Université de Lyon
Université Claude Bernard Lyon 1
CNRS
LIRIS

A BENCHMARK FOR GENERATING DYNAMIC COMMUNITIES WITH CUSTOM LIFECYCLE

Cazabet Rémy

COMMUNITY DETECTION

- Community detection or "graph clustering"
  - No formal definition
  - Two informal definitions:
    - groups of densely connected nodes, weakly connected to the rest of the network
    - groups of nodes that "make sense" in real networks

  - Too limited : Stochastic Block Models ?

COMMUNITY DETECTION

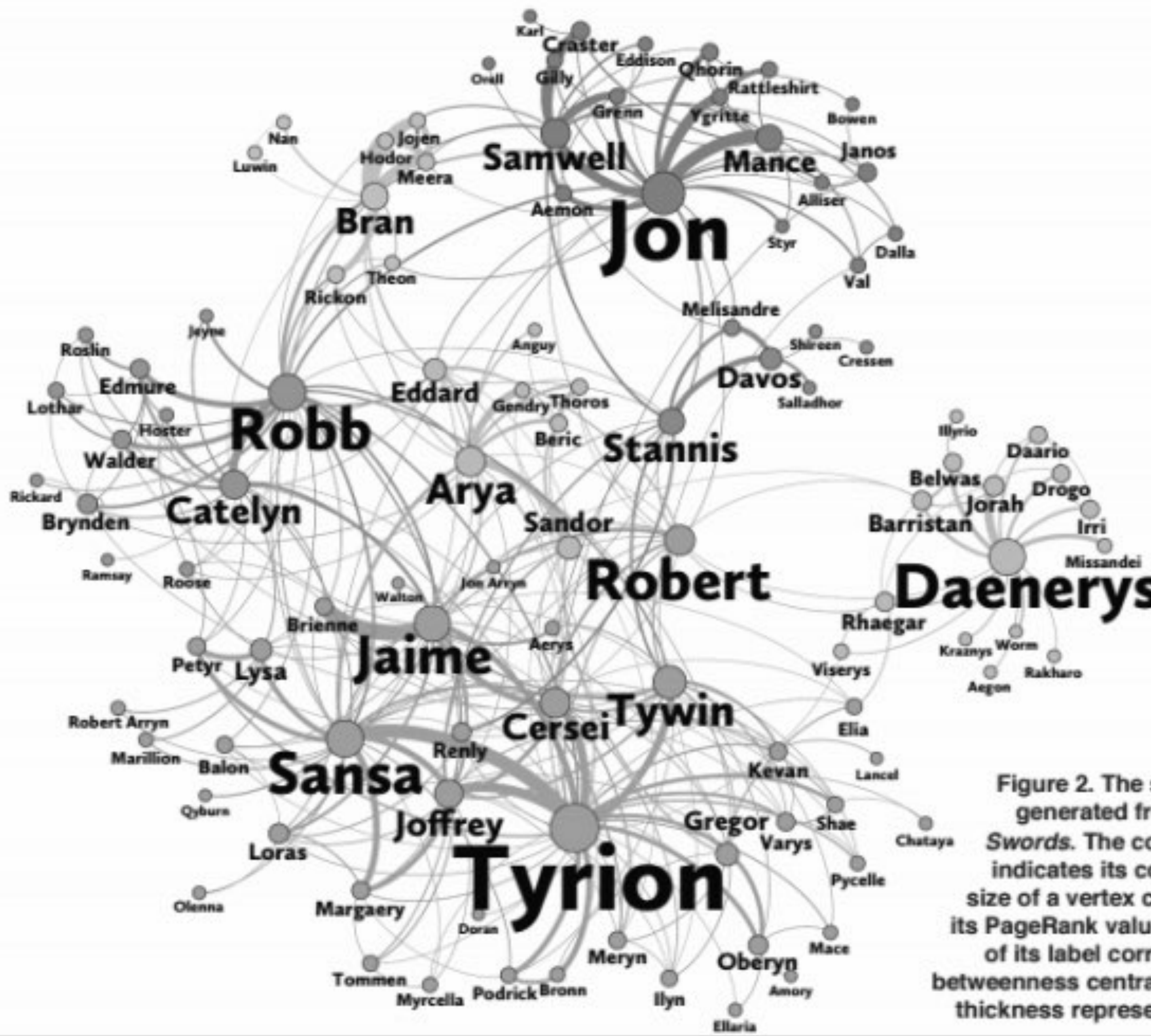


Figure 2. The social network generated from *A Storm of Swords*. The color of a vertex indicates its community. The size of a vertex corresponds to its PageRank value, and the size of its label corresponds to its betweenness centrality. An edge's thickness represents its weight.

COMMUNITY DETECTION

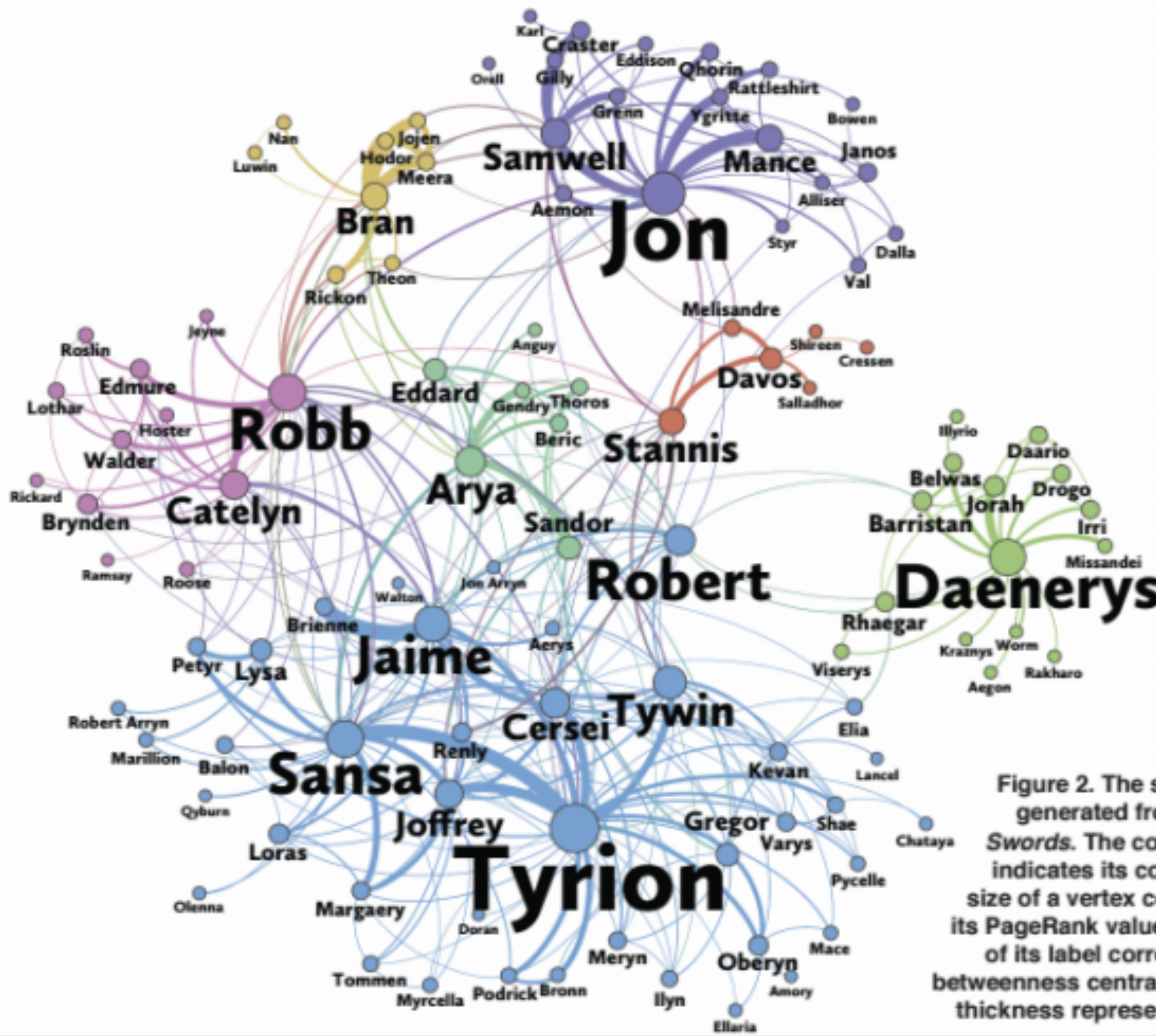


Figure 2. The social network generated from *A Storm of Swords*. The color of a vertex indicates its community. The size of a vertex corresponds to its PageRank value, and the size of its label corresponds to its betweenness centrality. An edge's thickness represents its weight.

COMMUNITY DETECTION

- Numerous applications:
  - groups of friends/colleagues in ego-networks
  - structure of an organisation (company, laboratory…)
  - topics in scientific networks
  - groups of interest in social medias (politics, opinions, etc.)
  - User de-anonymization
  - …

DYNAMIC NETWORKS

DYNAMIC NETWORKS

- Most real world networks evolve
  - Nodes can appear/disappear
  - Edges can appear/disappear
  - Nature of relations can change
- How to represent those changes?

