1 – Iteration 1
insert 0 in c[3]
insert 1 in c[4]
insert 2 in c[1,4]
insert 3 in c[0]
insert 4 in c[1]
insert 5 in c[7]
insert 6 in c[11]
insert 7 in c[5]
insert 8 in c[15]
insert 9 in c[12]
insert 10 in c[13]
insert 11 in c[10,13]
insert 12 in c[9]
insert 13 in c[10,11]
insert 14 in c[9,12]
insert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

An example

Pass 1 – Iteration 2

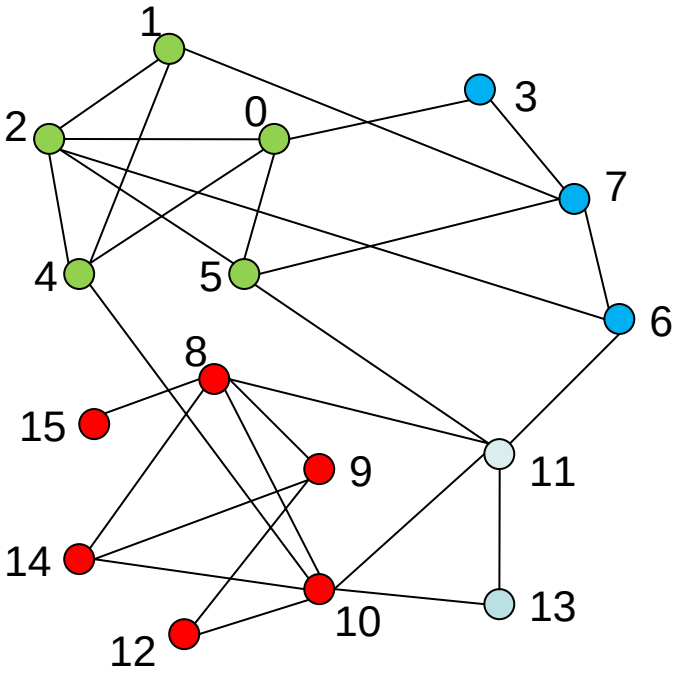


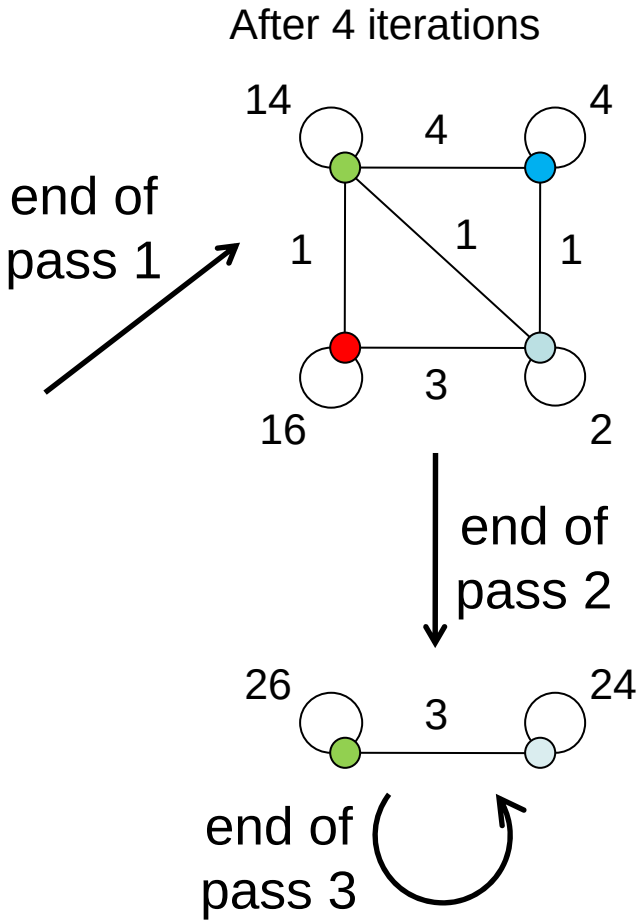
An example



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

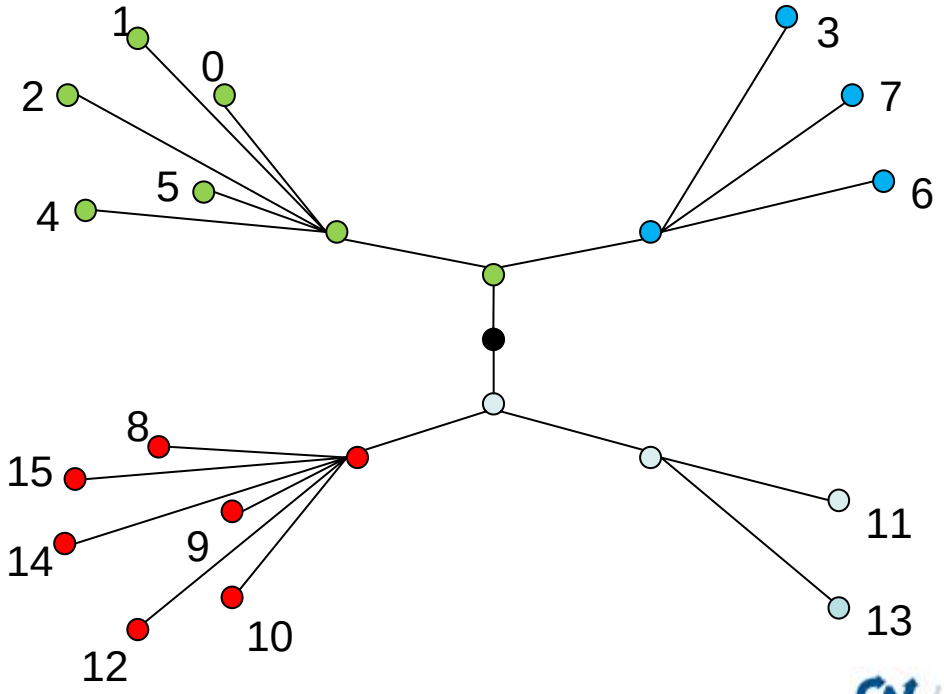
An example


1
2
0
3
7
4
5
6
8
15
9
11
14
13
12
10


After 4 iterations
end of
pass 1
14
4
4
1
1
1
16
3
2
end of
pass 2
26
3
24
end of
pass 3

An example

- Gives a tree (not a binary one):
  - each level is meaningful.



The algorithm formally

Sequence of passes:

- each pass computes one hierarchy level;
- input: (weighted) network;
- output: weighted network where nodes are “communities” of the original network;

- passes are applied recursively;
- stop when modularity cannot be increased.

The algorithm formally

One pass:

- initially each node forms a community;
- repeat iteratively for all nodes i:
  - remove i from its community;
  - insert i in a neighboring community of i so as to maximize modularity (local greedy approach);
- stop when a local maximum is attained.

Outline

- The algorithm
- Experimental results
- Case study:
  - Belgian phone call network

Experimental results

- High level networks are smaller:
  - first passes are the only costly ones;
  - in general 1st pass > 90% of computation time.
