DYNAMIC COMMUNITY DETECTION

Cazabet Rémy

COMMUNITY DETECTION

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

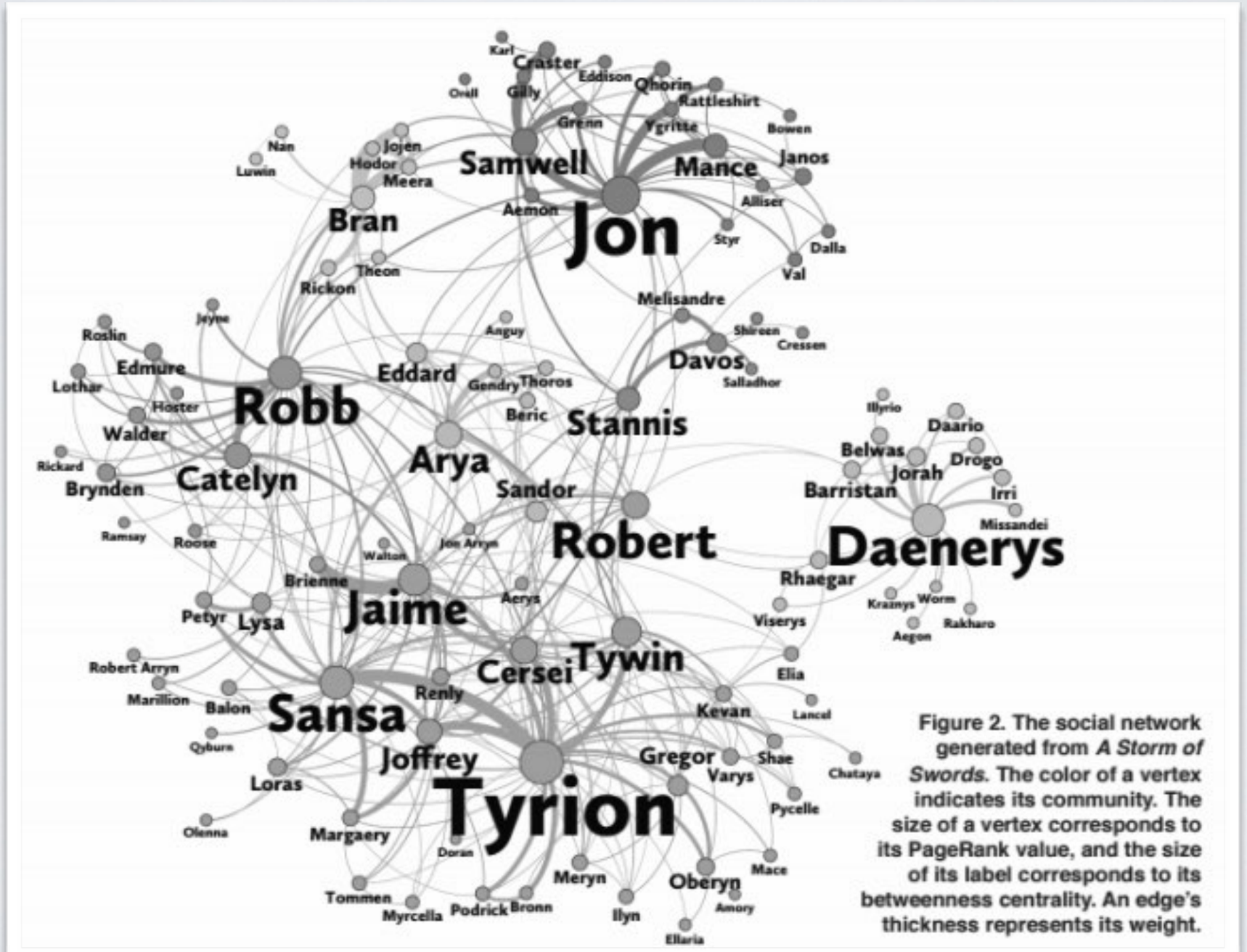


Figure 2. The social network generated from *A Storm of Swords*. The color of a vertex indicates its community. The size of a vertex corresponds to its PageRank value, and the size of its label corresponds to its betweenness centrality. An edge's thickness represents its weight.

COMMUNITY DETECTION

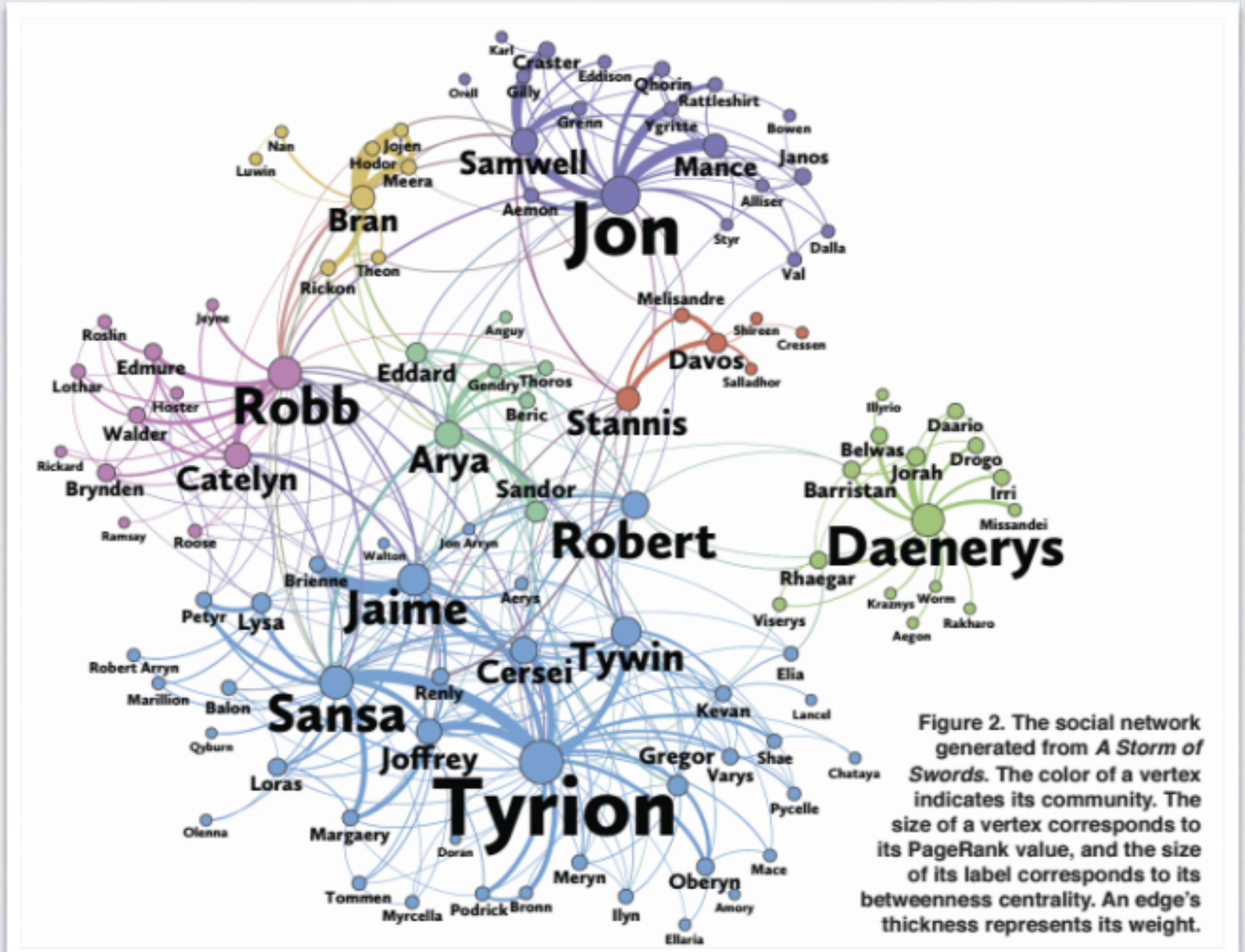


Figure 2. The social network generated from *A Storm of Swords*. The color of a vertex indicates its community. The size of a vertex corresponds to its PageRank value, and the size of its label corresponds to its betweenness centrality. An edge's thickness represents its weight.

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

Long term

-Friend
-Colleague
-Family relation
-…

Short term ?

-Collaborators in the same project
-Same team in a game
-Attendees of the same meeting
-…

