Community detection in fine-grained dynamical networks

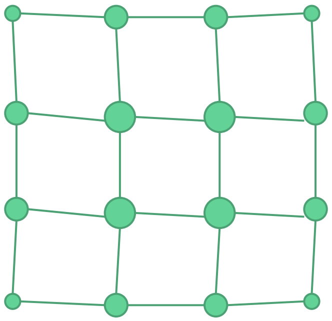
Fabrice Lécuyer
Supervised by Rémy Cazabet

2020

Outline

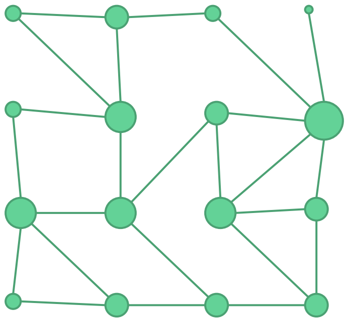
1. **Networks & communities**
2. Relevant algorithms
3. Static tests
4. Static method for dynamic results
5. Competitive random walks

From graphs to networks



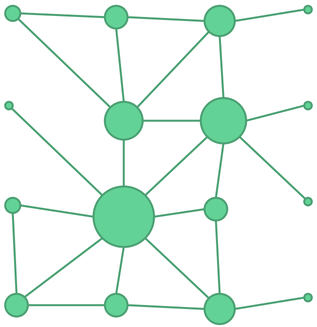
Regular graph

Structure
High diameter
Predefined degrees



Random graph

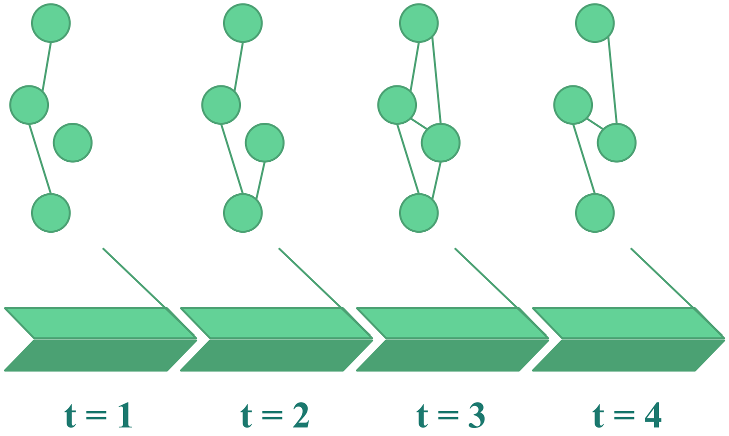
No structure
Small diameter
Poisson distribution