- There are few iterations for each pass:
  - only iterations on the first passes are costly;
  - <33 for all tested networks.
- Considering one node is simple.

Modularity

- A widely accepted measure:

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

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 k_i

One pass algorithm

```
Input: a (weighted) network
Variables: e, a, comm

for all nodes i do
   insert i in an atomic community (comm[i]=i)
   initialize e and a
while there is an increase of modularity do
   for all nodes i do
        remove(e,a) i from comm[i]
        compute DeltaQ(C,i,e,a) for all C in neigh_comm(i)
        insert(e,a) i in argmax(DeltaQ(C,i))

Output: weighted community graph
```

Experimental results (time)

	Karate	Arxiv	Internet	Web nd.edu	Belgian Phone Calls
	n=34/m=77	9k/24k	70k/351k	325k/1M	2.5M/6.3M
Newman Girvan Clauset Moore	0s	3.6s	799s	5034s	
Pons Latapy	0s	3.3s	575s	6666s	
Wakita Tsurumi (expected)	0s	0s	8s	52s	1279s

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**	**3s**	**134s**	**738s**	**152mn**
	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
	34/77	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**	**3s** **0.935**	**134s** **0.769**	**738s** **0.979**	**152mn** **0.984**
	3 passes	5 passes	5 passes	5 passes	5 passes	4 passes	5 passes

Data structures

- Need to keep in memory:
  - the adjacency lists (space complexity: 2m+n);
  - vectors ‘e’, ‘a’, node2comm (n each);
  - total = 2m+4n : 118M nodes, 1G links:
    - 8.472 GB for the network;
    - 1.416 GB for the vectors.