Interactions

Instantaneous

-e-mail
-Text message
-Co-authoring
…

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

Semantic level

Representation level

Relations

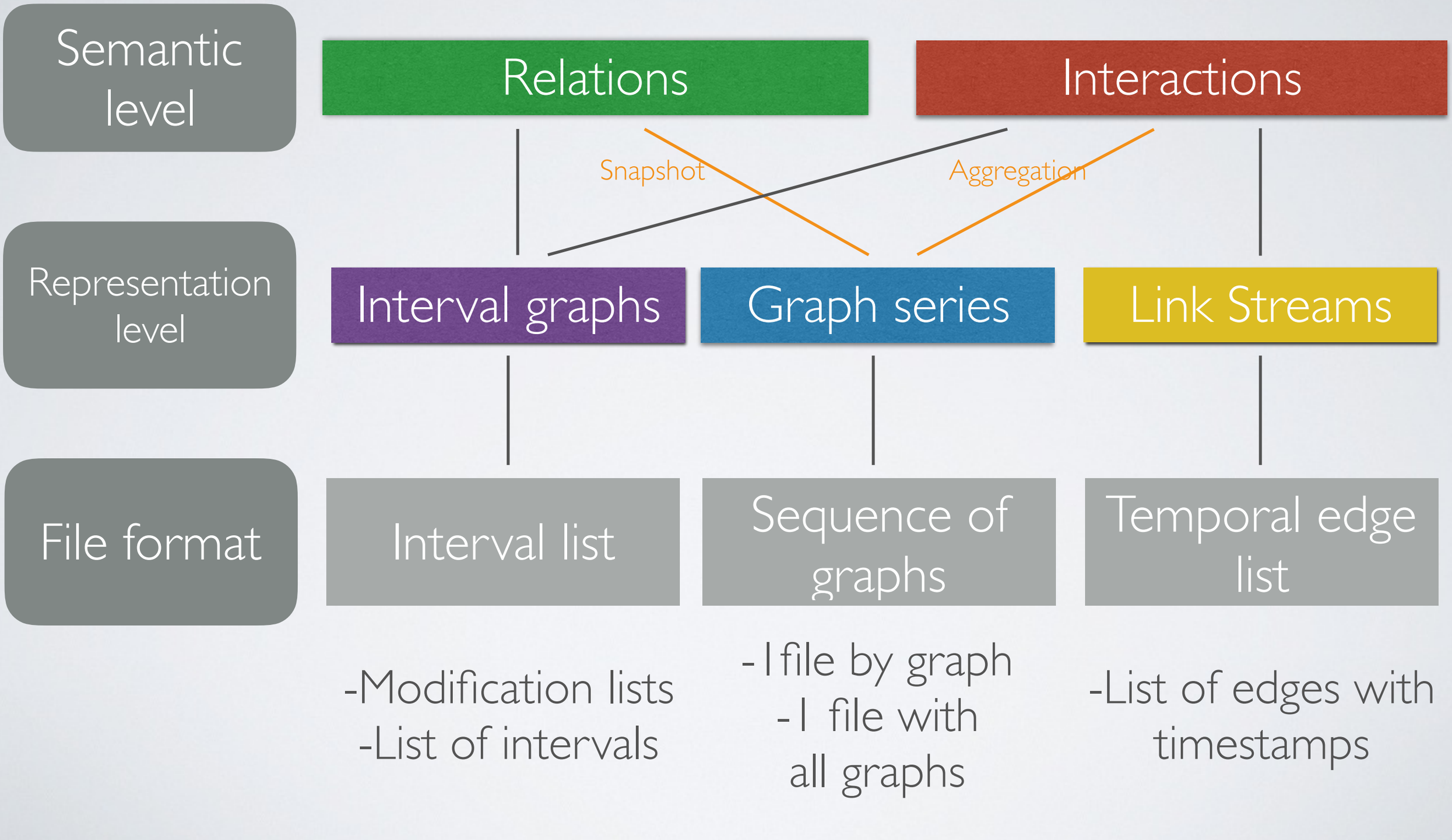
Interactions

Snapshot

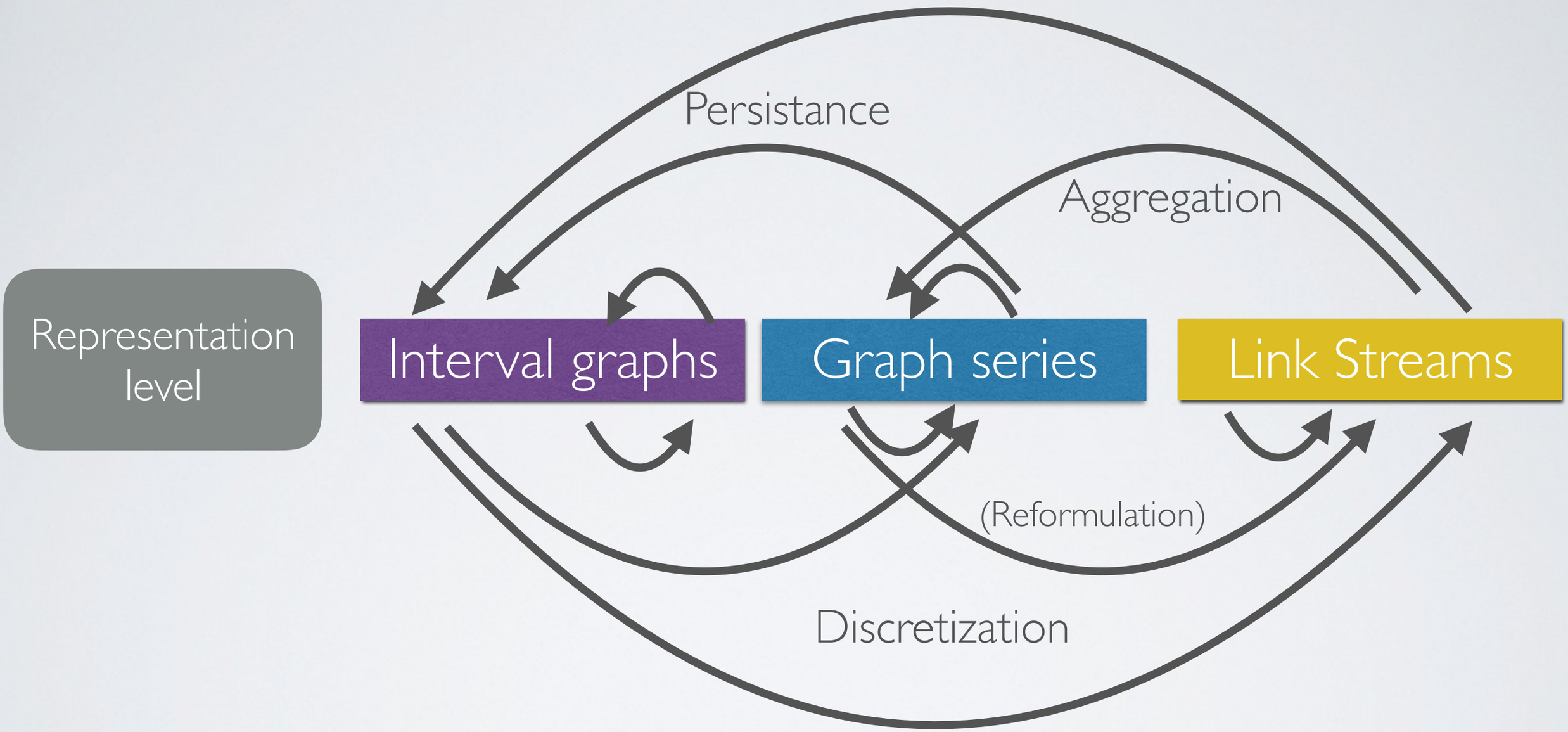
Aggregation

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



DYNAMIC NETWORKS



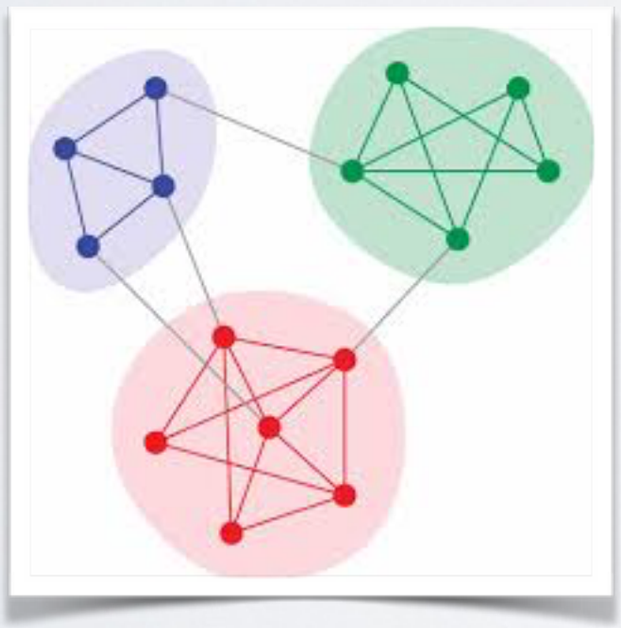
DYNAMIC COMMUNITY DETECTION

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

COMMUNITY DETECTION

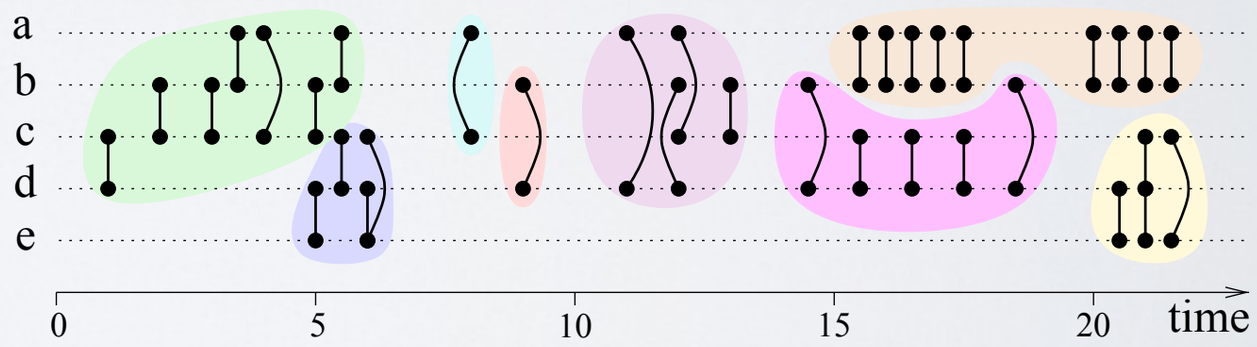
Static networks

Sets of nodes



Dynamic Networks

Sets of periods of nodes

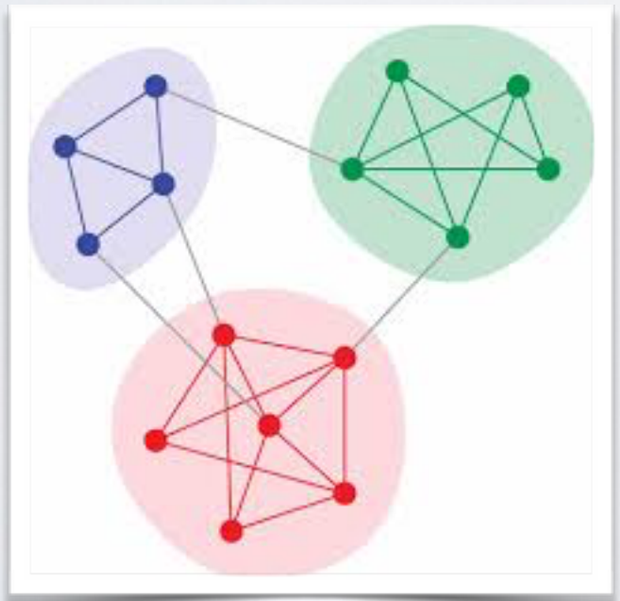


[Viard 2016]

