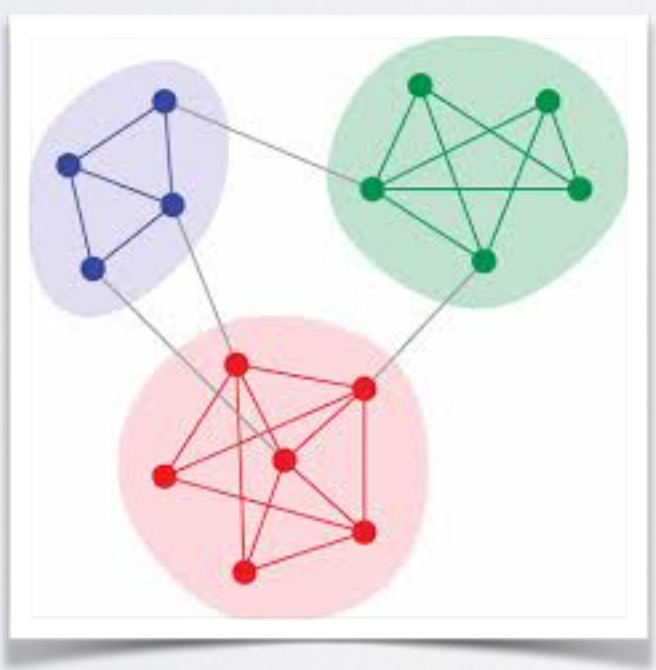
DYNAMIC COMMUNITY DETECTION

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

COMMUNITY DETECTION

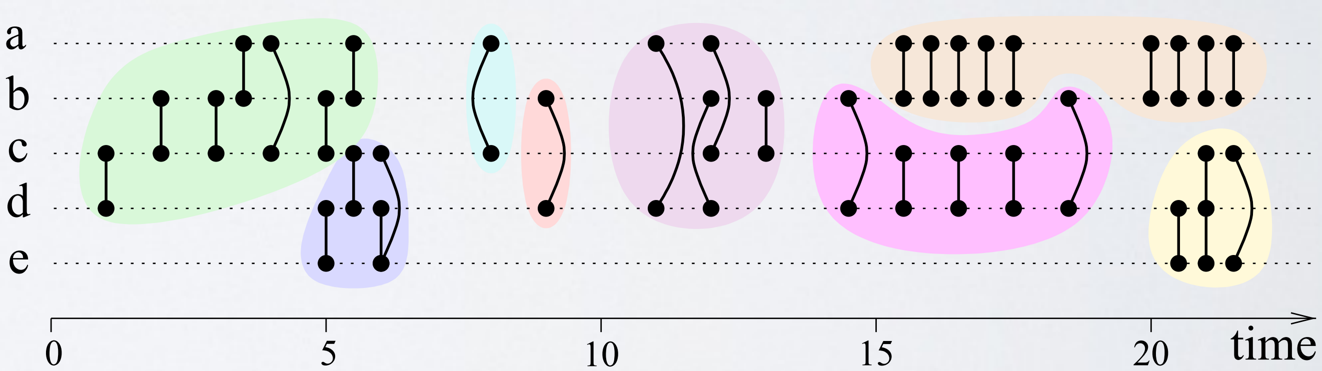
Static networks

Sets of nodes



Dynamic Networks

Sets of periods of nodes

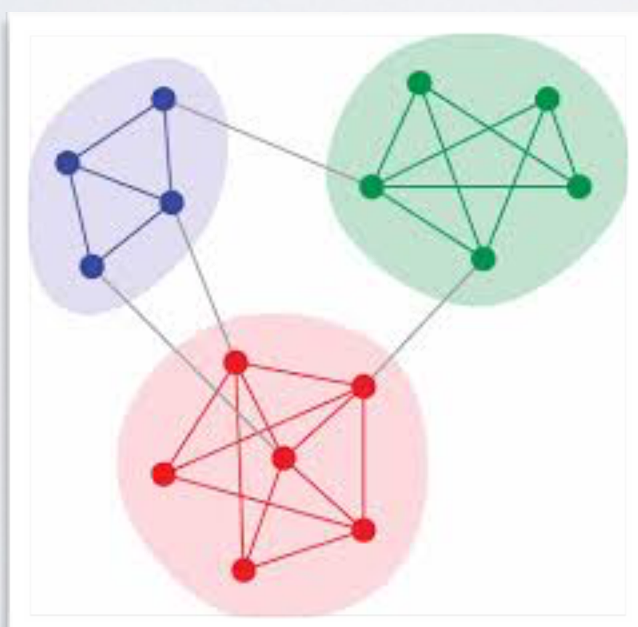


[Viard 2016]

COMMUNITY DETECTION

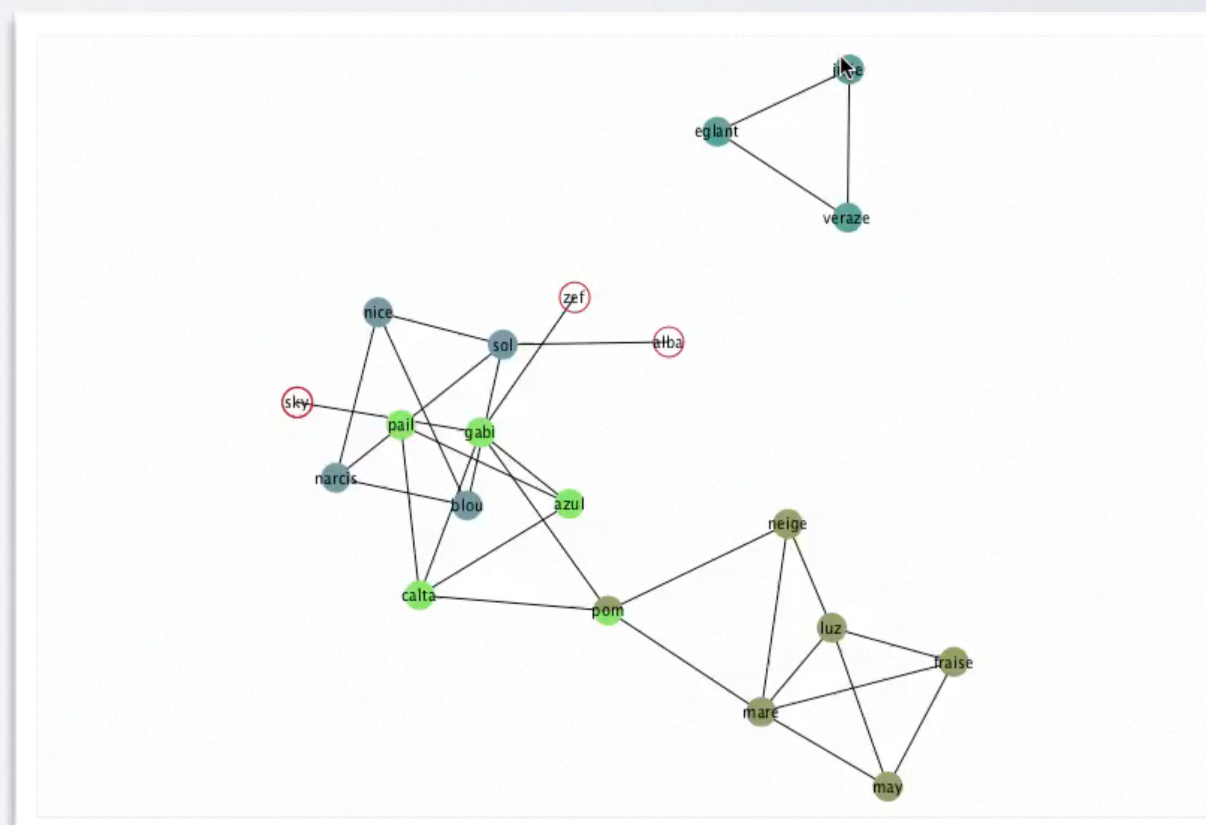
Static networks

Sets of nodes



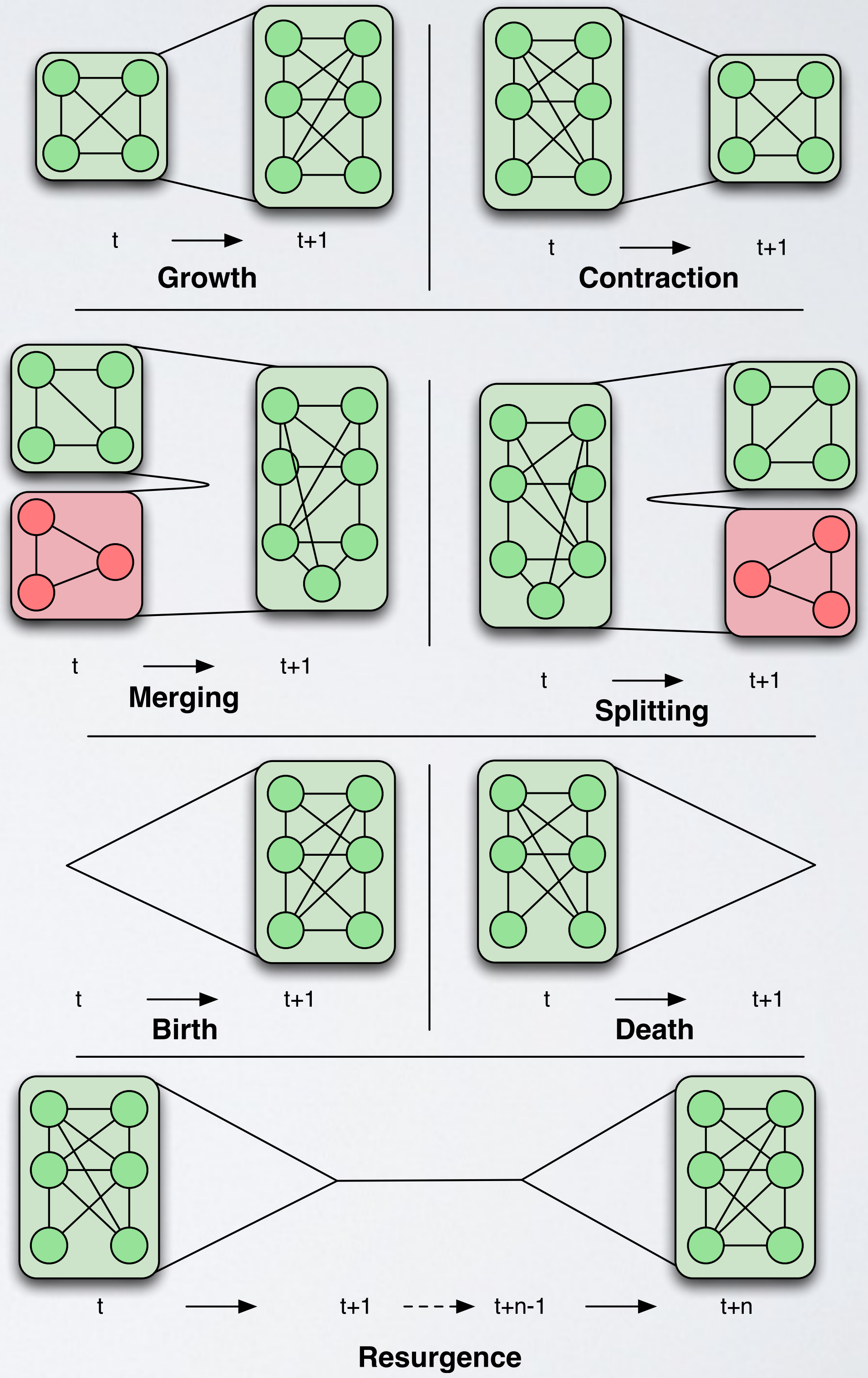
Dynamic Networks

Sets of periods of nodes

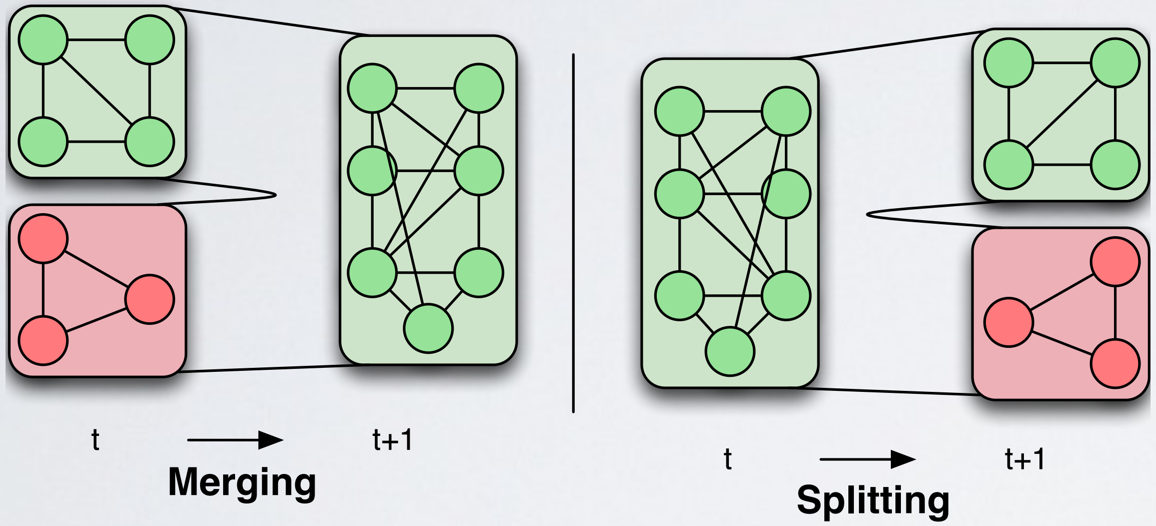


COMMUNITY DETECTION

Community events
(or operations)



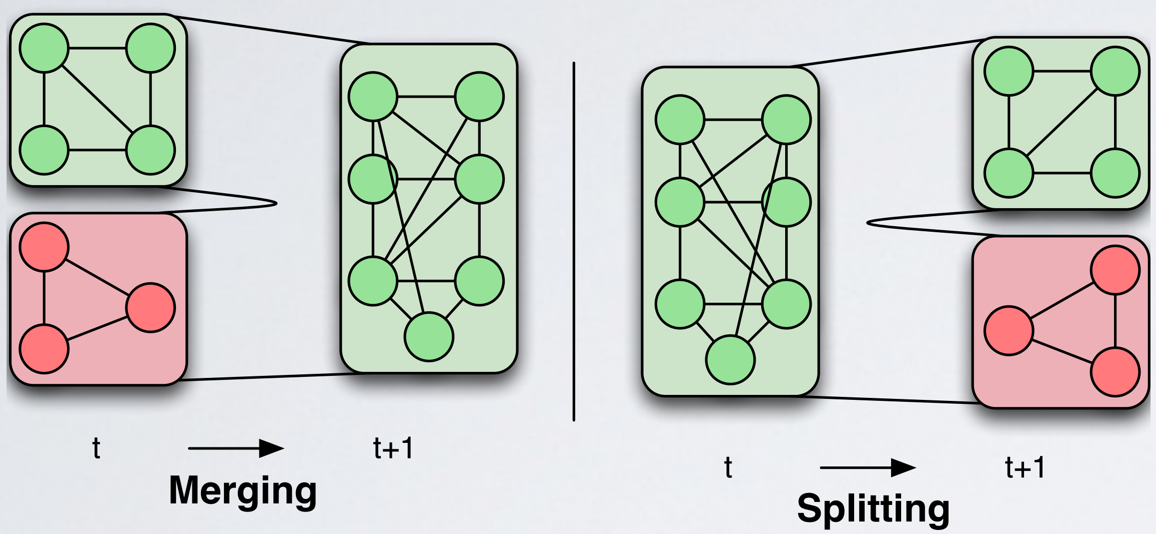
COMMUNITY DETECTION



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

Community events
(or operations)

