DYNAMIC NETWORKS

(Dynamic of networks)

DYNAMIC NETWORKS

- Most real world networks are dynamic
  - Facebook friendship
    - People joining/leaving
    - Friend/Unfriend
  - Twitter mention network
    - Each mention has a timestamp
    - Aggregated every day/month/year => still dynamic
  - World Wide Web
  - Urban network
  - …

DYNAMIC NETWORKS

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

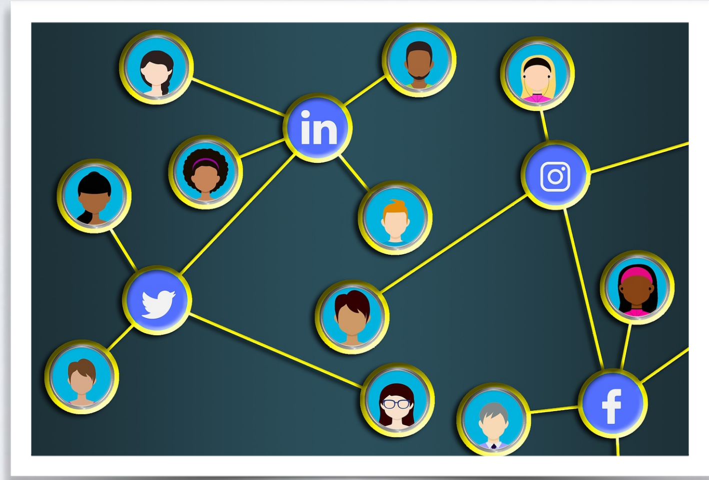
DYNAMIC NETWORKS

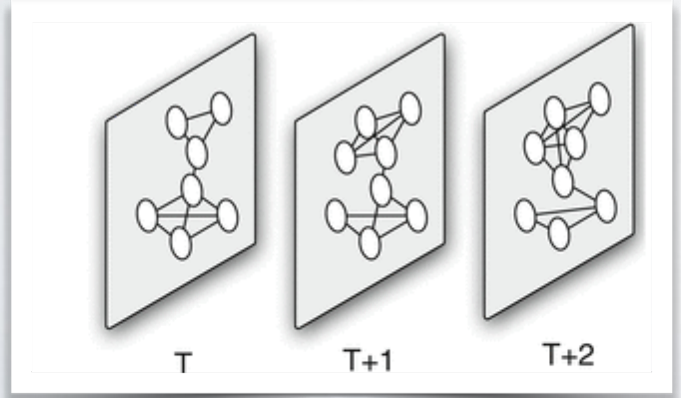
Dynamic Network Properties

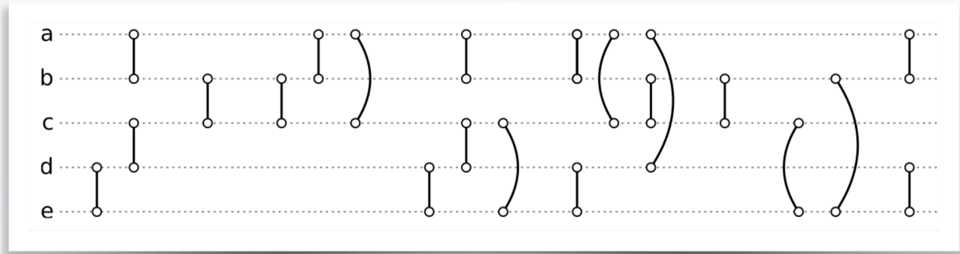
Independently of the studied data, dynamic networks can have various properties:

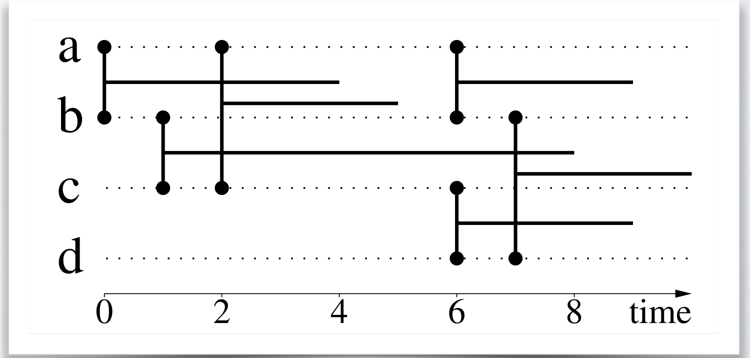
- **Edge** presence can be **punctual** or **with duration**
- **Node** presence can be **unspecified**, **punctual** or **contin-uous**
- If **time is continuous**, it can be **bounded** on a period of analysis or **ubounded**
- If **nodes** have attributes, they can be **constant** or **time-dependent**
- If **edges** have weights, they can be **constant** or **time-dependent**

SEVERAL FORMALISMS









TEMPORAL NETWORK

Collected dataset, for instance in (t,u,v) format

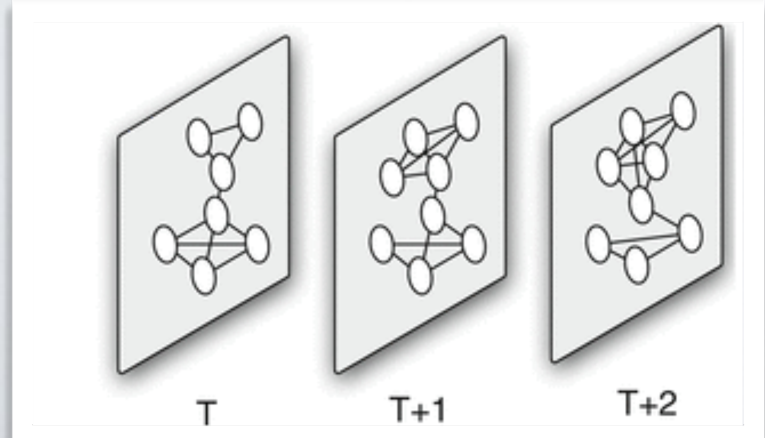
Time	u	v
1353304100	1148	1644
1353304100	1613	1672
1353304100	656	682
1353304100	1632	1671
1353304120	1492	1613
1353304120	656	682
1353304120	1632	1671
1353304140	1148	1644
1353304160	656	682
1353304160	1108	1601
1353304160	1632	1671
1353304160	626	698

Examples:
-SocioPatterns
-Enron
-…

TEMPORAL NETWORK

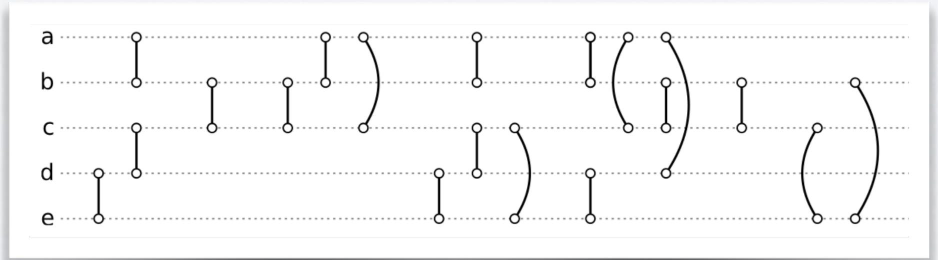
Snapshots

1353304100	1148	1644
1353304100	1613	1672
1353304100	656	682
1353304100	1632	1671
1353304120	1492	1613
1353304120	656	682
1353304120	1632	1671
1353304140	1148	1644
1353304160	656	682
1353304160	1108	1601
1353304160	1632	1671
1353304160	626	698



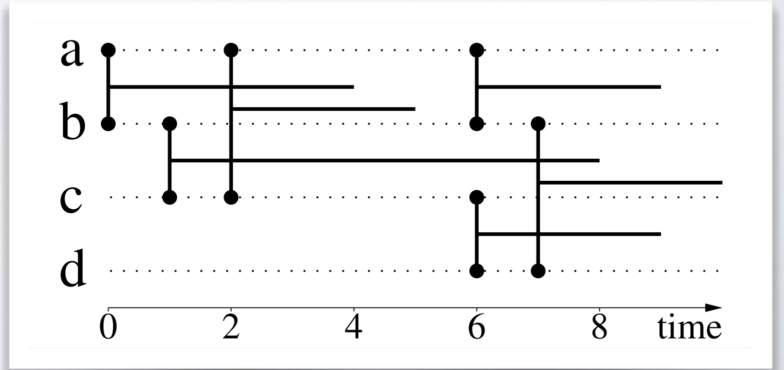
Link Stream

1353304100	1148	1644
1353304100	1613	1672
1353304100	656	682
1353304100	1632	1671
1353304120	1492	1613
1353304120	656	682
1353304120	1632	1671
1353304140	1148	1644
1353304160	656	682
1353304160	1108	1601
1353304160	1632	1671
1353304160	626	698