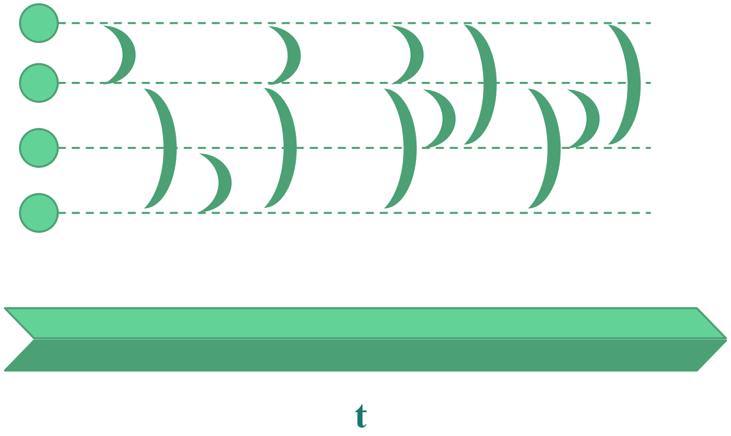


Real-world network

Hubs & clusters
Small diameter
Power-law distribution

Dynamic networks





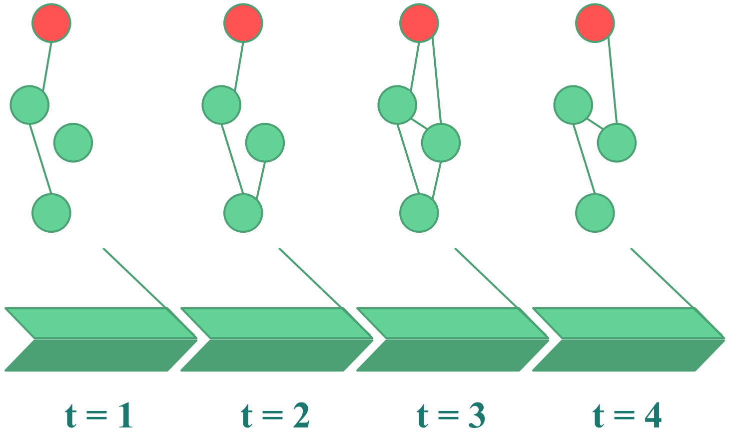
Snapshots

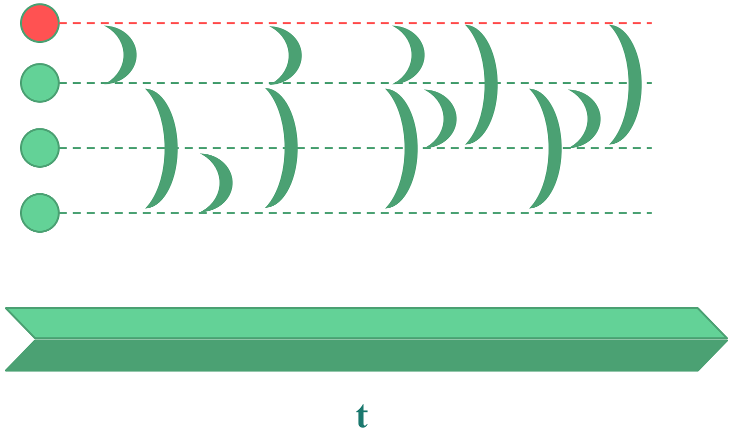
Network at each step
Relations (eg friendships)

Link stream

No network at a specific time
Interactions (eg emails)