COMMUNITY DETECTION

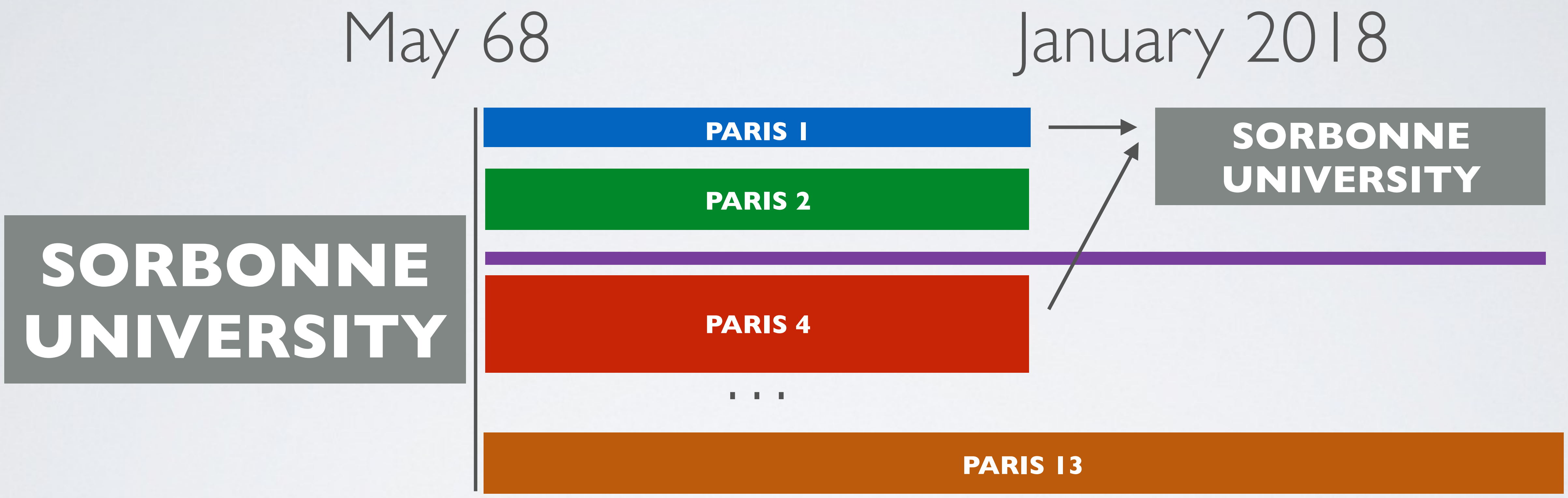


Which one persists ?
-Oldest ?
-Most similar ?
-Larger ?
-...

Ship of Theseus paradox

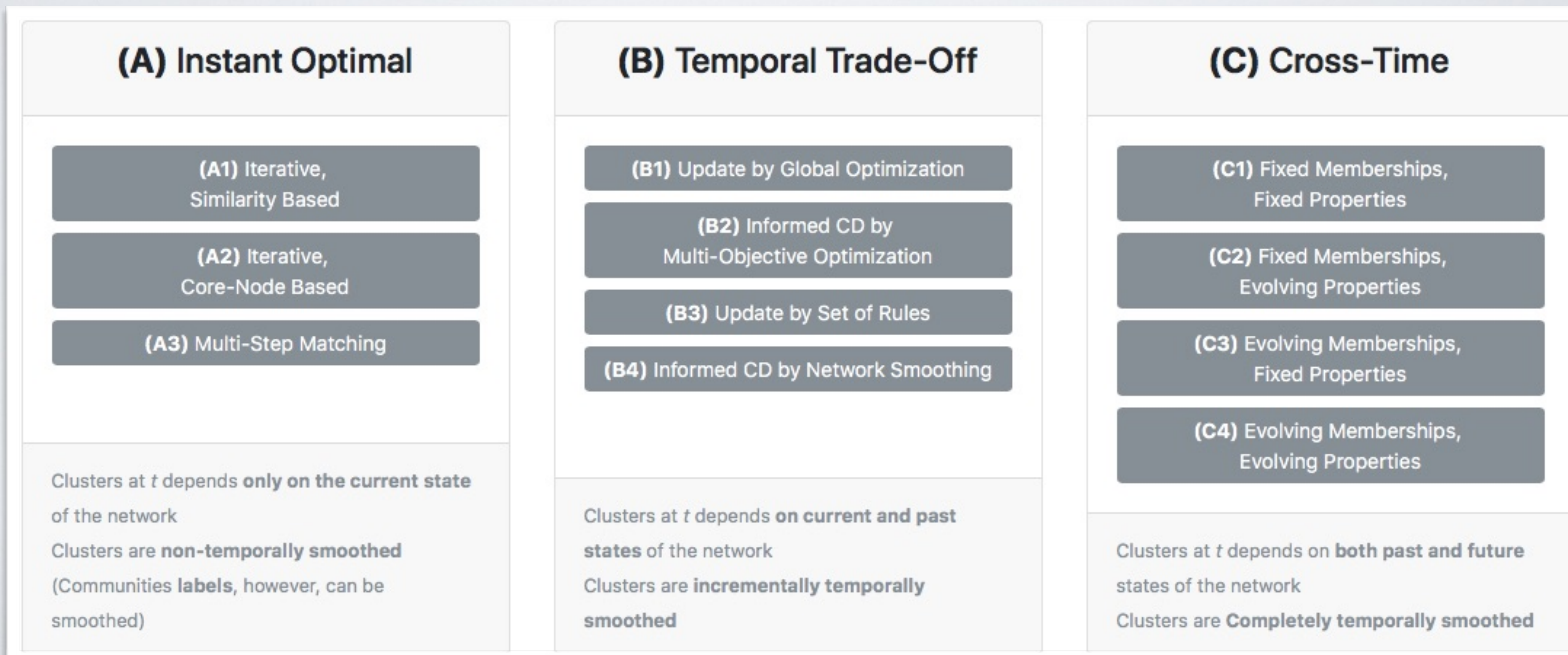
COMMUNITY DETECTION

A very concrete problem :



Who gets Marie Curie Nobel Prize points in the Shanghai University Ranking ?

COMMUNITY DETECTION



SYNTHETIC NETWORK

Instant Optimal:
Greene et al. 2011

- Input : a graph series
- Algorithm:
  - Detect communities on each snapshot using static algo
  - Compute Jaccard similarity between each pair of communities in successive graphs
  - Associate communities with similarity>Threshold

COMMUNITY DETECTION

(A) Instant Optimal

- **(A1)** Iterative, Similarity Based
- **(A2)** Iterative, Core-Node Based
- **(A3)** Multi-Step Matching

Clusters at *t* depends **only on the current state** of the network
Clusters are **non-temporally smoothed**
(Communities **labels**, however, can be smoothed)

(B) Temporal Trade-Off

- **(B1)** Update by Global Optimization
- **(B2)** Informed CD by Multi-Objective Optimization
- **(B3)** Update by Set of Rules
- **(B4)** Informed CD by Network Smoothing

Clusters at *t* depends **on current and past states** of the network
Clusters are **incrementally temporally smoothed**

(C) Cross-Time

- **(C1)** Fixed Memberships, Fixed Properties
- **(C2)** Fixed Memberships, Evolving Properties
- **(C3)** Evolving Memberships, Fixed Properties
- **(C4)** Evolving Memberships, Evolving Properties

Clusters at *t* depends on **both past and future** states of the network
Clusters are **Completely temporally smoothed**

SYNTHETIC NETWORK

Temporal trade-off:
Cazabet et al. 2010

- Input : an ordered list of modifications
- Algorithm:
  - For each edge creation:
    - Decide locally to update involved communities (density>Threshold)
    - Decide locally to create a new community (new clique size>k outside communities)
  - For each edge deletion:
    - Decide locally to update involved communities (density < Threshold)
    - Decide locally to delete communities (nb nodes < k)

COMMUNITY DETECTION

(A) Instant Optimal

- **(A1)** Iterative, Similarity Based
- **(A2)** Iterative, Core-Node Based
- **(A3)** Multi-Step Matching

Clusters at *t* depends **only on the current state** of the network
Clusters are **non-temporally smoothed**
(Communities **labels**, however, can be smoothed)

(B) Temporal Trade-Off

- **(B1)** Update by Global Optimization
- **(B2)** Informed CD by Multi-Objective Optimization
- **(B3)** Update by Set of Rules
- **(B4)** Informed CD by Network Smoothing

Clusters at *t* depends **on current and past states** of the network
Clusters are **incrementally temporally smoothed**

(C) Cross-Time

- **(C1)** Fixed Memberships, Fixed Properties
- **(C2)** Fixed Memberships, Evolving Properties
- **(C3)** Evolving Memberships, Fixed Properties
- **(C4)** Evolving Memberships, Evolving Properties

Clusters at *t* depends on **both past and future** states of the network
Clusters are **Completely temporally smoothed**

SYNTHETIC NETWORK

Cross-Time:
Mucha et al. 2010

- Input : a graph series
- Optimise a global quality function, with two parts:
  - A weighted average of the modularity at each snapshot
  - A metric of node stability (max when all nodes always in the same community)
- A parameter **ω** allows to tune which aspect is more important

COMMUNITY DETECTION

(A) Instant Optimal

- **(A1)** Iterative, Similarity Based
- **(A2)** Iterative, Core-Node Based
- **(A3)** Multi-Step Matching

Clusters at *t* depends **only on the current state** of the network
Clusters are **non-temporally smoothed**
(Communities **labels**, however, can be smoothed)

(B) Temporal Trade-Off

- **(B1)** Update by Global Optimization
- **(B2)** Informed CD by Multi-Objective Optimization
- **(B3)** Update by Set of Rules
- **(B4)** Informed CD by Network Smoothing

Clusters at *t* depends **on current and past states** of the network
Clusters are **incrementally temporally smoothed**

(C) Cross-Time

- **(C1)** Fixed Memberships, Fixed Properties
- **(C2)** Fixed Memberships, Evolving Properties
- **(C3)** Evolving Memberships, Fixed Properties
- **(C4)** Evolving Memberships, Evolving Properties

Clusters at *t* depends on **both past and future** states of the network
Clusters are **Completely temporally smoothed**