- The algorithm is iterative:
  - adjacency lists can be read from disk iteratively;
  - passes can be made one at a time;
  - can deal with very large networks or to use laptops.

Heuristics

- Last iterations and passes offer a marginal gain:
  - stop when the gain is lower than a given epsilon.
- Leaves can be removed before the computation:
  - only useful if networks are very large (>M nodes).
- Only few nodes (<10%) are moved at a given iteration:
  - a standing node is not considered at the following iteration.

Previous results have been obtained using the first one.

Outline

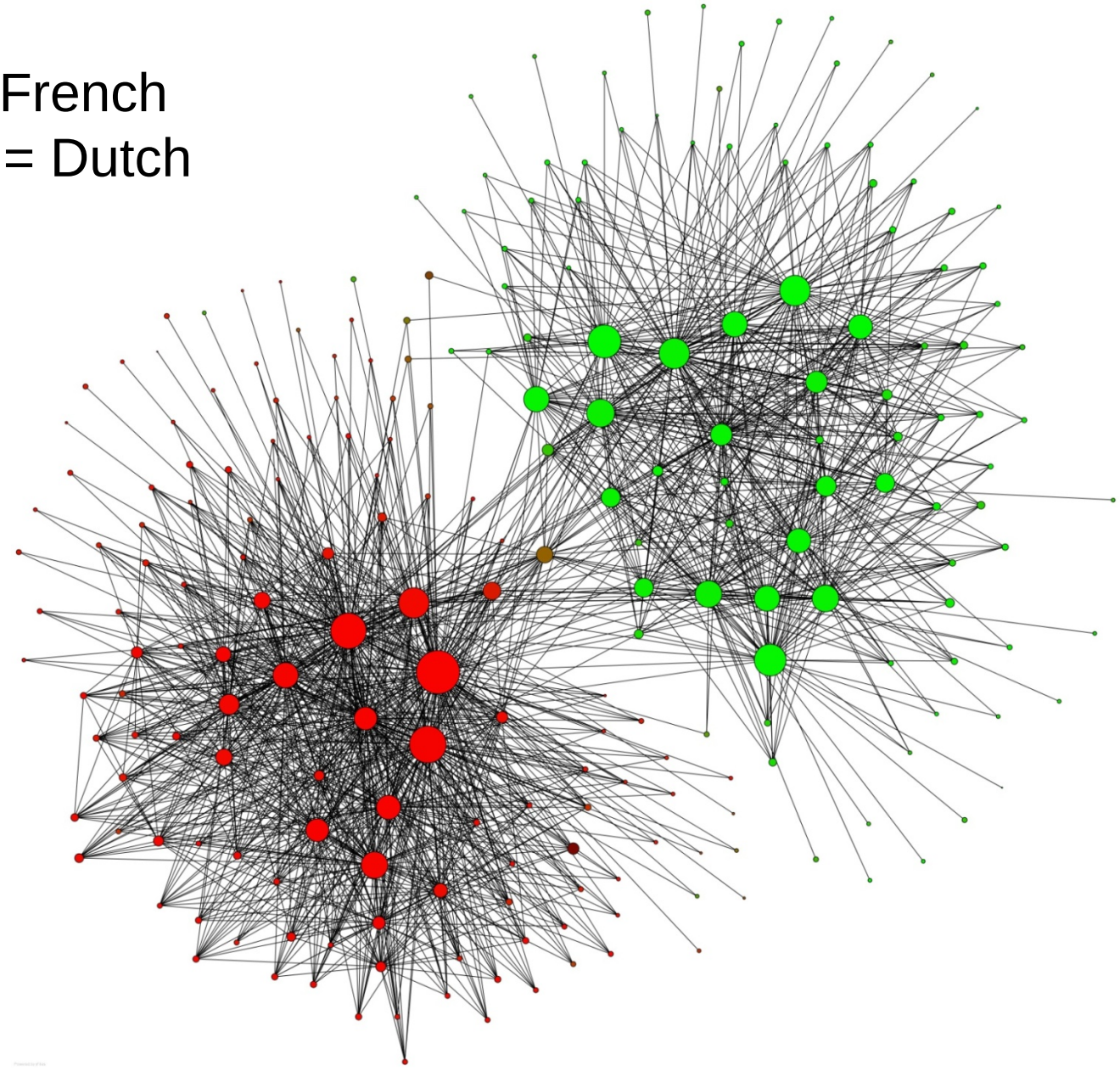
- The algorithm
- Experimental results
- Case study:
  - Belgian phone call network