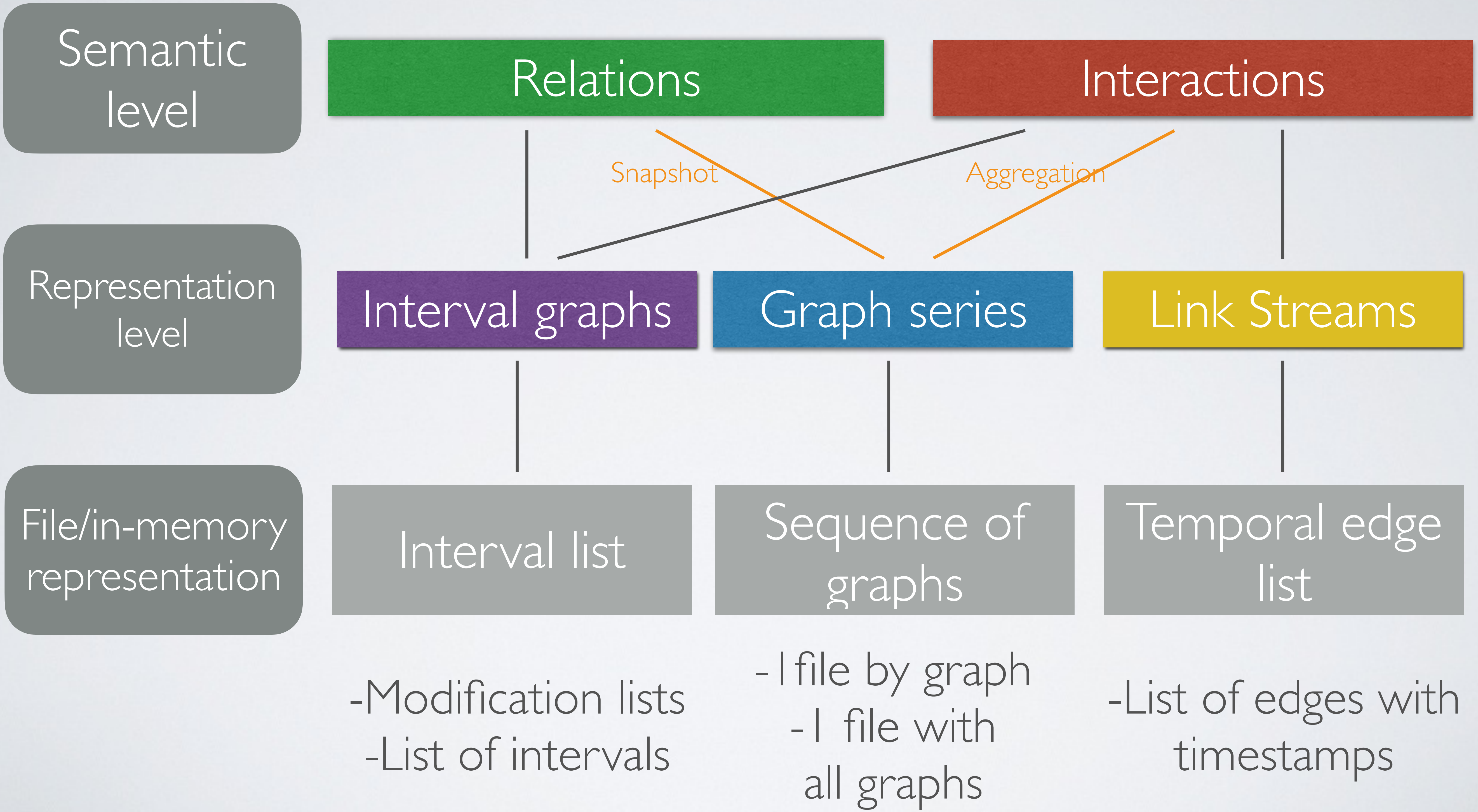


Interval Graph

1353304100	1148	1644
1353304100	1613	1672
1353304100	656	682
1353304100	1632	1671
1353304120	1492	1613
1353304120	656	682
1353304120	1632	1671
1353304140	1148	1644
1353304160	656	682
1353304160	1108	1601
1353304160	1632	1671
1353304160	626	698



DYNAMIC NETWORKS



DYNAMIC NETWORKS

Vocabulary

Many different names have been used to for networks changing with time, but there is no broad consensus in the literature on the meaning of those terms, unless they are used with an explicit reference to a paper defining those terms. Here is a list of the most popular:

- **Dynamic Networks** and **Dynamic Graphs**
- **Longitudinal Networks**
- **Evolving Graphs**
- **Link Streams** & **Stream Graphs** (Latapy, Viard, and Magnien 2018)
- **Temporal Networks**, **Contact Sequences** and **Interval Graphs** (Holme and Saramäki 2012)
- **Time Varying Graphs** (Casteigts et al. 2012)

ANALYZING DYNAMIC NETWORKS

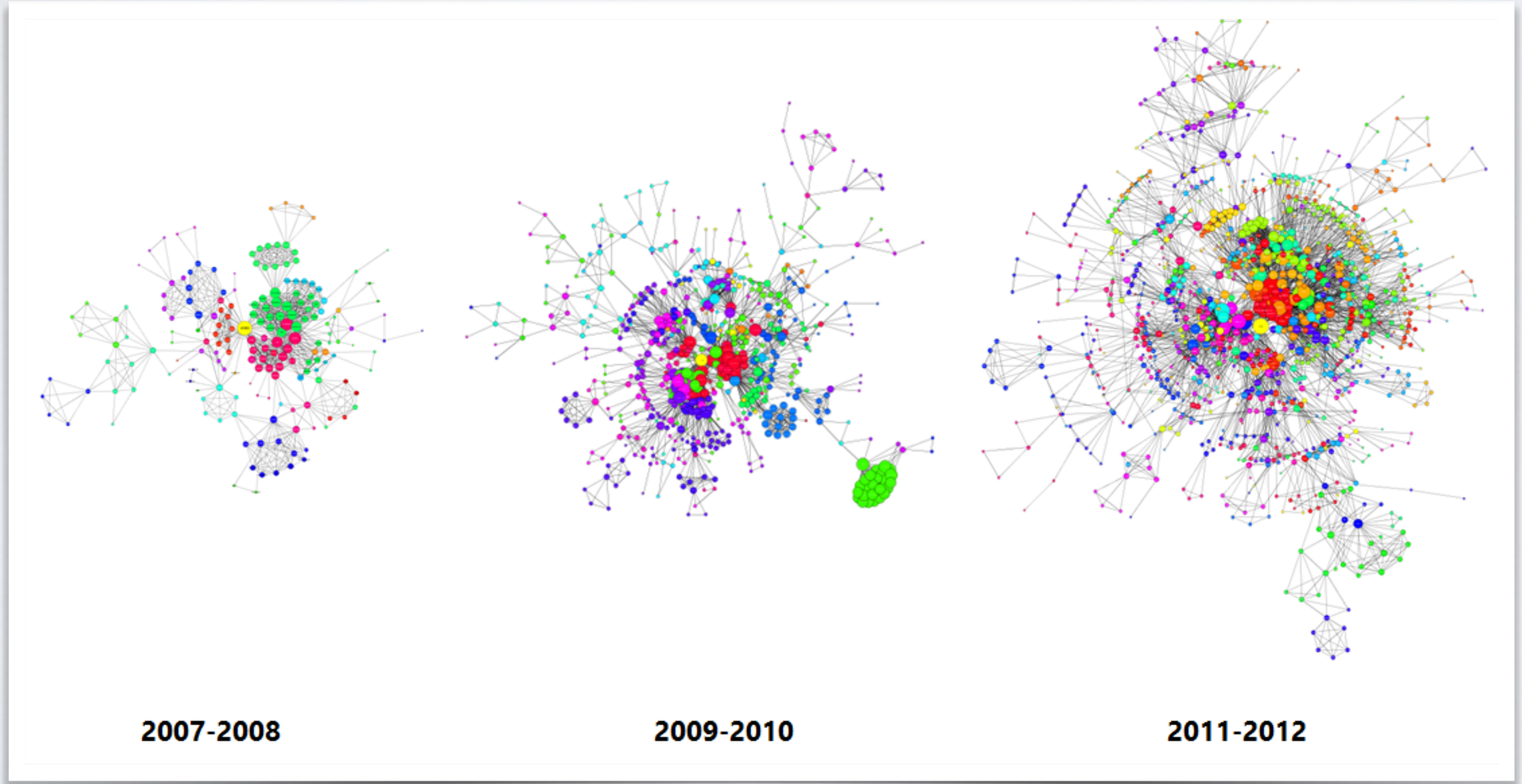
ANALYZING DYNAMIC NETWORKS

- Few snapshots
- Slowly Evolving Networks (SEN)
- Degenerate/Unstable temporal networks

FEW SNAPSHOTS

FEW SNAPSHOTS

- The evolution is represented as a series of *a few* snapshots.
- Many changes between snapshots
  - Cannot be visualized as a "movie"



FEW SNAPSHOTS

- Each snapshot can be studied as a static graph
- The evolution of the properties can be studied "manually"
- "Node X had low centrality in snapshot t and high centrality in snapshot t+n"

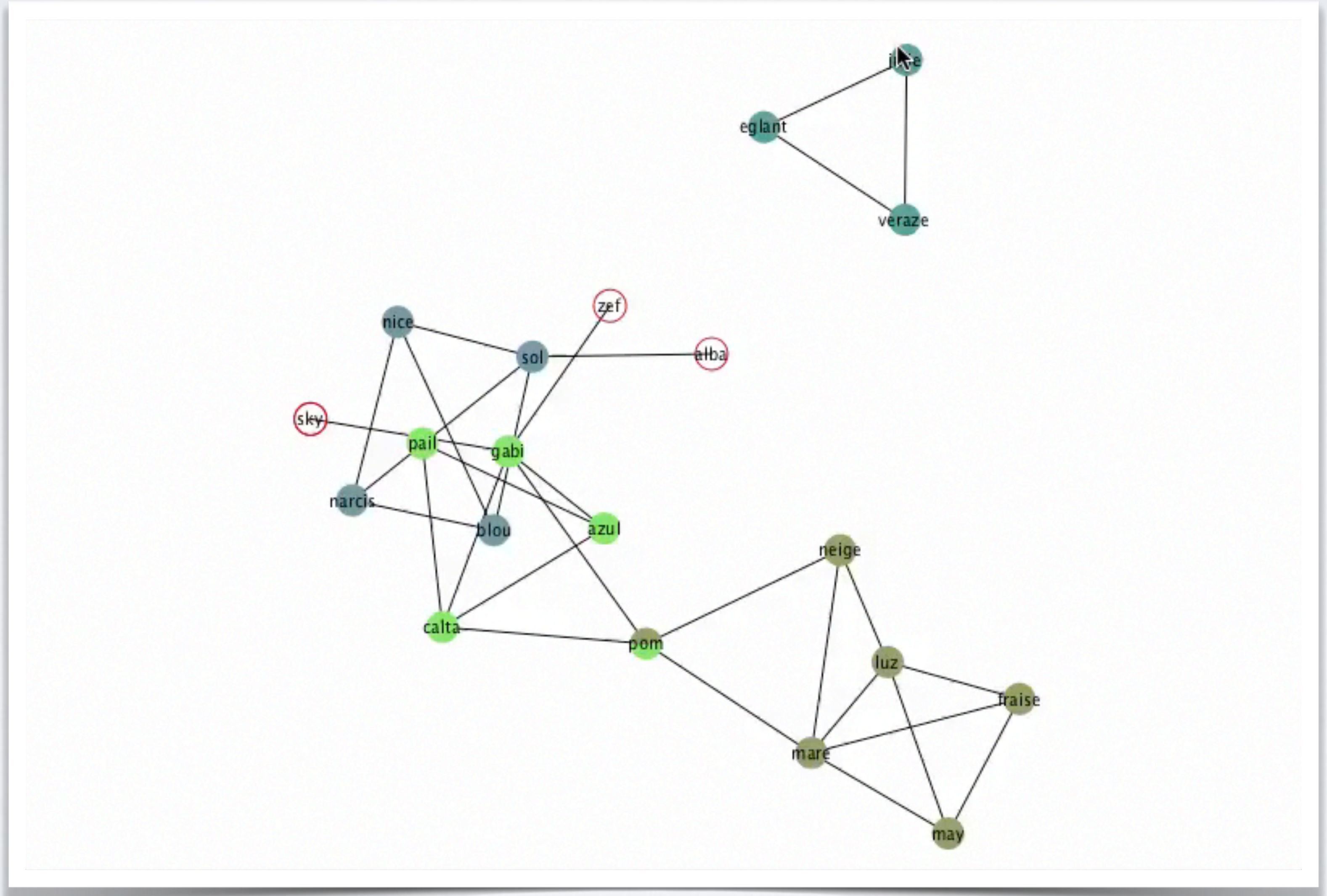
SLOWLY EVOLVING NETWORKS (SEN)