COMMUNITY DETECTION

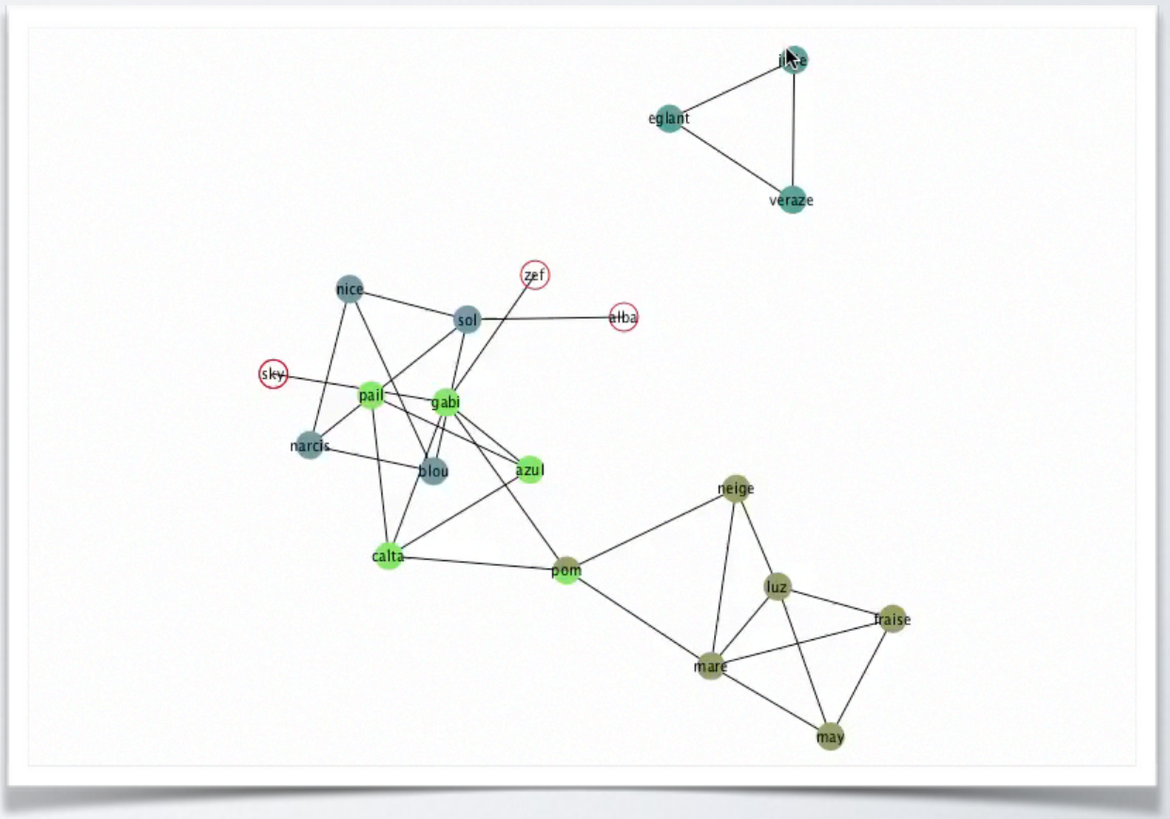
Static networks

Sets of nodes



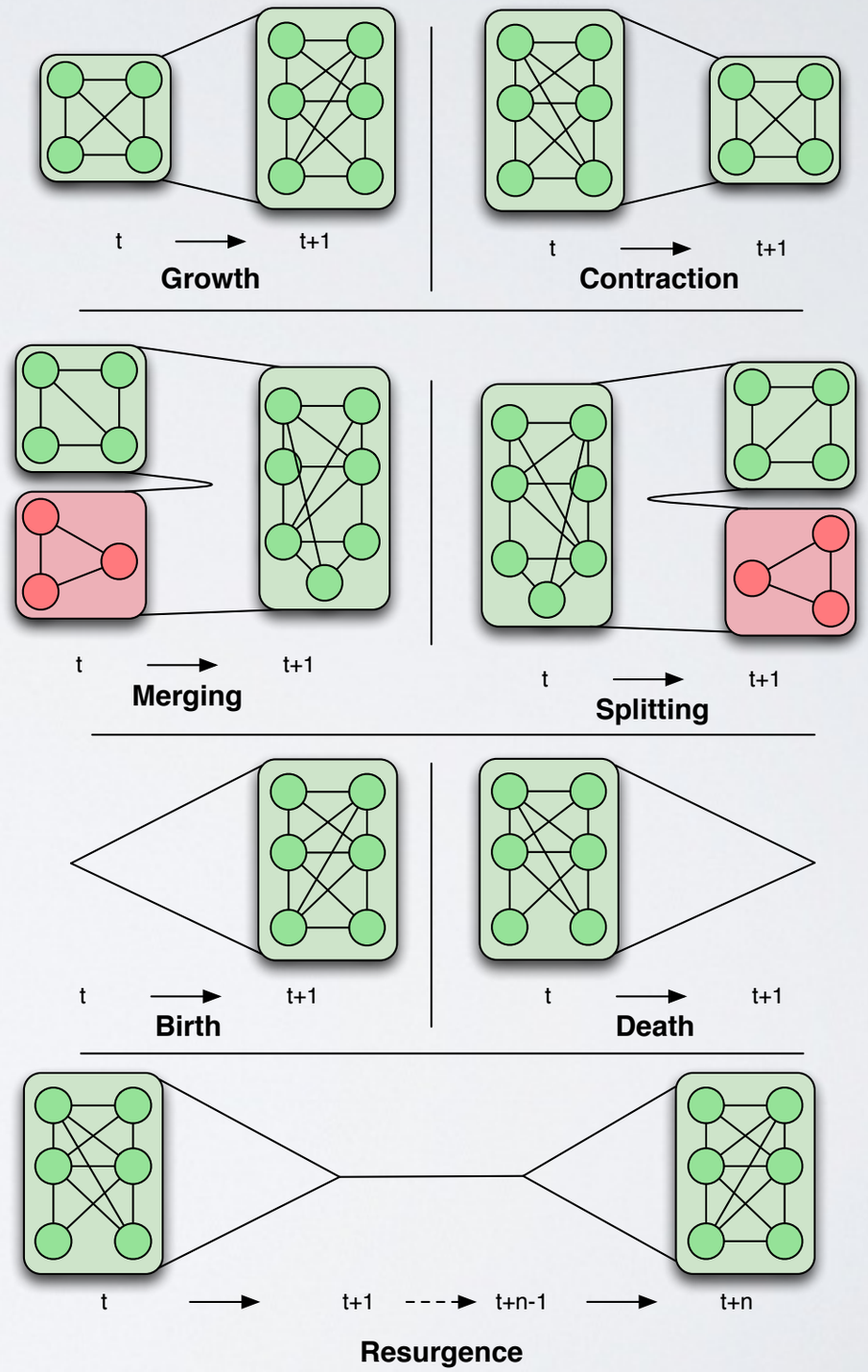
Dynamic Networks

Sets of periods of nodes

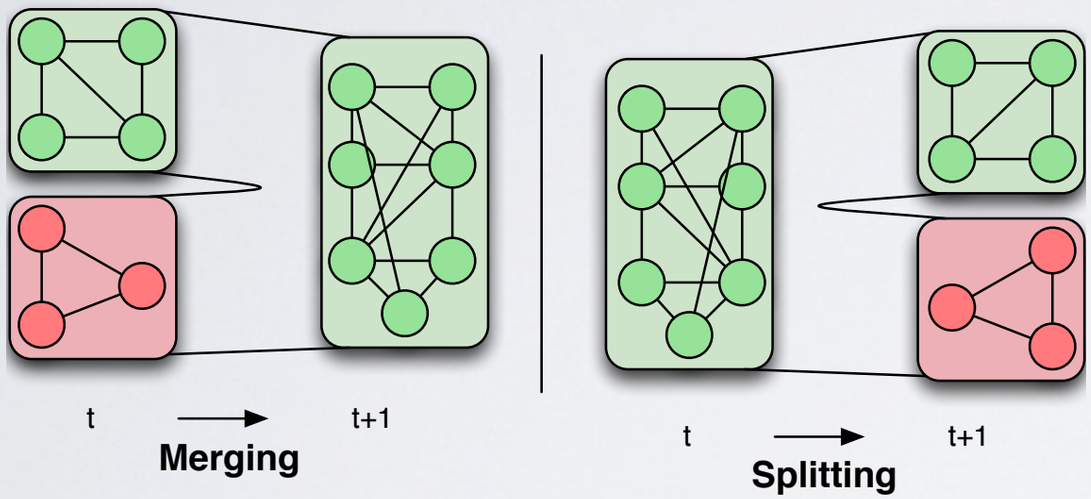


COMMUNITY DETECTION

Community events
(or operations)



COMMUNITY DETECTION



Which one persists ?

-Oldest ?

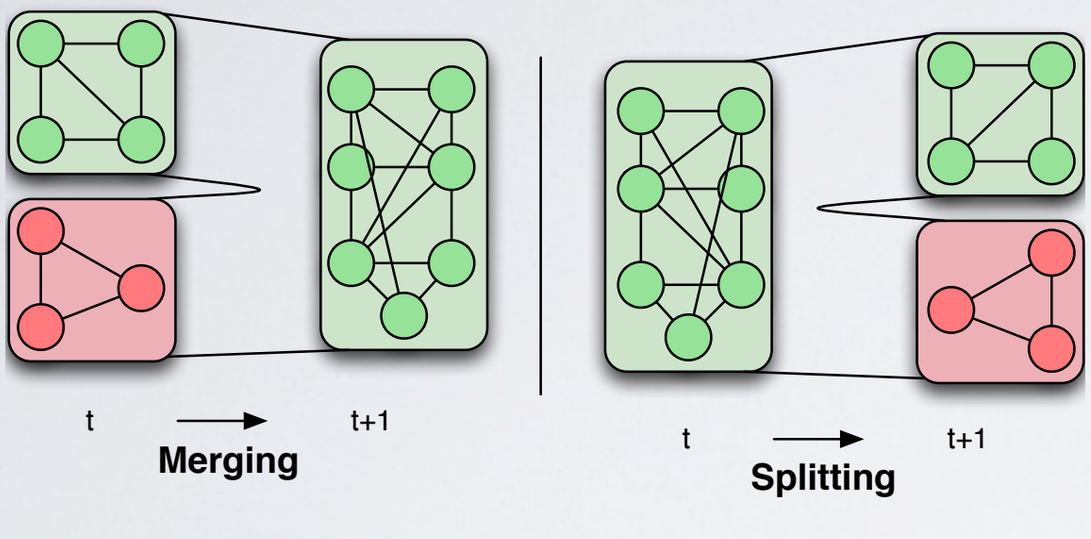
-Most similar ?

-Larger ?

-...

Community events
(or operations)

COMMUNITY DETECTION



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

Ship of Theseus paradox

Sequence of small modification
=>Complete change

COMMUNITY DETECTION

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

(A) Instant Optimal	**(B)** Temporal Trade-Off	**(C)** Cross-Time
(A1) Iterative, Similarity Based	**(B1)** Update by Global Optimization	**(C1)** Fixed Memberships, Fixed Properties
(A2) Iterative, Core-Node Based	**(B2)** Informed CD by Multi-Objective Optimization	**(C2)** Fixed Memberships, Evolving Properties
(A3) Multi-Step Matching	**(B3)** Update by Set of Rules	**(C3)** Evolving Memberships, Fixed Properties
	(B4) Informed CD by Network Smoothing	**(C4)** Evolving Memberships, Evolving Properties
Clusters at *t* depends **only on the current state** of the network Clusters are **non-temporally smoothed** (Communities **labels**, however, can be smoothed)	Clusters at *t* depends **on current and past states** of the network Clusters are **incrementally temporally smoothed**	Clusters at *t* depends on **both past and future** states of the network Clusters are **Completely temporally smoothed**

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

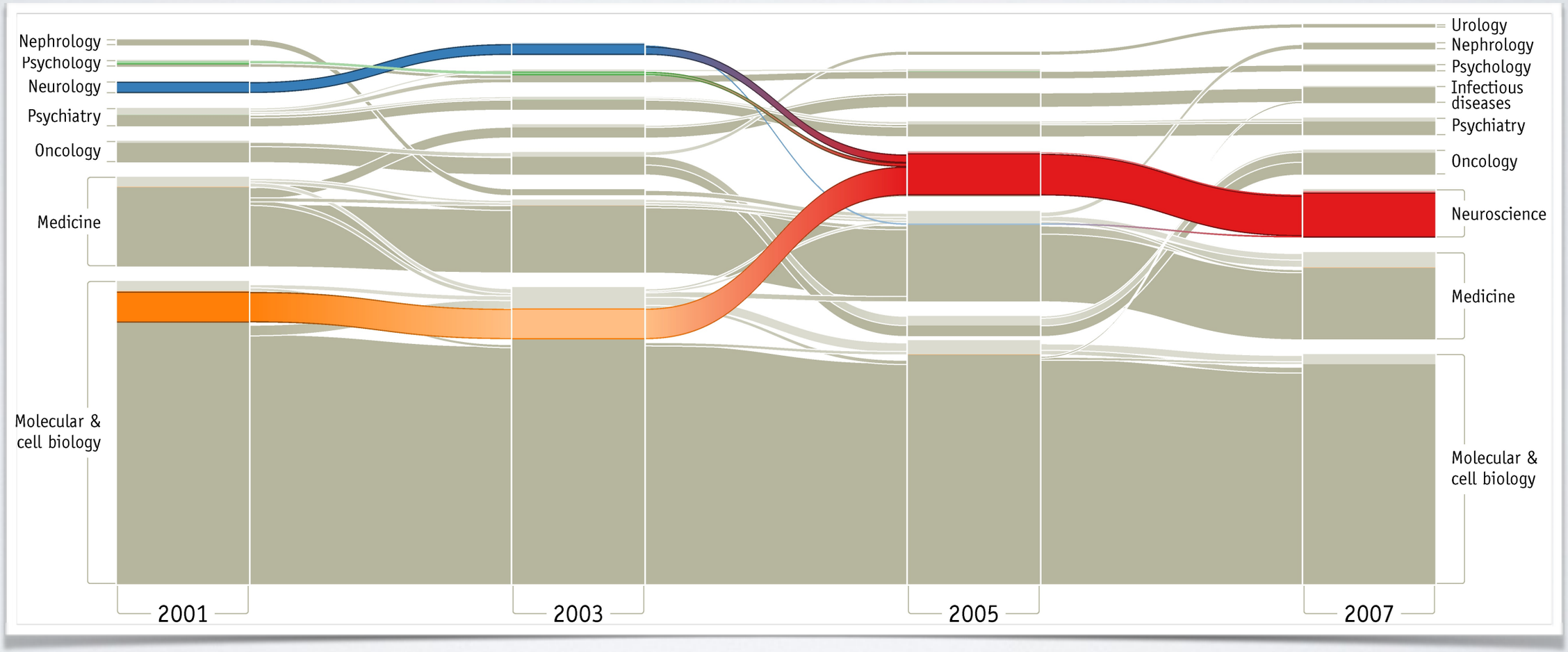
SCALABILITY

- Several types of complexity:
  - Snapshots approaches: Complexity for a snapshot x #snapshots + added cost (matching…)
  - Temporal networks: Complexity proportional to the number of modifications

- Can scale up to some limits:
  - Snapshots approaches: Hard part can be parallelized, but limit #snapshots
  - Temporal networks: cannot be parallelized, but can study fast dynamic at low cost