Dynamic networks



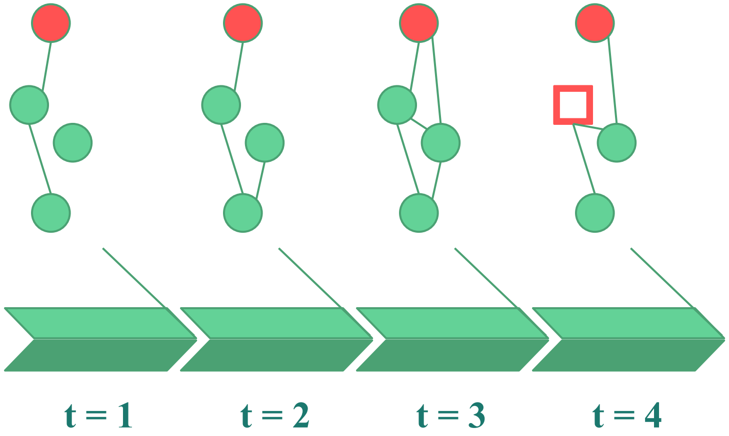


Snapshots

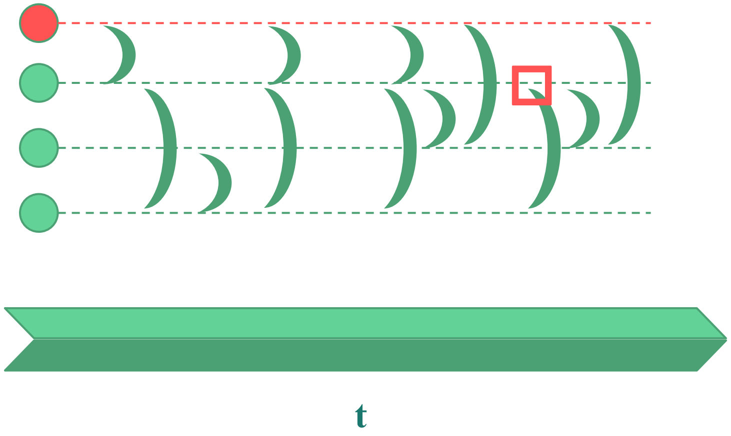
Link stream

node

Dynamic networks



Snapshots

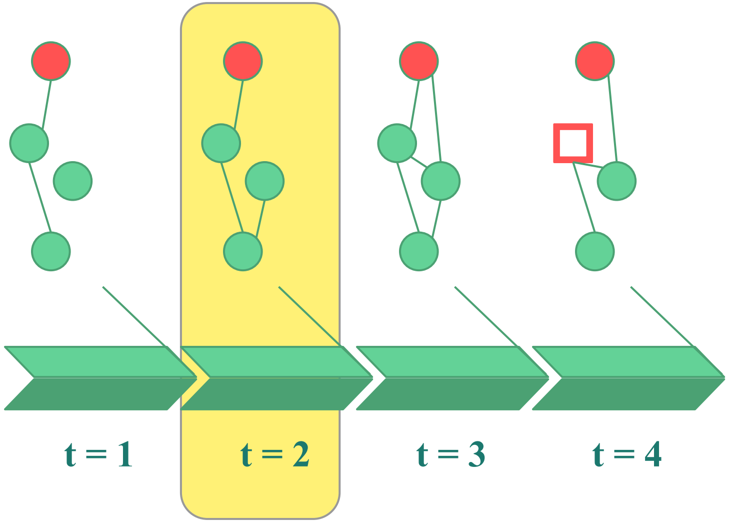


Link stream

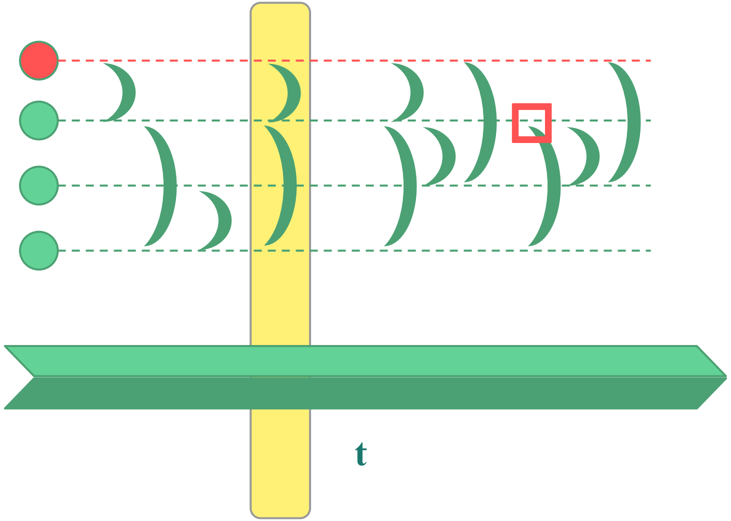
node

tnode

Dynamic networks



Snapshots



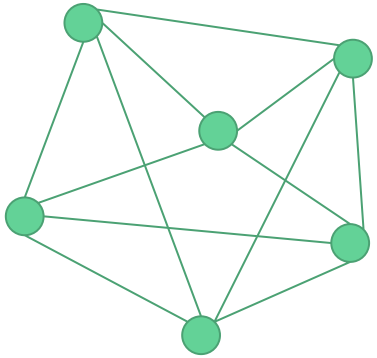
Link stream

node

tnode

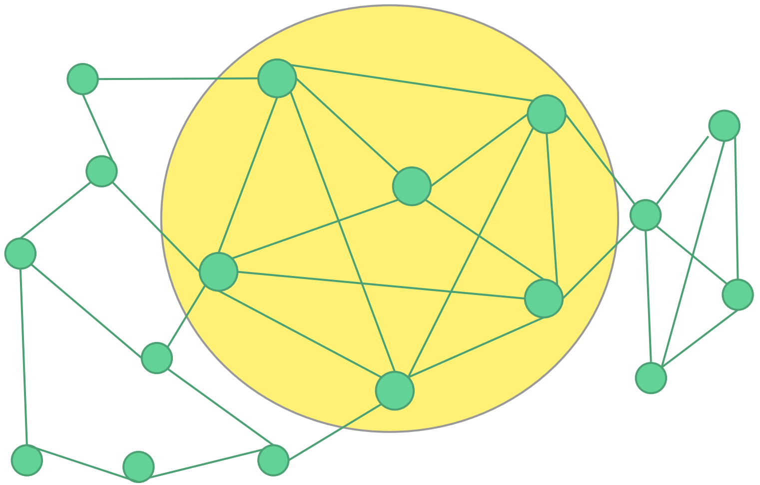
frame

Communities



Dense subset…

at best: a clique



… with sparse cut

at best: isolated

Quality functions: modularity

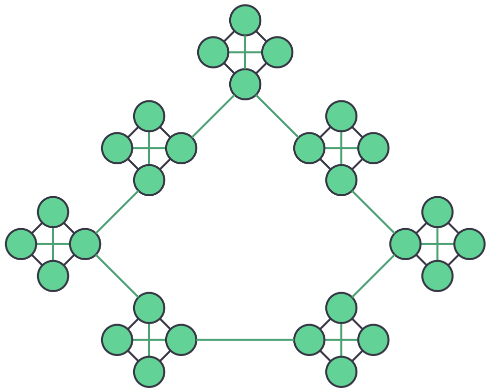
$$Q(C) = \frac{1}{2m} \sum_{u,v \in C} \left(1_{(u,v) \in E} - \frac{d_u d_v}{2m} \right)$$

Existing edges in the community

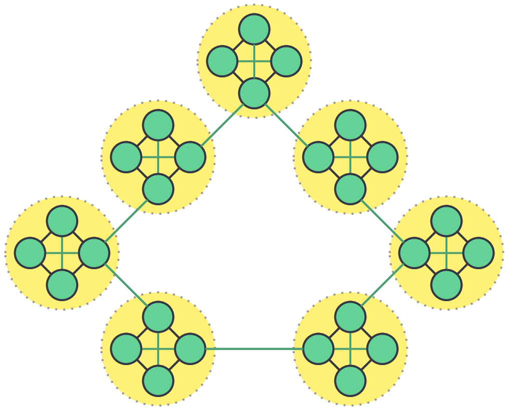
Expected edges in the configuration model

Quality functions: modularity

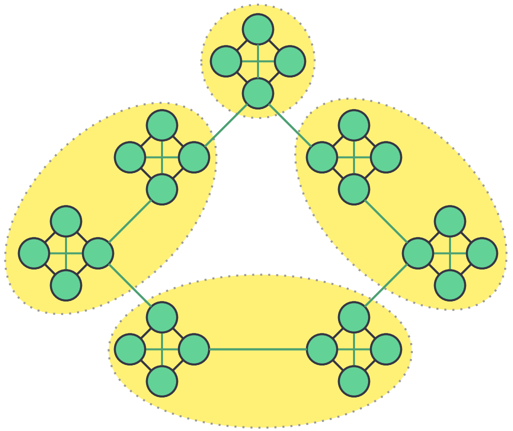
$$\mathcal{Q}(C) = \frac{1}{2m}\sum_{u,v\in C}\left(1_{(u,v)\in E} - \frac{d_u d_v}{2m}\right)$$