DYNAMIC NETWORKS

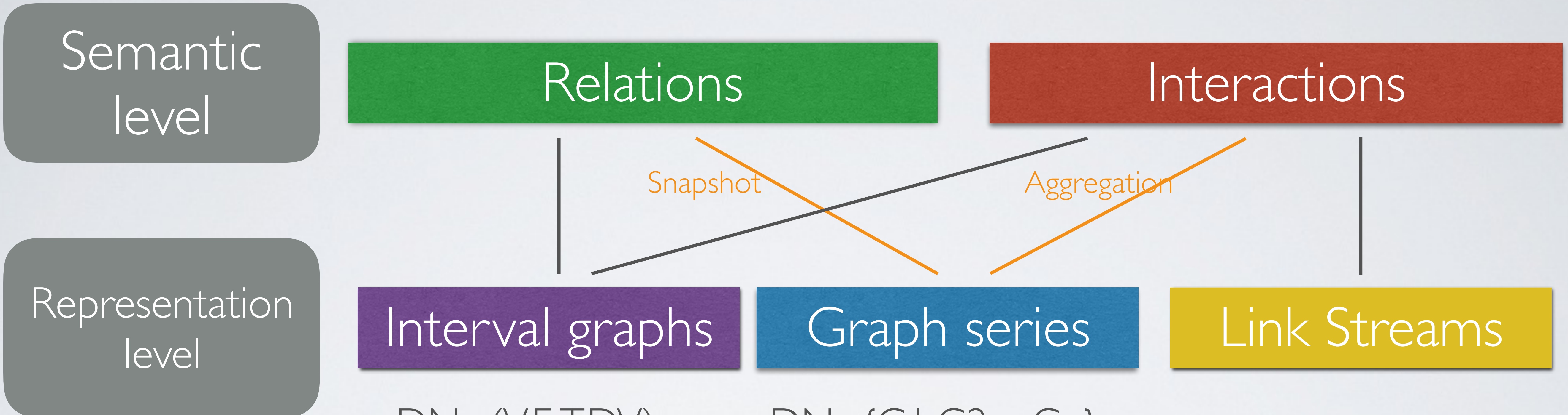
Semantic level

Relations	Interactions
Long term -Friend -Colleague -Family relation -...	**Instantaneous** -e-mail -Text message -Co-authoring ...
Short term ? -Collaborators in the same project -Same team in a game *-Attendees of the same meeting* -...	**With duration** -Phone call -Discussion in real life *-Participate in a same meeting*

DYNAMIC NETWORKS

Semantic level	Relations		Interactions
Representation level	Interval graphs	Graph series	Link Streams
	DN=(V,E,T,DV) DV:VxTxT E:VxVxTxT	DN={G1,G2…Gn} Gi=(V,E) E :VxV	DN=(V,E,T) E:VxVxT