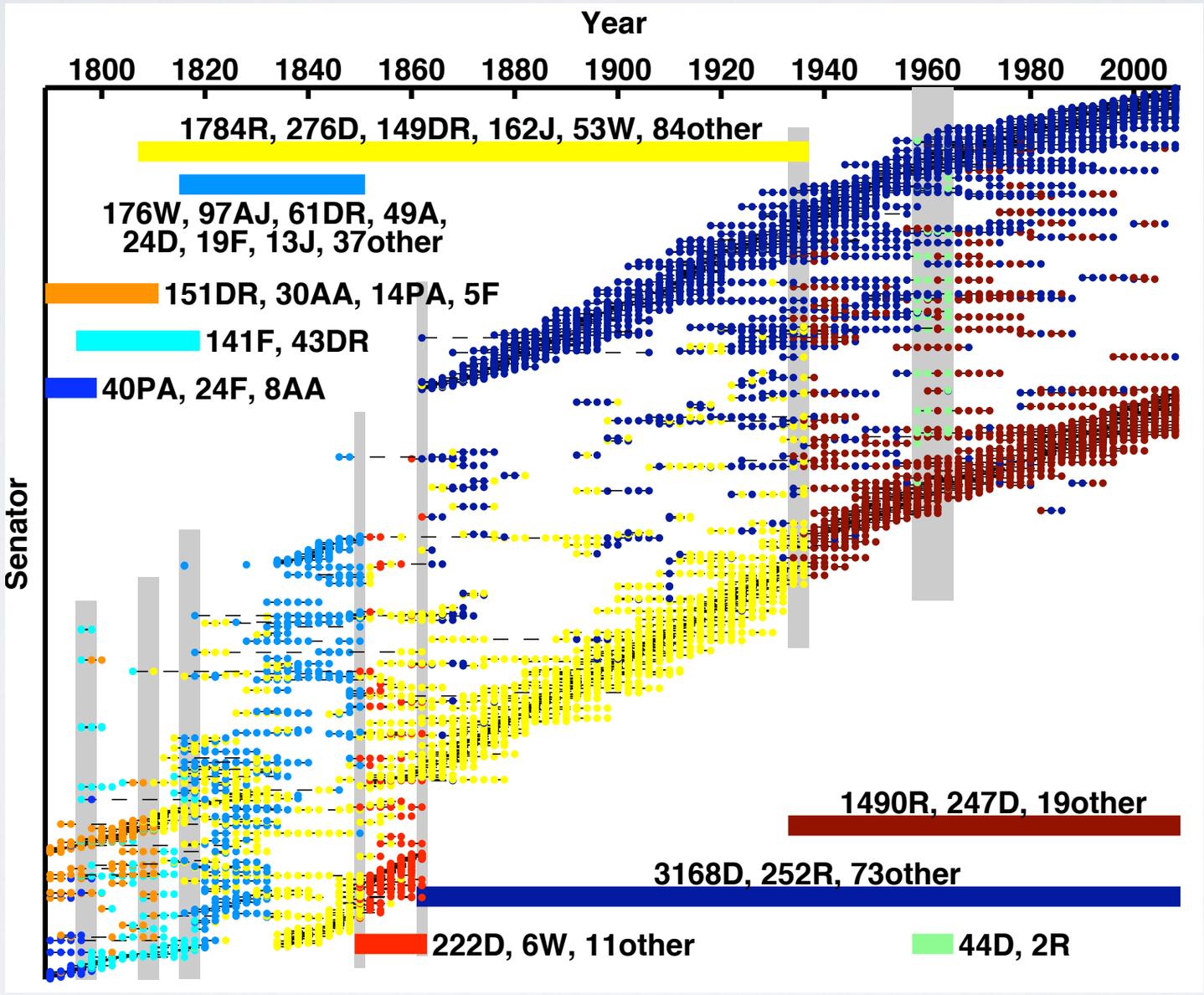
COMMUNITY DETECTION

Some examples of applications



Rosvall et al. 2010

COMMUNITY DETECTION

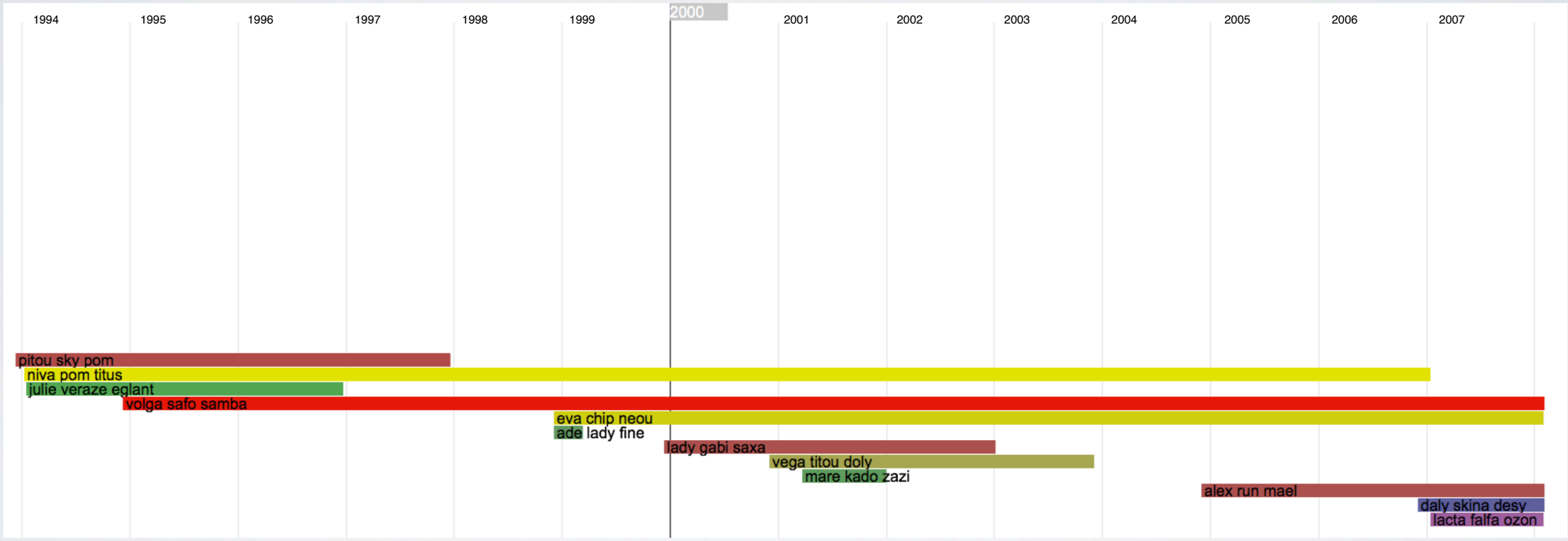


R : Républicains

D : Démocrates

Mucha et al. 2010

COMMUNITY DETECTION



DCD IN PRACTICE

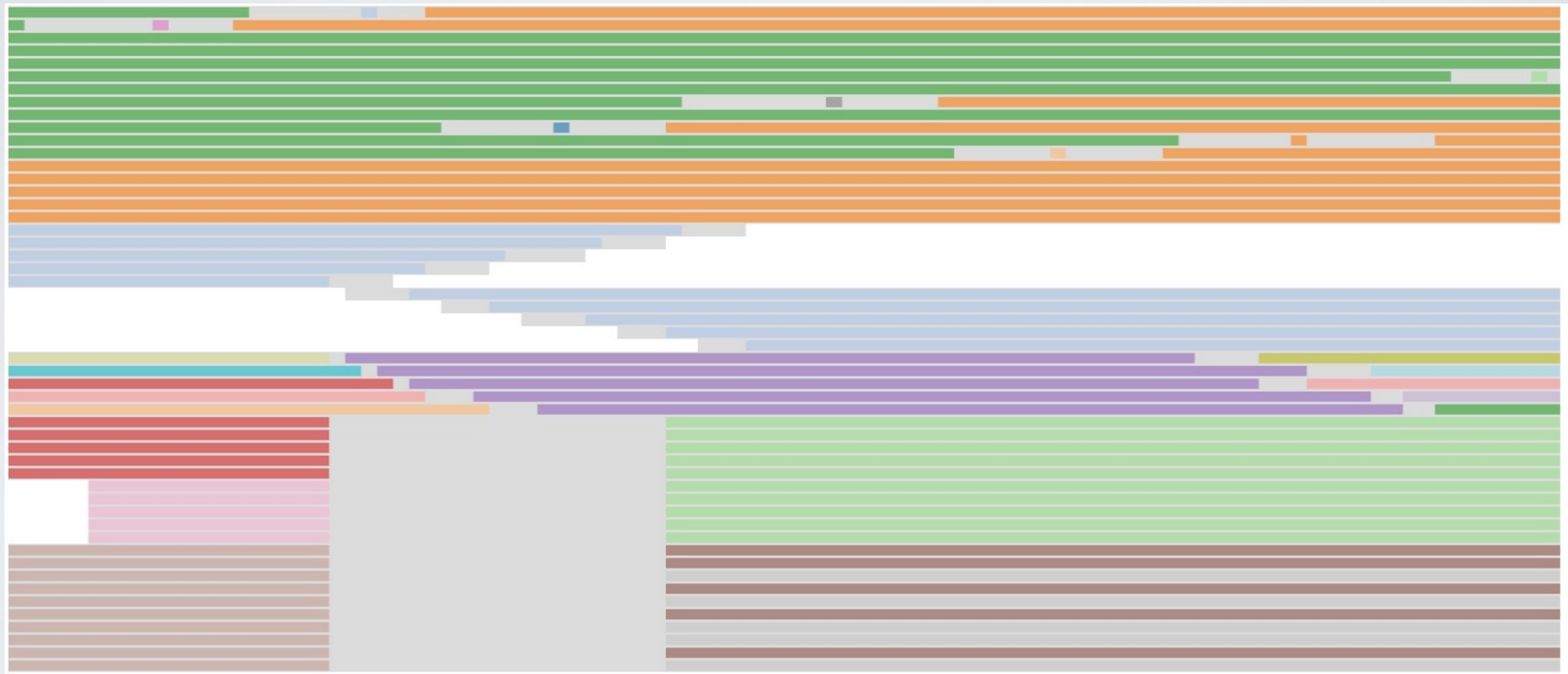
DCD IN PRACTICE

- 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

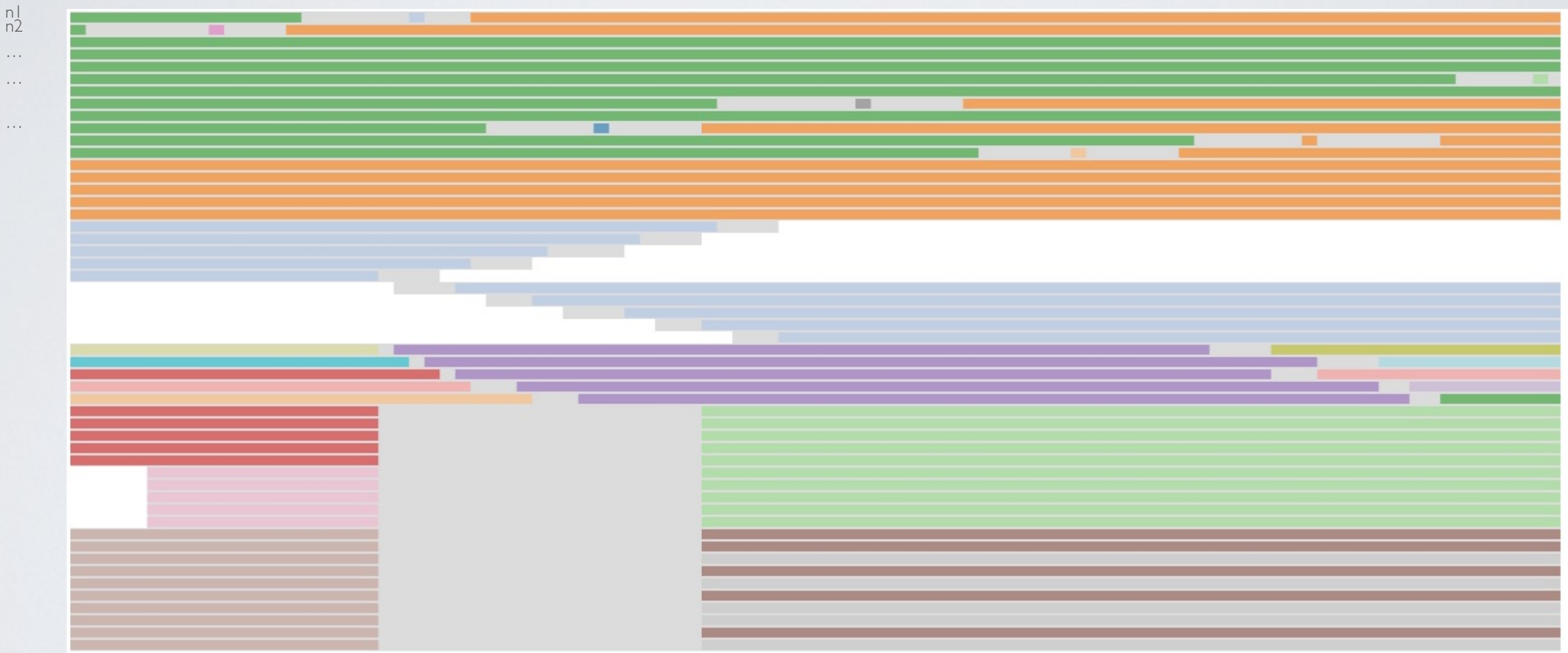
SYNTHETIC NETWORK

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

SYNTHETIC NETWORK

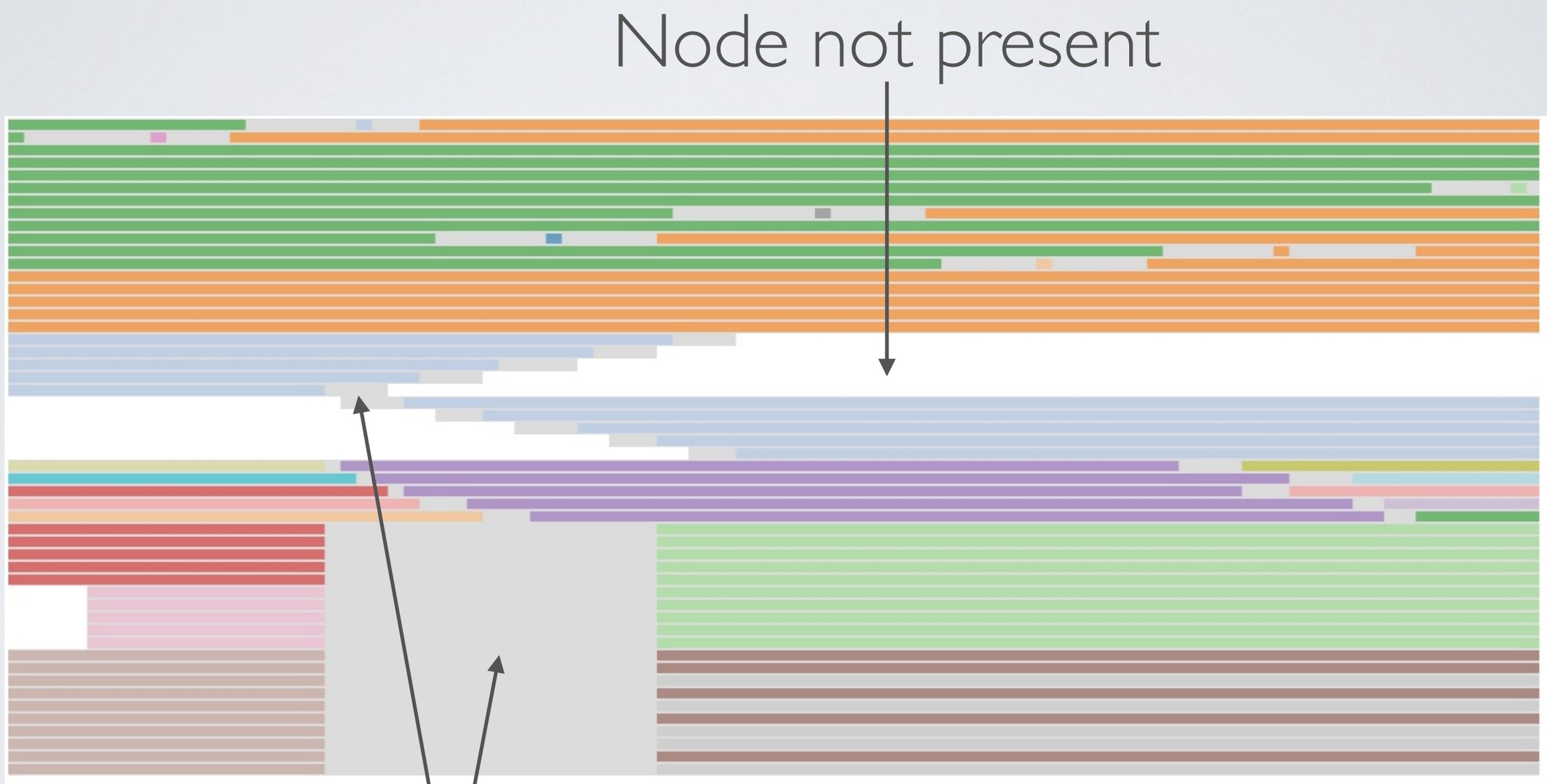


SYNTHETIC NETWORK



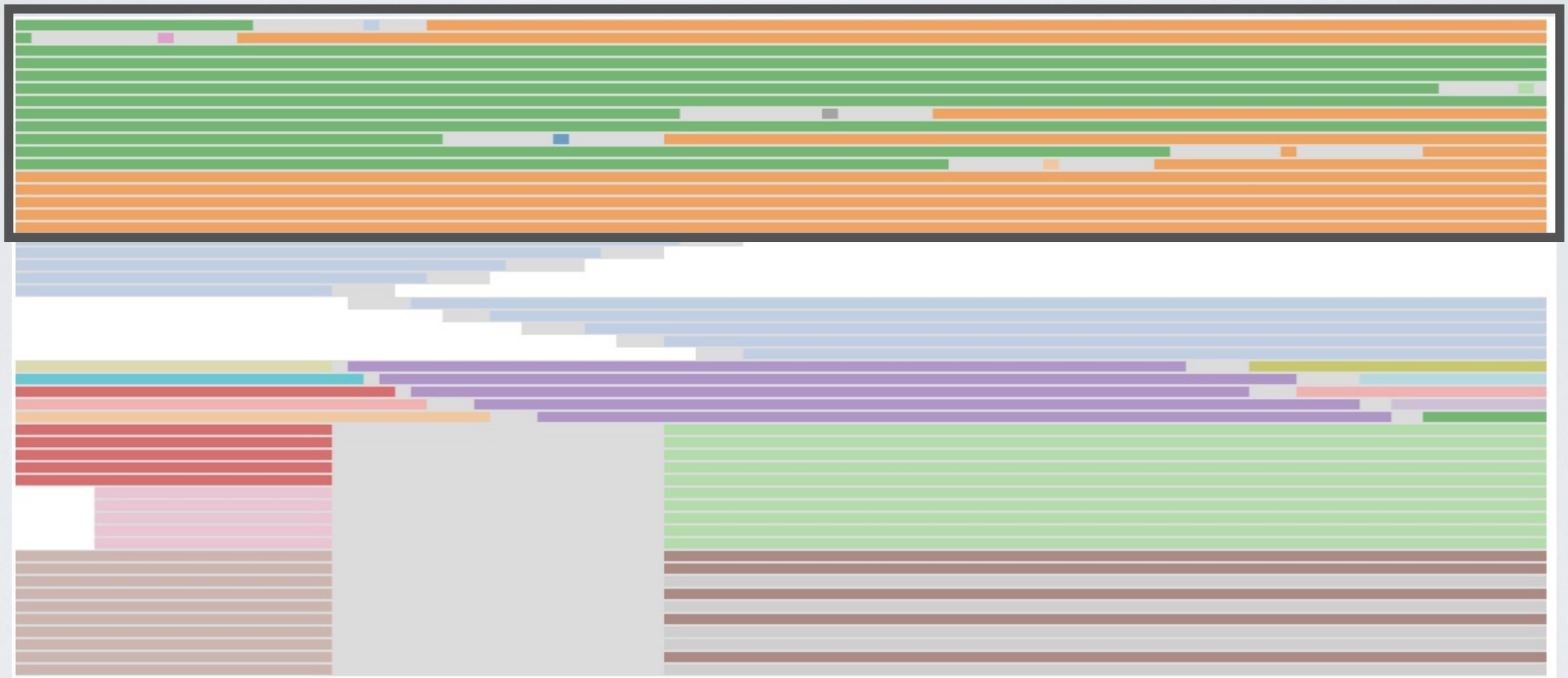