Ring of cliques



Intuitive communities

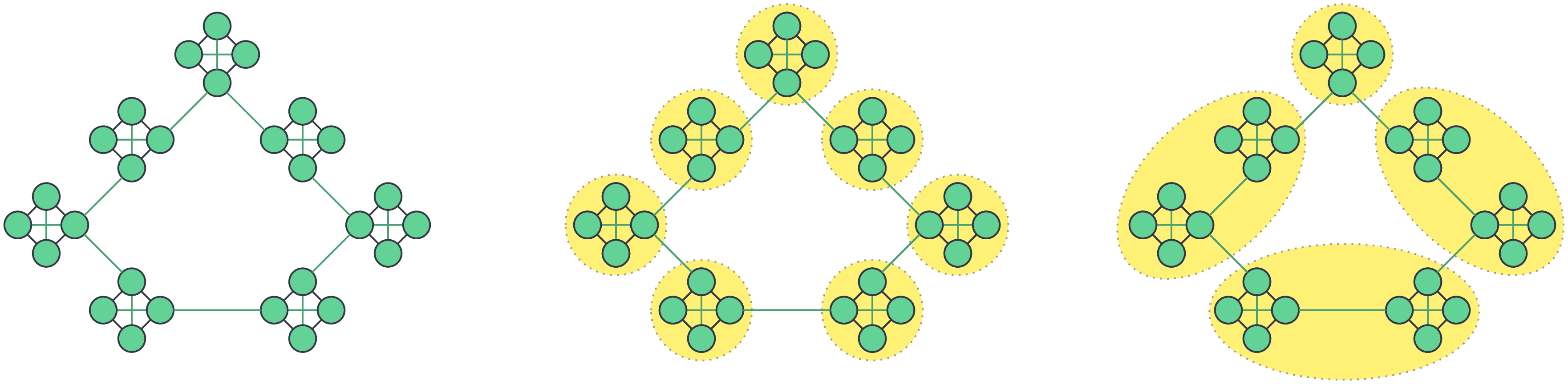


Resolution limit

Quality functions: modularity

Resolution parameter

$$\mathcal{Q}(C) = \frac{1}{2m} \sum_{u,v \in C} \left(1_{(u,v)\in E} - \lambda \frac{d_u d_v}{2m} \right)$$



Ring of cliques Intuitive communities Resolution limit

Quality functions: many more

Conductance

Surprise

Expansion

Clustering coefficient

Significance

Permanence

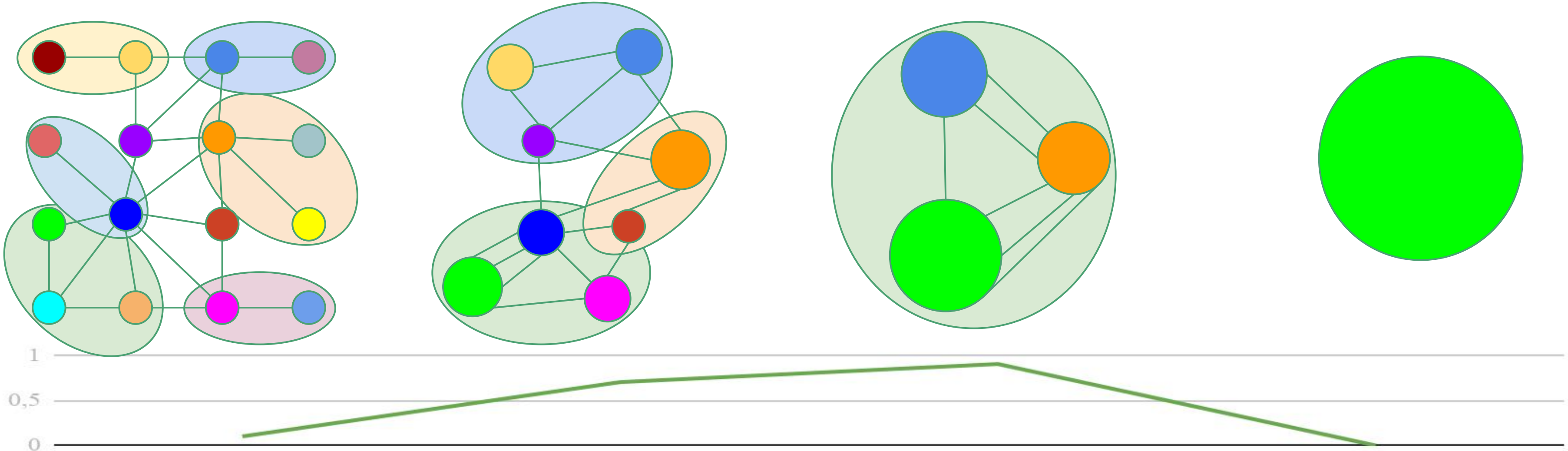
Separability

No unique standard definition

Outline

1. Networks & communities
2. **Relevant algorithms**
3. Static tests
4. Static method for dynamic results
5. Competitive random walks