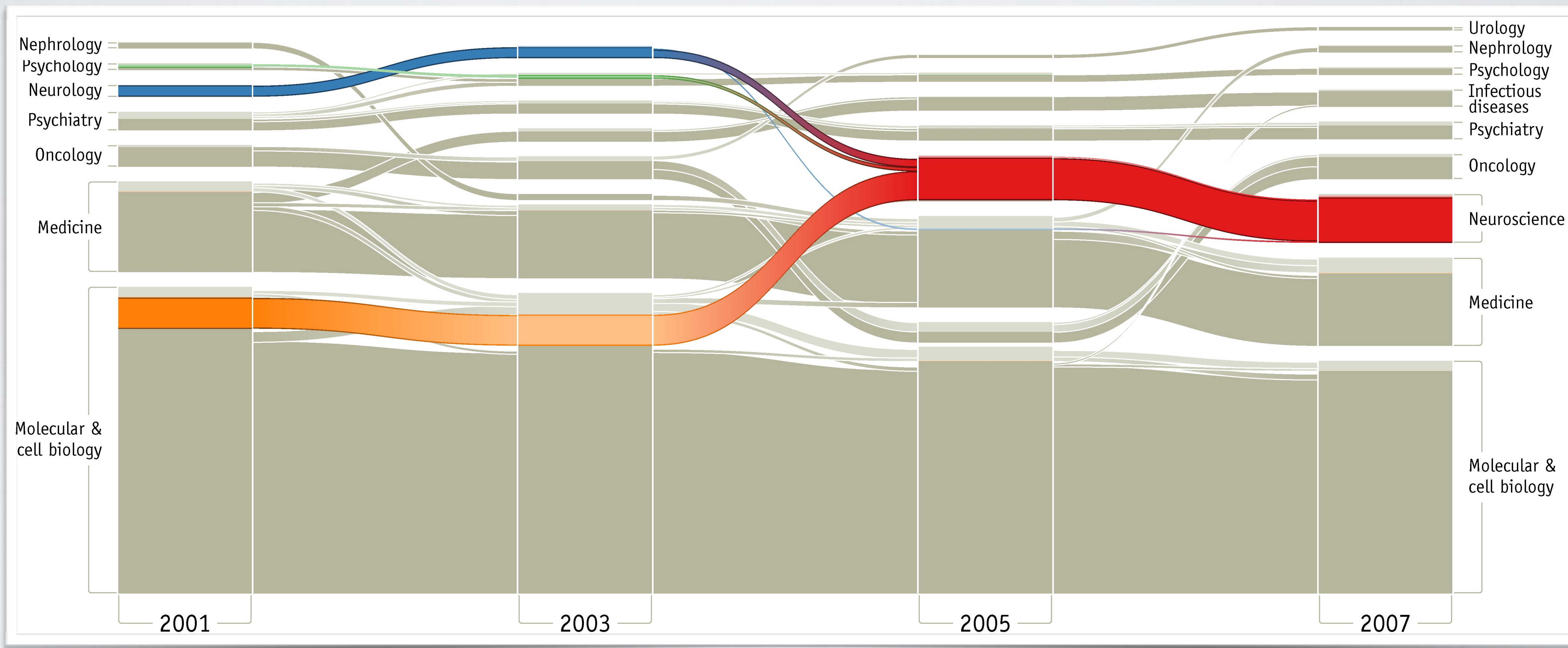
Snapshots/Temporal networks

SBM, Modularity, Conductance, …

Overlapping YES/NO

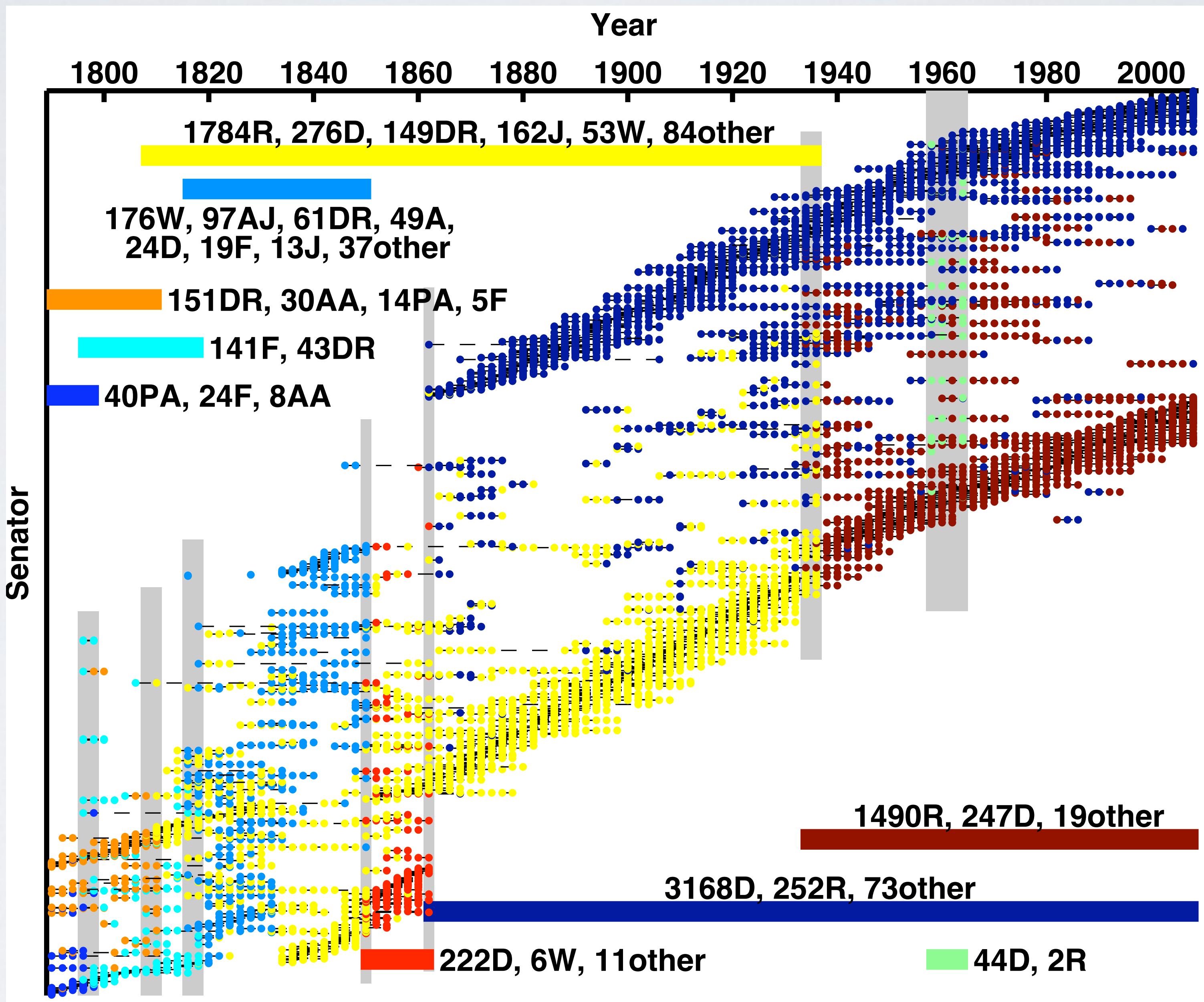
COMMUNITY DETECTION

Some examples of applications



Rosvall et al. 2010

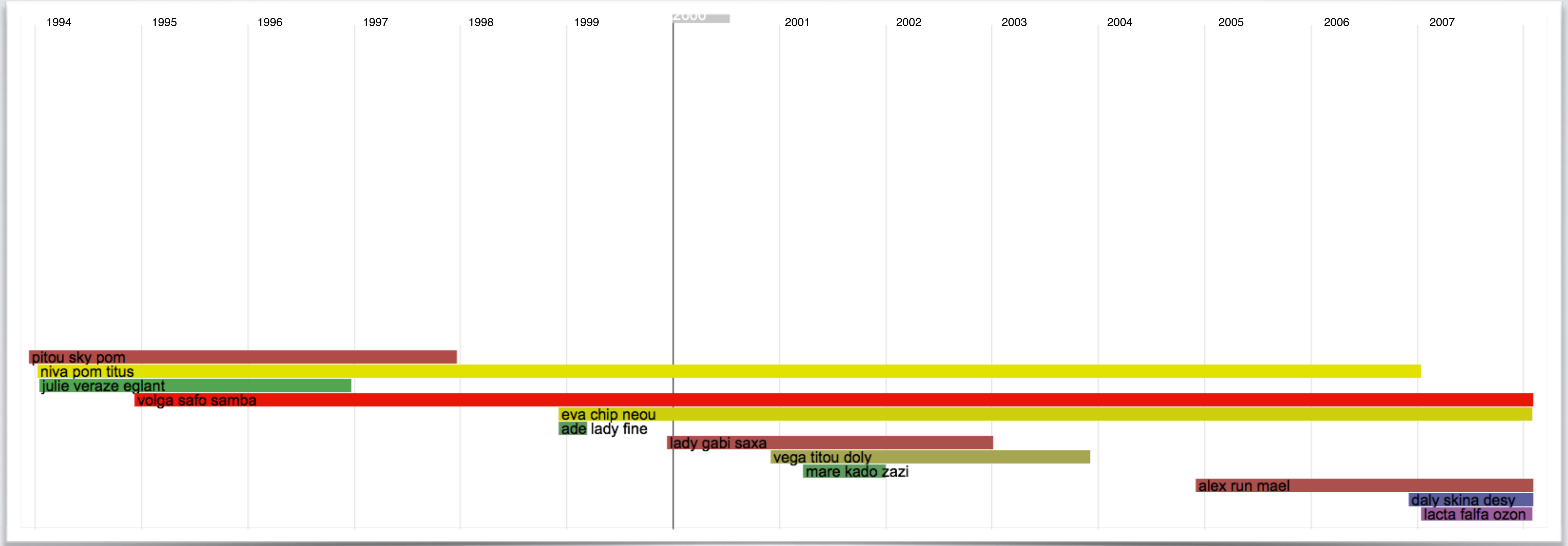
COMMUNITY DETECTION



R : Républicains
D : Démocrates

Mucha et al. 2010

COMMUNITY DETECTION



DCD IN PRACTICE

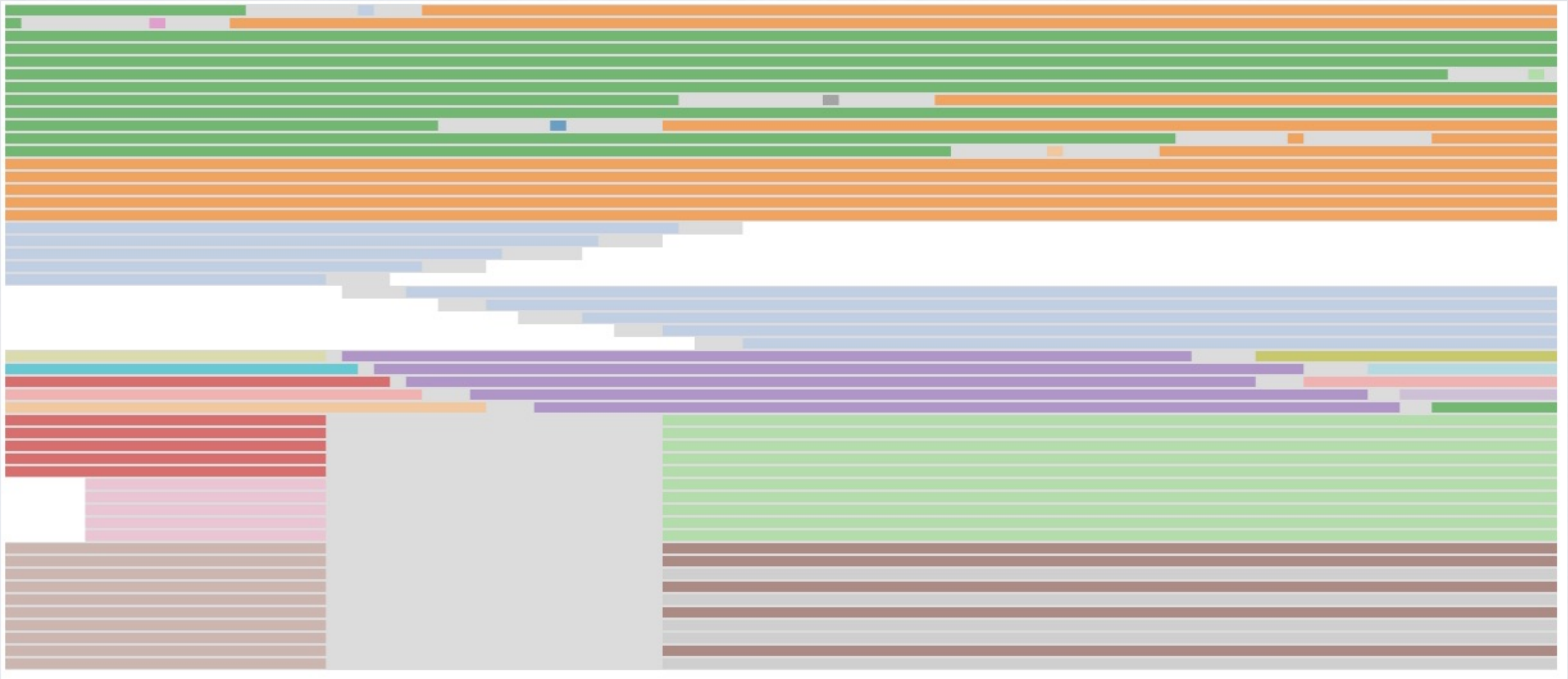
DCD IN PRACTICE

- Tests on synthetic networks
  - We know what we want to find
  - We run algorithms and check the results
- Tests on real networks
  - Start from a real dataset
  - Transform into an appropriate dynamic network (if needed)
  - Run algorithms and try to interpret results

SYNTHETIC NETWORK

- Using a dynamic network generator
- Testing several cases:
  - Continuation
  - Growth / Shrink
  - Merge
  - Division
  - Birth / Death
  - Theseus boat
  - Migration