Case study

- Belgian phone call network :
  - 6 months of communications;
  - One Belgian major operator.

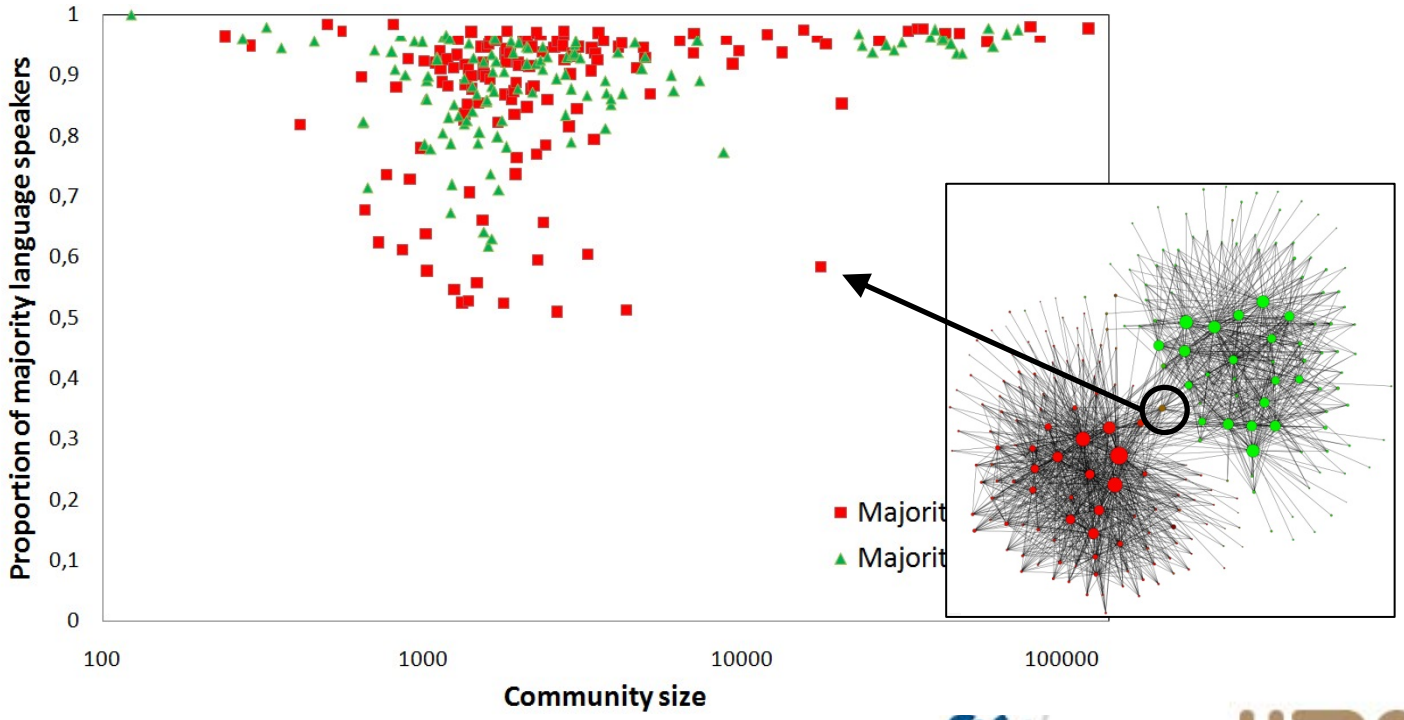
- Flat weighted network :
  - 2.6 millions customers;
  - language information (Dutch, English, French or German);
  - 6.3 millions links:
    - weight : number of calls + sms;
    - only stable calls are kept.