SLOWLY EVOLVING NETWORKS

- Edges change (relatively) slowly
- The network is well defined at any t
  - Nodes/edges described by (long lasting) intervals
  - Enough snapshots to track nodes
- A static analysis at every (relevant) t gives a dynamic vision
- No formal distinction with previous case (higher observation frequency)

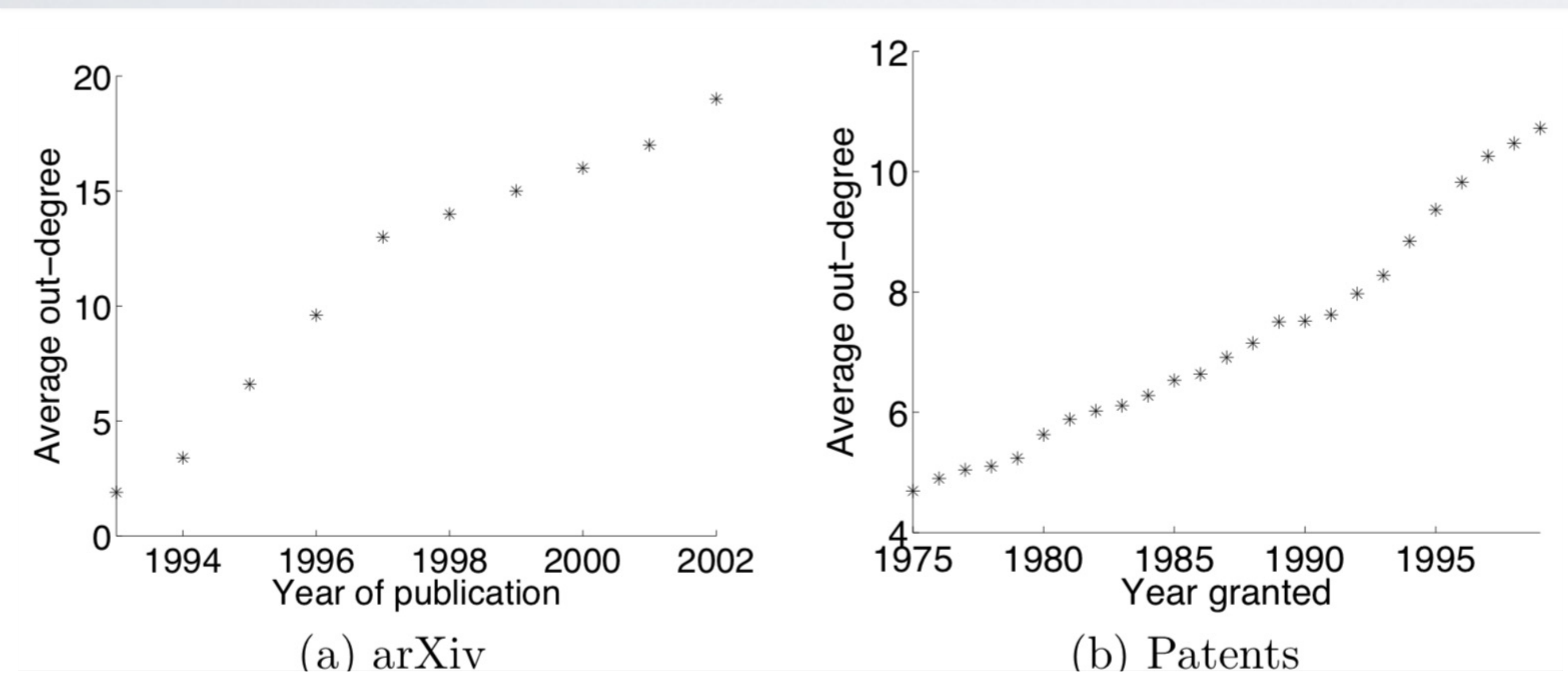
SLOWLY EVOLVING NETWORKS

- Visualization
  - Problem of stability of node positions

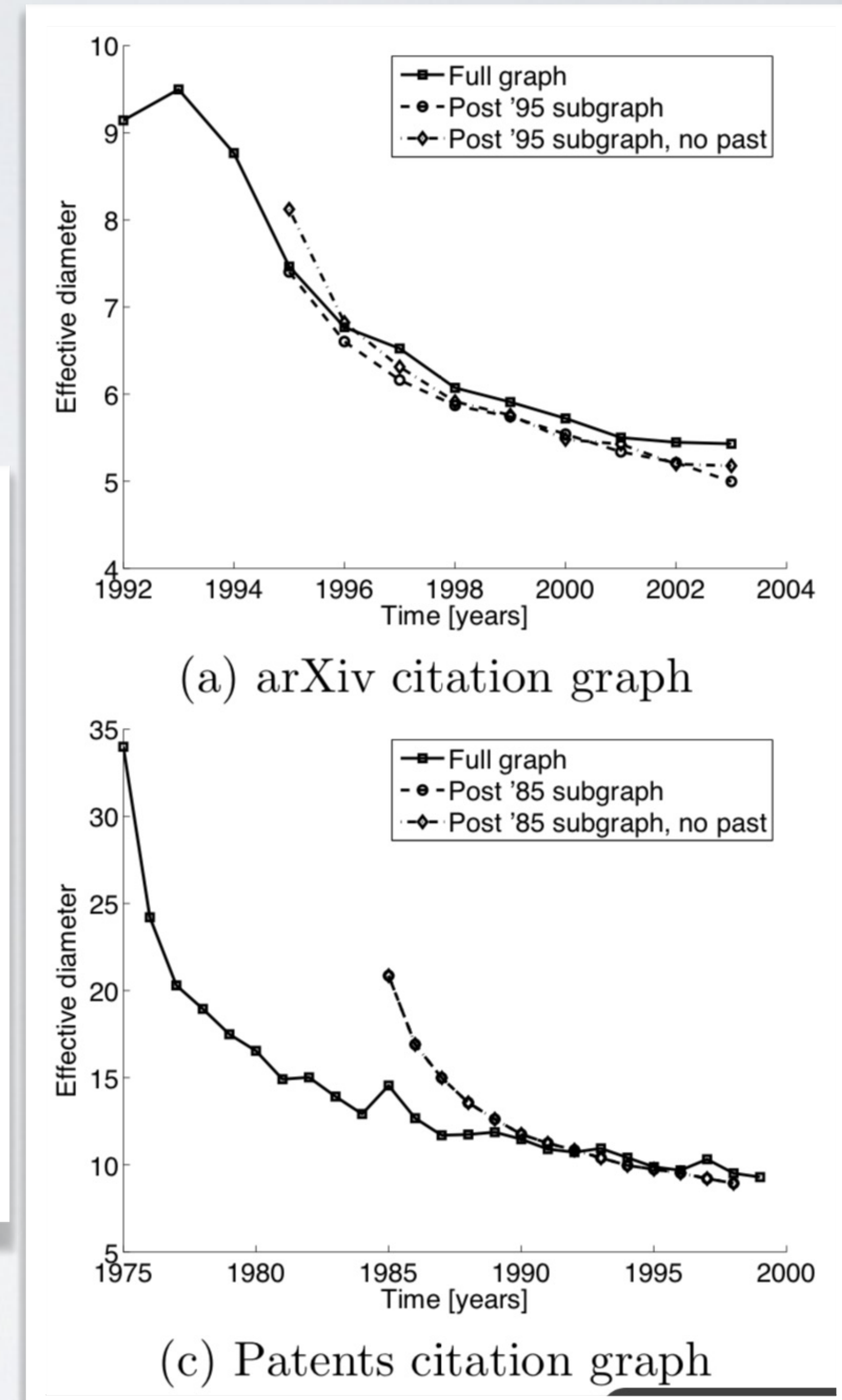


SLOWLY EVOLVING NETWORKS

- Global graph properties



(a) arXiv (b) Patents



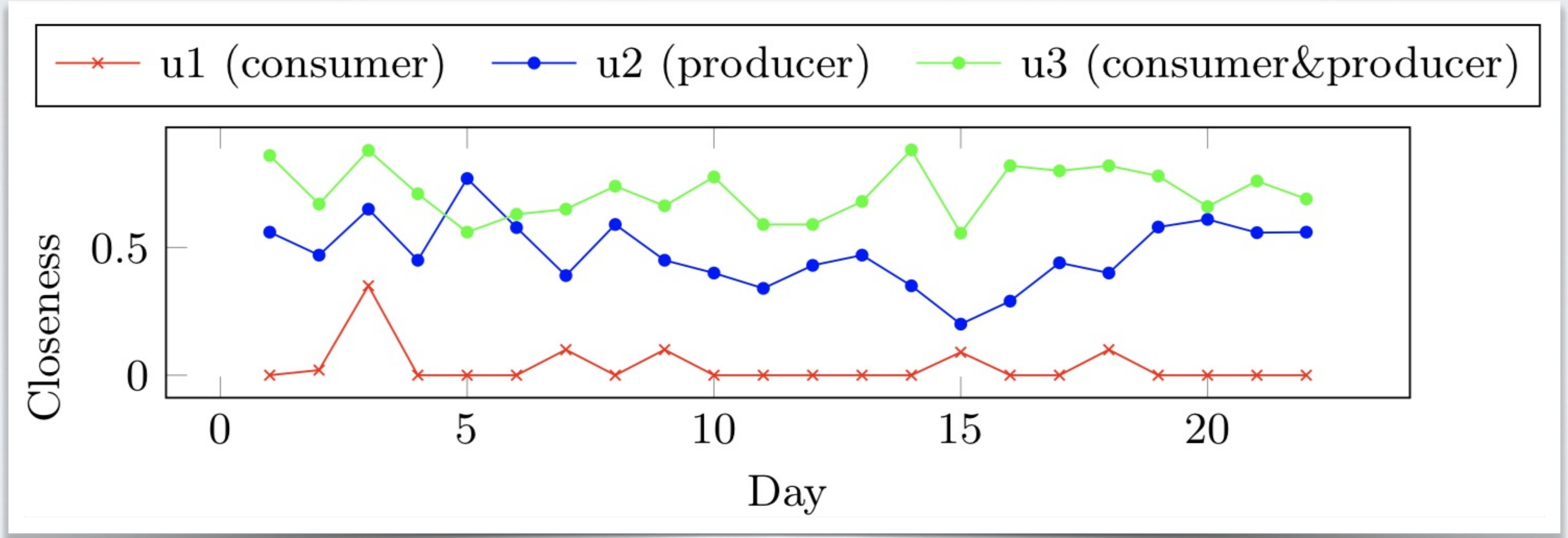
(a) arXiv citation graph

(c) Patents citation graph

Leskovec, Jure, Jon Kleinberg, and Christos Faloutsos. "Graph evolution: Densification and shrinking diameters." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007): 2.