Time

SYNTHETIC NETWORK



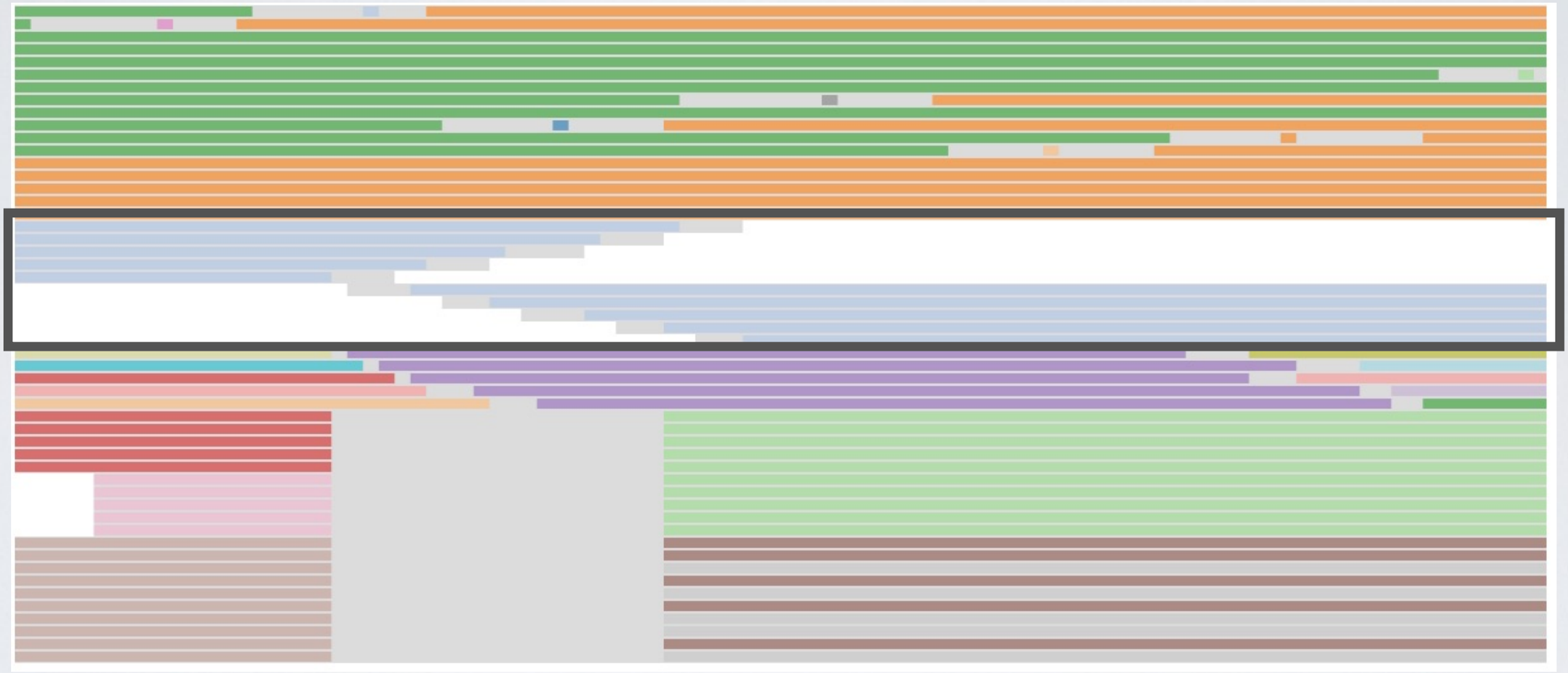
SYNTHETIC NETWORK

Migration



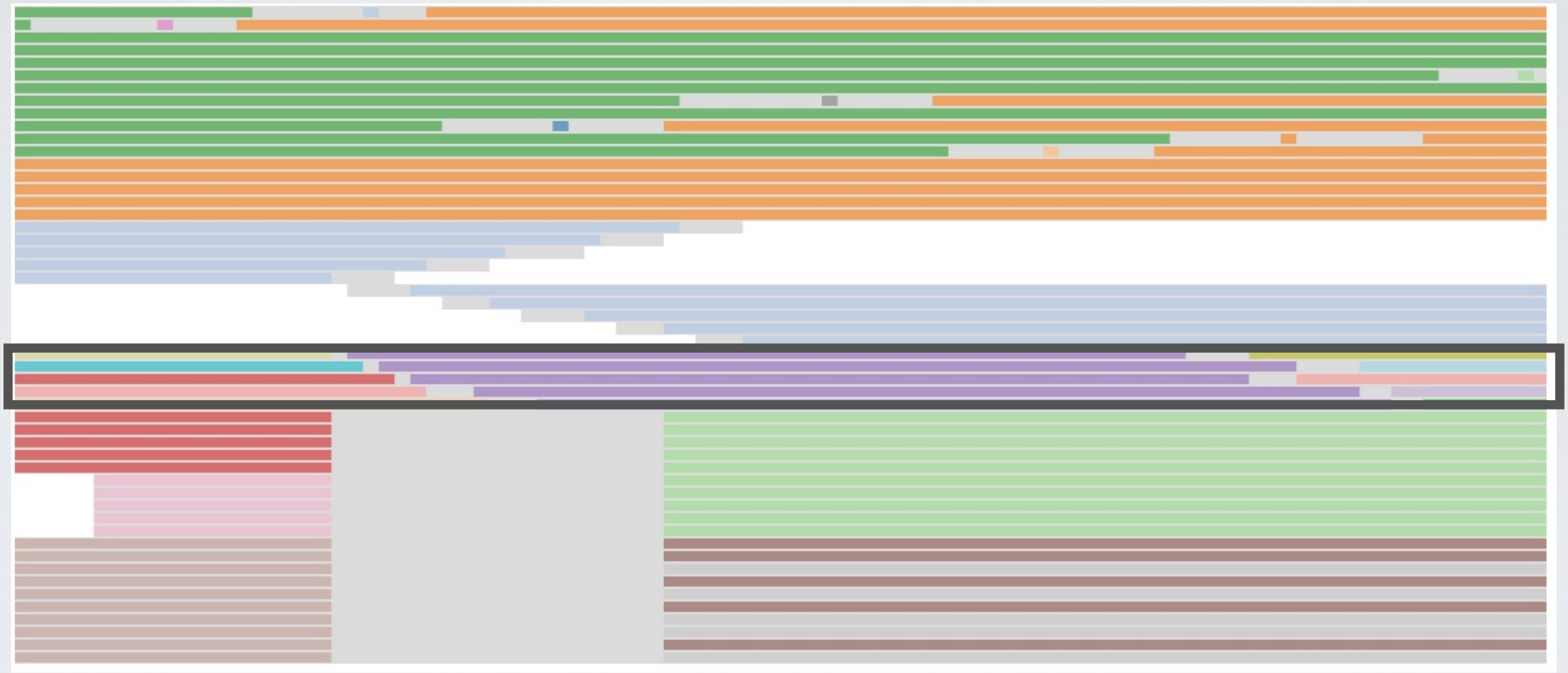
SYNTHETIC NETWORK

Theseus boat



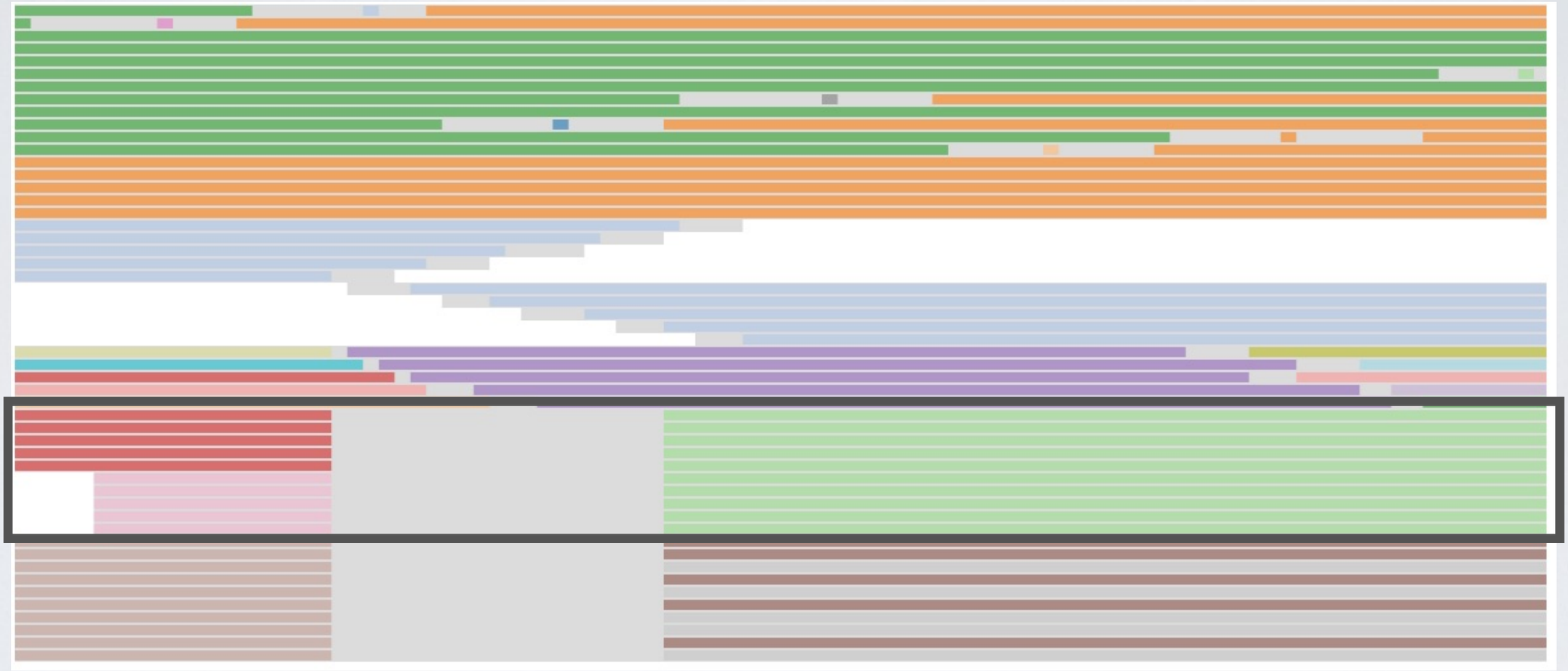
SYNTHETIC NETWORK

Birth and death



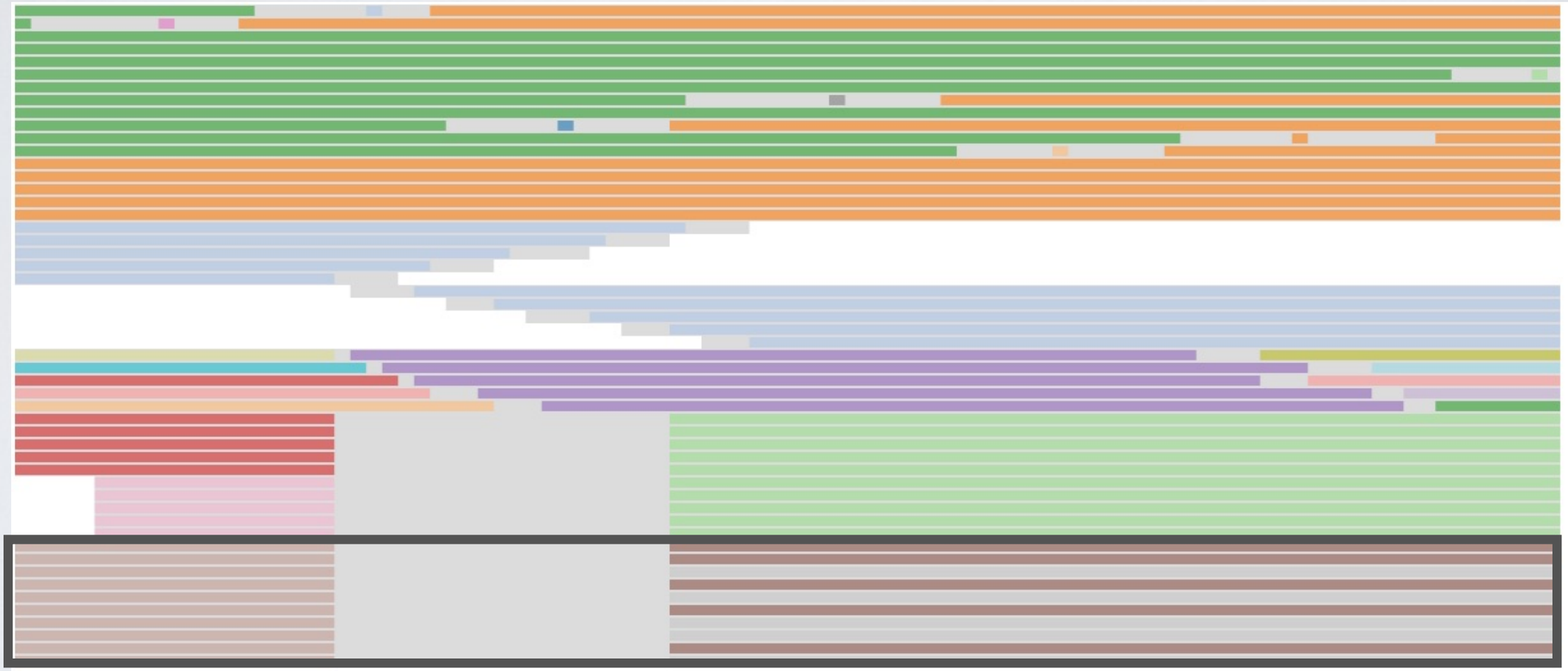
SYNTHETIC NETWORK

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

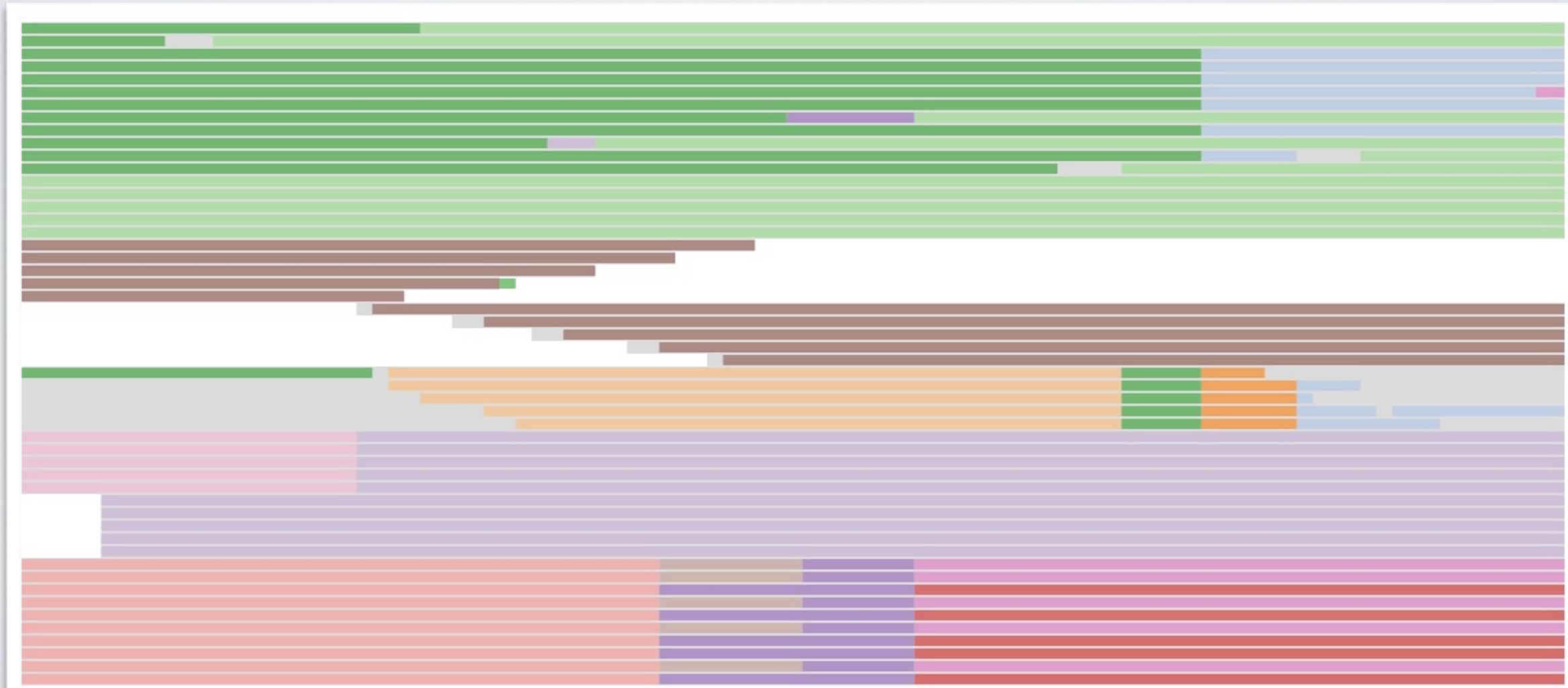
SYNTHETIC NETWORK

Instant Optimal:
Greene et al. 2011

- 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

SYNTHETIC NETWORK

Instant Optimal:
Greene et al. 2011



SYNTHETIC NETWORK