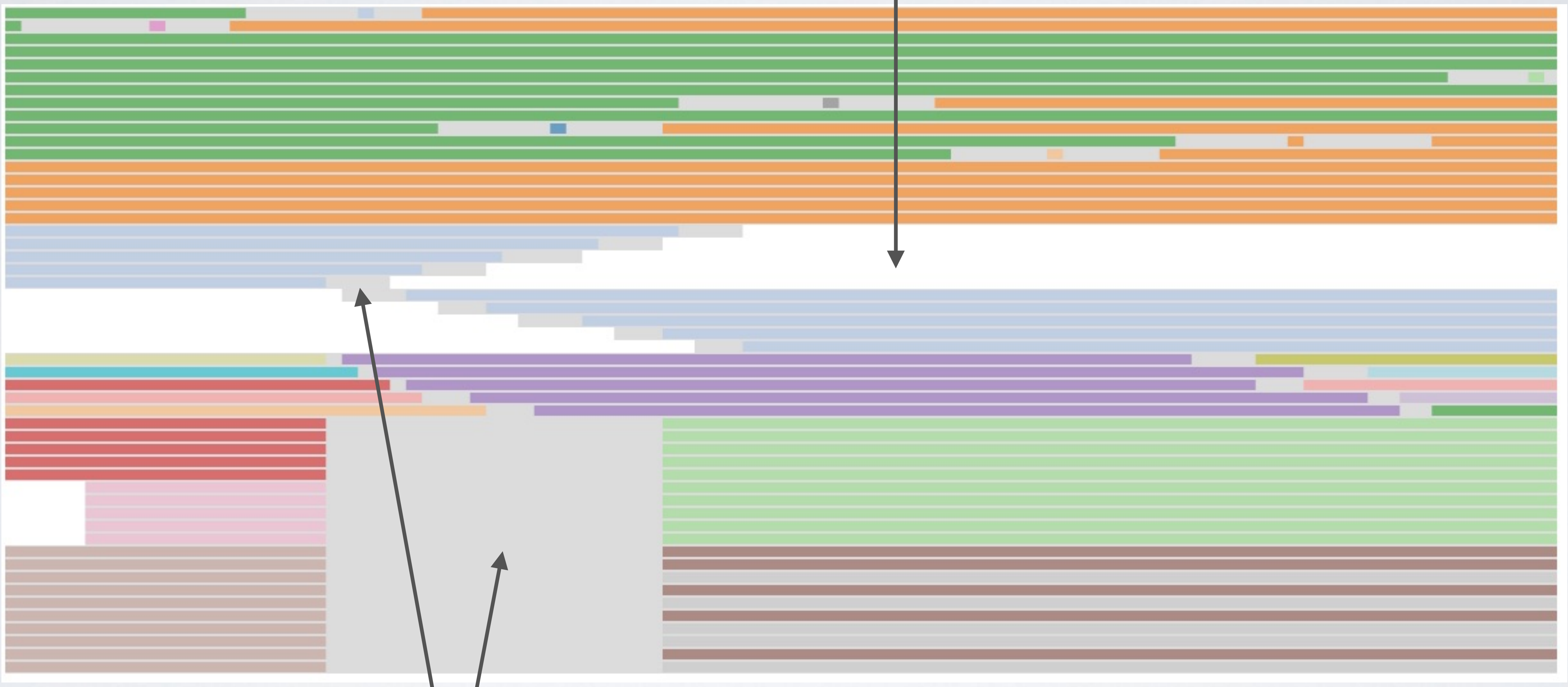
SYNTHETIC NETWORK



SYNTHETIC NETWORK

SYNTHETIC NETWORK

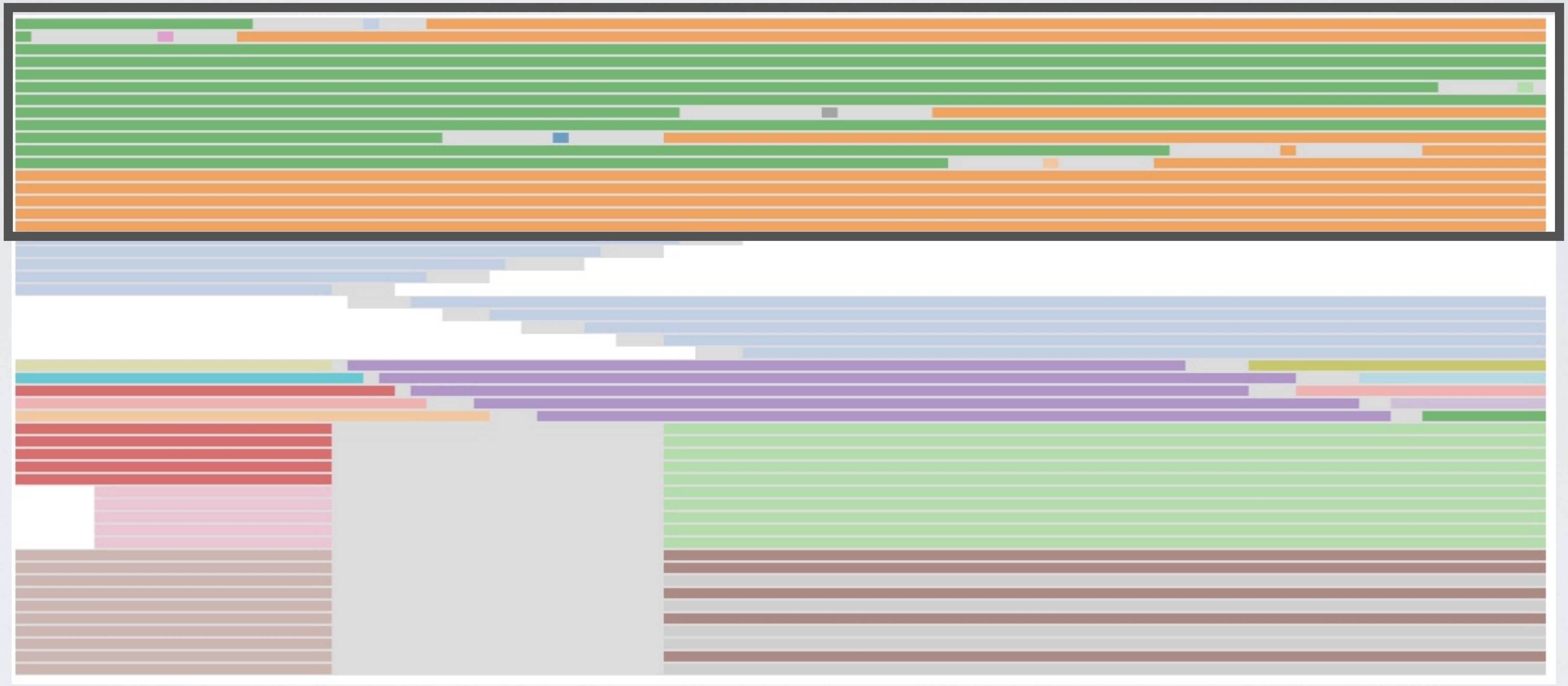
Node not present



Alive node, no known community

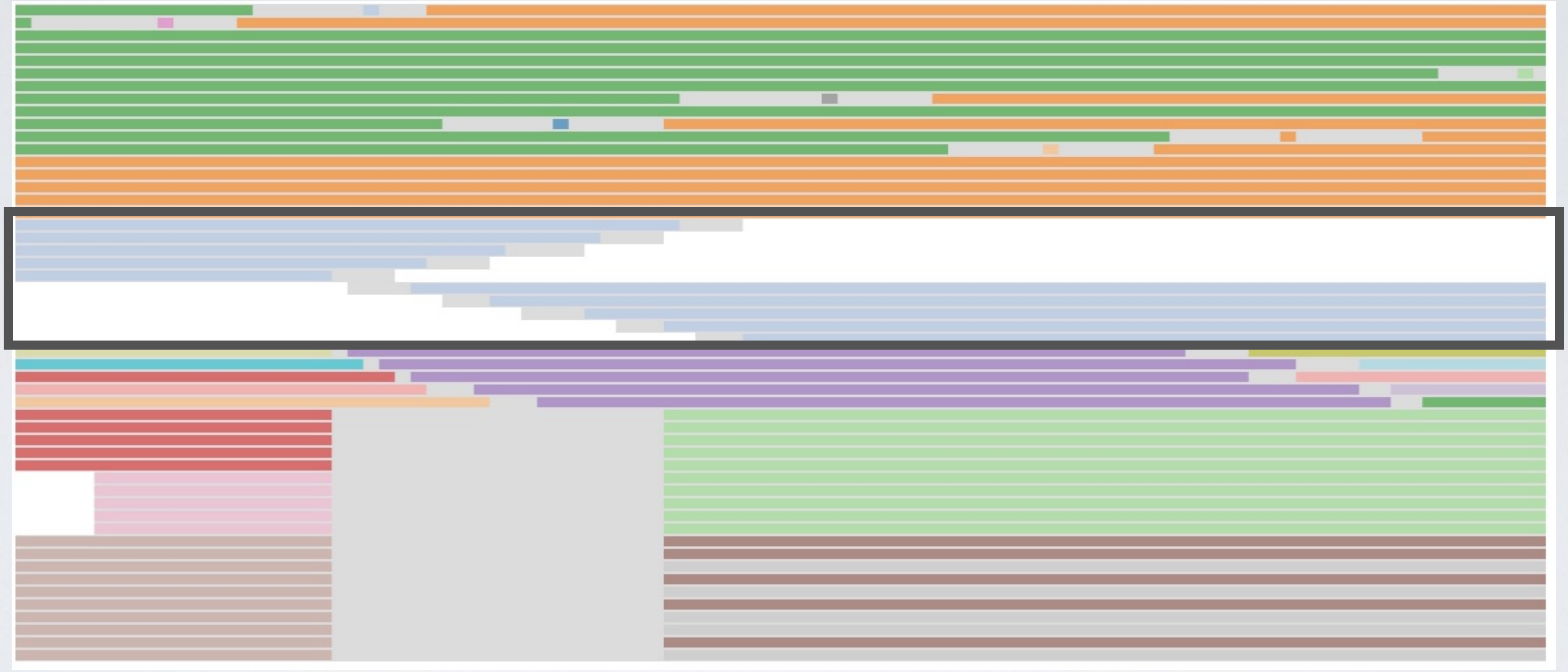
SYNTHETIC NETWORK

Migration



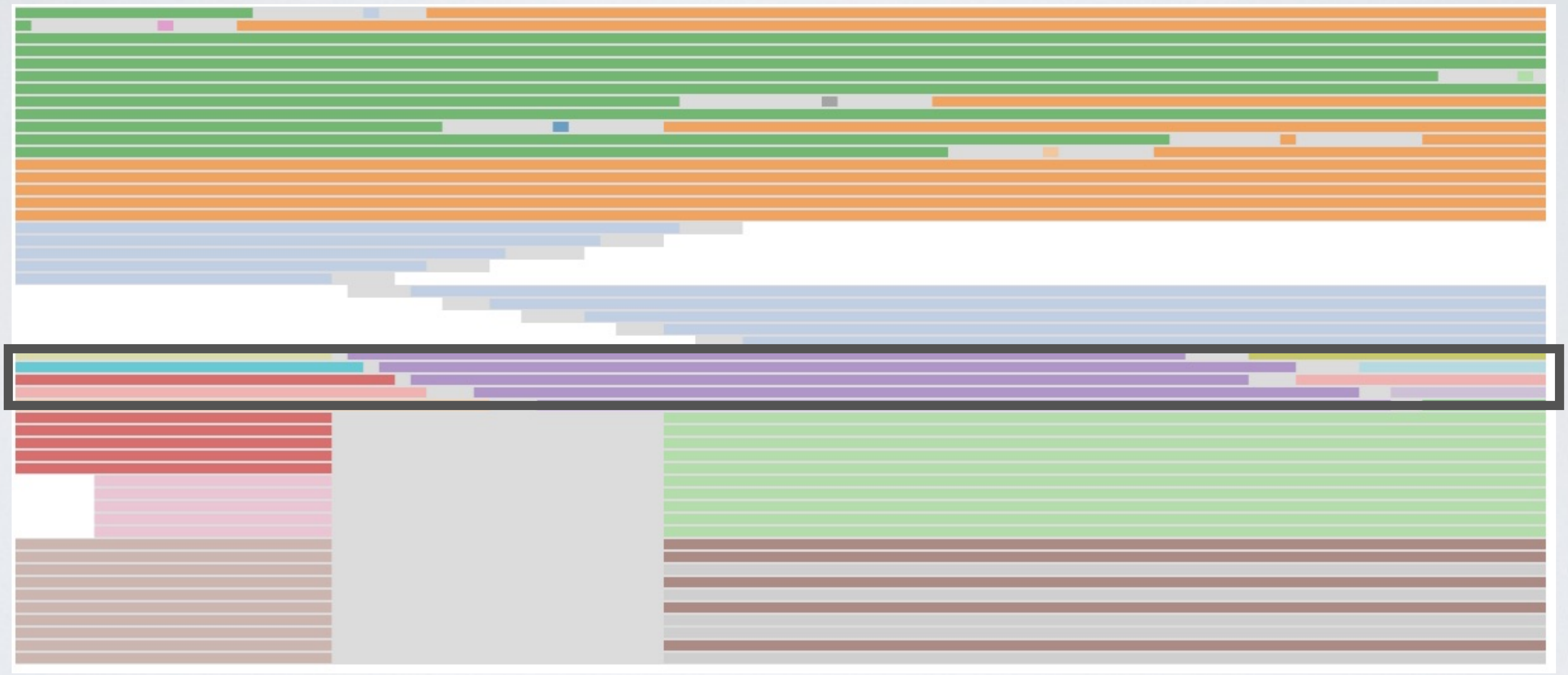
SYNTHETIC NETWORK

Theseus boat



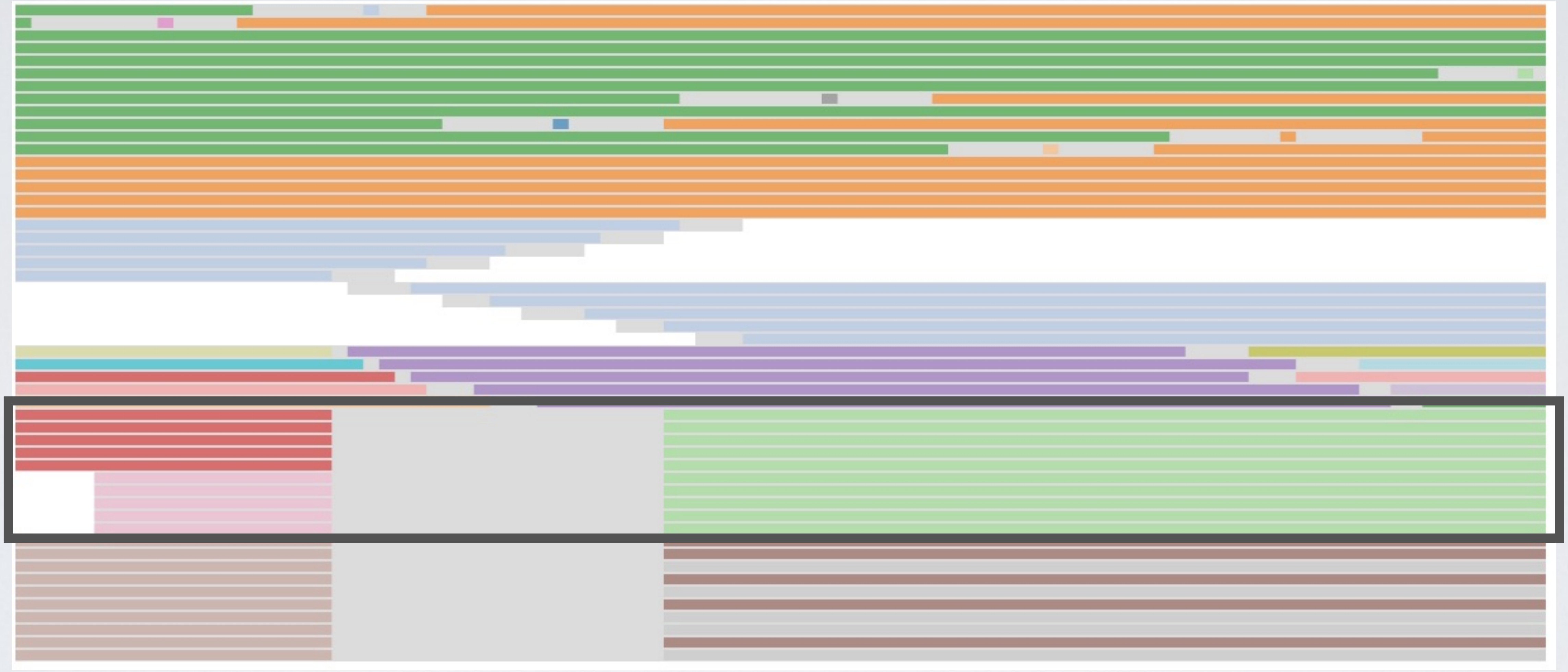
SYNTHETIC NETWORK

Birth and death



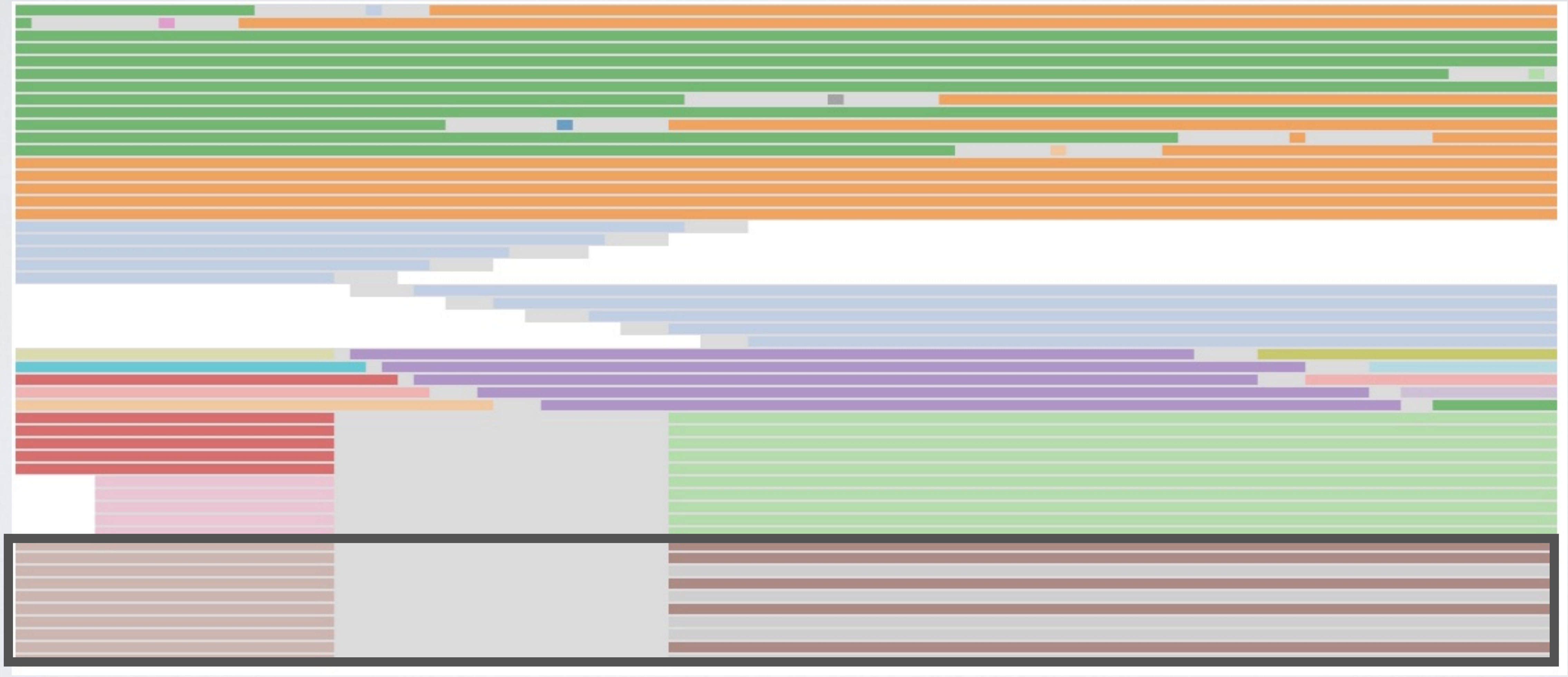
SYNTHETIC NETWORK

Merge



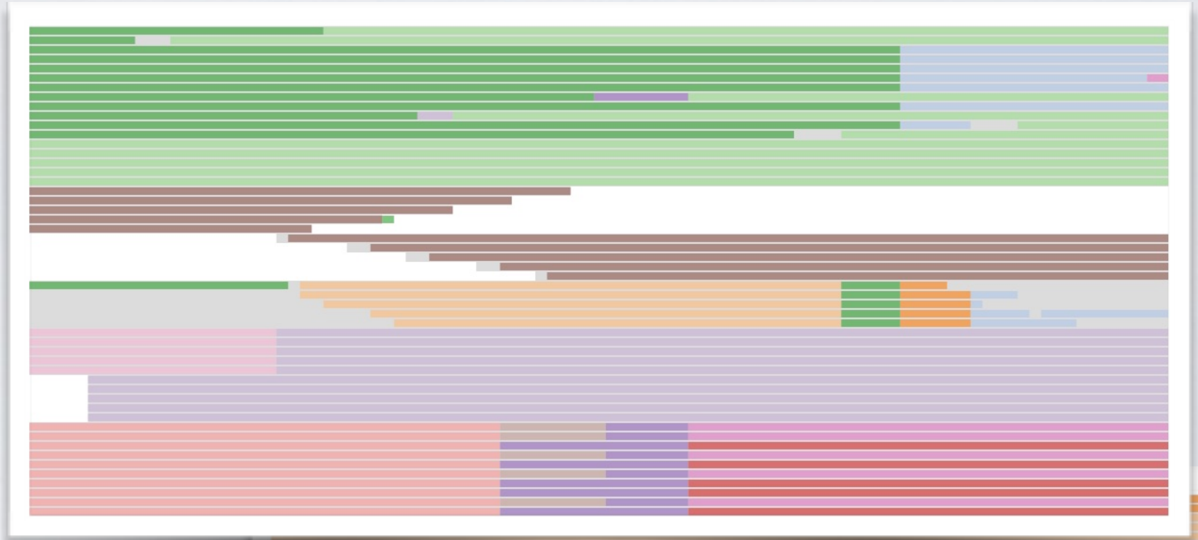
SYNTHETIC NETWORK

Division