SLOWLY EVOLVING NETWORKS

- Centralities



TIME SERIES ANALYSIS

- TS analysis is a large field of research

- Time series: evolution of a value over time
  - Stock market, temperatures…

- Typical questions:
  - Detection of periodic patterns
  - Detection of anomalies
  - Identification of global trends
  - Measure of auto-correlation
  - Prediction of future values

- e.g. ARIMA (Autoregressive integrated moving average)

https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

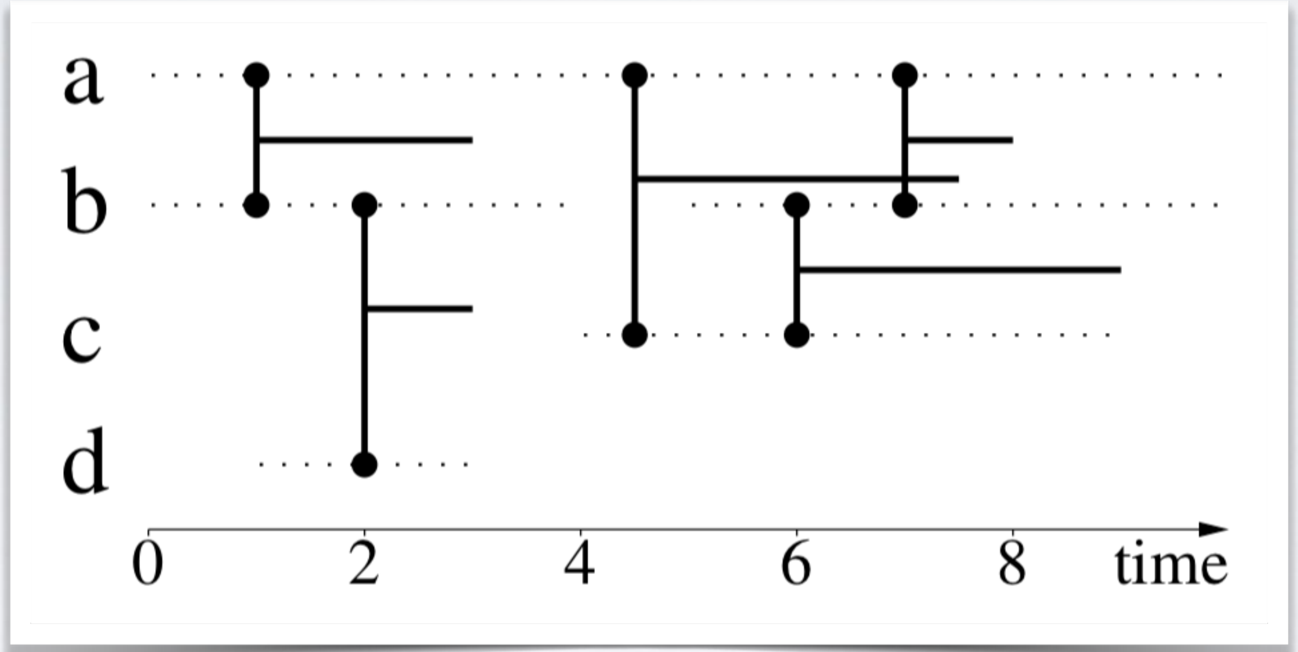
UNSTABLE/DEGENERATE TEMPORAL NETWORKS

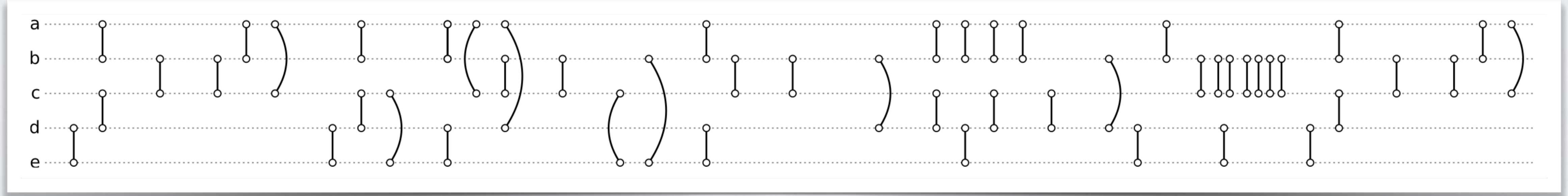
Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. "Stream graphs and link streams for the modeling of interactions over time". In: *Social Network Analysis and Mining* 8.1 (2018), p. 61.

UNSTABLE TEMPORAL NETWORK

- The network at a given *t* is not meaningful
- How to analyze such a network?

UNSTABLE TEMPORAL NETWORK





UNSTABLE TEMPORAL NETWORK

- Common solution: transform into SEN using aggregation/ sliding windows
  - Information loss
  - How to chose a proper aggregation window size?

- New theoretical tools developed to deal with such networks

> - **Link Streams** & **Stream Graphs** (Latapy, Viard, and Magnien 2018)
> - **Temporal Networks**, **Contact Sequences** and **Interval Graphs** (Holme and Saramäki 2012)
> - **Time Varying Graphs** (Casteigts et al. 2012)

CENTRALITIES
&
NETWORK PROPERTIES
IN STREAM GRAPHS

STREAM GRAPHS

Stream Graph (SG)- Definition

Stream Graphs have been proposed in[a] as a generic formalism – it can represent any type of dynamic networks, continuous, discrete, with or without duration, with the objective or redefining typical notions of graphs on dynamic networks, including degenerate ones.
Let's define a Stream Graph

$$S = (T, V, W, E)$$

T	**Set of Possible times** (Discrete or Time intervals)
V	**Set of Nodes**
W	**Vertices presence time** $V \times T$
E	**Edges presence time** $V \times V \times T$

[a] Latapy, Viard, and Magnien 2018.

STREAM GRAPHS

SG - Time-Entity designation

It is useful to work with Stream Graphs to introduce some new notions mixing entities (nodes, edges) and time:

V_t	**Nodes At Time**: set of nodes present at time t
E_t	**Edges At Time**: set of edges present at time t
G_t	**Snapshot**: Graph at time t, $G_t = (V_t, E_t)$
v_t	**Node-time**: v_t exists if node v is present at time t
$(u,v)_t$	**Edge-time**: $(u,v)_t$ exists if edge (u,v) is present at time t
T_u	**Times Of Node**: the set of times during which u is present
T_{uv}	**Times Of Edge**: the set of times during which edge (u,v) is present

STREAM GRAPHS

N_u	**Node presence**: The fraction of the total time during which u is present in the network $\frac{\lvert T_u \rvert}{\lvert T \rvert}$
L_{uv}	**Edge presence**: The fraction of the total time during which (u, v) is present in the network $\frac{\lvert T_{uv} \rvert}{\lvert T \rvert}$

STREAM GRAPHS

SG - Redefining Graph notions

The general idea of redefining static network properties on Stream Graphs is that if the network stays unchanged along time, then properties computed on the stream graph should yield the same values as the same property computed on the aggregated graph.

STREAM GRAPHS

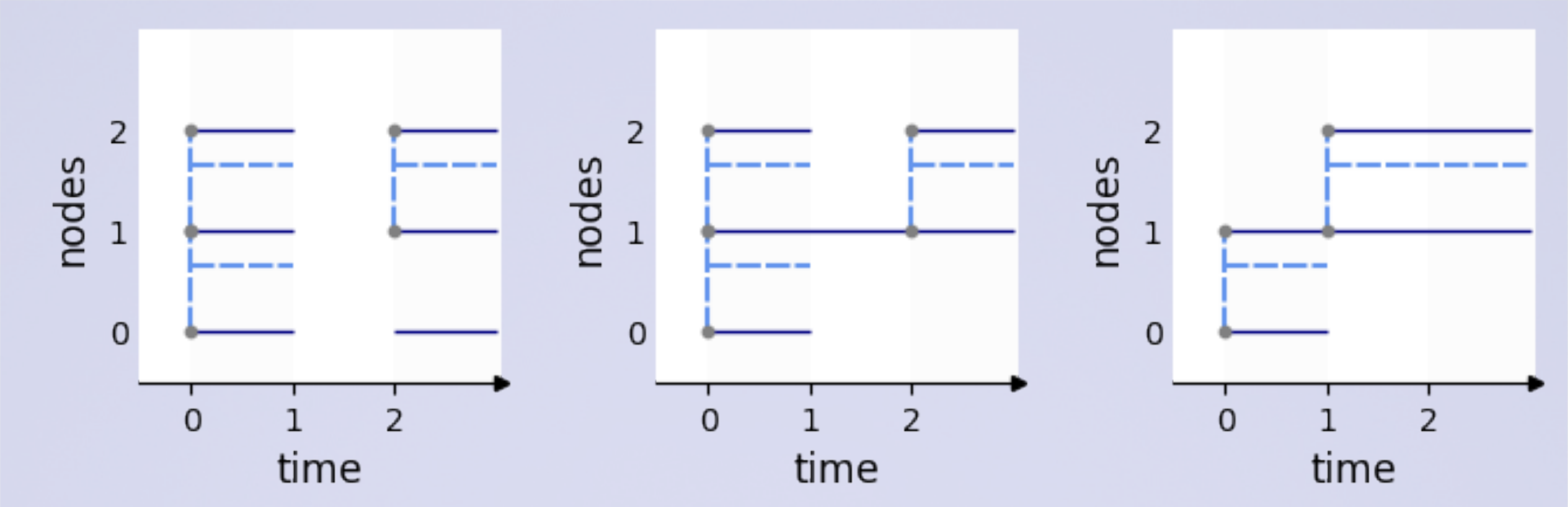
SG - N & L

The number/quantity of nodes in a stream graph is defined as the total presence time of nodes divided by the dataset duration. In general, it isn't an integer.
More formally:

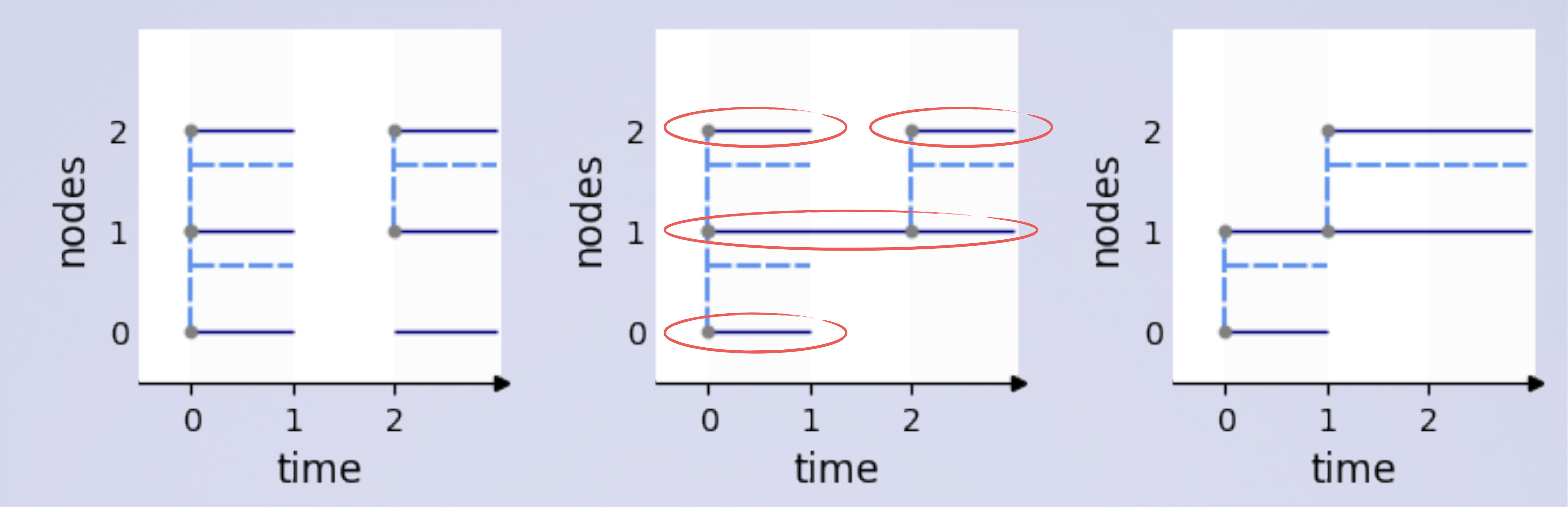
$$N = \sum_{v \in V} N_v = \frac{|W|}{|T|}$$

For instance, $N = 2$ if there are 4 nodes present half the time, or two nodes present all the time.

STREAM GRAPHS



STREAM GRAPHS



$$N = 2$$

STREAM GRAPHS

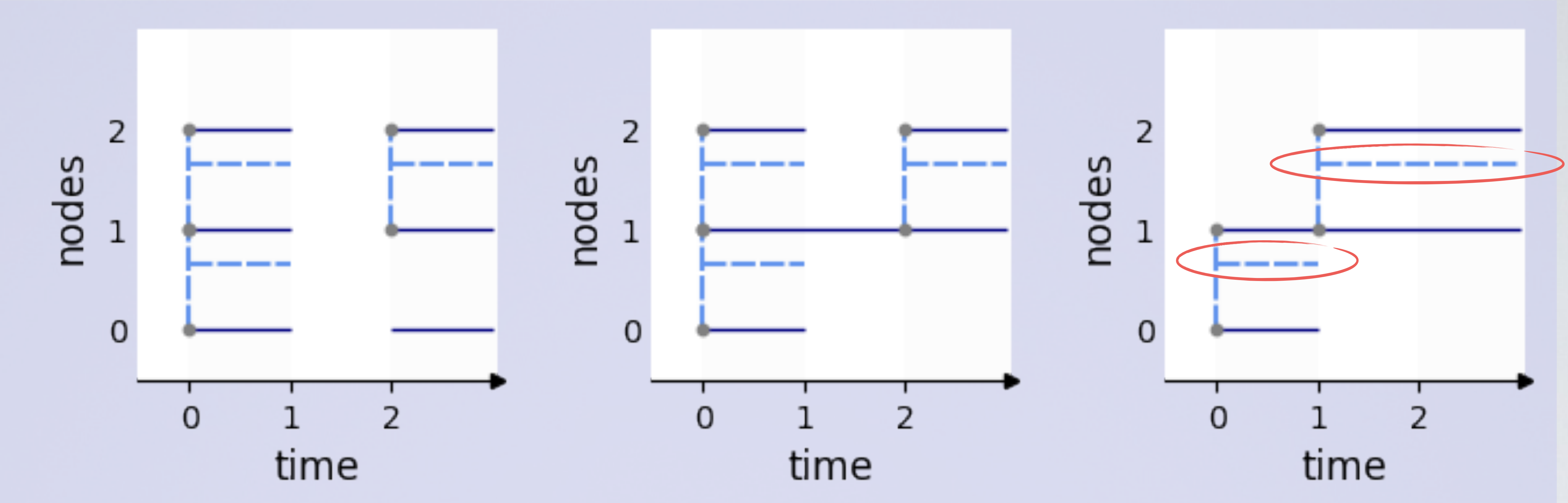
SG - L

The number of edges is defined as the total presence of edges divided by the total dataset duration.
More formally:

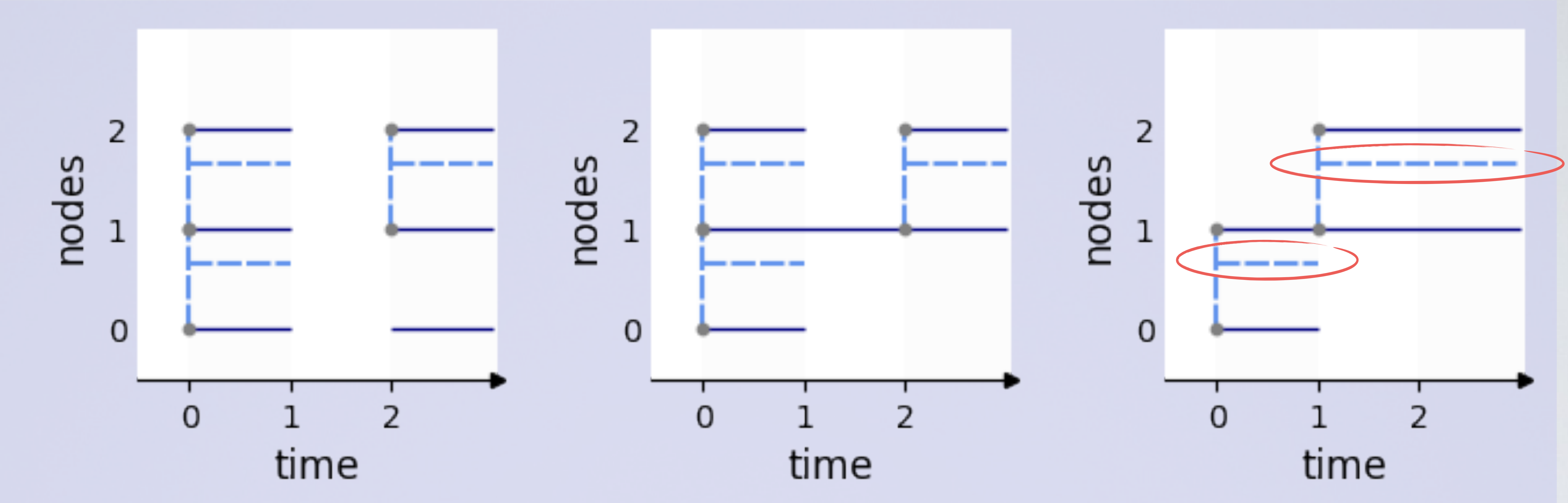
$$L = \sum_{(u,v),u,v \in V} L_{uv} = \frac{|E|}{|T|}$$

For instance, $L = 2$ if there are 4 edges present half the time, or two edges present all the time.

STREAM GRAPHS



STREAM GRAPHS



$$L = 1$$

STREAM GRAPHS

SG - Edge domain - $L_{\max}$

In Stream Graphs, several possible definitions of $L_{\max}$ could exist:

- Ignoring nodes duration: $L^1_{\max} = |V|^2$
- Ignoring nodes co-presence $L^2_{\max} = N^2$
- Taking nodes co-presence into account:
$L_{\max}{}^3 = \sum_{(u,v),u,v \in V} |T_u \bigcap T_v|$

STREAM GRAPHS

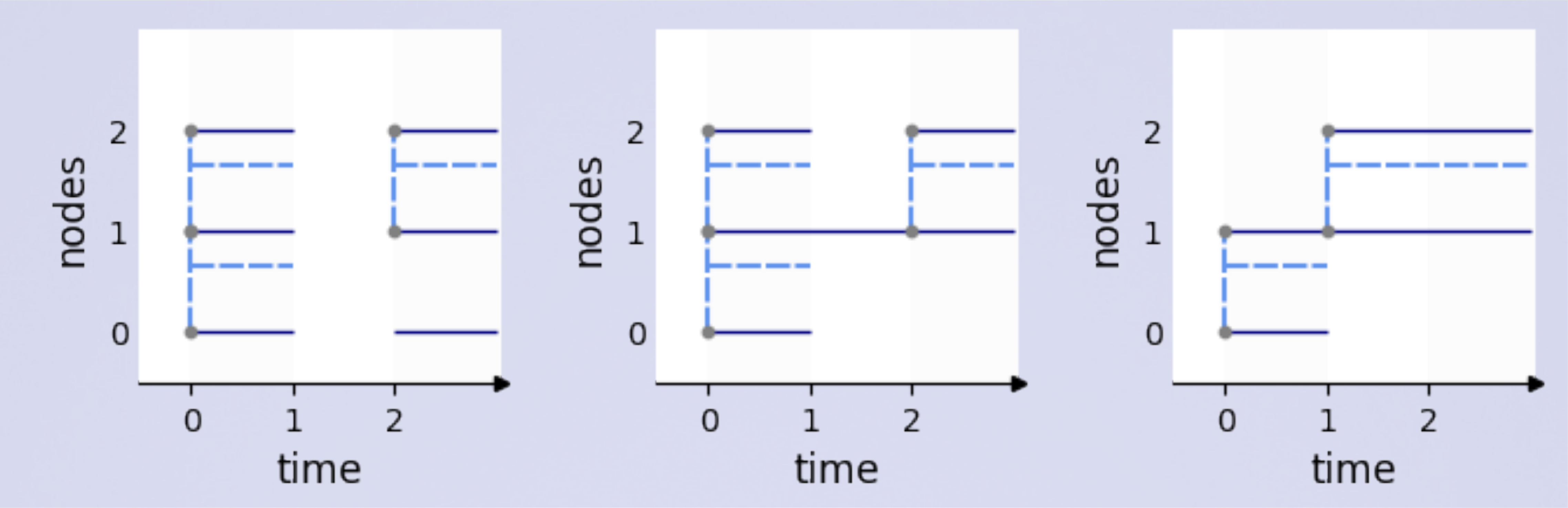
The density in static networks can be understood as the fraction of existing edges among all possible edges,

$$d = \frac{L}{L_{\max}}.$$

In the following, we will use L_{max}^3, as in Latapy et al.