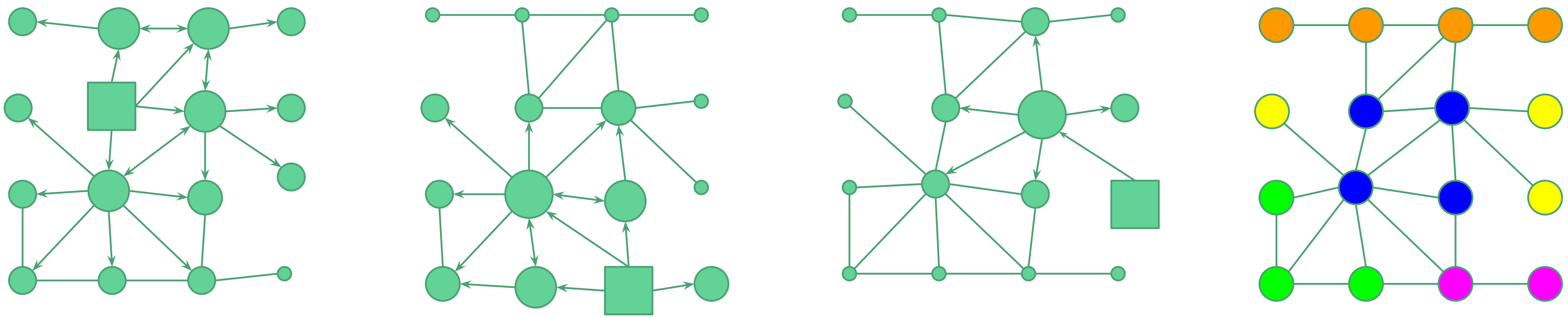
Static algorithm: Louvain & Leiden



Merge nodes to locally optimise modularity

+	very fast (linear in m)	–	global modularity optimisation
+	relevant results	–	sparse communities

Static algorithm: Walktrap

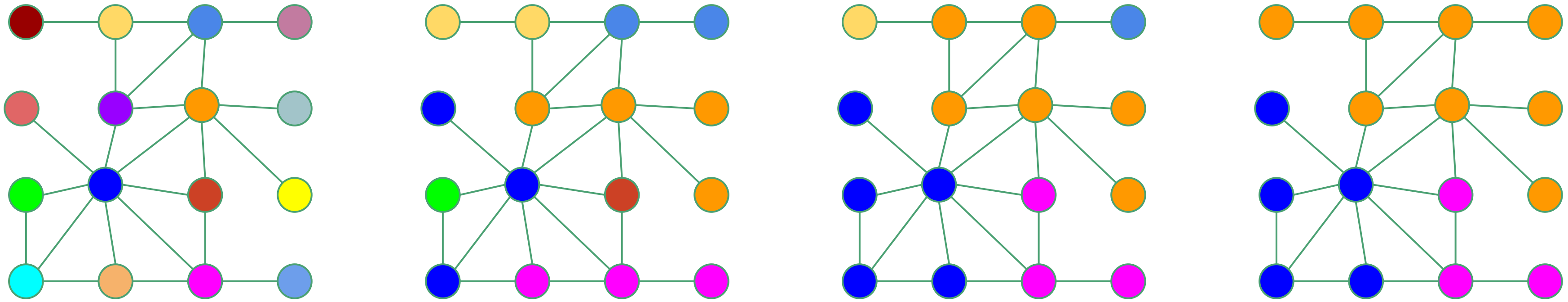


Merge nodes having similar results of short random walks

+ ideal for small diameters
+ local computation

– modularity optimisation
– sparsely connected communities

Static algorithm: Label propagation



Update by taking neighbours' favourite label

+	matrix definition	–	quadratic
+	local computation	–	many different results

Dynamic algorithms on snapshots

Aim: good instantaneous communities that evolve smoothly

- Dynamic Louvain: use degeneracy to smooth communities
- Multi-objective optimisation
- Detect and follow core nodes
- Match communities with similarity index

Dynamic algorithms on link streams

Aim: local scalable algorithm with mathematical grounding

- Aggregate the stream on a sliding time-window
- Link tedges having consecutive tnodes
- Put a Gaussian weight around tedges
- Extend and prune an initial community