Instant Optimal:
Greene et al. 2011

SYNTHETIC NETWORK

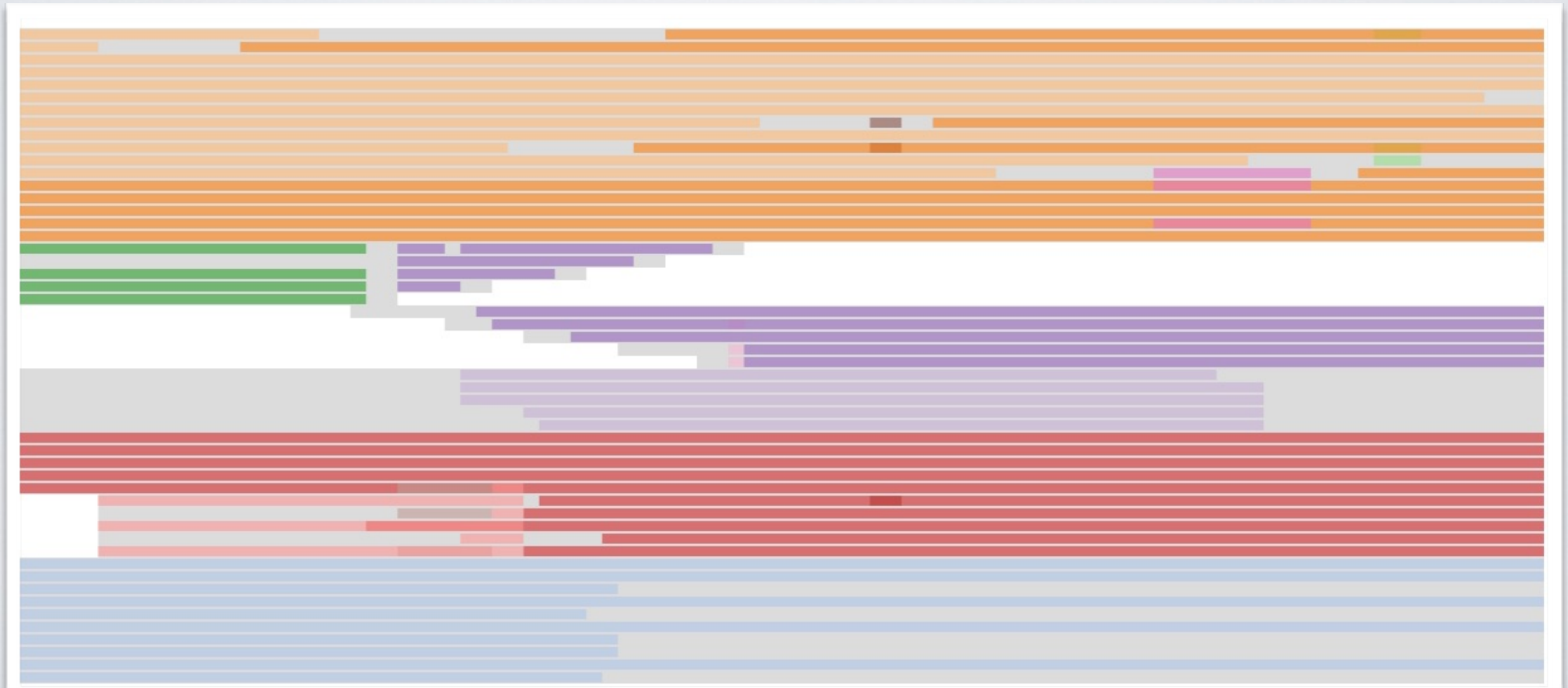
Temporal trade-off:
Cazabet et al. 2010

- 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)

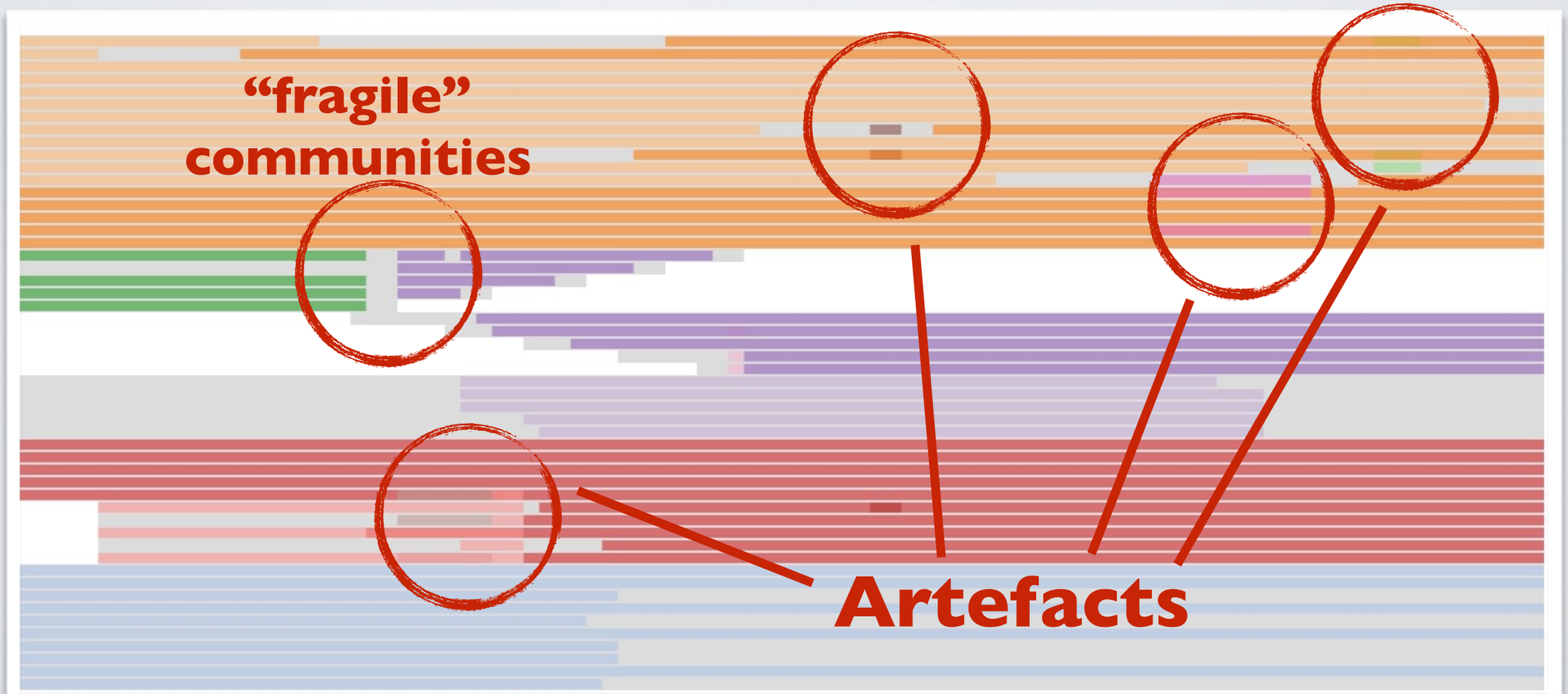
SYNTHETIC NETWORK

Temporal trade-off:
Cazabet et al. 2010



SYNTHETIC NETWORK

Temporal trade-off:
Cazabet et al. 2010



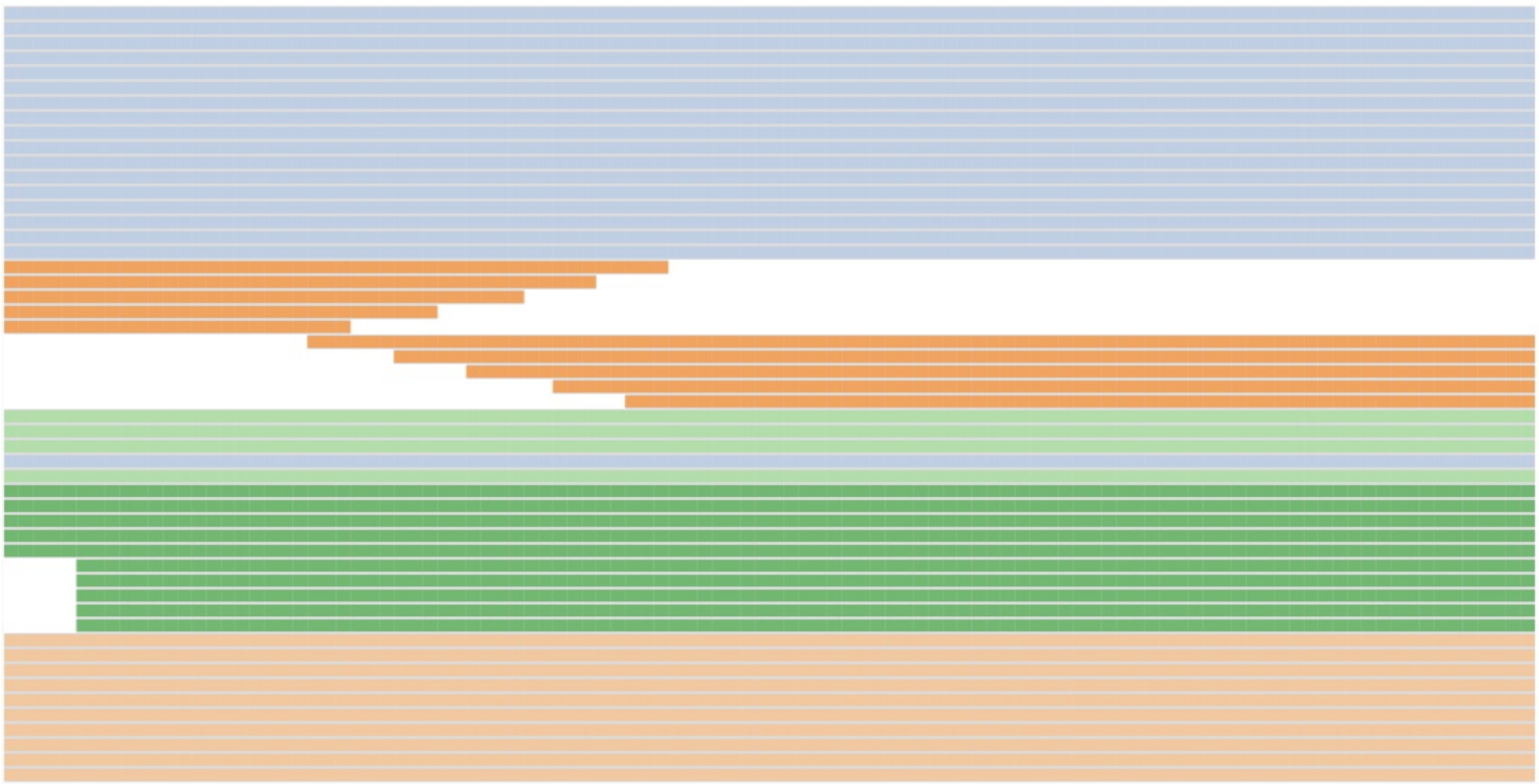
SYNTHETIC NETWORK

Cross-Time:
Mucha et al. 2010

- 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

SYNTHETIC NETWORK

Cross-Time:
Mucha et al. 2010



CONCLUSION

- Work in progress :
  - compare more methods
  - test on more datasets

GRAPH EMBEDDING FOR DYNAMIC COMMUNITY DETECTION

NETWORK EMBEDDING

- Have attracted a lot of attention in the last 3 years
- Deepwalk: 2014: 765 citations
- Node2vec: 2016: 536 citation
- Survey by Goyal/ferrara: end of 2017, 43 citations
- Methods using matrix factorization, random walks, deep learning…

IN CONCRETE TERMS

- A graph is composed of
  - Nodes (possibly with labels)
  - Edges (possibly directed, with labels)

- A graph embedding technique in **d** dimension will assign a vector of length **d** to each node, that will be useful for **what we want to do with the graph**.