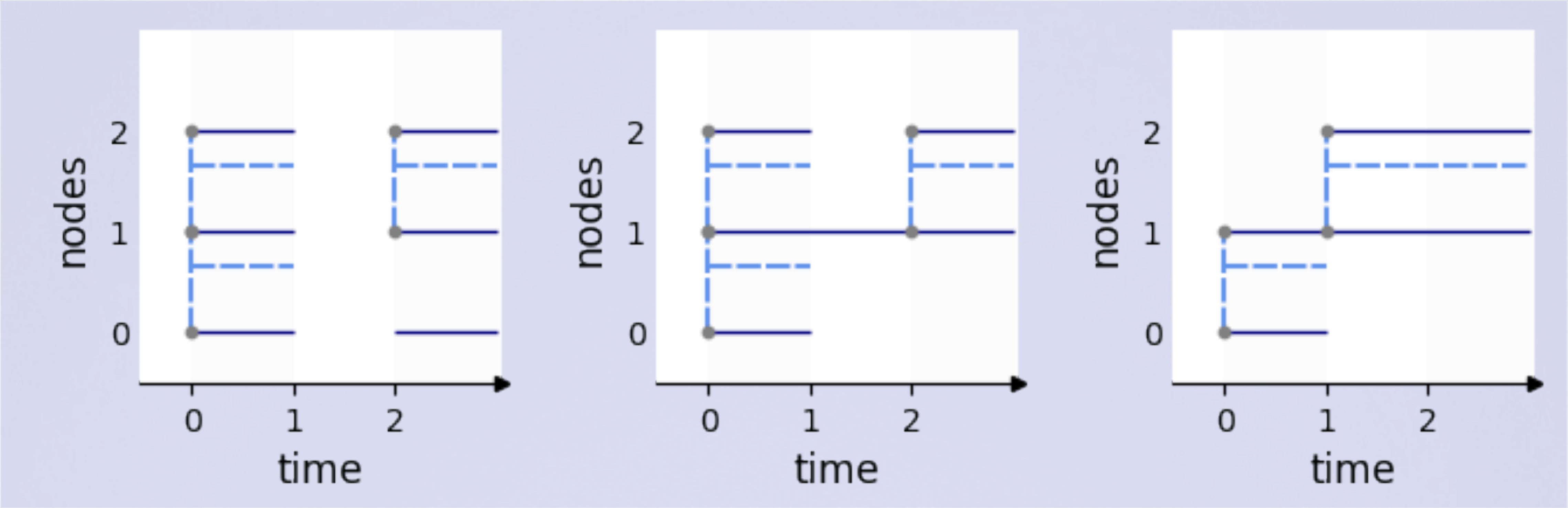
STREAM GRAPHS

$N = 2$ $L = 1$



STREAM GRAPHS

$N = 2$ $L = 1$

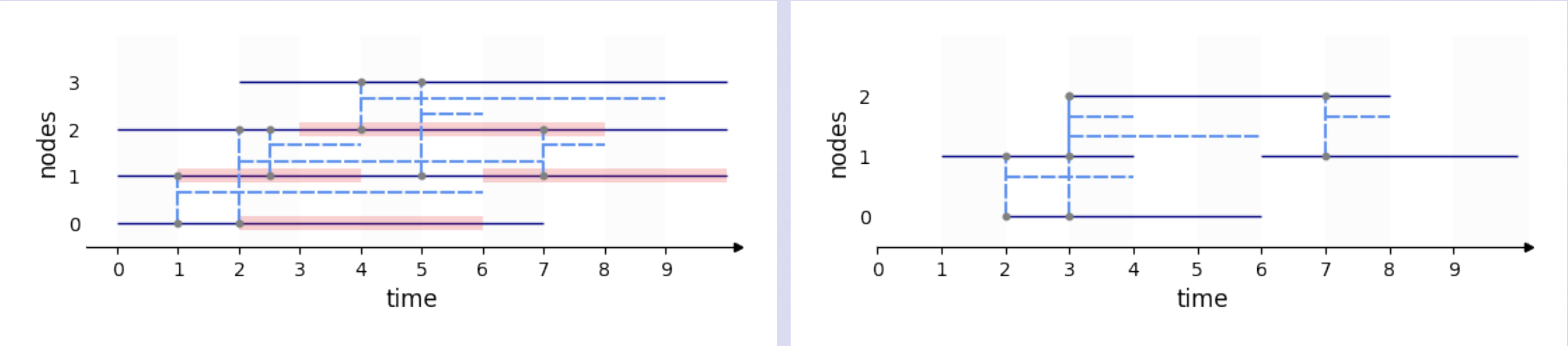


$$d = \frac{3}{6} = \frac{1}{2}$$ $$d = \frac{3}{4}$$ $$d = \frac{3}{3} = 1$$

STREAM GRAPHS

SG - Clusters & Substreams

In static networks, a cluster is a set of nodes, and we have defined an (induced) subgraph of this cluster as a graph composed of the nodes of the cluster and the edges existing between those nodes. In Stream Graphs, a clusters C is as subset of W, and the corresponding (induced) substream $S(C) = (T, V, C, E(C))$, with $E(C) = \{(t,(u,v)) \in E, (t,u),(t,v) \in C\}$.

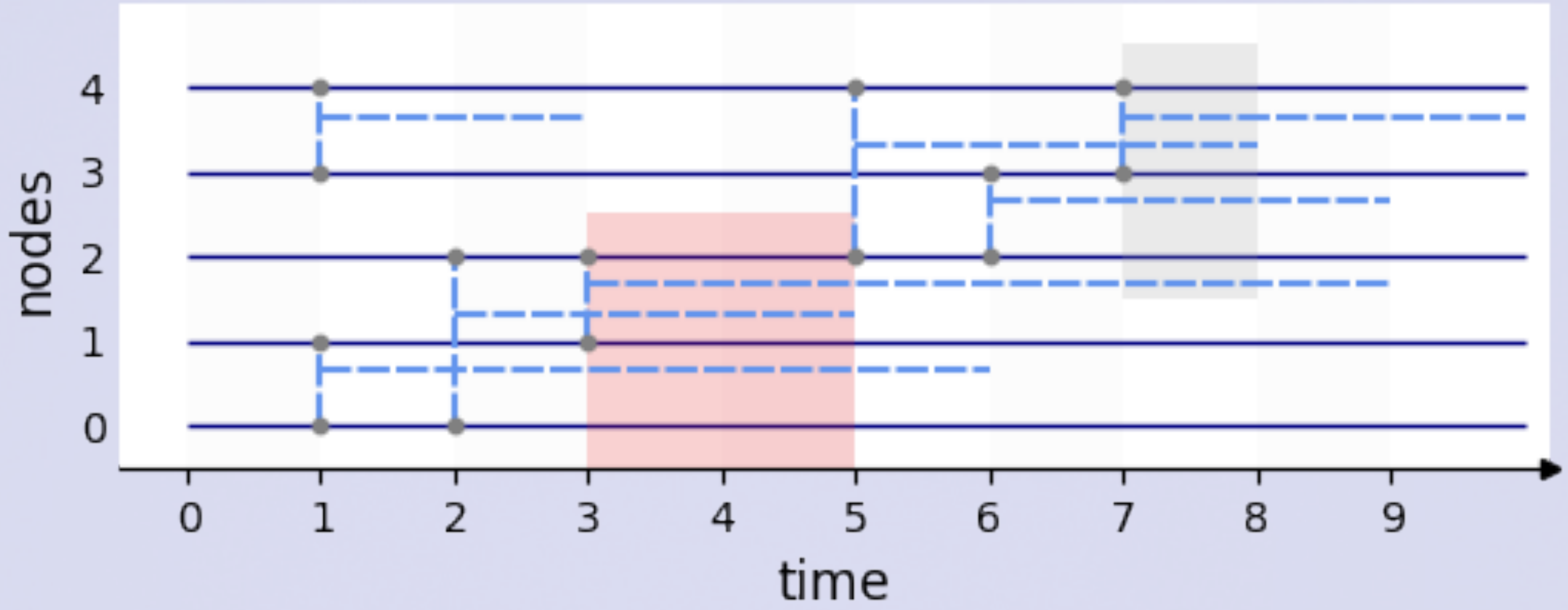


Example of subgraph (red,left) and induced substream (right).

STREAM GRAPHS

SG - Cliques

Having defined substreams and density, we can now naturally define a clique by analogy with static networks as a substream of density 1. A clique is said to be a **maximal clique** if it is not included in any other clique.



Red and Grey are the two maximal cliques of size three in this Stream Graph.