DYNAMIC NETWORKS

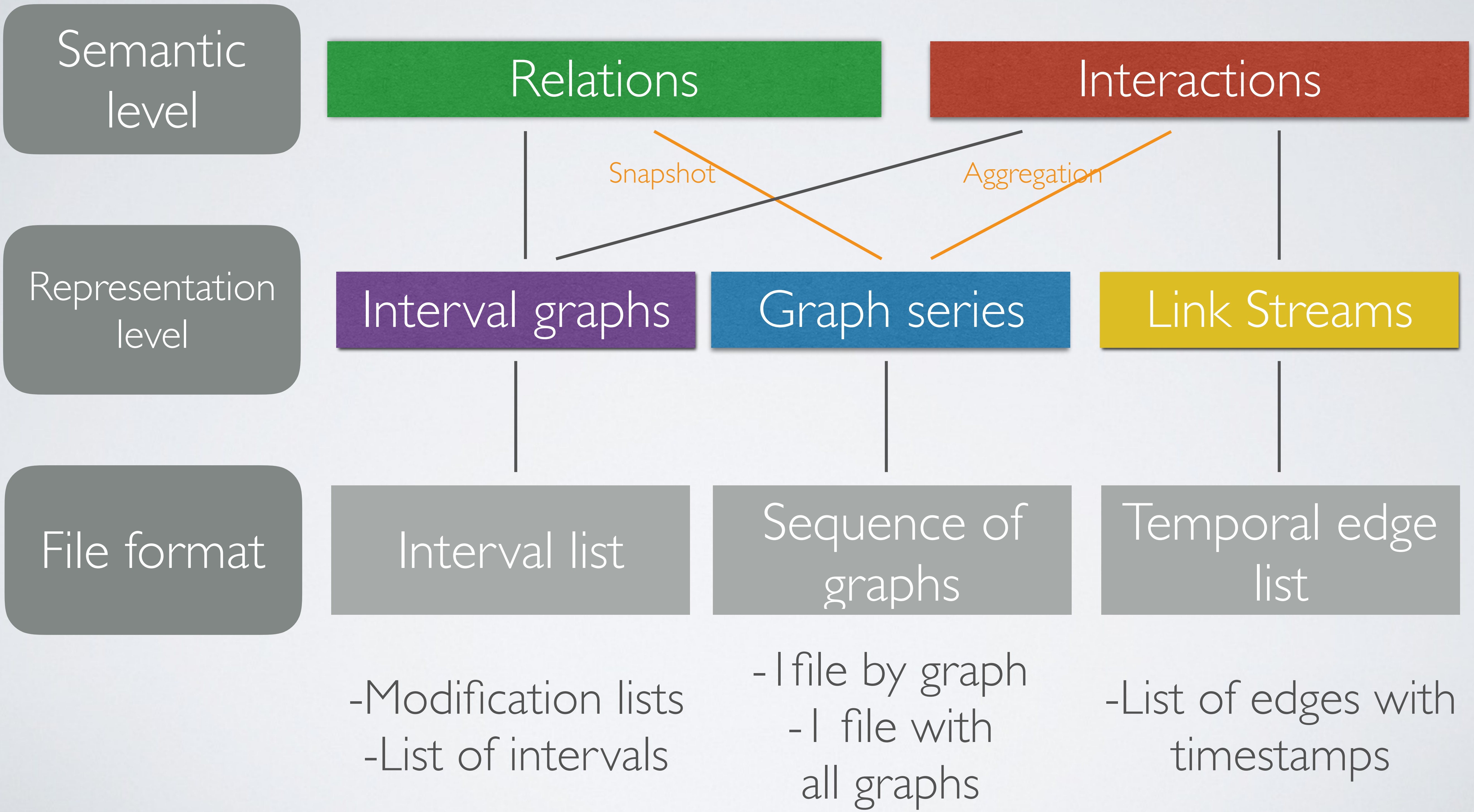


DN=(V,E,T,DV)
DV:VxTxT
E:VxVxTxT

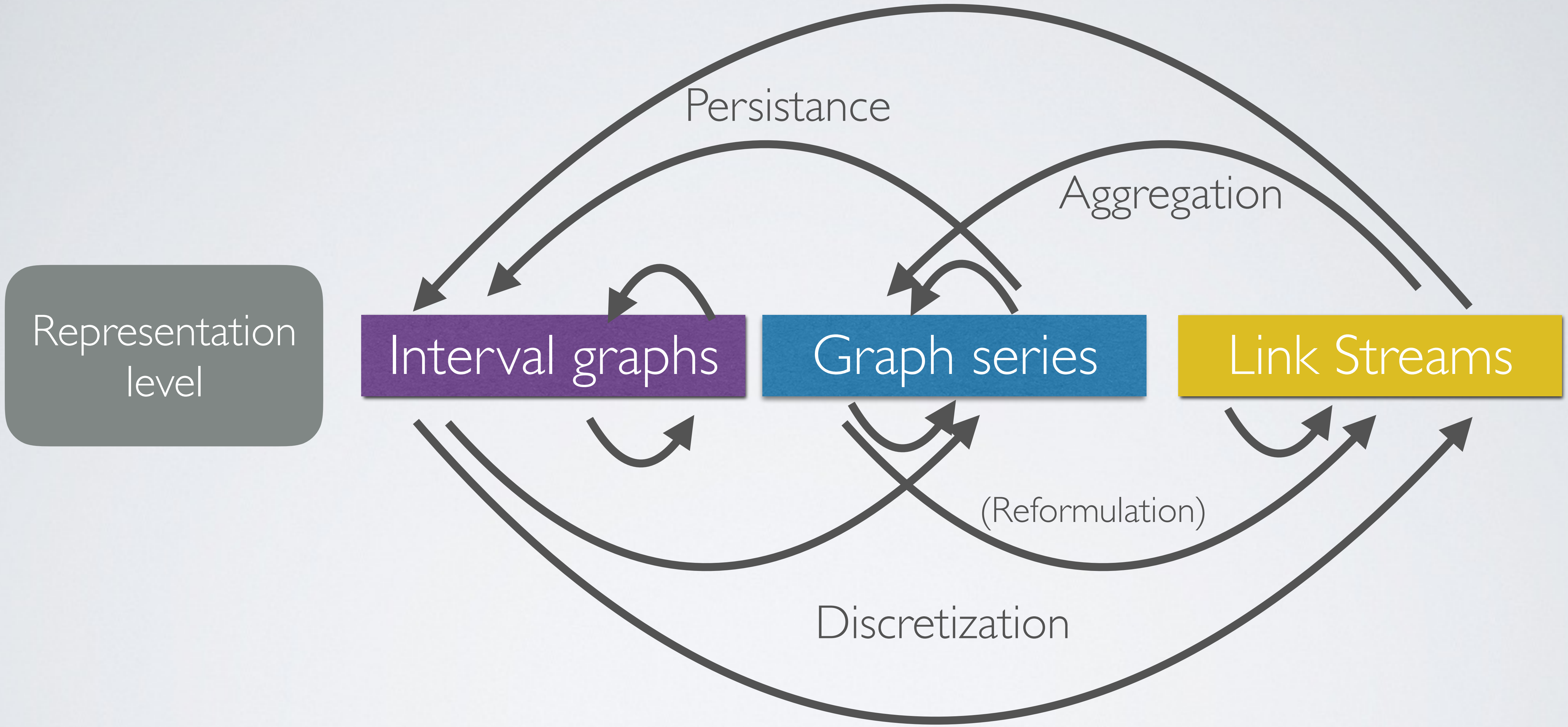
DN={G1,G2…Gn}
Gi=(V,E)
E :VxV

DN=(V,E,T)
E:VxVxT

DYNAMIC NETWORKS



DYNAMIC NETWORKS



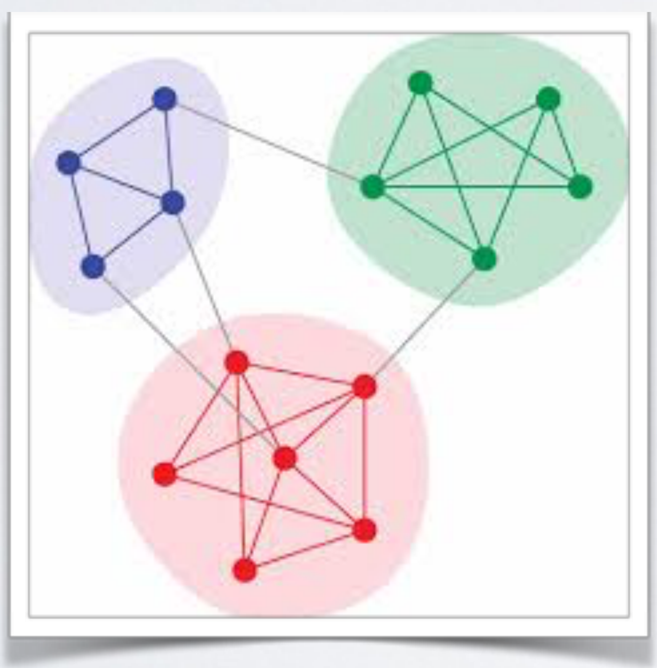
DYNAMIC COMMUNITY DETECTION

Source : Dynamic community detection: a Survey
[Rossetti, Cazabet, 2018]

COMMUNITY DETECTION

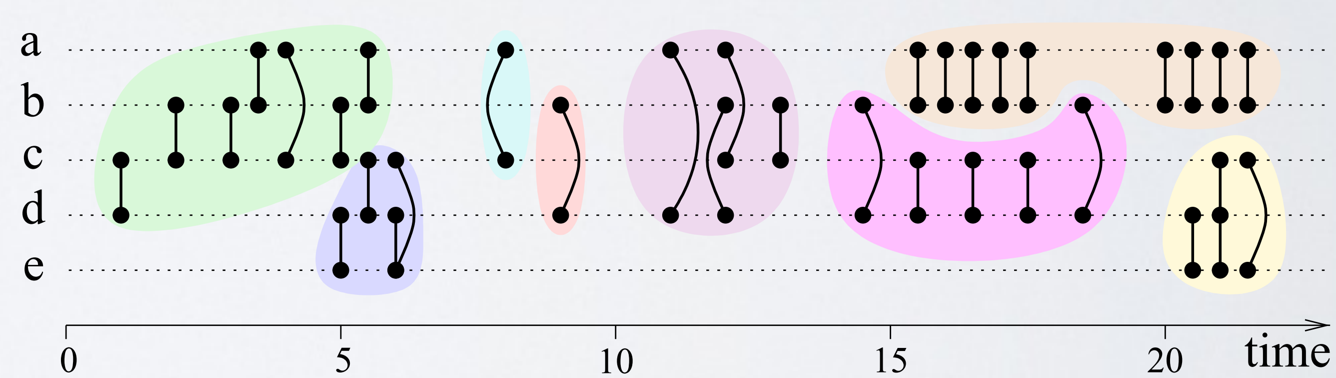
Static networks

Sets of nodes



Dynamic Networks

Sets of periods of nodes



[Viard 2016]