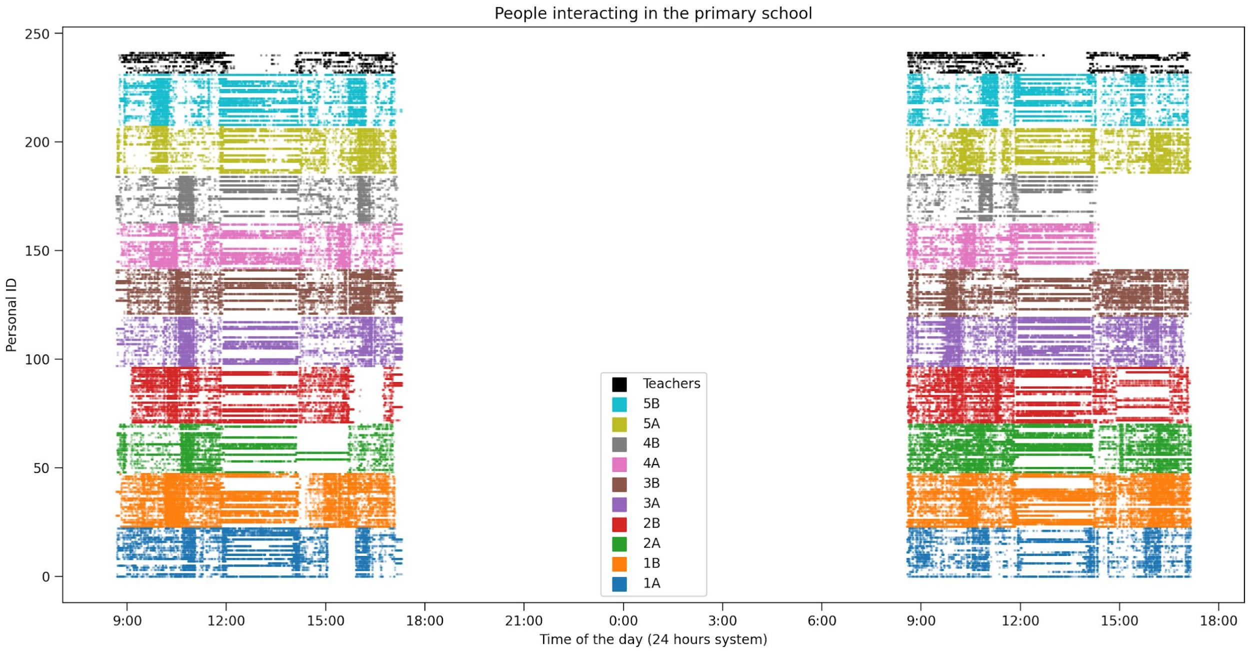
Problem: no unique definition in static — worse in dynamic

Outline

1. Networks & communities
2. Relevant algorithms
3. **Static tests**
4. Static method for dynamic results
5. Competitive random walks