STREAM GRAPHS

SG - Neighborhood $N(u)$

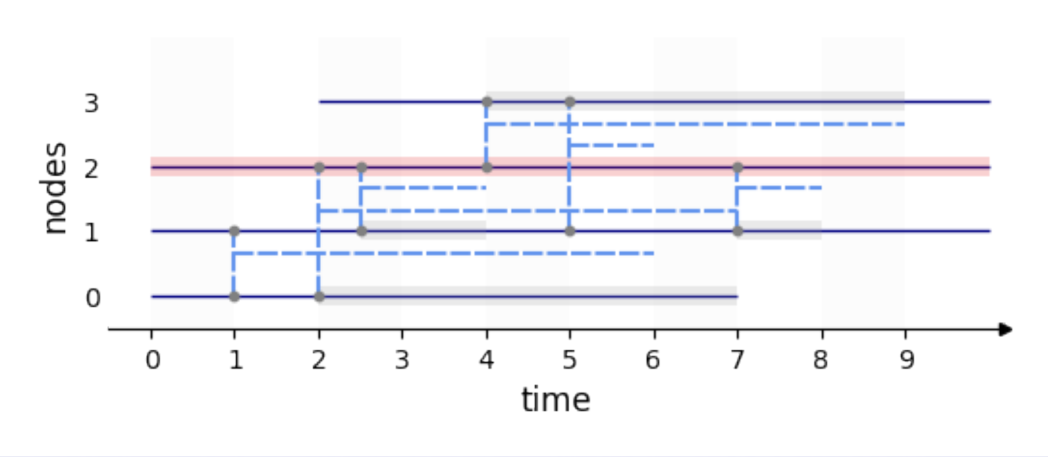
The neighborhood $N(u)$ of node u is defined as the cluster composed of node-times such as an edge-time exists between it and a node-time of u, i.e.,

$$N(u) = \{v_t, (u, v)_t \in E\}$$

SG - Degree $k(u)$

The degree $k(u)$ of node u is defined as the quantity of node in the Neighborhood of node u, i.e.

$$k(u) = |N(u)|$$



Example, the neighborhood of node 2 is highlighted in grey.
$k(c) = \frac{5+2.5+5}{10} = 1.25.$

STREAM GRAPHS

SG - Ego-network

The Ego network G_u of node u is defined as the substream induced by its neighborhood, i.e., $G_u = (T, V, N(u), E(N(u)))$.

SG - Clustering coefficient

The clustering coefficient $C(u)$ of node u is defined as the density of the ego-network of u, i.e.,

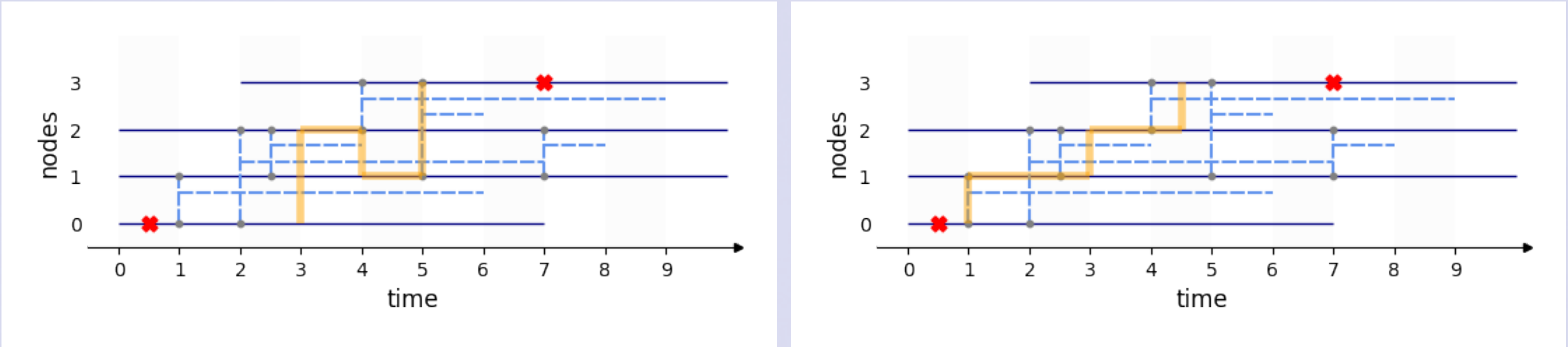
$$C(u) = d(N(u))$$

PATHS AND DISTANCES IN STREAM GRAPHS

PATHS

SG - Paths

In a Stream Graph S=(T,V,W,E), a **path** P from node-time x_α to node-time y_ω is a sequence $(t_0, x, v_0), (t_1, v_0, v_1), ..., (t_k, v_k, y)$ of elements of $T \times V \times V$ such that $t_0 \geq \alpha$,$t_k \leq \omega$, $((t_i, u_i, v_i)) \in E$. We say that P **starts at** t_0, **arrives at** t_k, has **length** $k+1$ and **duration** $t_k - t_0$.



Examples of two paths from (node 0, t=0.5) to (node 3, t=1). The left one starts at 3, arrives at 5, has length 3 and duration 2. The right one starts at 1, arrives at 4.5, has length 3 and duration 3.5.

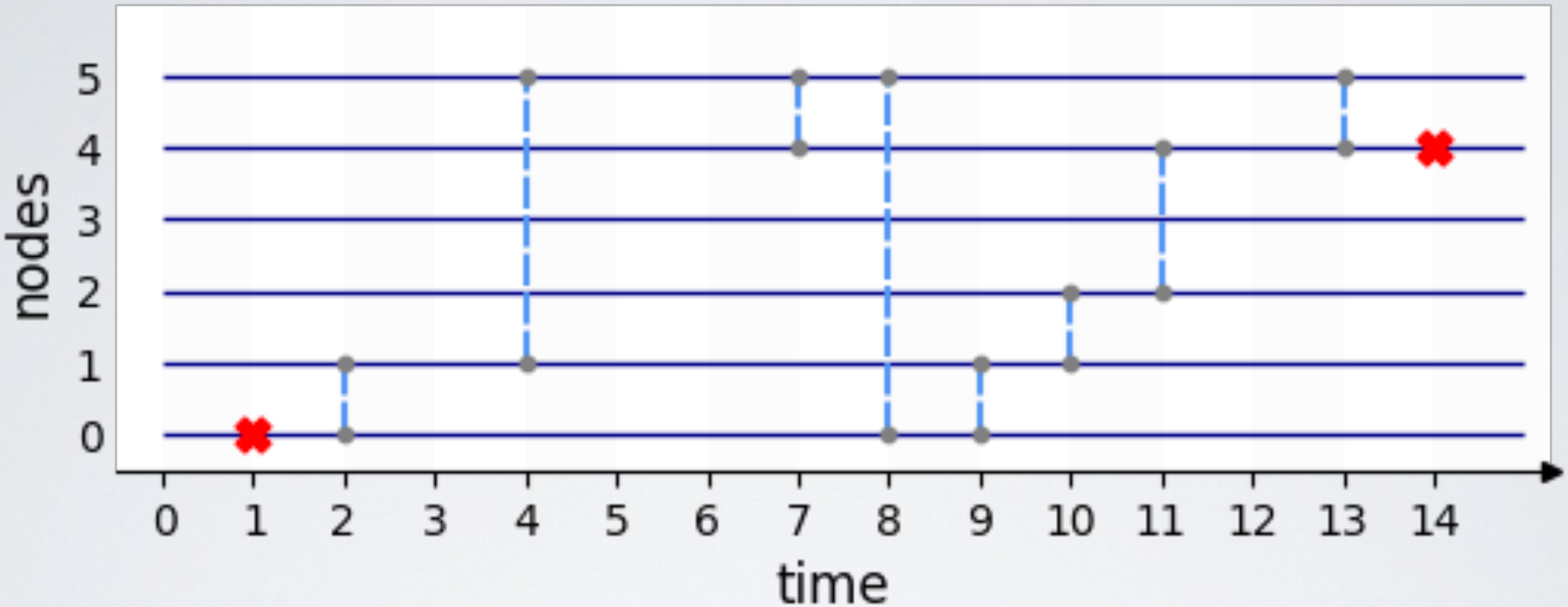
PATHS

SG - Shortest - Fastest - Foremost

- **Shortest Paths**, as in static networks, are paths of **minimal length**.
- **Fastest Paths** are paths of **minimal duration**.
- **Foremost Paths** are paths **arriving first**.

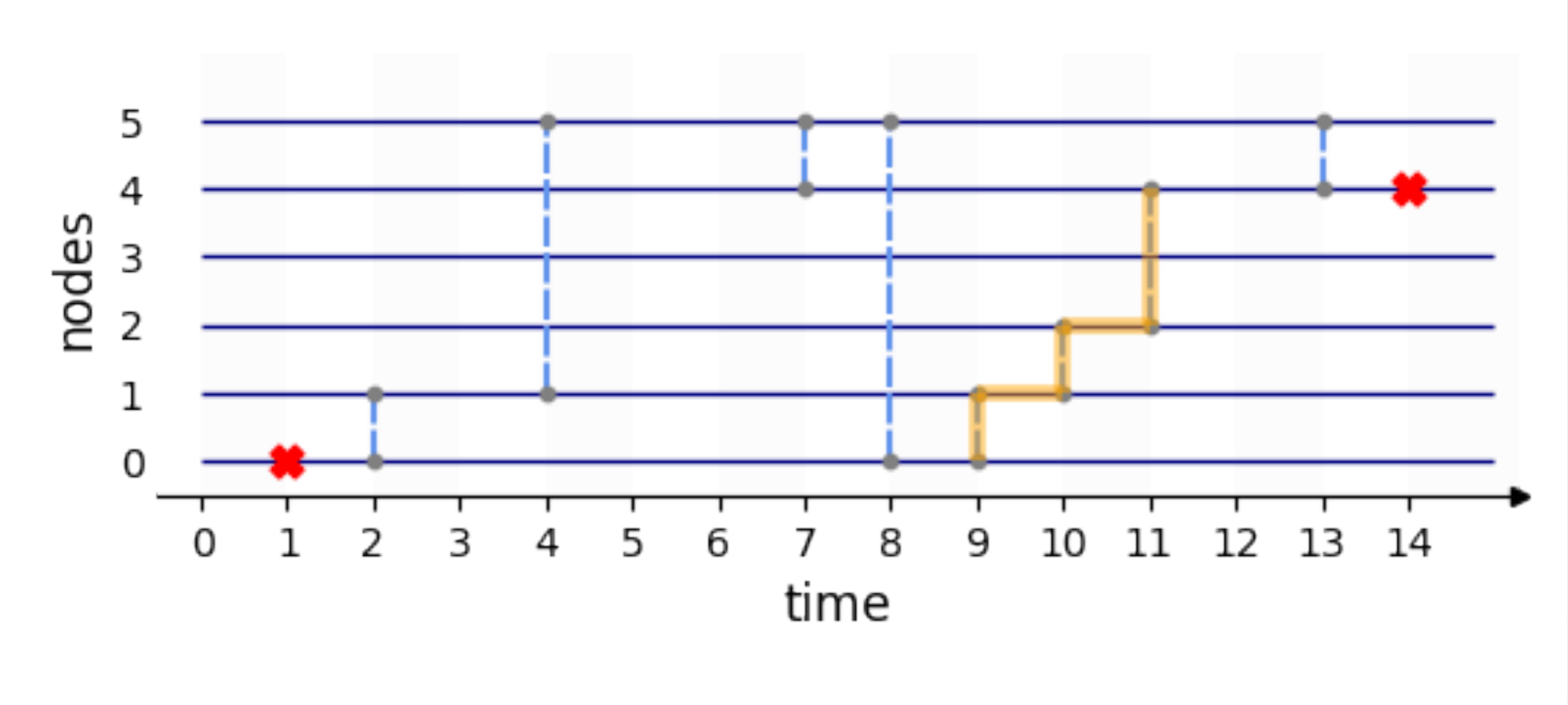
Furthermore, one can combine those properties, defining for instance:
Fastest shortest paths (paths of minimum duration among those of minimal length)
Shortest fastest paths (paths of minimal length among those of minimal duration)