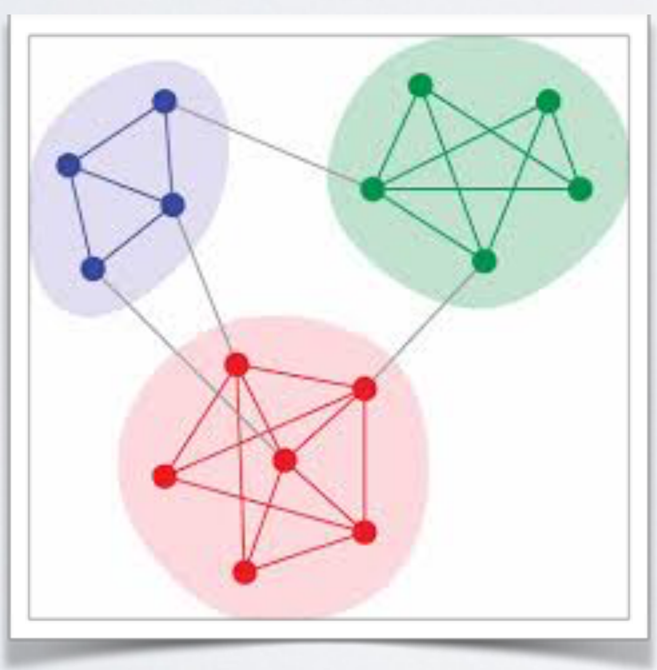
COMMUNITY DETECTION

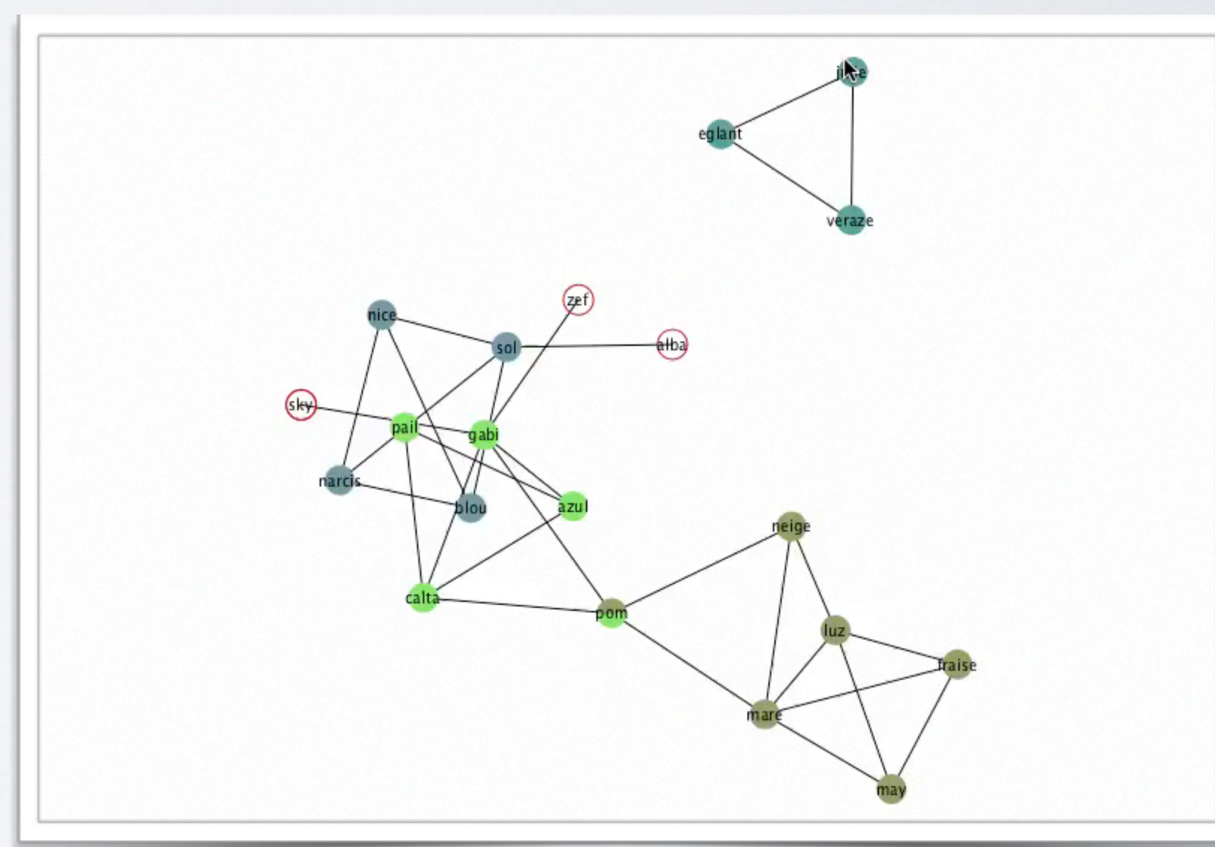
Static networks

Dynamic Networks

Sets of nodes

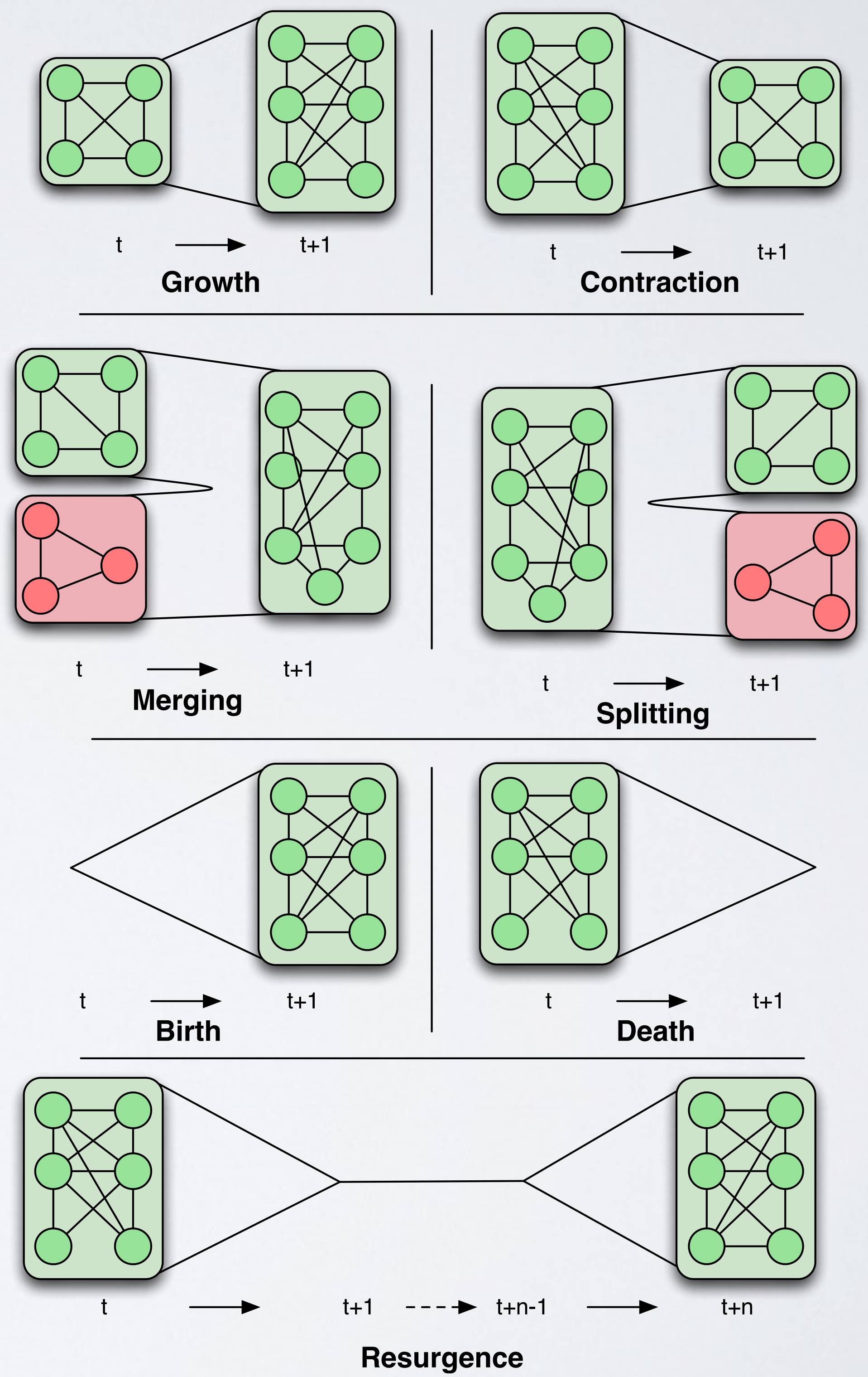
Sets of periods of nodes



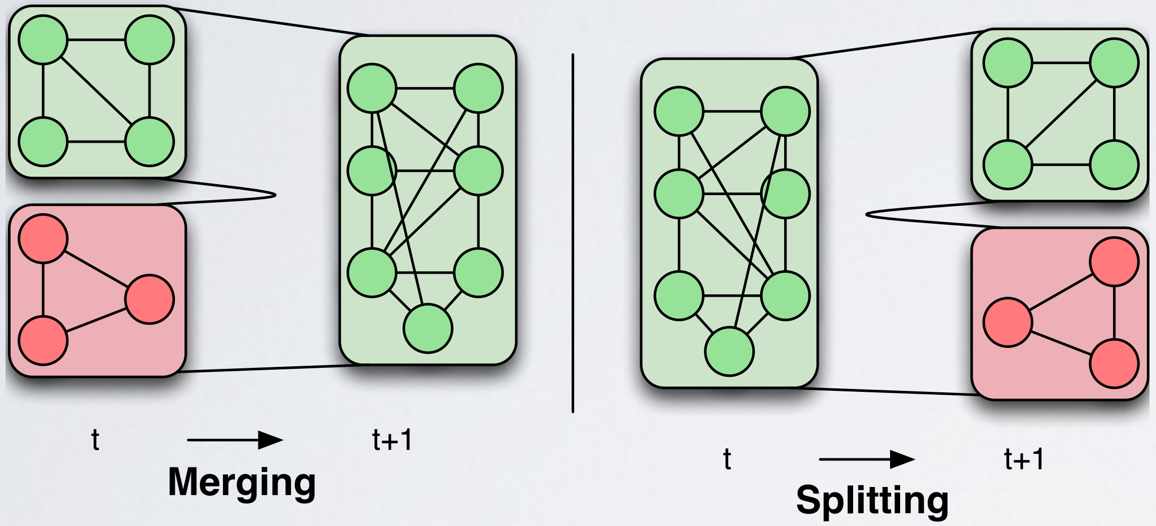


COMMUNITY DETECTION

Community events
(or operations)



COMMUNITY DETECTION

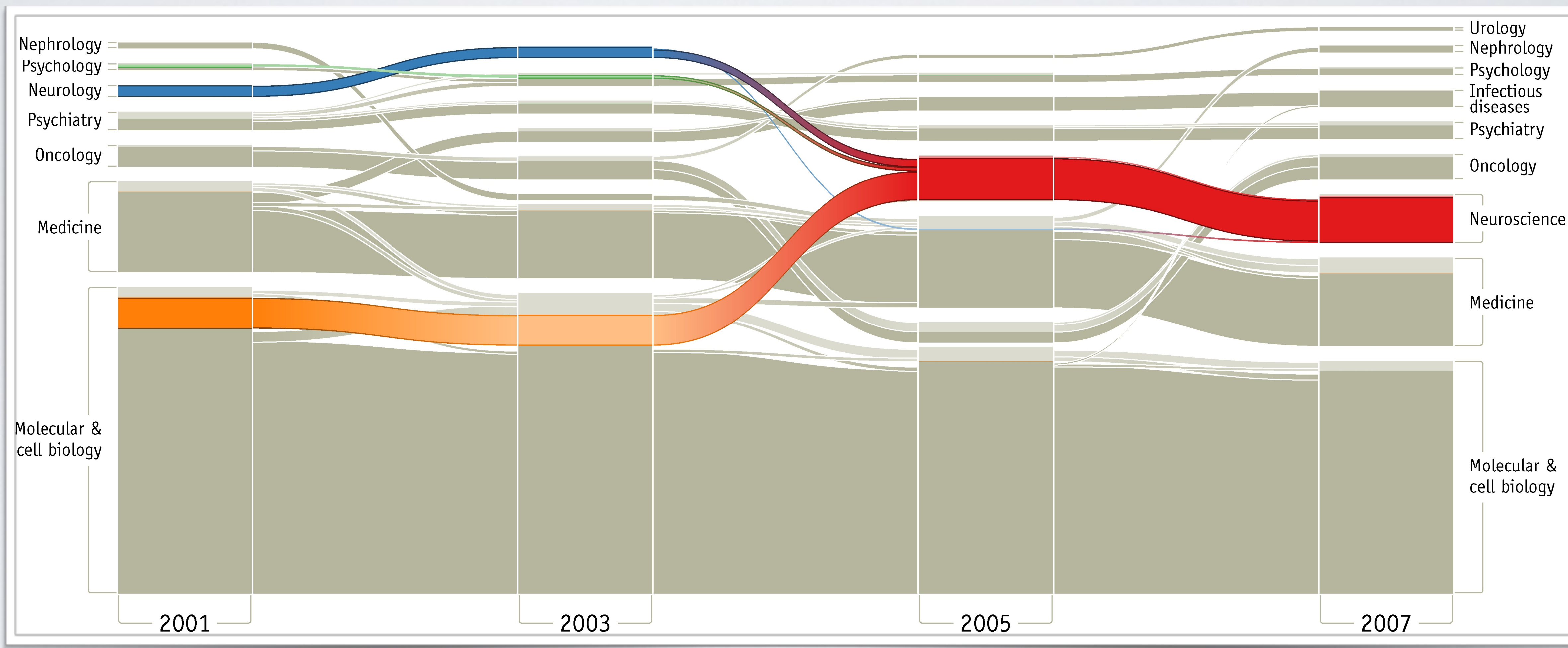


Which one persists ?
-Oldest ?
-Most similar ?
-Larger ?
-…

Community events
(or operations)