Link stream datasets: Sociopatterns

Interactions between people, recorded with RFID sensors



See also: hospital ward, traditional household, conference attendees, wild herd…

Big available data

Bitcoin

Blockchain is public by design

150 million transactions
7 years
175Gb of data

Track anonymous addresses
Communities may be recovered

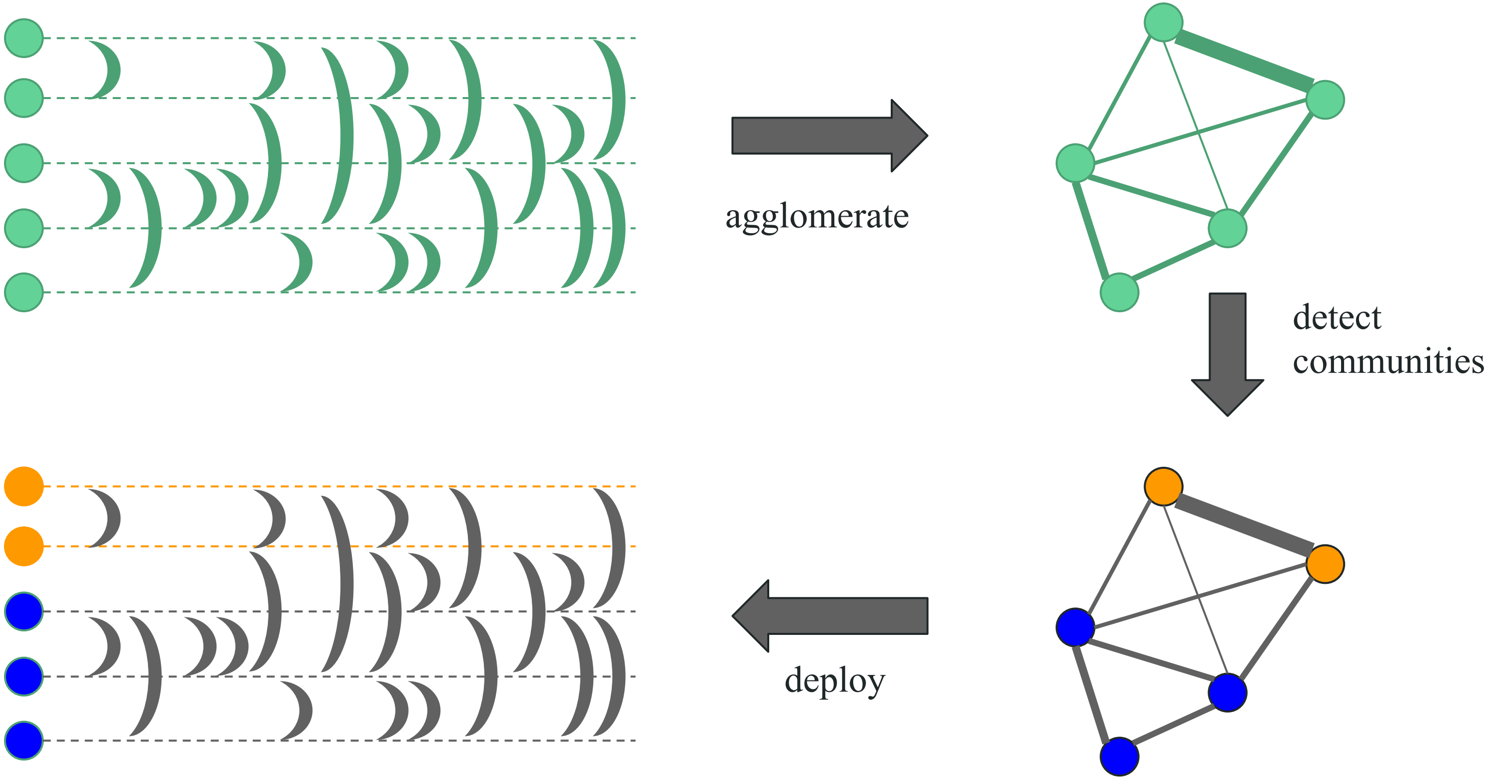
Twitter

Private but partially released

300 million users
500 million posts a day

Track posts of the same user
Communities may be recovered

Contribution: test static algorithms on agglomerated graphs



Contribution: test static algorithms on agglomerated graphs

Graph with 242 nodes and 2340 edges.

Method	#	aNMI%	Modul%	Conduc%	Surp	Stab%	Time (s)
groundtruth	11.0	100	68	33	2274	100	2.004000000965789e-06
louvain	7.2	83	72	18	2375	98	0.05116954990000124
leidenW	6.6	81	72	16	2432	96	0.04283884670000191
leiden	6.9	83	72	16	2535	99	0.040756389099999527
label	8.0	85	70	19	2468	100	0.015577493500001083
walktrap	8.0	86	72	18	2650	100	0.04235497800000019
infomap	8.0	87	72	18	2657	100	0.01987894130000143

Decent results compared to groundtruth

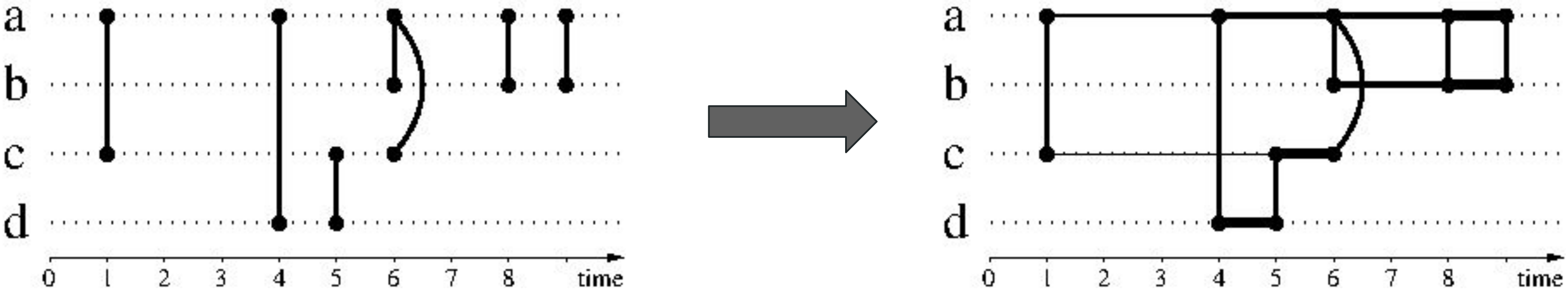
Quality functions are not the truth

All very fast and stable on small networks

Outline

1. Networks & communities
2. Relevant algorithms
3. Static tests
4. **Static method for dynamic results**
5. Competitive random walks

Contribution: topochrone graphs



Build chronological edges

Apply time decay

Detect communities using static algorithms

$$\omega^{\text{const}}_{(t,u),(t+\delta t,u)} = \alpha$$

$$\omega^{\text{lin}}_{(t,u),(t+\delta t,u)} = \frac{\alpha}{\frac{\delta t}{\tau} + 1}$$

$$\omega^{\text{exp}}_{(t,u),(t+\delta t,u)} = \alpha \exp -\frac{\delta t}{\tau}$$

Contribution: topochrone graphs

175’000 nodes, 300’000 edges

Label propagation is too slow (30 min)
Walktrap is 30x slower than Leiden (9 sec)

≃100 communities found
Pupils together during class time
School-wide blending during breaks

Outline

1. Networks & communities
2. Relevant algorithms
3. Static tests
4. Static method for dynamic results
5. **Competitive random walks**

Contribution: competitive random walks

Ants of different species navigate:

– jump *k* times (topology)

– survive *t* frames (chronology)

Community accepts **remarkable** tnodes:

$$\mathcal{R}_{b|a} = \frac{|\mathcal{D}_a| \cdot p_{b|a}}{|\mathcal{D}| \cdot p_b} > 1$$

$p_{b|a}$ = portion of pheromones from species *a*

p_b = portion of total pheromones

Input: link stream L, depth k,
walk duration τ, precision π
Output: Non-overlapping farms
Create $f = \pi \frac{m}{\tau}$ farms of 1 random tnode
for k times **do**
  for $\omega = \pi \frac{T \cdot n}{\tau}$ times **do**
    Choose F proportionally to $|F|$
    Ant starts at random in F
    Ant walks τ frames and k jumps, laying down F-pheromones
  end for
  Tnodes choose most remarkable farm
  Delete small farms when $|F| < \tau$
end for