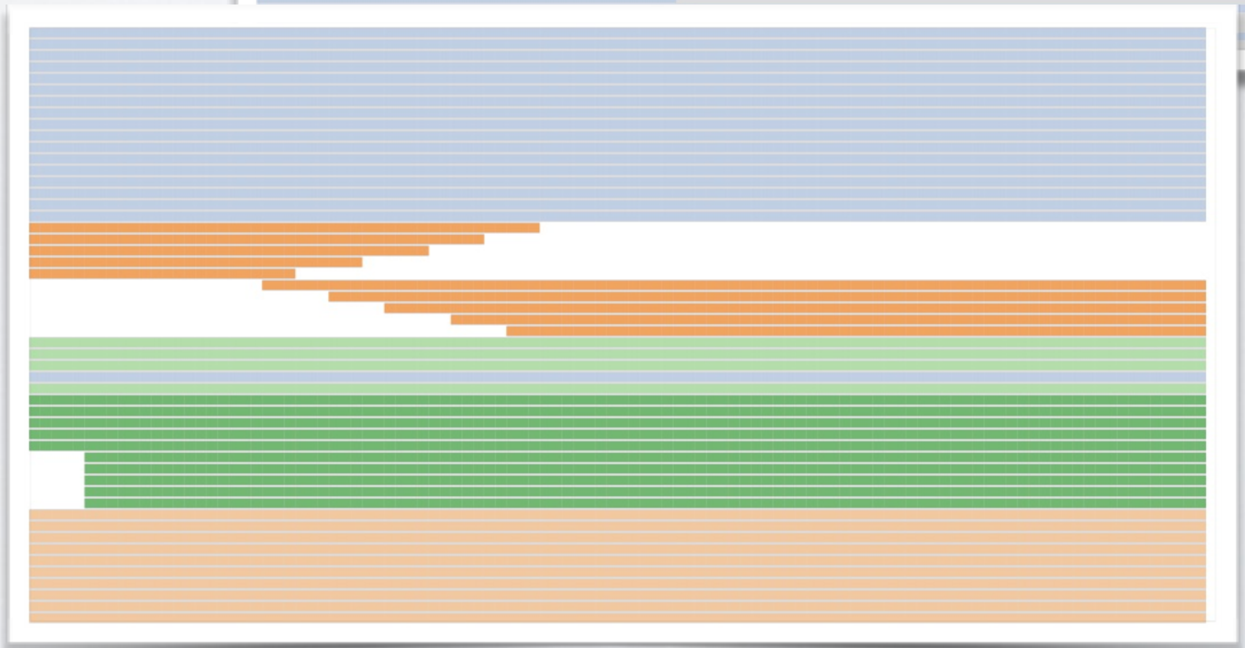
SYNTHETIC NETWORK

Instant Optimal:
Greene et al. 2011



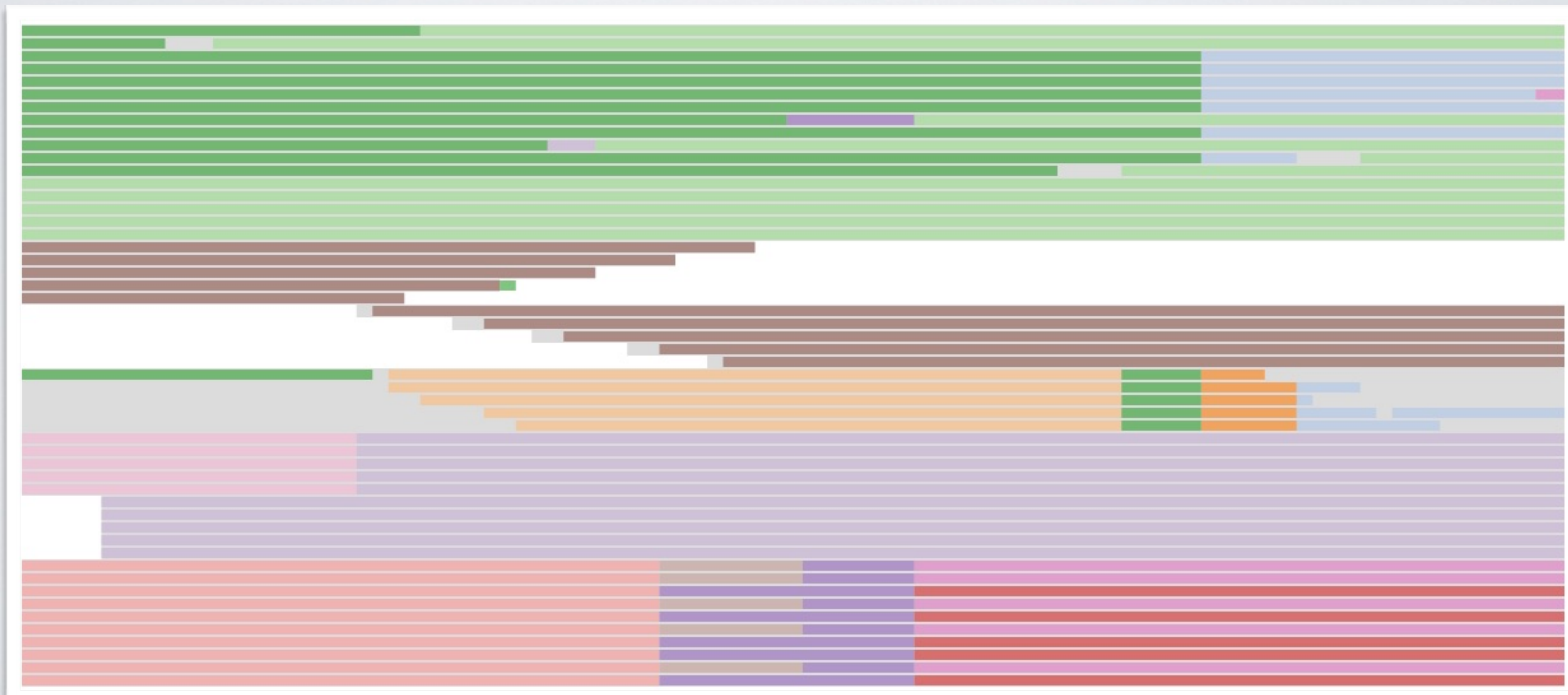
Temporal trade-off:
Cazabet et al. 2010

Cross-Time:
Mucha et al. 2010



SYNTHETIC NETWORK

Instant Optimal:
Greene et al. 2011

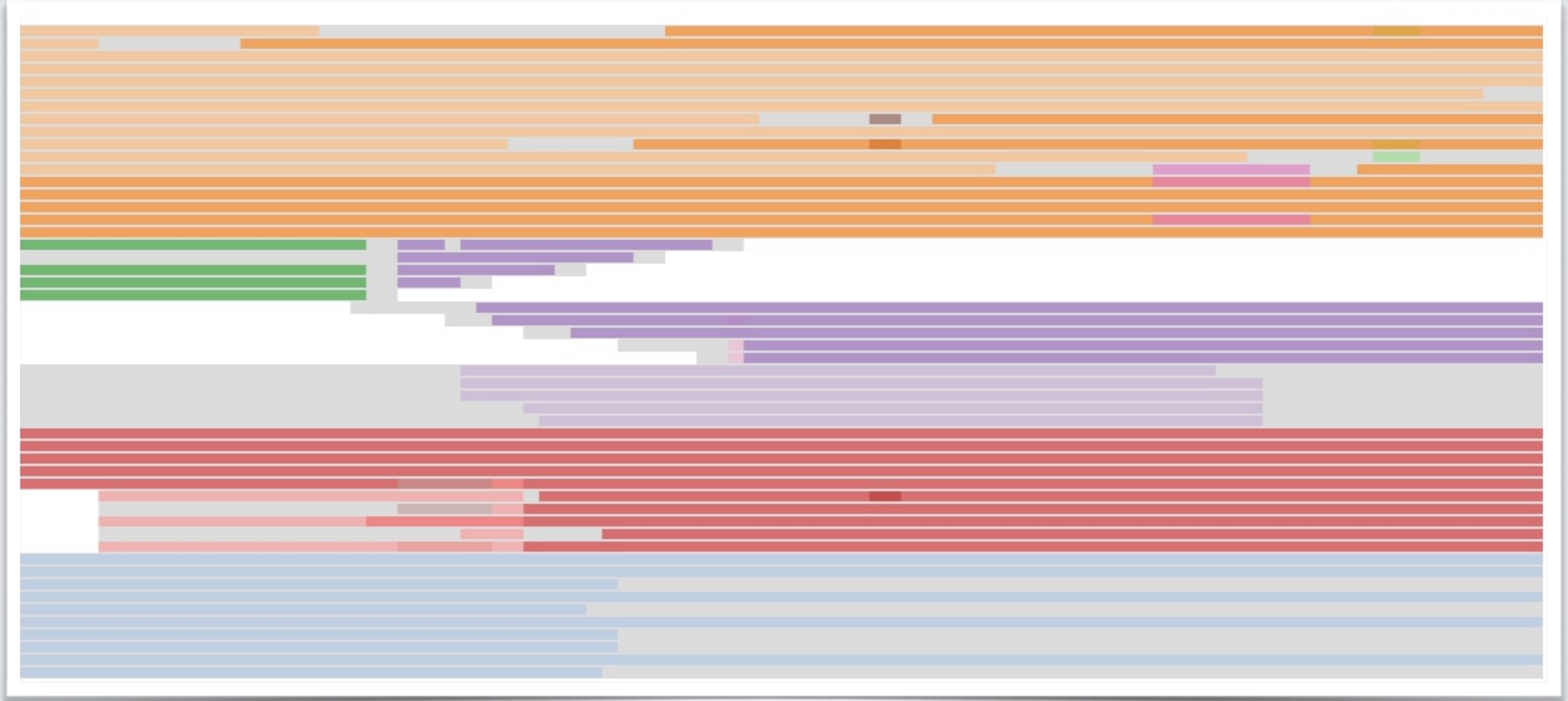


SYNTHETIC NETWORK

Instant Optimal:
Greene et al. 2011

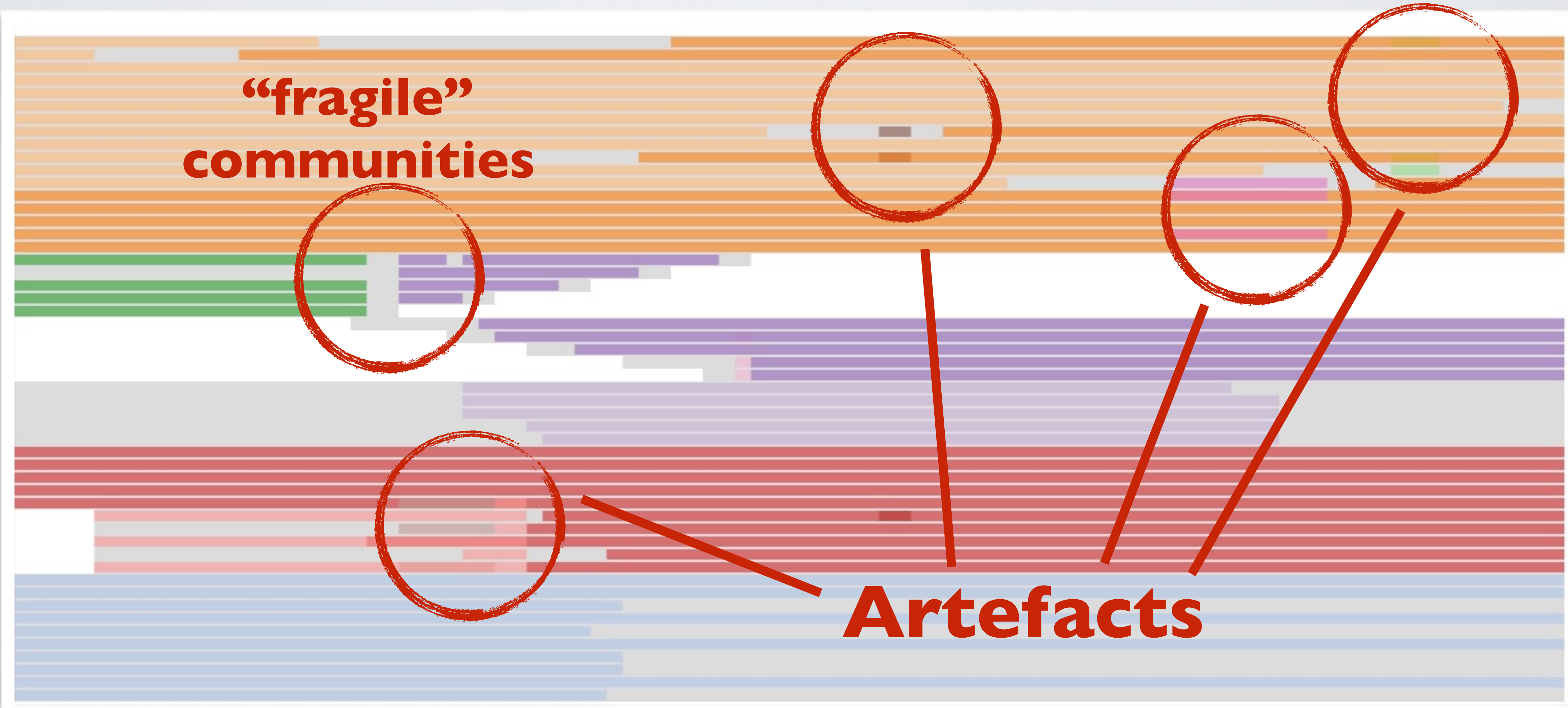
SYNTHETIC NETWORK

Temporal trade-off:
Cazabet et al. 2010



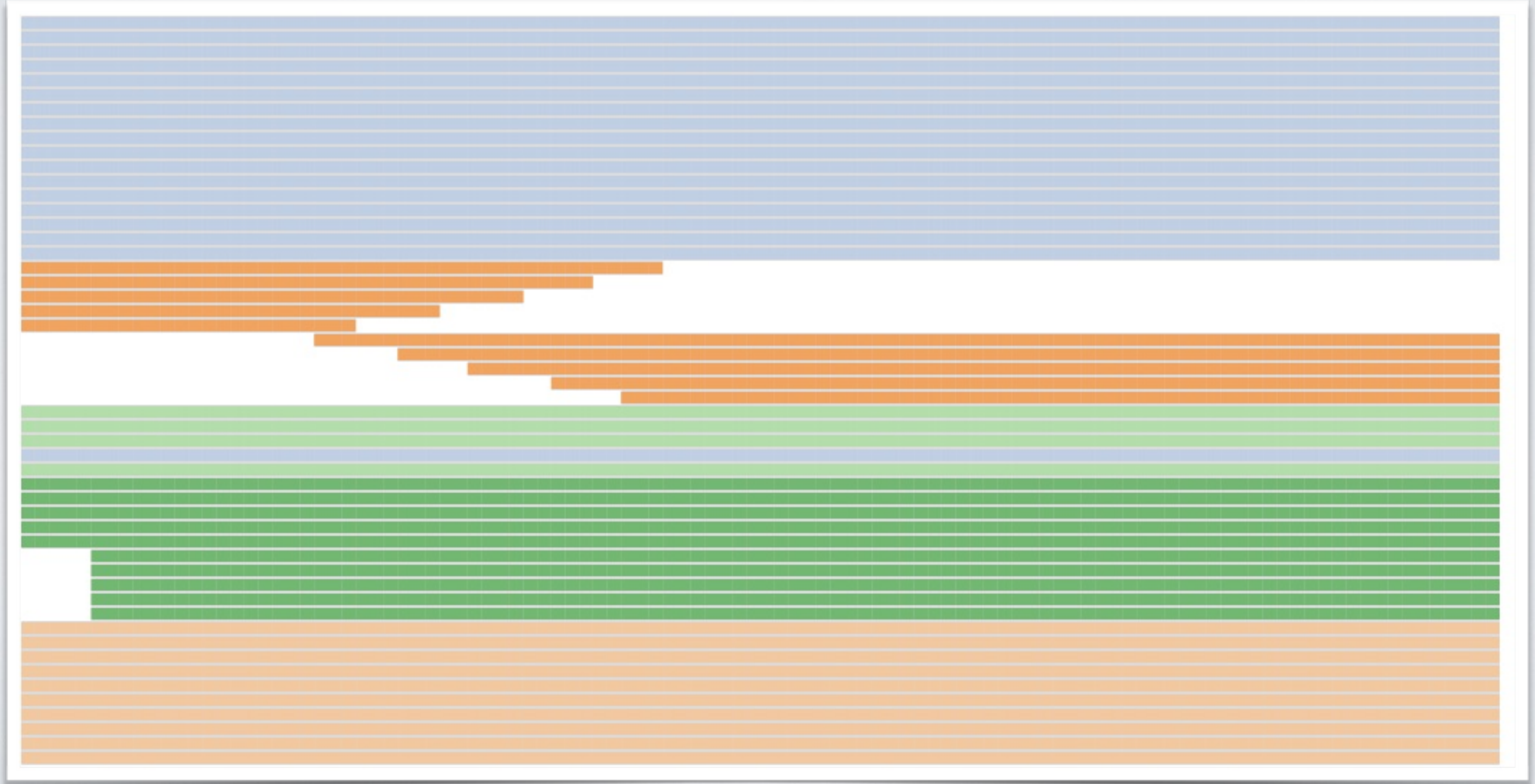
SYNTHETIC NETWORK

Temporal trade-off:
Cazabet et al. 2010



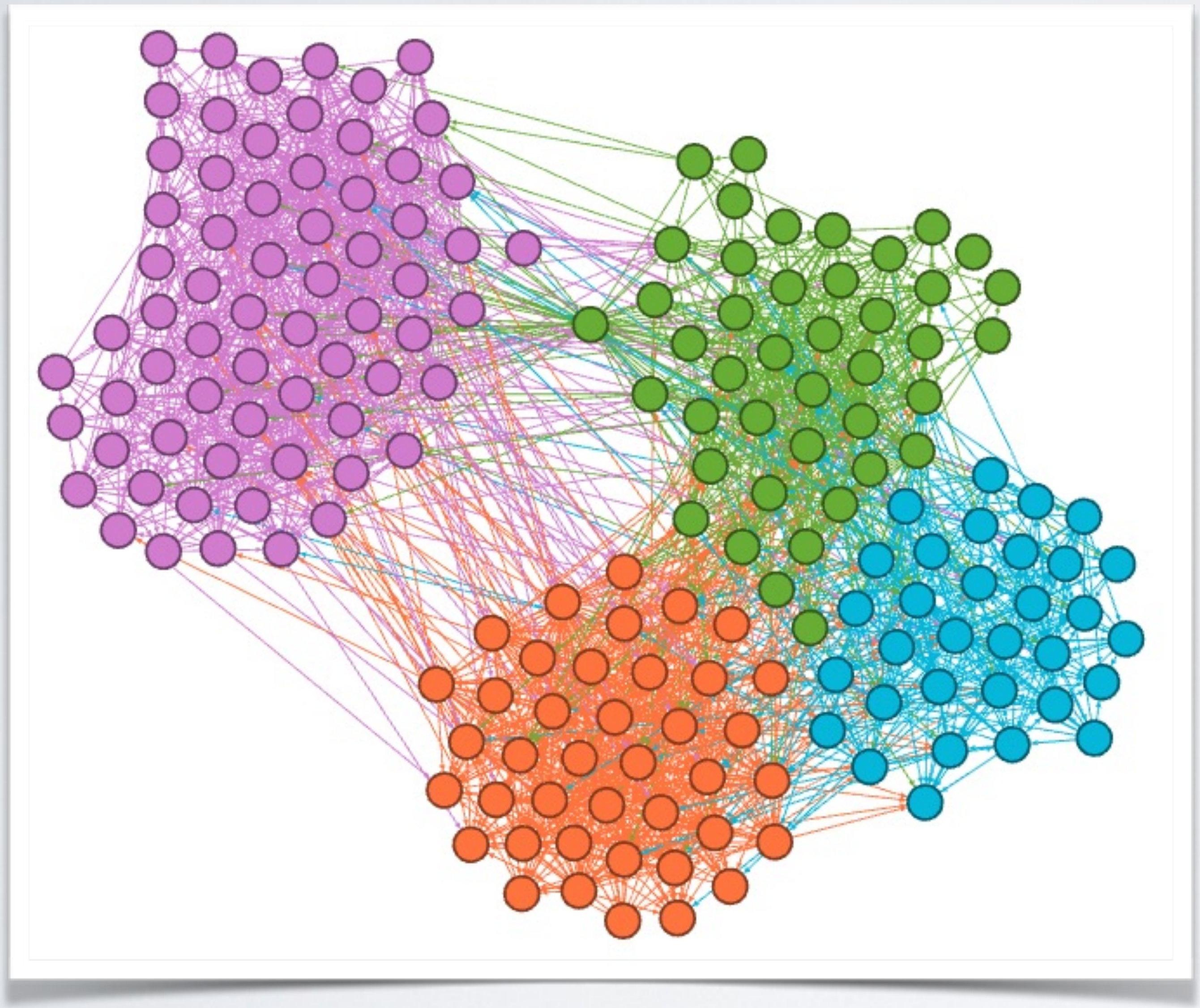