Red = French
Green = Dutch

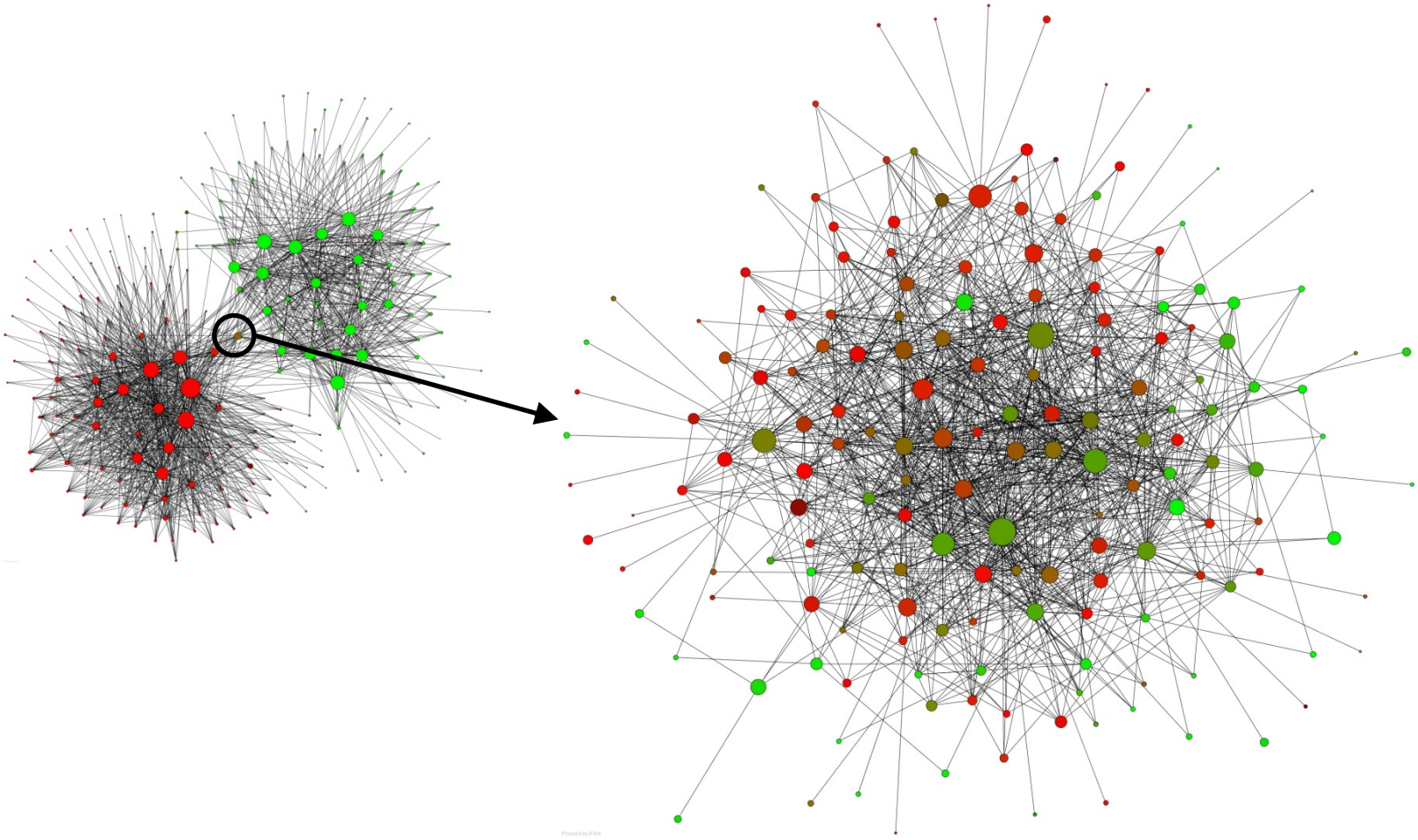


Language segregation

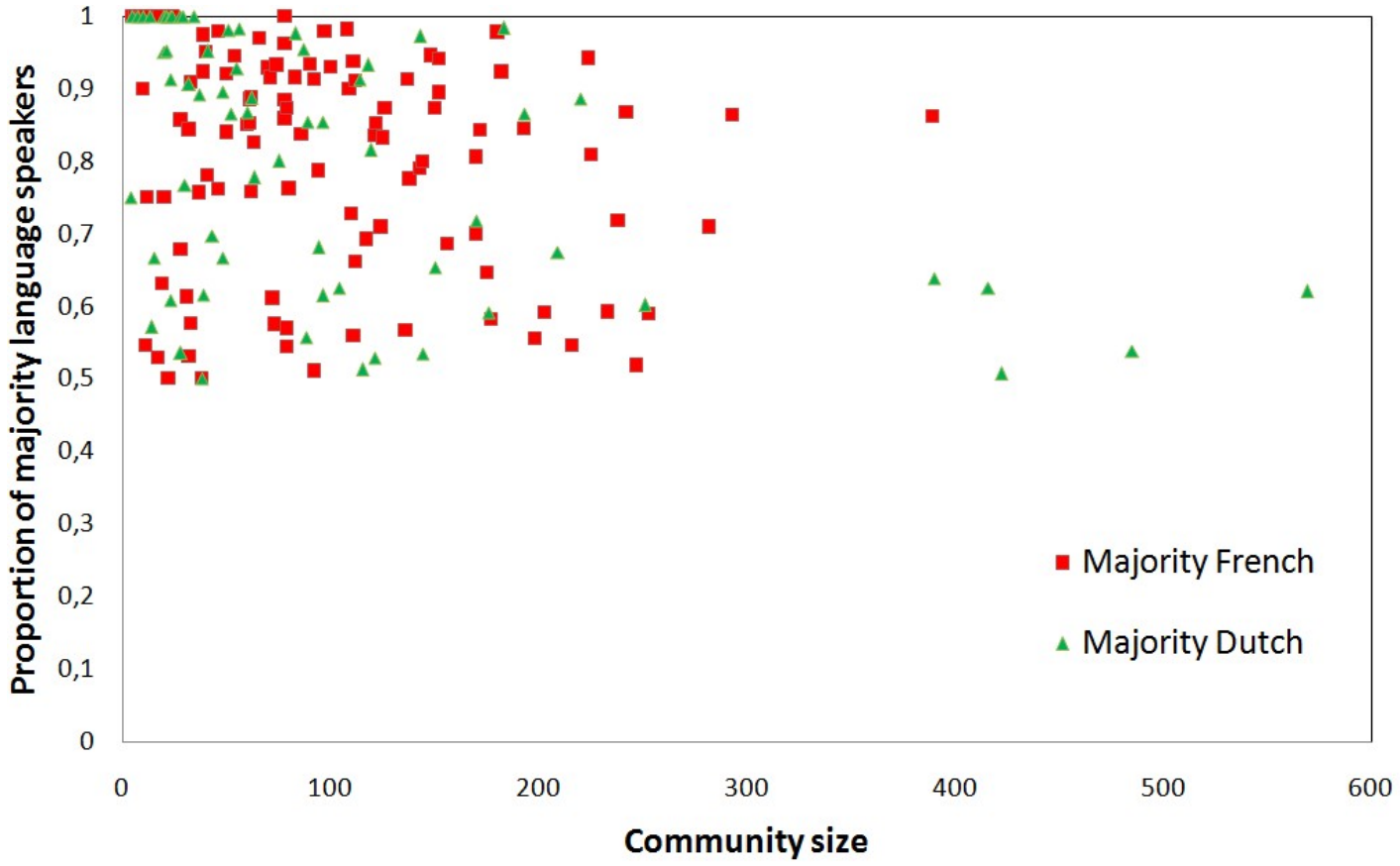
- All but two communities of size >10k are >93% segregated.
- One community contains more than 60% of all German speaking Belgians.