Contribution: competitive random walks

Classes during lessons are still discovered
80 to 100 communities

Some tnodes have no community
Easy to find overlapping communities

Noisy results (nodes with few tnodes)
Unstable results (different at every run)

Contribution: competitive random walks

Complexity:

- runs: $k \simeq 5$
- walkers: $w = tnodes \,/\, t \simeq 2m \,/\, t$
- walk duration: t

Overall: $O(k \cdot m)$

Assign tnodes to most remarkable community

+ quite fast (linear in m)
+ detects several scales
+ overlapping / incomplete communities

– unstable and noisy
– based on emergence instead of maths
– sensitivity to constant t

Conclusion

Link streams = high precision temporal graphs.
Community detection has standard algorithms for the static case; not for link streams.

Data of any size can be found.
We propose 2 methods with linear complexity.

Further work:

- extensive tests
- mathematical grounding
- optimisation

Any question?

Fabrice Lécuyer
Supervised by Rémy Cazabet

2020

Synthetic datasets

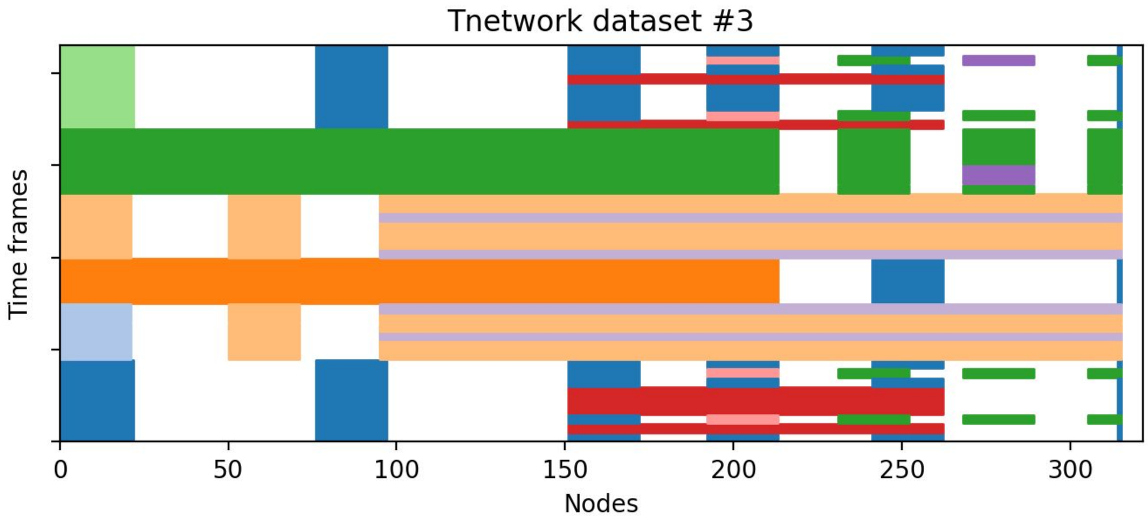
Python library *tnetwork*

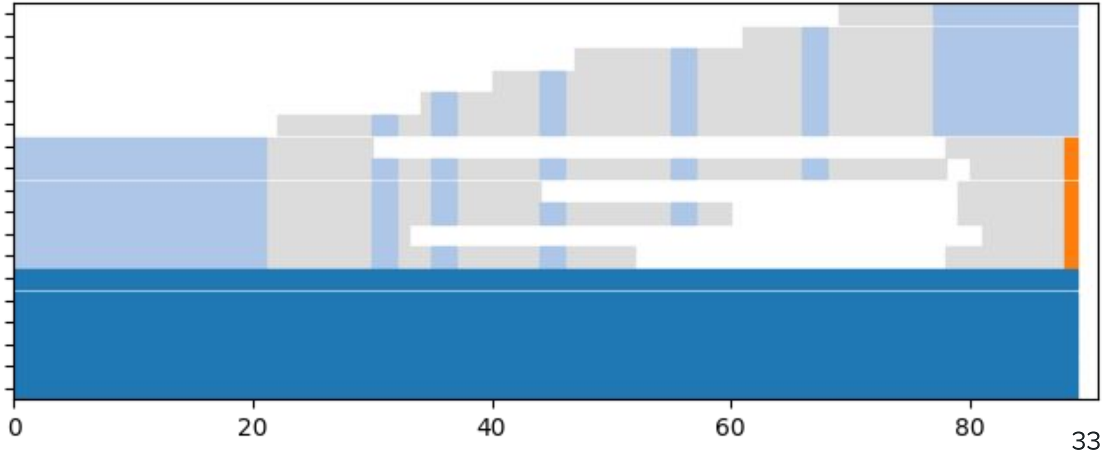
Create predefined scenarios

Create random events

Test tricky situations

Compare average situations





Quality functions: modularity

$$\mathcal{Q}(C) = \frac{1}{2m} \sum_{u,v \in C} \left(1_{(u,v) \in E} - \frac{d_u d_v}{2m} \right)$$