SYNTHETIC NETWORK

Cross-Time:
Mucha et al. 2010



REAL NETWORK

SOCIOPATTERNS



180 nodes

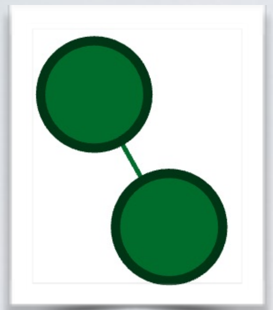
9 days (Monday to Tuesday)

5 classes

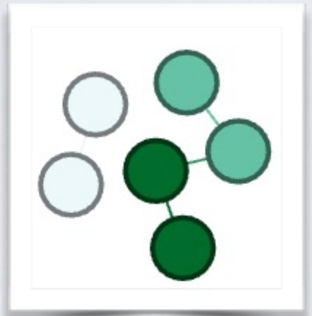
Face to face captured
every 20s

AGGREGATED GRAPHS

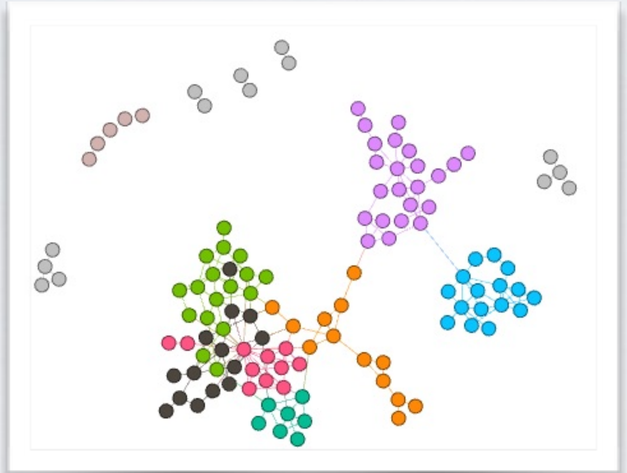
Monday
11:30



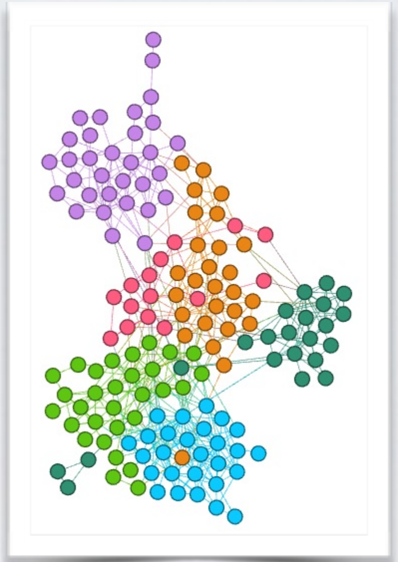
Monday
11:30 - 11:31



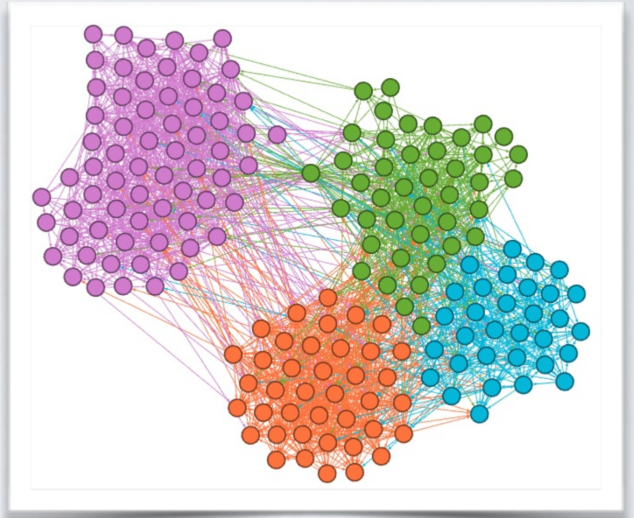
Monday
11:00 - 12:00



Monday

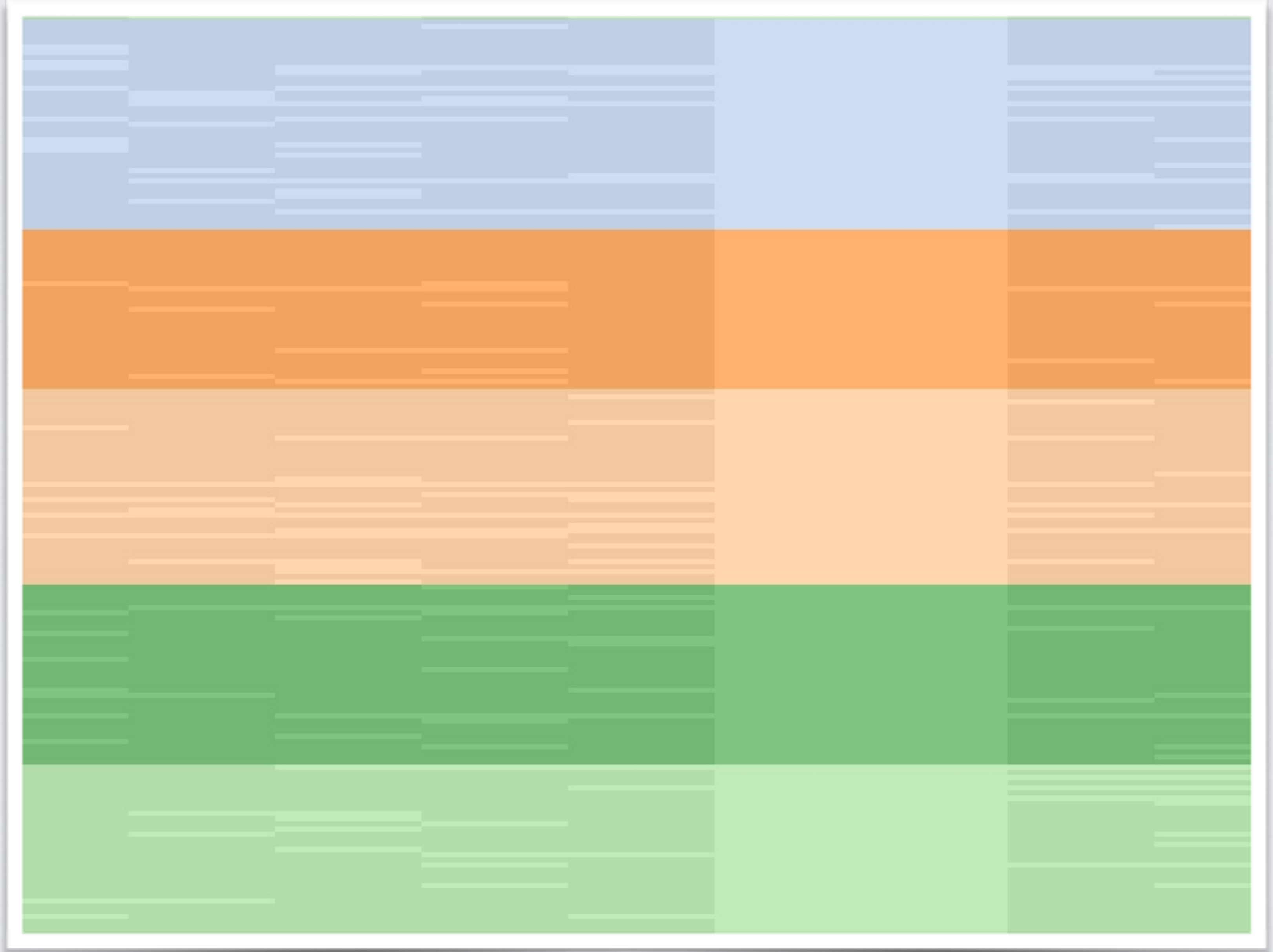


All periods



SOCIOPATTERNS

"Ground truth" (classes)