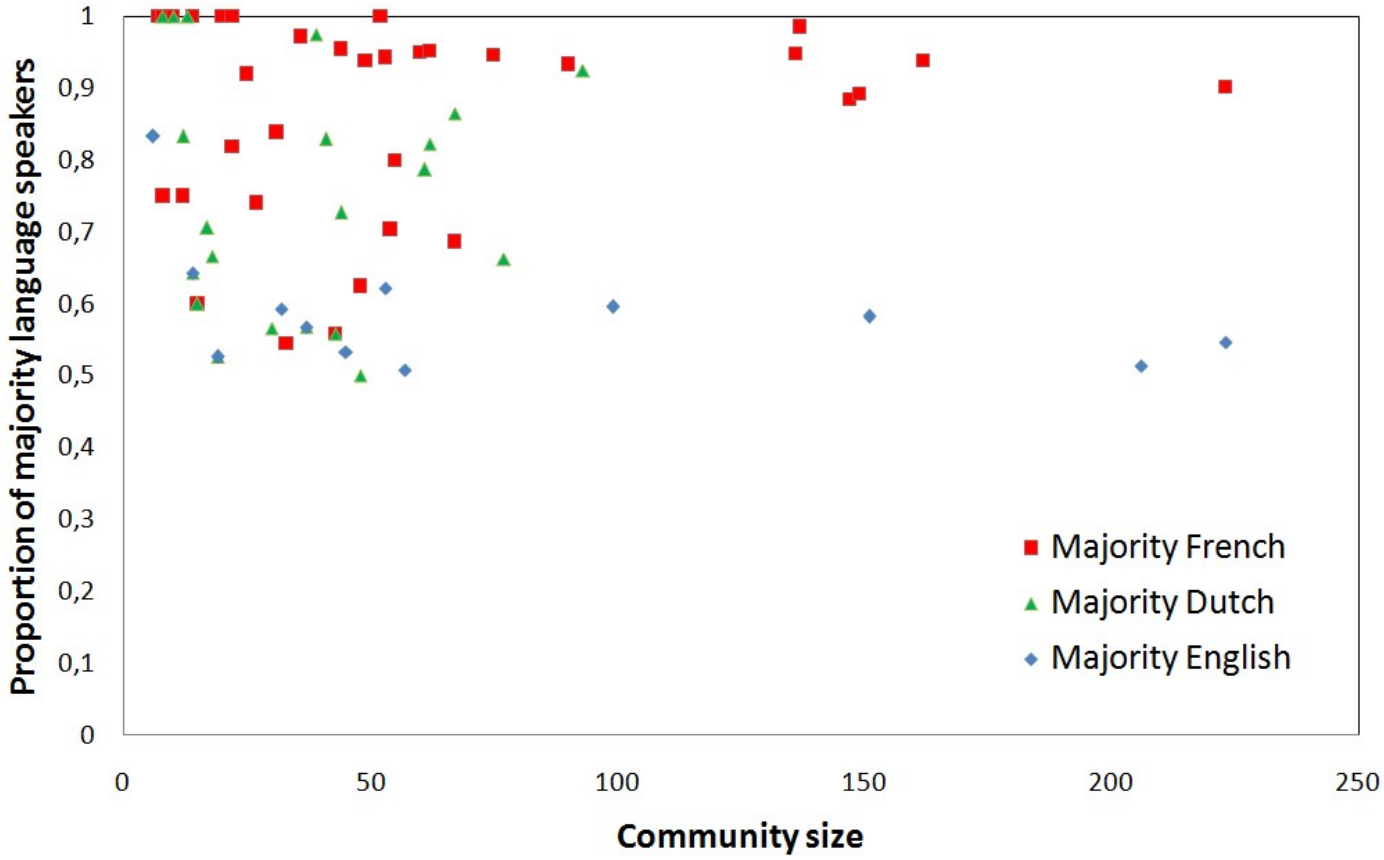
Largest bilingual community



Largest bilingual community



Second largest “bilingual”



Conclusion

can deal with millions/billions nodes/links
achieves very good modularity

- Moreover:
  - directly produces a hierarchy structure;
  - is strikingly simple;
  - can work on external memory;
  - can use other local quality functions.

Open issues

- Use more heuristics:
  - Allow non increasing modularity choices?
  - Simulated annealing like approaches?
- Understand the community structure:
  - use more information (language) to understand/validate.
- Overlapping communities
  - good quality “overlapping partition”?
- Evolving networks/communities?

Post-doc position for 1 year

- LIP6, NPA team, University Paris 6, France.
- Complex networks.
- Open to signal processing, data-mining, distributed computing, etc. in relation with complex networks.