- A vector can be assigned to an edge *(u,v)* by combining vectors of *u* and *v using *your favorite operation**

WHY EMBEDDINGS ?

- Machine Learning/Data mining/IA techniques => very popular and quite successful for supervised and unsupervised tasks.

- These techniques take as input **vectors** (and only vectors).

- Vectors must contain all relevant information and be as small as possible

- ==>Transform graphs into vectors (of low dimensions).

- Some methods are highly scalable, for instance based on skipgram

NAIVE EXEMPLE OF EMBEDDING

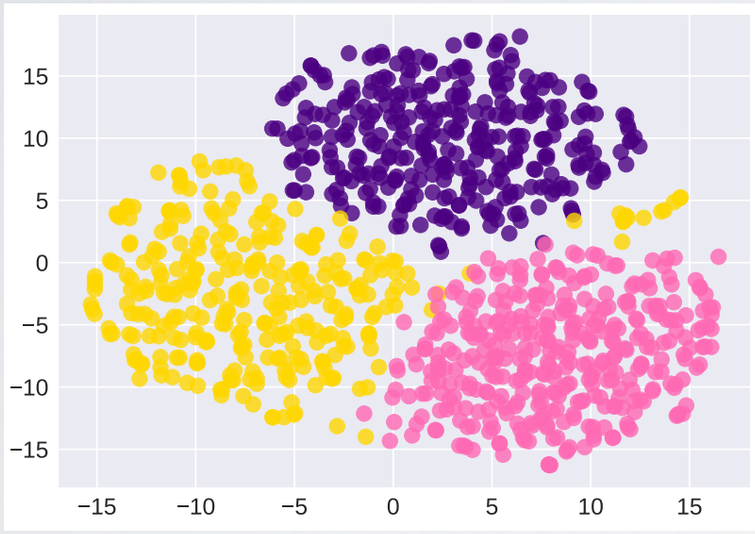
Optimize a cost function defined as:

Distance in the graph- Distance in the embedding

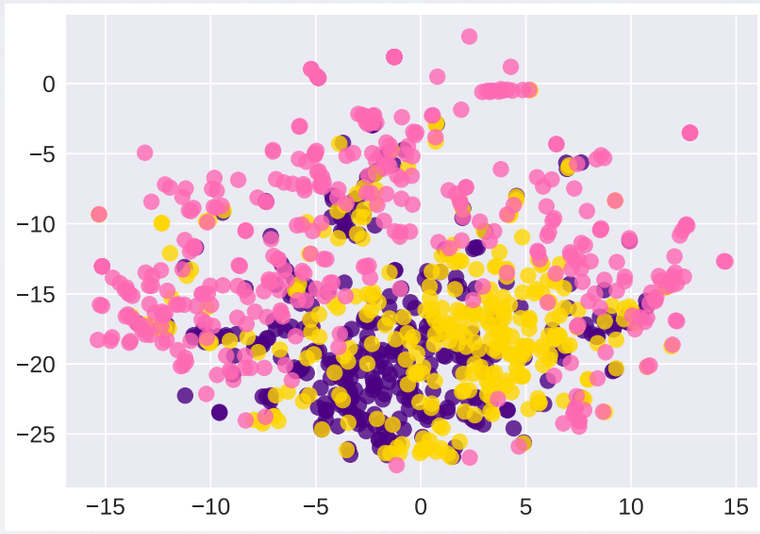
COMMUNITY DETECTION USING EMBEDDINGS

VISUALLY

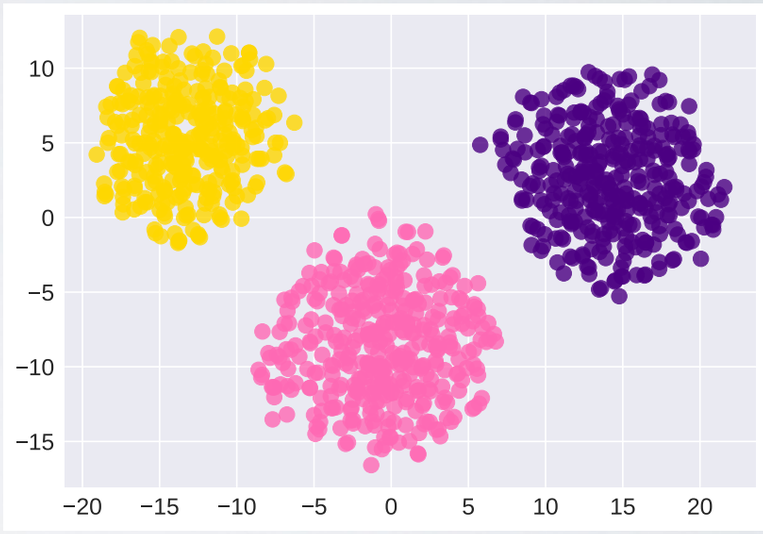
SBM, 3 communities, intern:0.1, extern 0.01



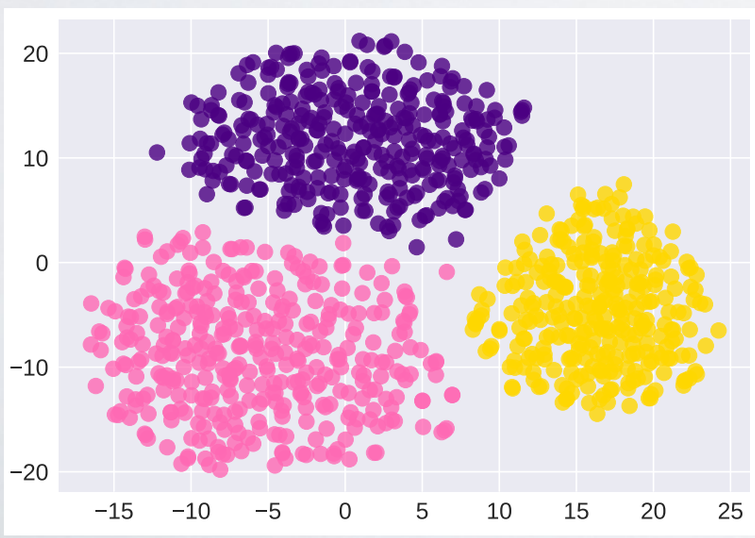
(a) LLE



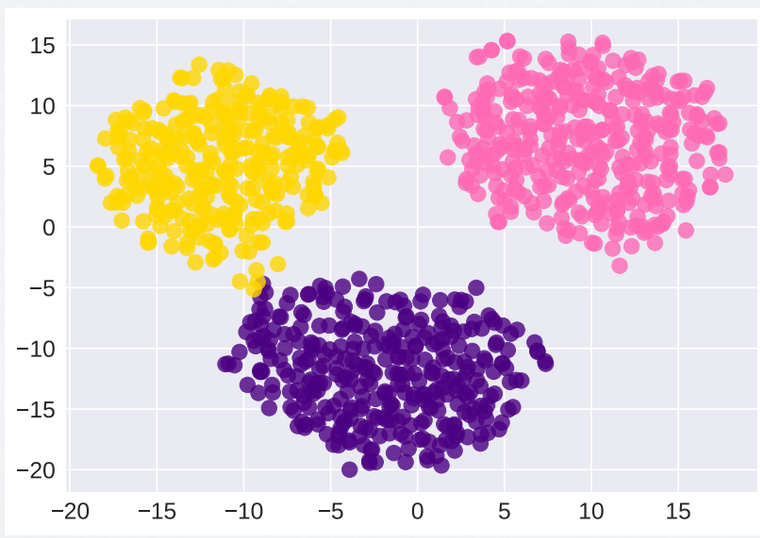
(b) GF



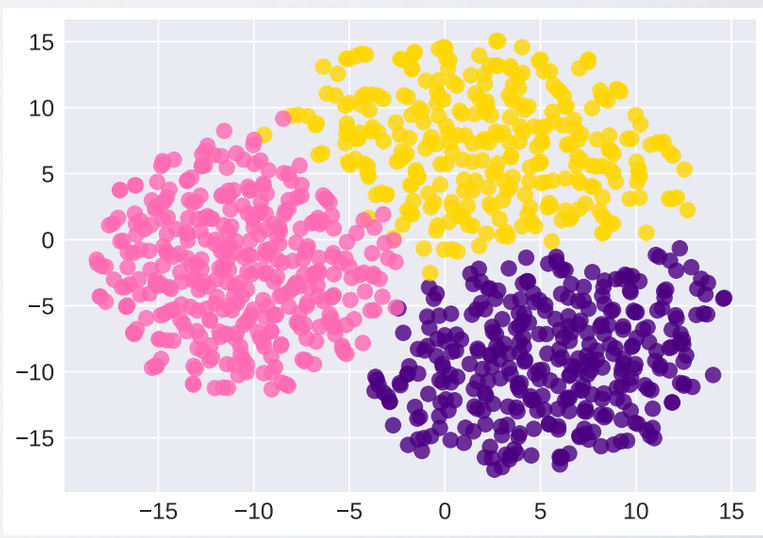
(c) *node2vec*



(d) HOPE



(e) SDNE



(f) LE

MAIN IDEA

- 1) Embed a graph
- 2) Use clustering method to find communities

- It is also possible to to "supervised" cluster detection, which corresponds to node classification (discover labels of nodes)

EMPIRICALLY

Real networks, try k-means++ with 2<=k<=50, Keep highest modularity

method	CoCit	CoAuthor	VK	YouTube	Orkut
FVERSE	70.12	80.95	44.59	—	—
VERSE	69.43	79.25	45.78	67.63	42.64
DEEPWALK	70.04	73.83	43.30	58.08	44.66
LINE	60.02	71.58	39.65	63.40	42.59
GRAREP	67.61	77.40	—	—	—
HOPE	42.45	69.57	21.70	37.94	—
HSVERSE	69.81	79.31	45.84	69.13	—
NODE2VEC	70.06	75.78	44.27	—	—
Louvain	72.05	84.29	46.60	71.06	—

Table 12: Node clustering results in terms of modularity.

Argument: "more scalable than Louvain"

EMBEDDING DYNAMIC NETWORKS

OVERVIEW

- The embedding captures both
  - Structural similarity
  - Temporal similarity / temporal accessibility

STATE OF THE ART

- Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change (2016) (snapshot by snapshot)

- Scalable Temporal Latent Space Inference for Link Prediction in Dynamic Social Networks (2016)(smoothed snapshots)

- DynGEM: Deep Embedding Method for Dynamic Graphs (2018)(Deep auto encoder, incremental learning on snapshots

- Combining Temporal Aspects of Dynamic Networks with node2vec for a more Efficient Dynamic Link Prediction (To Be Published/2018) (node2vec using time-aware random walks)

- To be published: Marton Karsai et al.

THANK YOU