COMMUNITY DETECTION

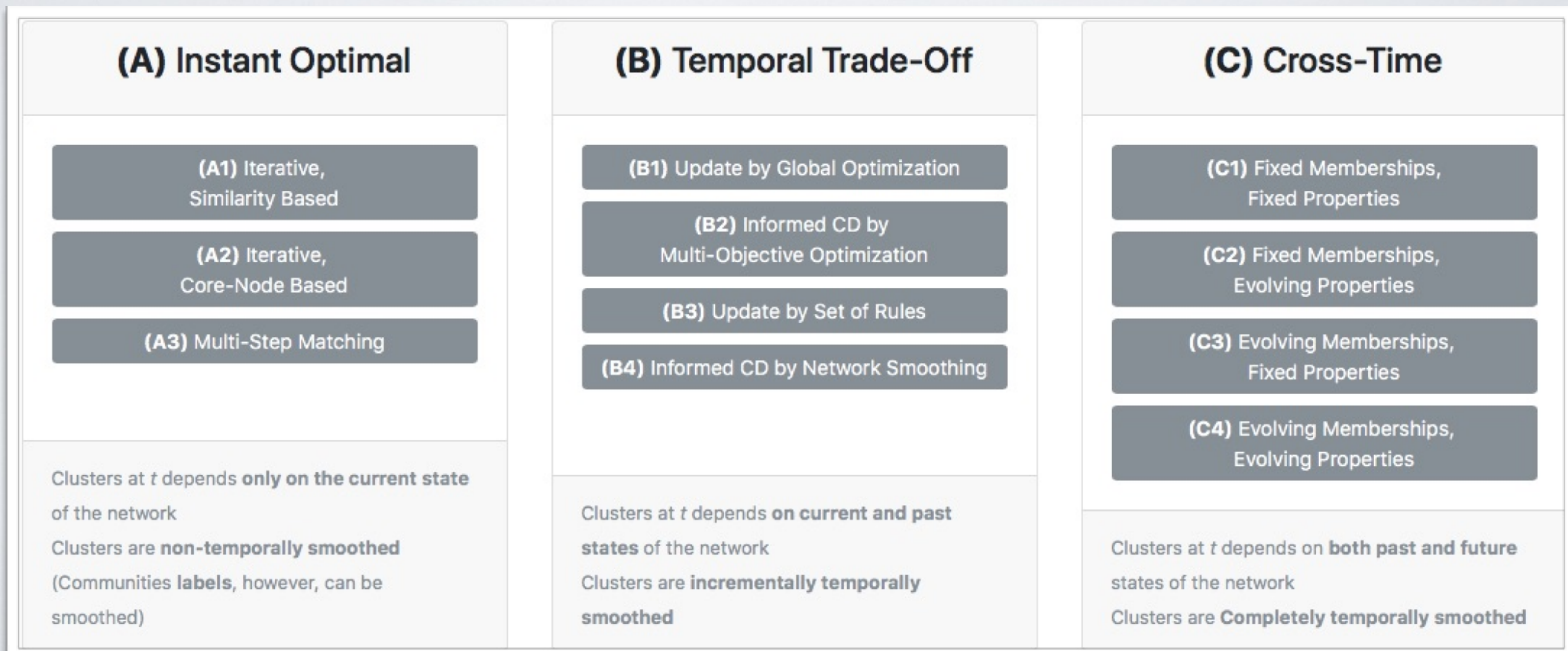
An example



Rosvall et al. 2010

COMMUNITY DETECTION

Over 40 methods published,
but barely any systematic comparison
(nor re-use)



BENCHMARK FOR DYNAMIC COMMUNITIES

STATE OF THE ART

N1
(e.g. with LFR)

N2
(e.g. with LFR)

Permutations

STATE OF THE ART

N1
(e.g. with LFR)

N2
(e.g. with LFR)