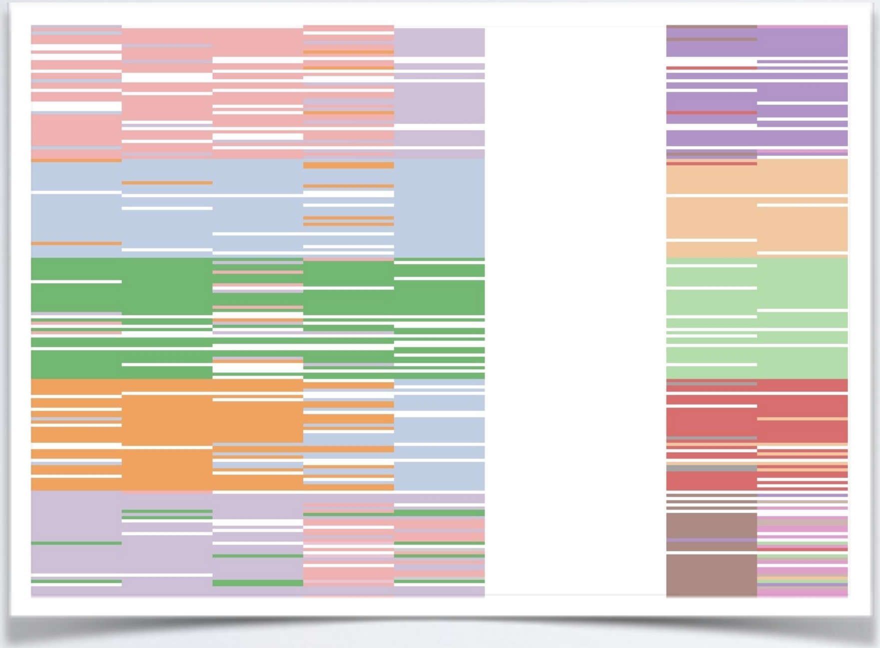


SOCIOPATTERNS

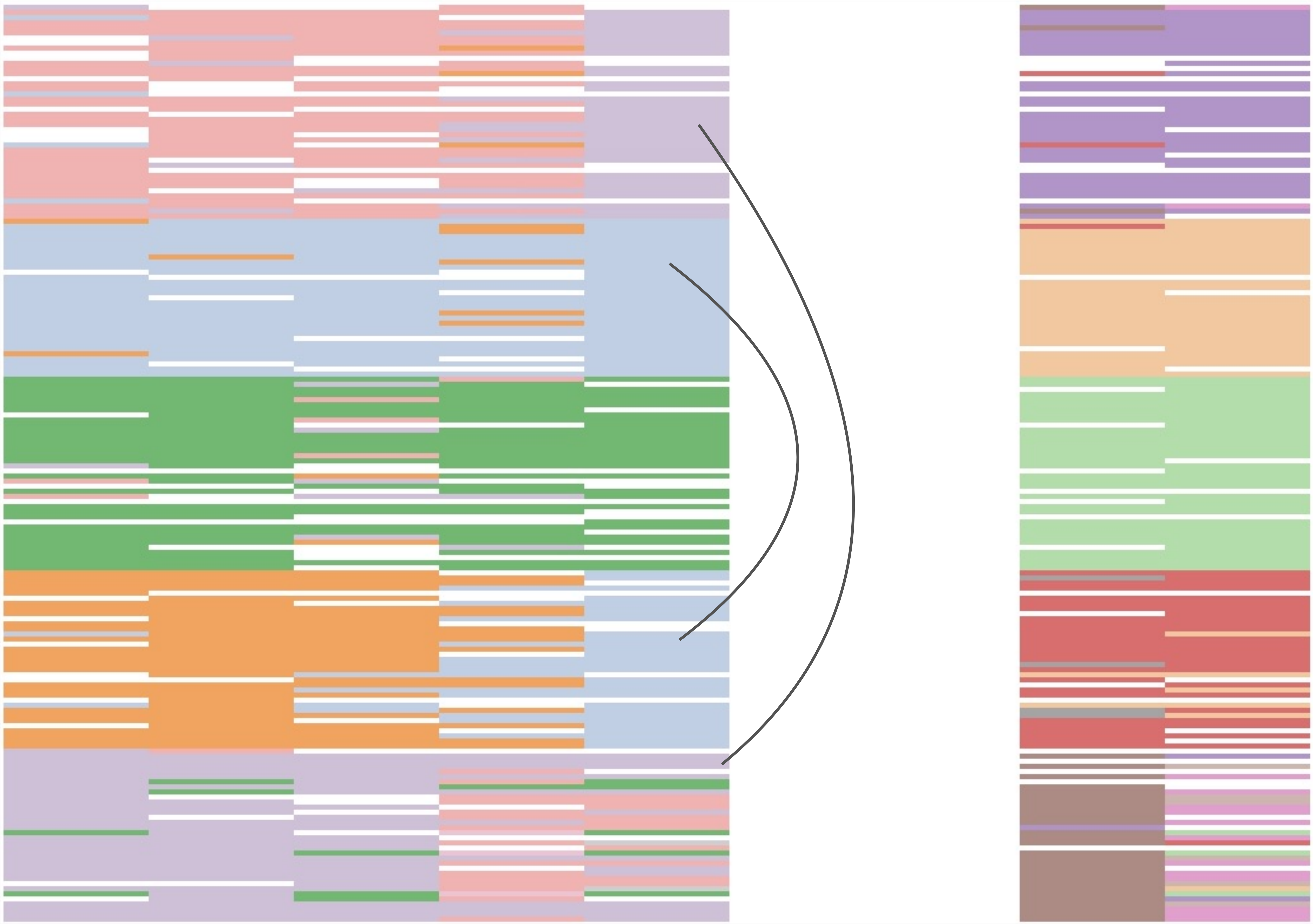
Instant Optimal:
Greene et al. 2011

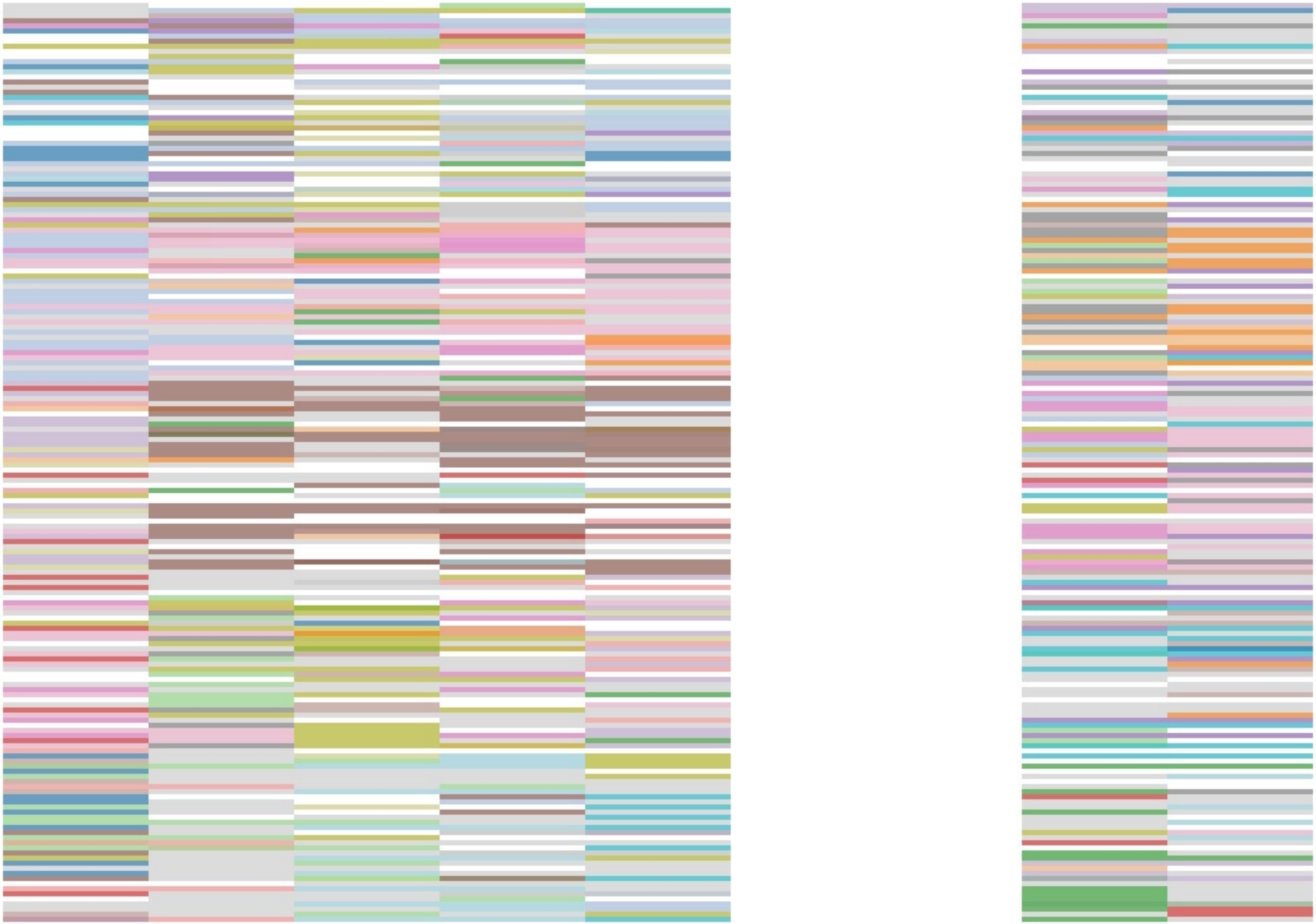
Temporal trade-off:
Cazabet et al. 2010

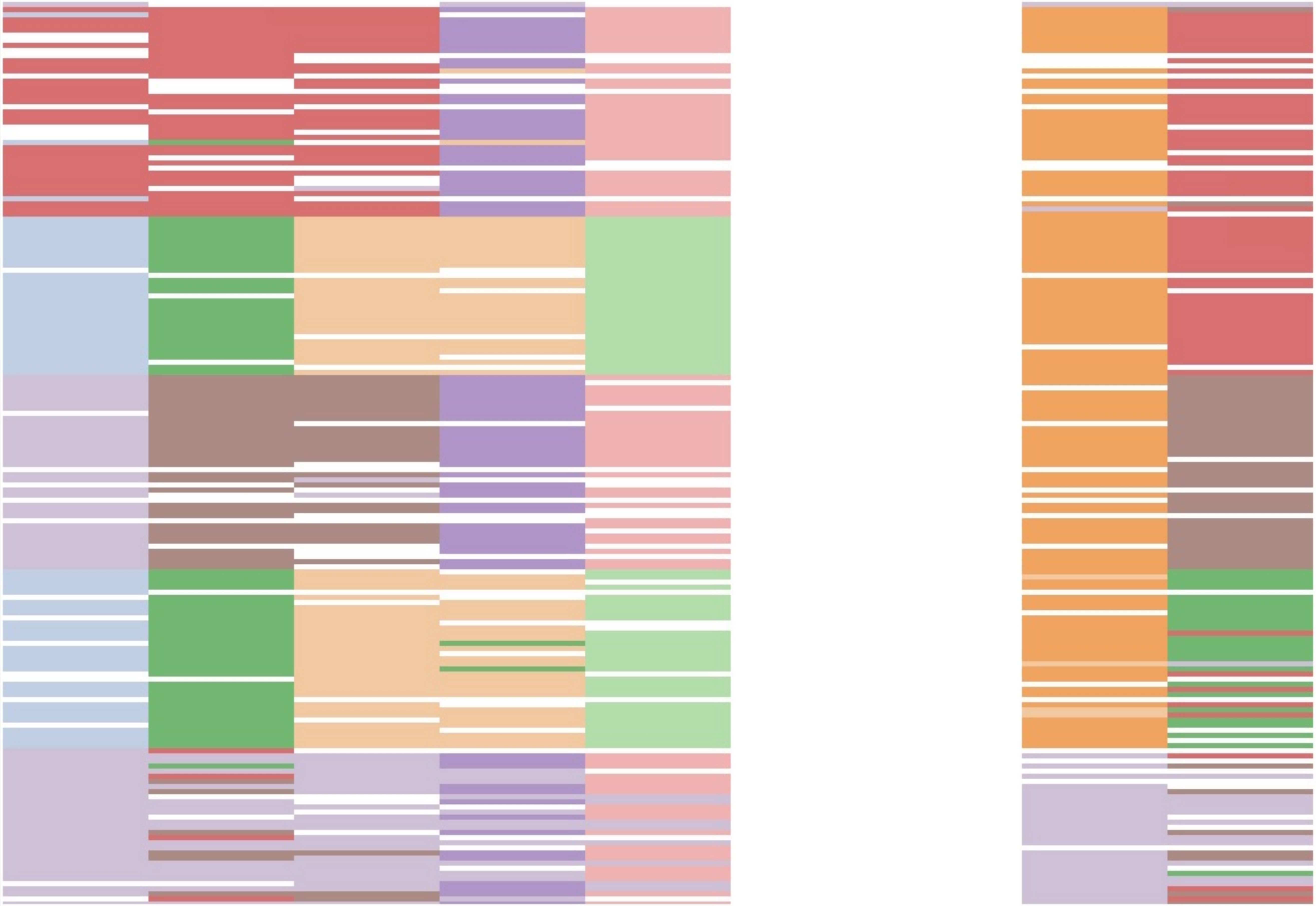
Cross-Time:
Mucha et al. 2010



Daily snapshots

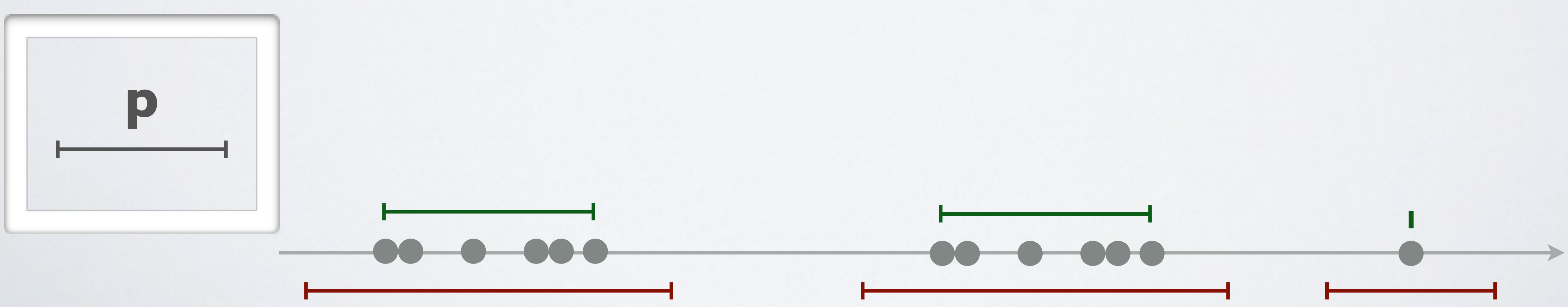






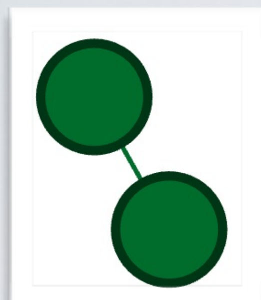
AGGREGATED GRAPHS

- Persistance computation:
  - Choose a *granularity* **p**
  - Sliding window :
    - element alive a **t** if appears in dataset in **[t-p/2,t+p/2]**
  - Truncated sliding windows
    - same as before, but truncate **p/2** from extremities