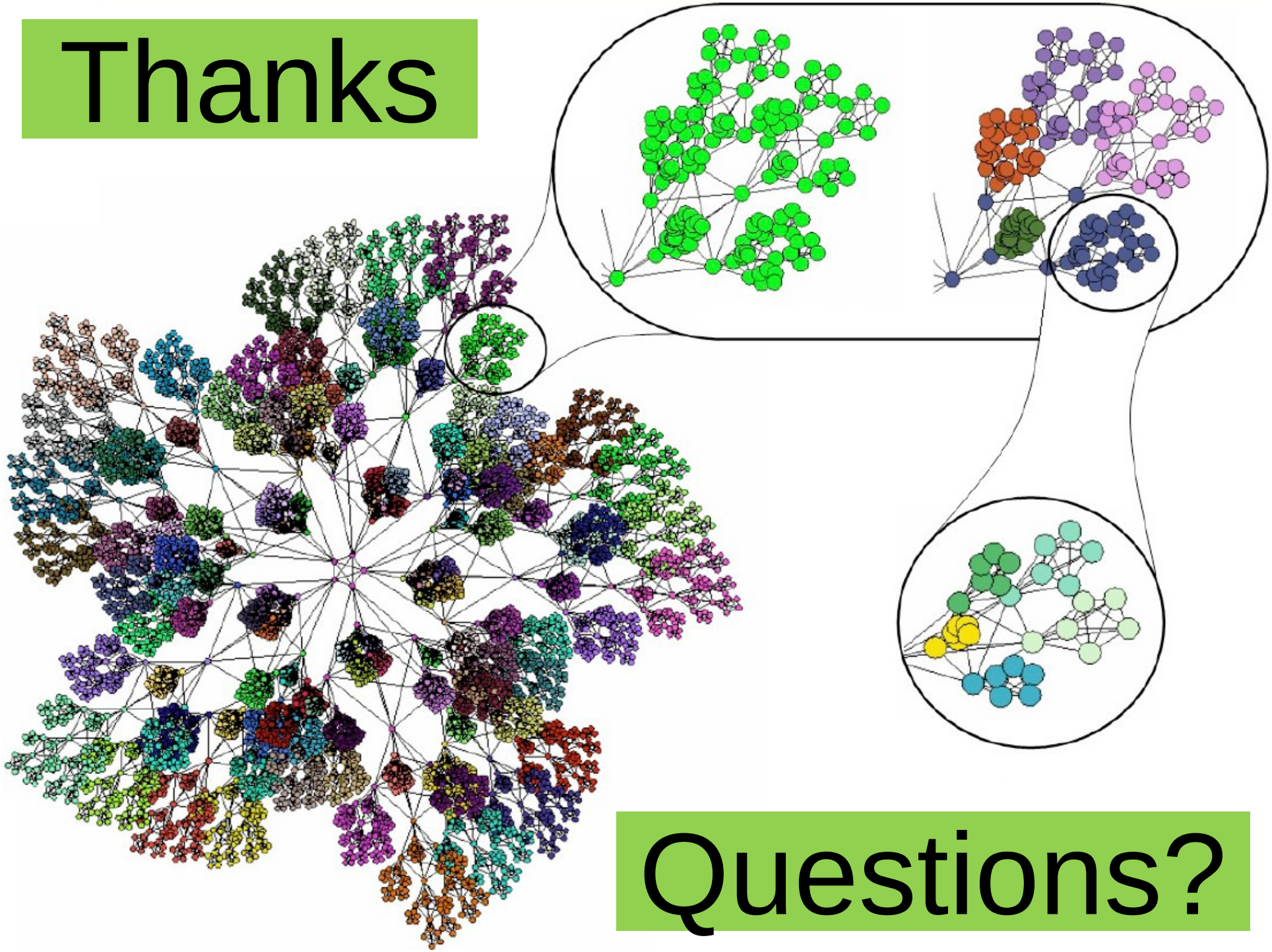
Deadline March 30th

Simple application form

Remember https://www2.cnrs.fr/DRH/post-docs08/?pid=1&action=view&id=597 !!!

Or ask me

Thanks



Questions?