Permutations

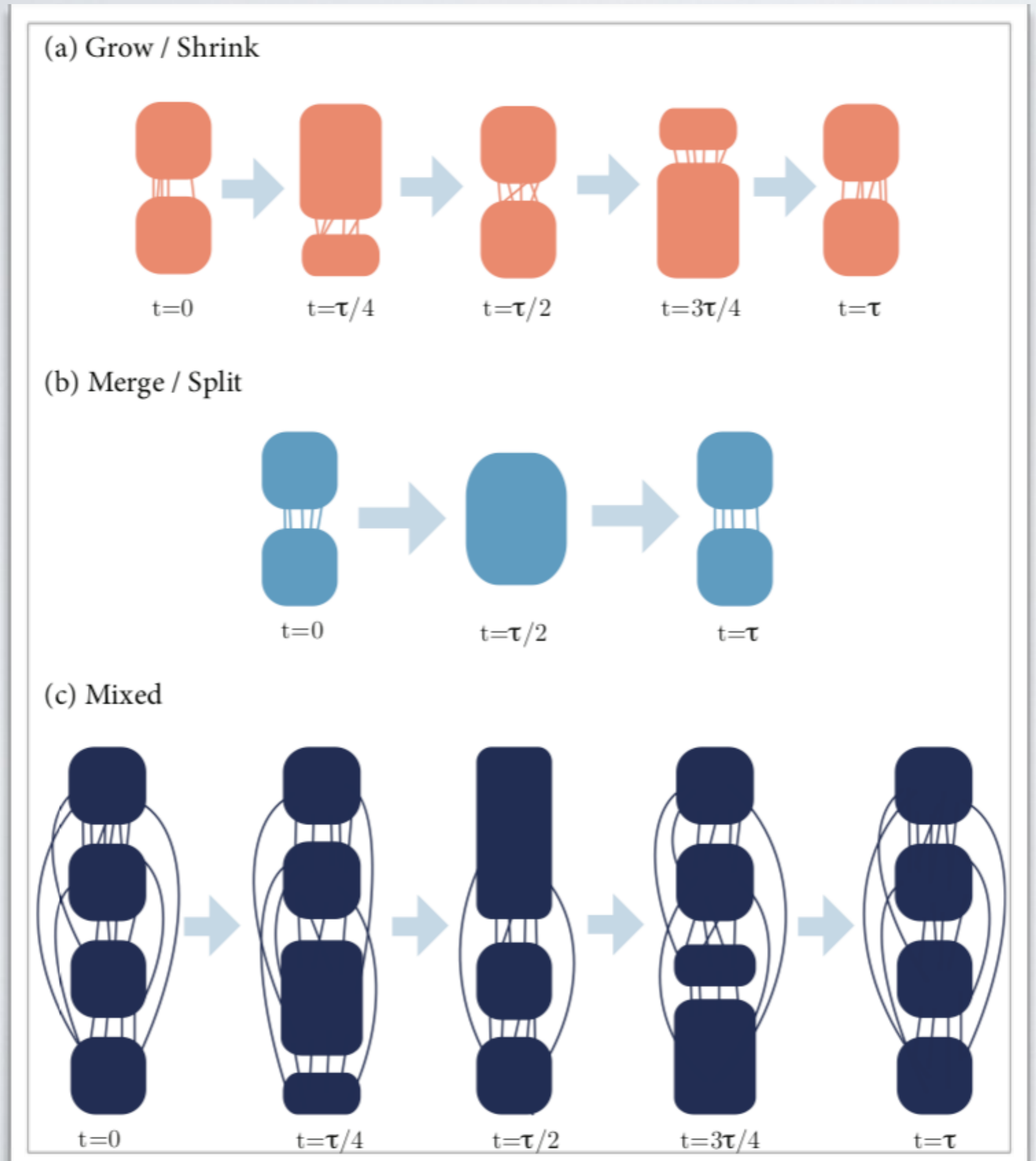
Good, but too simple

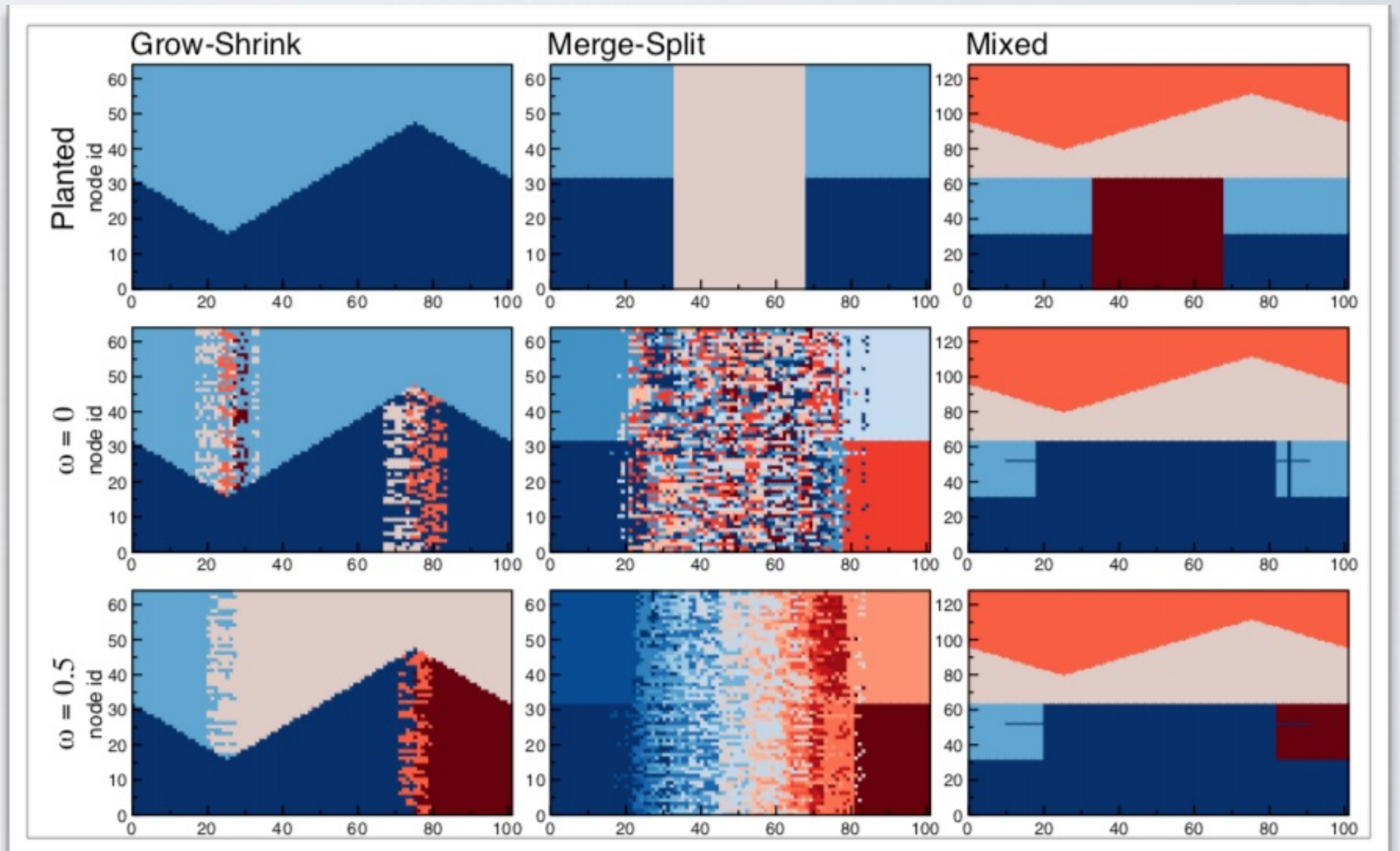
INDEPENDENT SNAPSHOTS

N1 → N2 → … → N3

Nx: a network whose edges are generated independently

SBM BASED MODELS



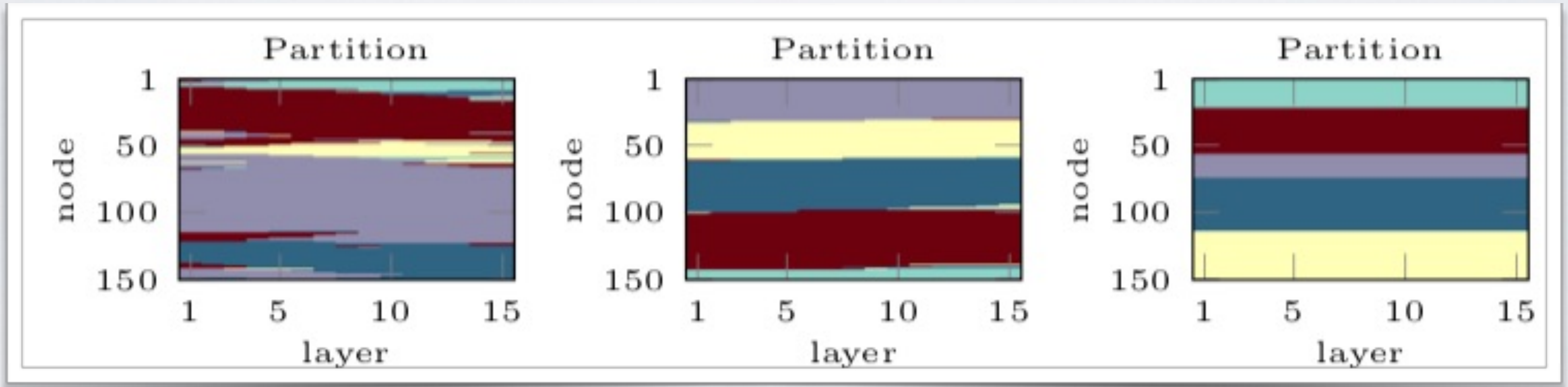


(Ad hoc structure evolution)

- [Granell et al. 2015]

SBM BASED MODELS

[Bazzi et al. 2016]



Model based on a dependency layer
(how similar are affiliations between steps)

OVERLAPPING RANDOM GRAPHS

Evolution configured by numerical parameters

Param.	Meaning	Recomm. Value
N	Number of nodes in G_0	-
T	Number of time steps	-
x_{min}	Minimum node membership	1
x_{max}	Maximum node membership	$N/10$
β_1	Community Membership Exponent	2.5
n_{min}	Minimum size of community	$min(N/100, 20)$
n_{max}	Maximum size of community	$N/10$
β_2	Community Size Exponent	2.5
α	Intra Community Edge Prob. $= \frac{\alpha}{n^\gamma}$	2
γ	Intra Community Edge Prob. $= \frac{\alpha}{n^\gamma}$	0.5
ϵ	Inter Community Edge Probability	$2N^{-1}$
p	Community Event Probability	0.1
p'	Node Event Probability	10^{-2}
λ	Community event sharpness	0.2
t_{effect}	Time steps for community events to take effect	-

TABLE I: Parameters used in the Graph Generator

[Sengupta et al. 2017]

INDEPENDENT SNAPSHOTS

Good community structures, but:

- Limited (# parameters, …)

- Edges are redrawn for each snapshot
  - =>Strong assomption (methods not based on SBM won't see any community)

PROGRESSIVE CHANGE ALGORITHM

Positional arguments

Name	Type	Description	Default
nodes	Integer	Number of nodes	1000
iterations	Integer	Number of iterations	1000
simplified	Boolean	Simplified execution	True

Optional arguments

Flag	Extended Name	Type	Description	Default
-d	--avg_degree	Integer	Average node degree	15
-s	--sigma	Float	Percentage of node's edges within a community	.7
-l	--lbd	Float	Lambda community size distribution	1
-a	--alpha	Float	Alpha degree distribution	3
-p	--prob_action	Float	Probability of node action	1
-r	--prob_renewal	Float	Probability of edge renewal	.8
-q	--quality_threshold	Float	Conductance quality threshold	.3
-n	--new_nodes	Float	Probability of node appearance	0
-j	--delete_nodes	Float	Probability of node vanishing	0
-e	--max_events	Integer	Max number of community events for stable iteration	1

[RDyn, Rossetti et al. 2015]

PROGRESSIVE CHANGE ALGORITHM

Solve the problem of independent snapshots, but:

- Is the evolution of community structure realistic ?
- Crude control on generated communities

OUR PROPOSITION

TWO CONTRIBUTIONS

- Easy way to control community lifycles
- Good edge evolution properties:
  - Stable edges
  - Preserved random structure of communities

EASY COMMUNITY LIFECYCLE

- Atomic events for nodes (and communities)
  - Birth (new community with 1 or several nodes => new node)
  - Death (End of a community with 1 or several nodes =>kill node)
  - Migration (affiliation change)

EASY COMMUNITY LIFECYCLE

- Composed events
  - Merge
    - Migration of nodes
    - Death of merged community (with 0 node)
  - Split
    - Birth (community with 0 nodes)
    - Migration of nodes