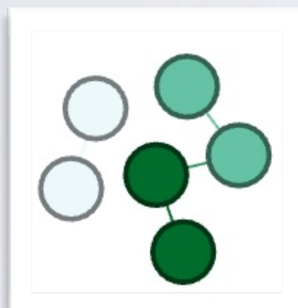


AGGREGATED GRAPHS

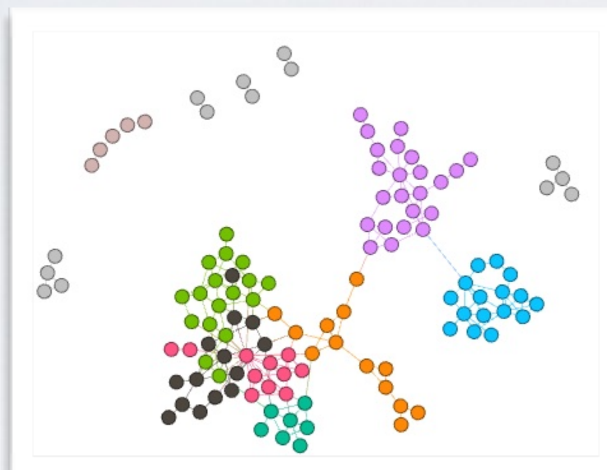
Monday
11:30



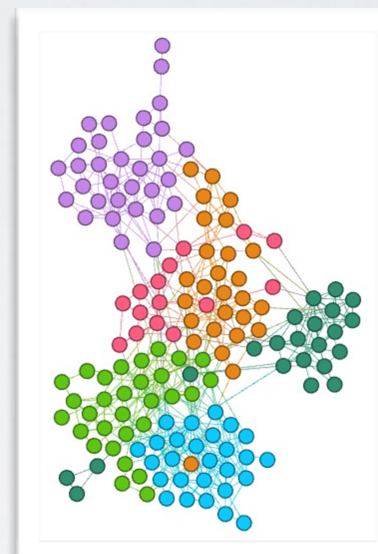
Monday
11:30 - 11:31



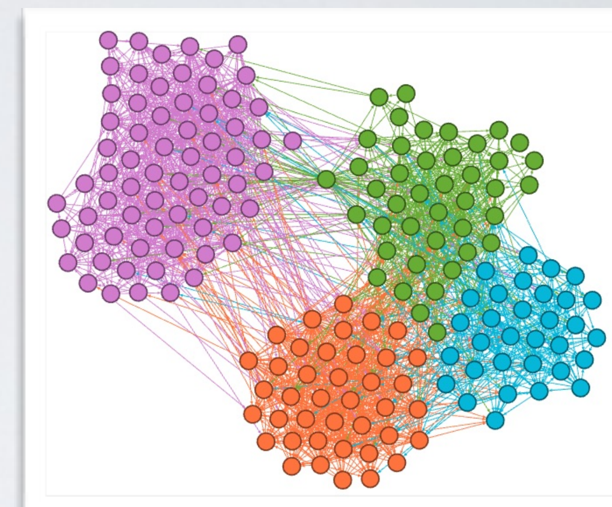
Monday
11:00 - 12:00

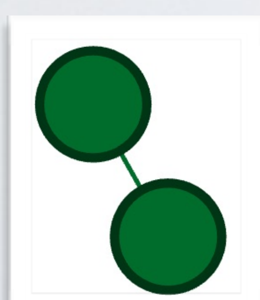


Monday



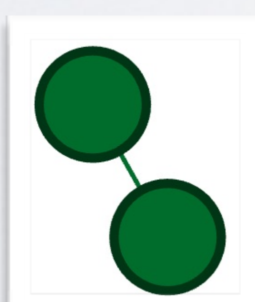
All periods





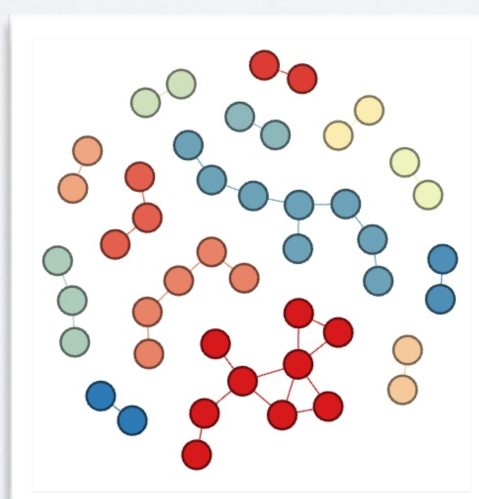
Monday 11:30

20s



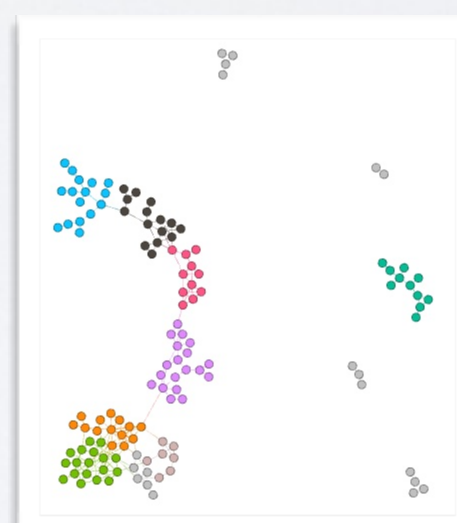
Monday 11:30

60s



Monday 11:30

1h



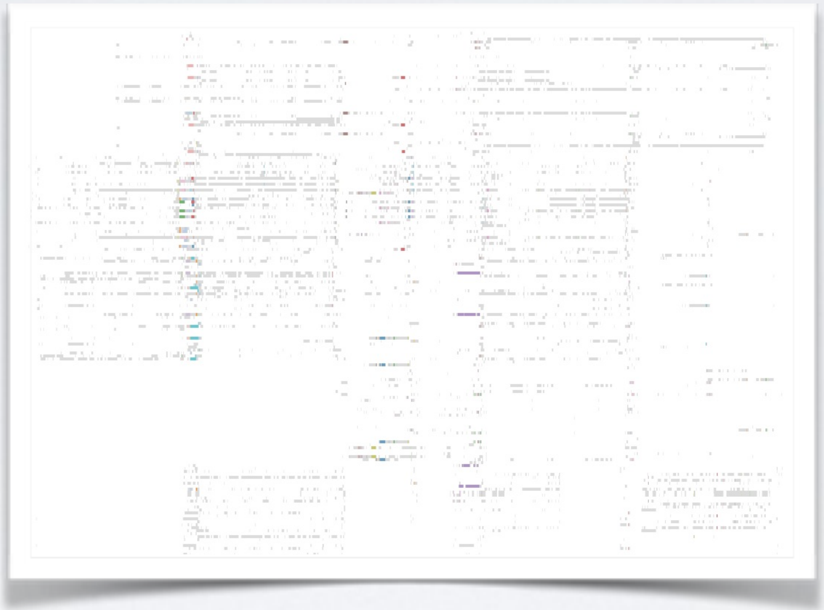
Monday 11:30

24h

SOCIOPATTERNS

Instant Optimal:
Greene et al. 2011

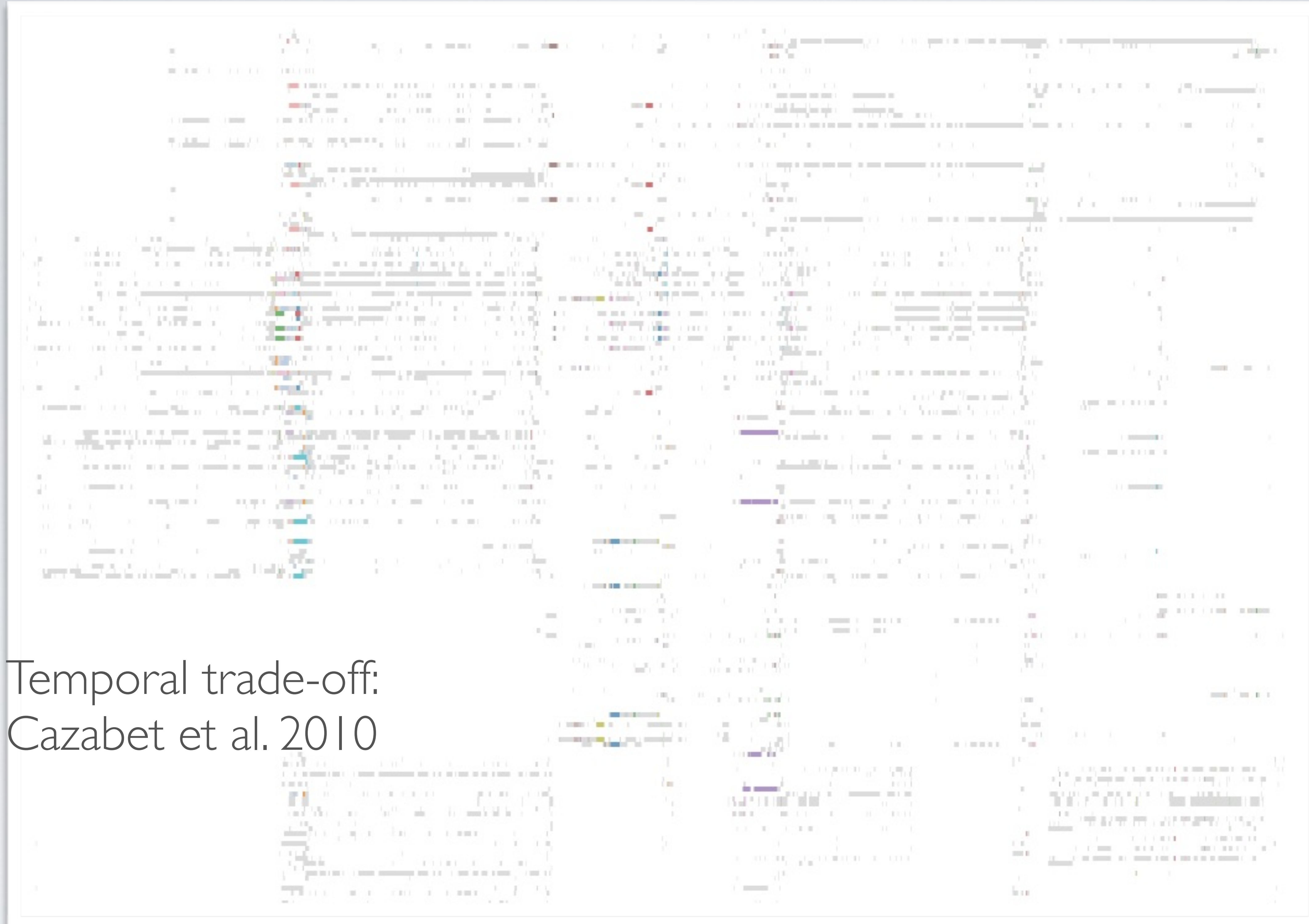
Temporal trade-off:
Cazabet et al. 2010



Cross-Time:
Mucha et al. 2010

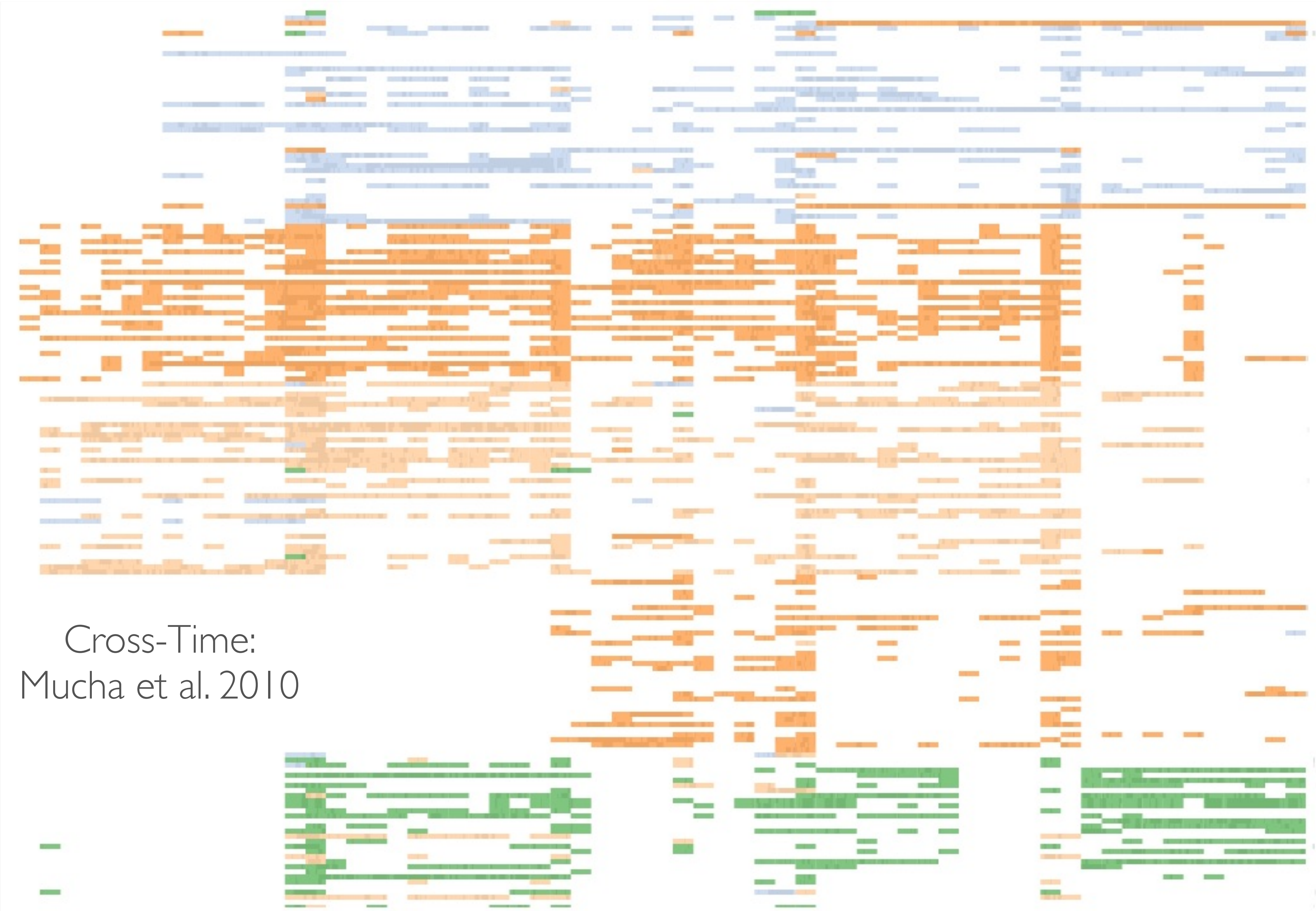
One minute Persistence, Monday

One minute Persistence, Monday



Temporal trade-off:
Cazabet et al. 2010

One minute Persistence, Monday



Cross-Time:
Mucha et al. 2010

PRACTICALS

- On the dynamic network of your choice (Game of Thrones for instance)
  - 1)Compute communities using your favorite algorithm at each step (at least 4).
  - 2)Compute the Jaccard coefficient between each pair of community in T and T+1. Choose a threshold to decide which community must be matched with which
    - (You can have split or merge if more than 2 matchs to the same community)
  - 3)Find examples of a stable community, unstable, and a community that appear or disappear
  - 4)Stable version: create a network in which each node is a community in a snapshot, and a link is a jacquard coefficient >0. Apply a community detection algorithm to this network to find dynamic communities.
  - 5)Any difference compared with matching only T and T+1 ?
  - 6)(Advanced) Use the library available at (https://github.com/Yquetzal/dynetx) and the example available at (http://cazabetremy.fr/Teaching/DCDclass/Intro_dynnetx.html) to use state of the art dynamic community detection algorithms