EASY COMMUNITY LIFECYCLE

- Complex events
  - Gradual Growth

for $i \leftarrow 0$ **to** n **do**
    Birth(1,i);
    Merge([c,i],[c]);
end

EASY COMMUNITY LIFECYCLE

- Complex events
  - Theseus Ship

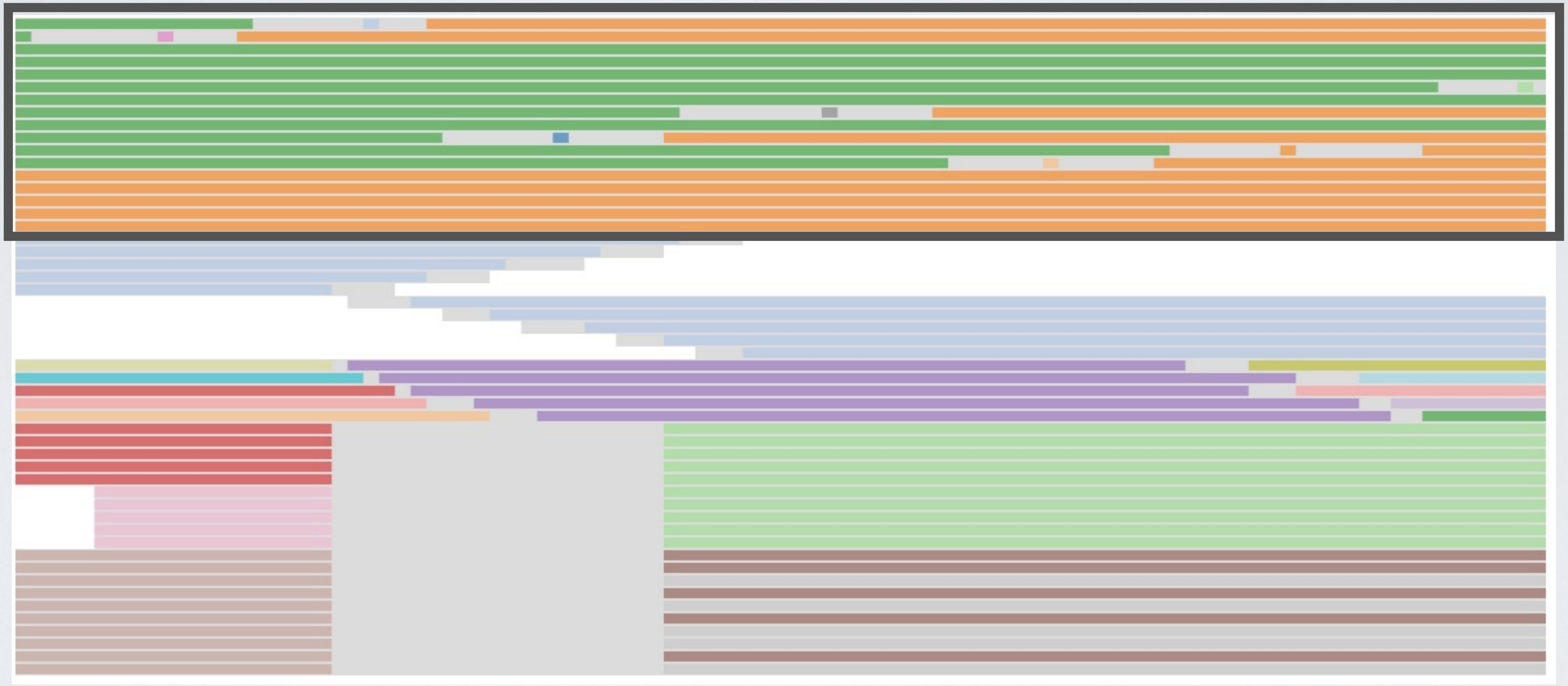
```
initNodes← c.nodes;
for i ← 0 to n do
    Birth(1, ”born”+i);
    Merge([c, ”born”+i], [c]);
    toRemove←initNodes.pop();
    Split(c, [c, ”killed”+i], [c.nodes-toRemove, toRemove]);
    Death(”killed”+i);
end
```

EASY COMMUNITY LIFECYCLE

- One can programmatically design a complex scenario, or design a specific one

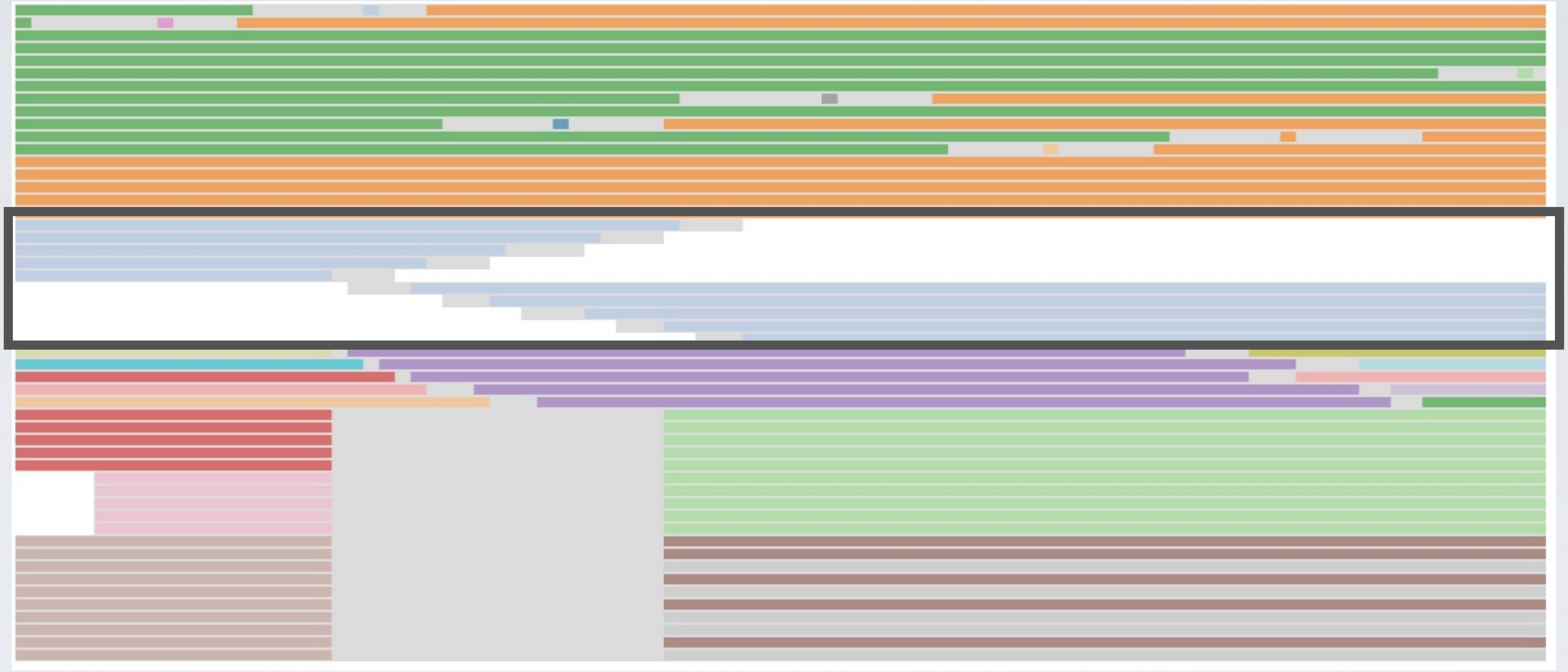
SYNTHETIC NETWORK

Migration



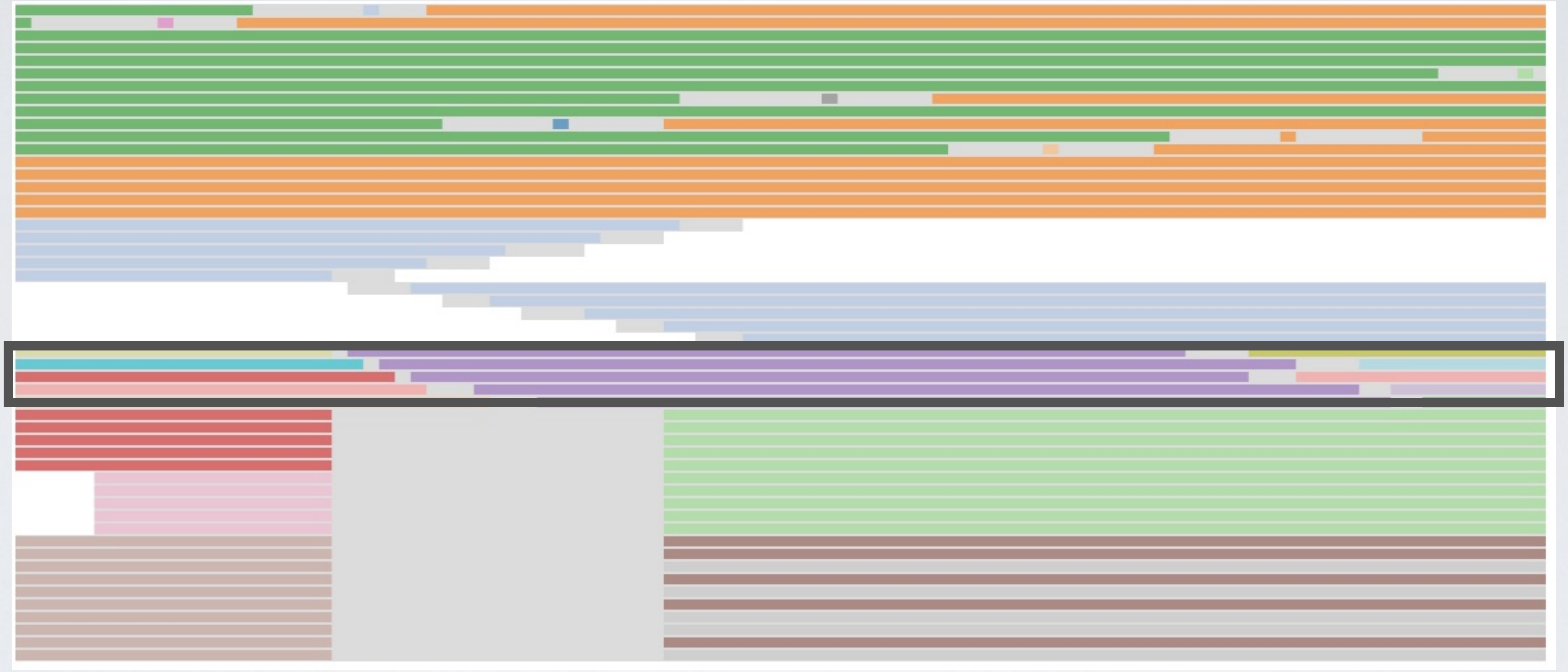
SYNTHETIC NETWORK

Theseus boat



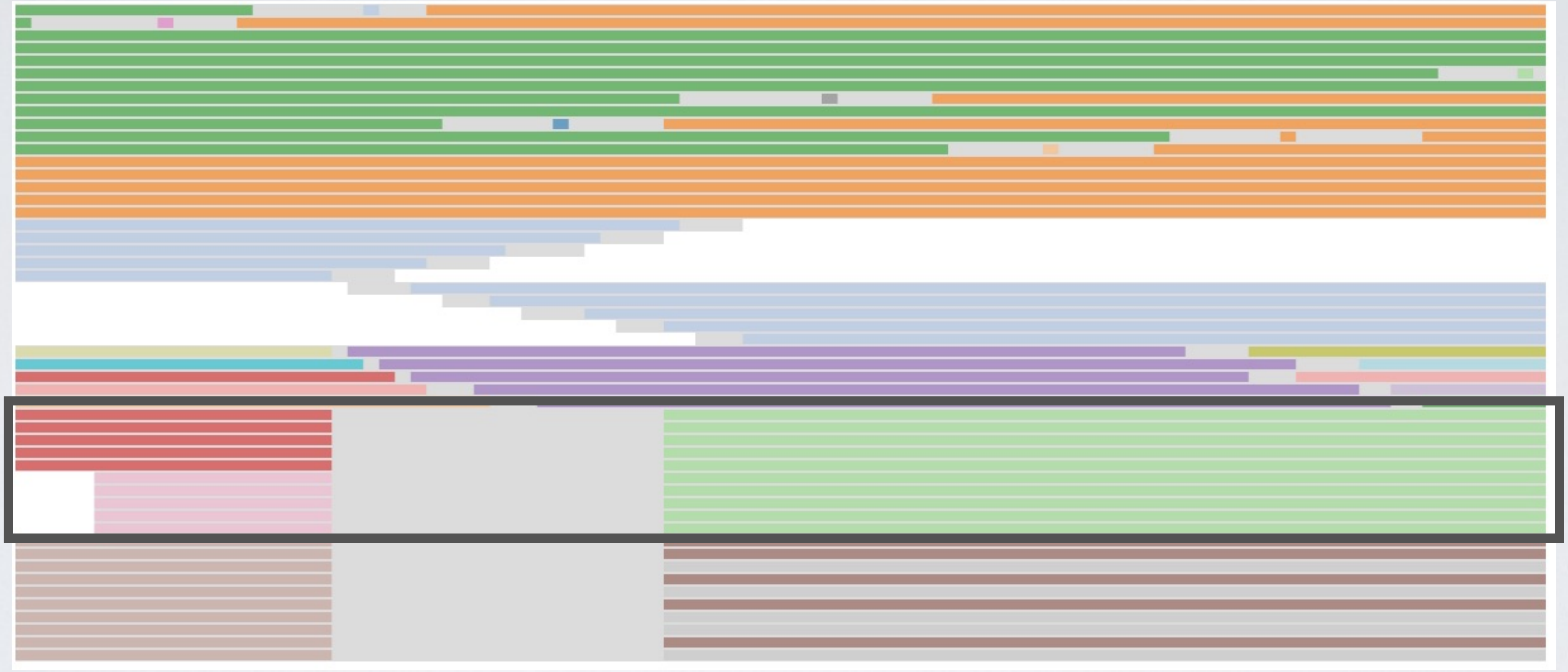
SYNTHETIC NETWORK

Birth and death



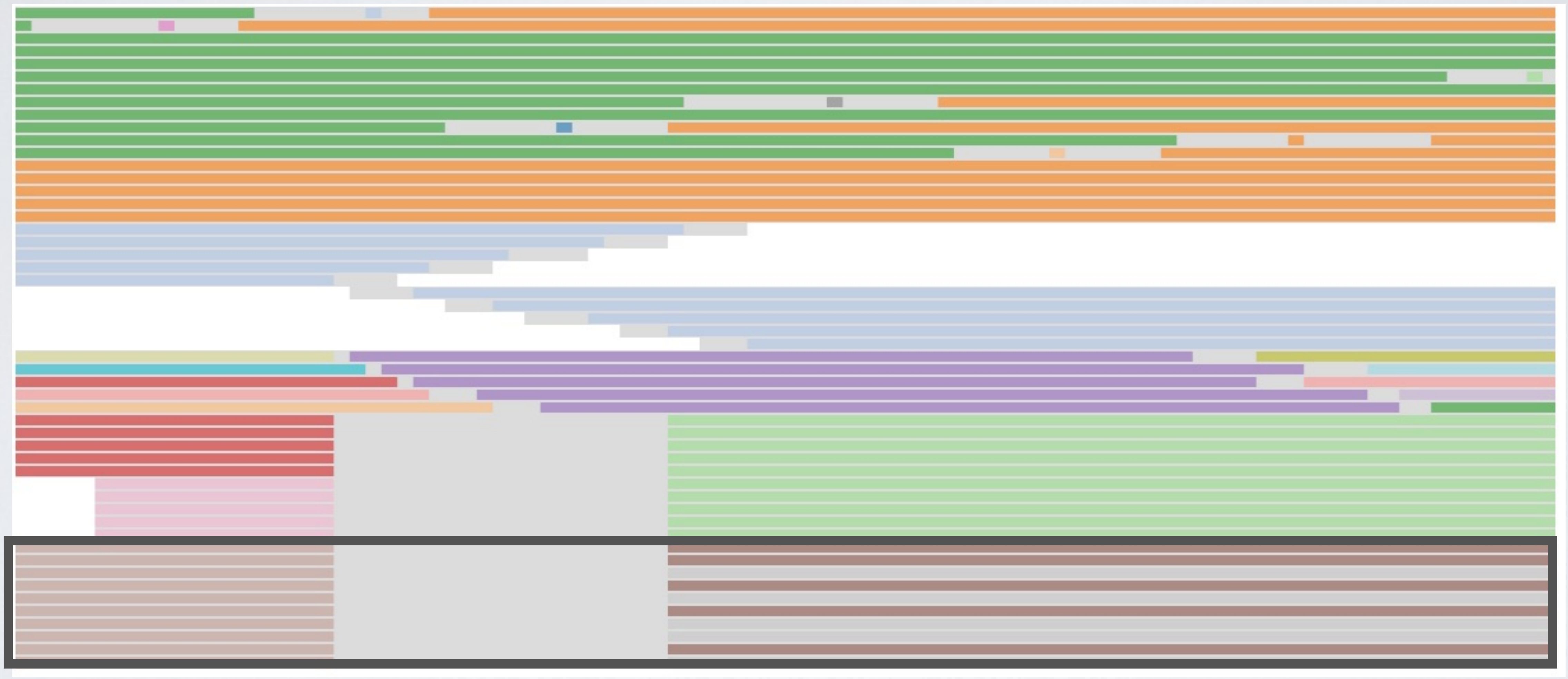
SYNTHETIC NETWORK

Merge



SYNTHETIC NETWORK

Division



CONTRIBUTION 2: EDGE GENERATION

EDGE GENERATION

- We want edges to be stable: not an independent process
- We assume an SBM-like process:
  - We know the affiliations
  - We know how many edges we want (for each community)

EDGE GENERATION

- The problem: community C at t1 must differ from C at t2:
  - Different nodes
  - Different # of edges
- Two simple ideas to do that:
  - Keep the current structure and modify only at the margin
  - Pick edges of the new community randomly with a probability $p(\alpha,\delta,\phi)$
    - With δ the previous presence of the edge, and α a tunable parameter and ϕ the objective density

EDGE GENERATION

- Problem: Any method that preserves the previous structure introduces a **historical bias**:
- Imagine 2 communities of size 4 and density 1
  - 12 edges
- Merge into 1 community (of size 8), and desired density of 0.23
  - 13 edges
- Resulting community: 2 cliques connected by a single edge

SOLUTION

- For each pair of node, associate a random value a∈[0,1]
  - Interpreted as the “affinity” of each pair of node
- When a community change
  - 1)compute the desired number of edges x
  - 2)pick the x pairs of nodes of highest *affinity* value inside the community
- When several modifications needed, add/remove edges in a random order until reaching the new state

FUTURE WORK

- Is there a flow ?
- Make everything work well
- Compare existing methods empirically
  - How to define properly the ground truth ?
  - On what scenario ?
  - Should I add a random generator of scenario ?
  - Which methods ? (Currently about 10 runnable ones)
- Evaluate
  - How exactly ?

FUTURE WORK

- If anyone wants to help, you're welcome !
  - (hard to find time to code…)

UNRELATED AD :)

UNIVERSITÉ DE LYON

LIRIS

IXXI

ANR — PROJET FINANCÉ PAR L'ANR — PROJECT FUNDED BY THE ANR

MACHINE LEARNING AND NETWORK ANALYSIS FOR UNDERSTANDING THE NATURE OF ACTIVITIES IN CRYPTOCURRENCIES

Positions opening:

1) One **Master Internship** (5/6 months, from Feb. 2019 -)
2) One **PhD position** (3 years funding, Sept. 2019 -)

KEYWORDS

CRYPTO-CURRENCIES | COMPLEX NETWORK ANALYSIS | MACHINE LEARNING | INTERDISCIPLINARY RESEARCH

QUESTIONS?
RECOMMENDATIONS?
IDEAS?