PATHS



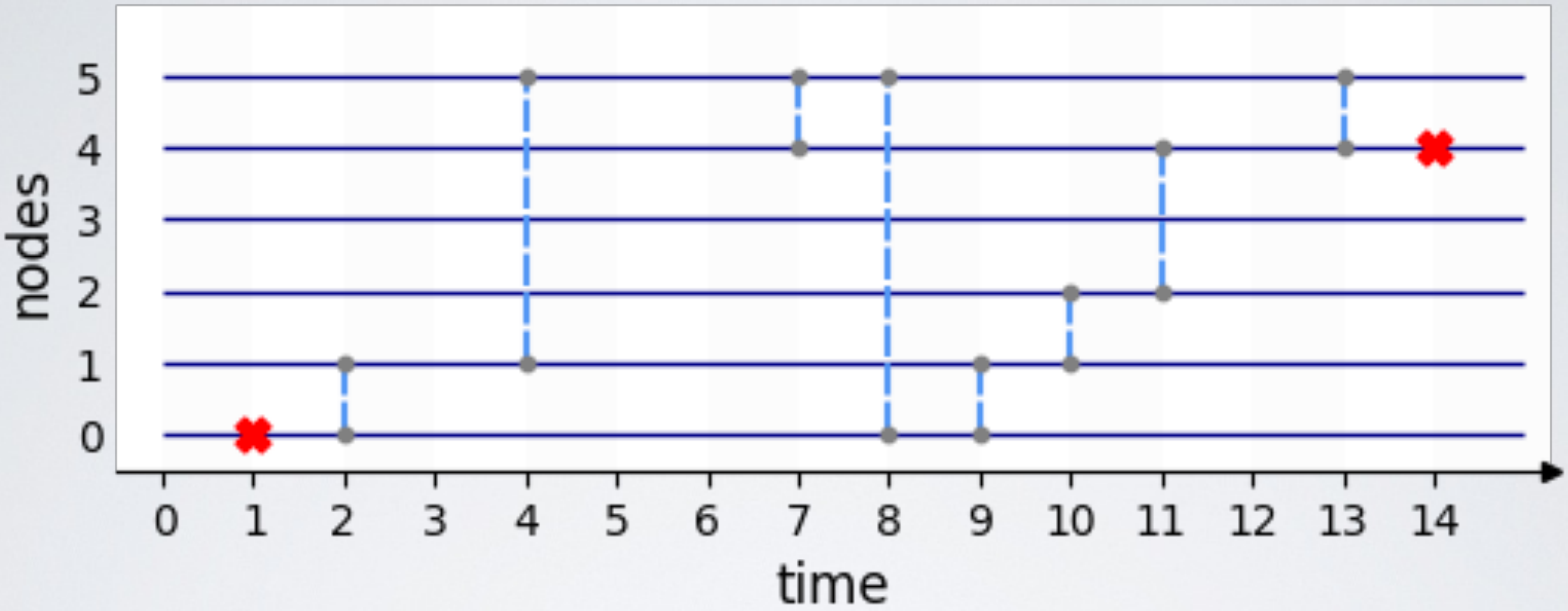
Fastest ?

PATHS



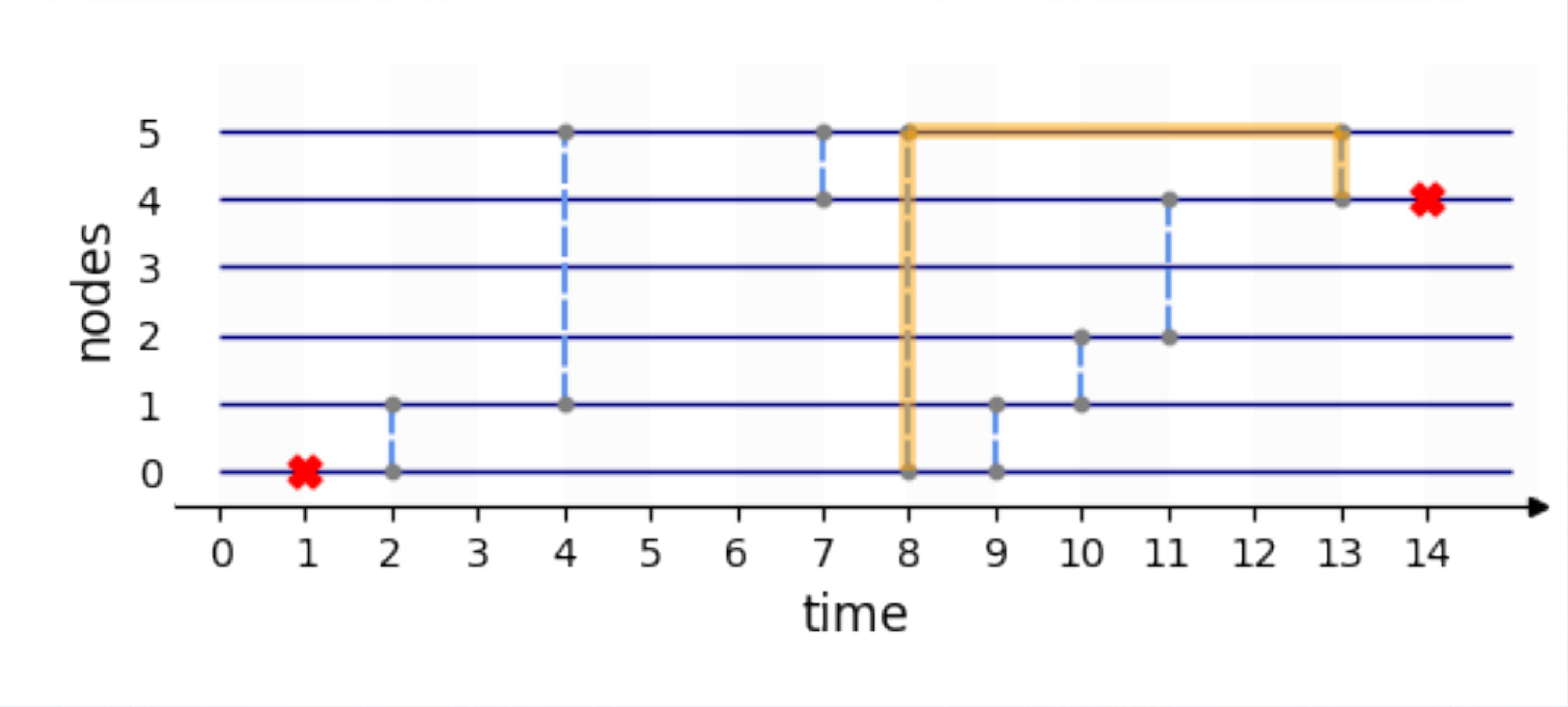
Fastest

PATHS



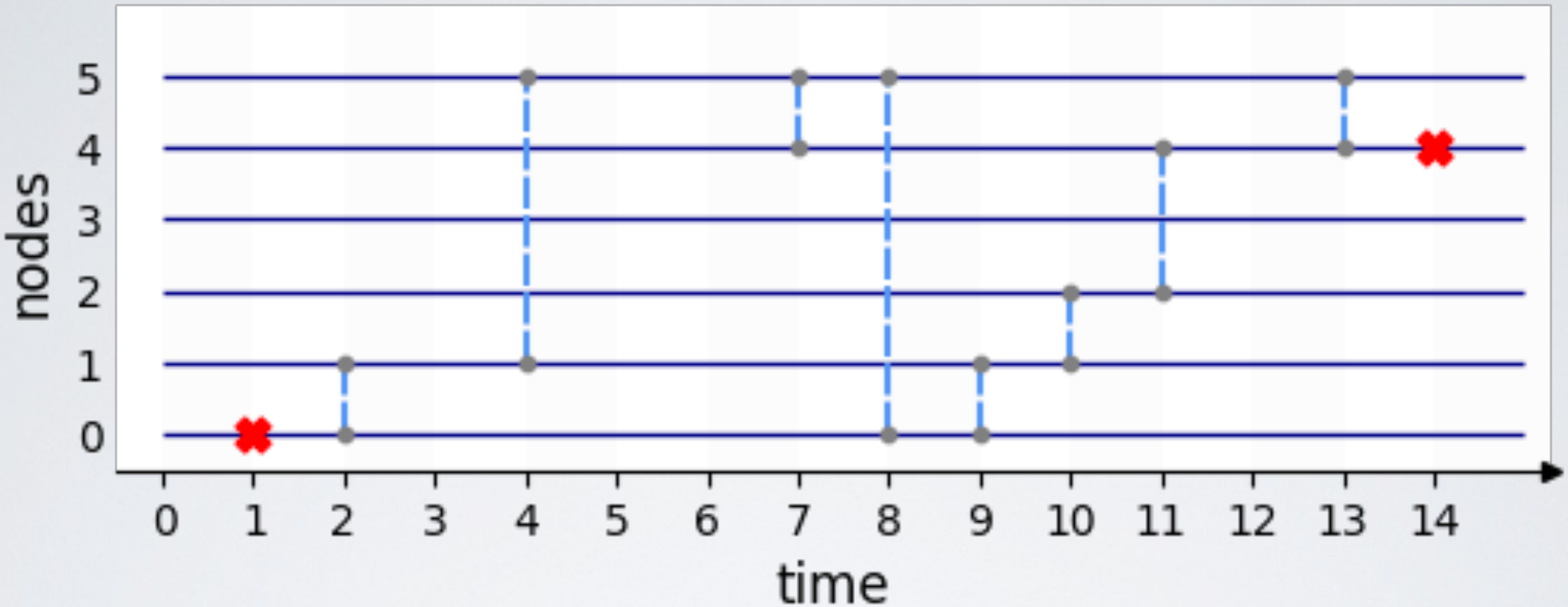
Shortest ?

PATHS



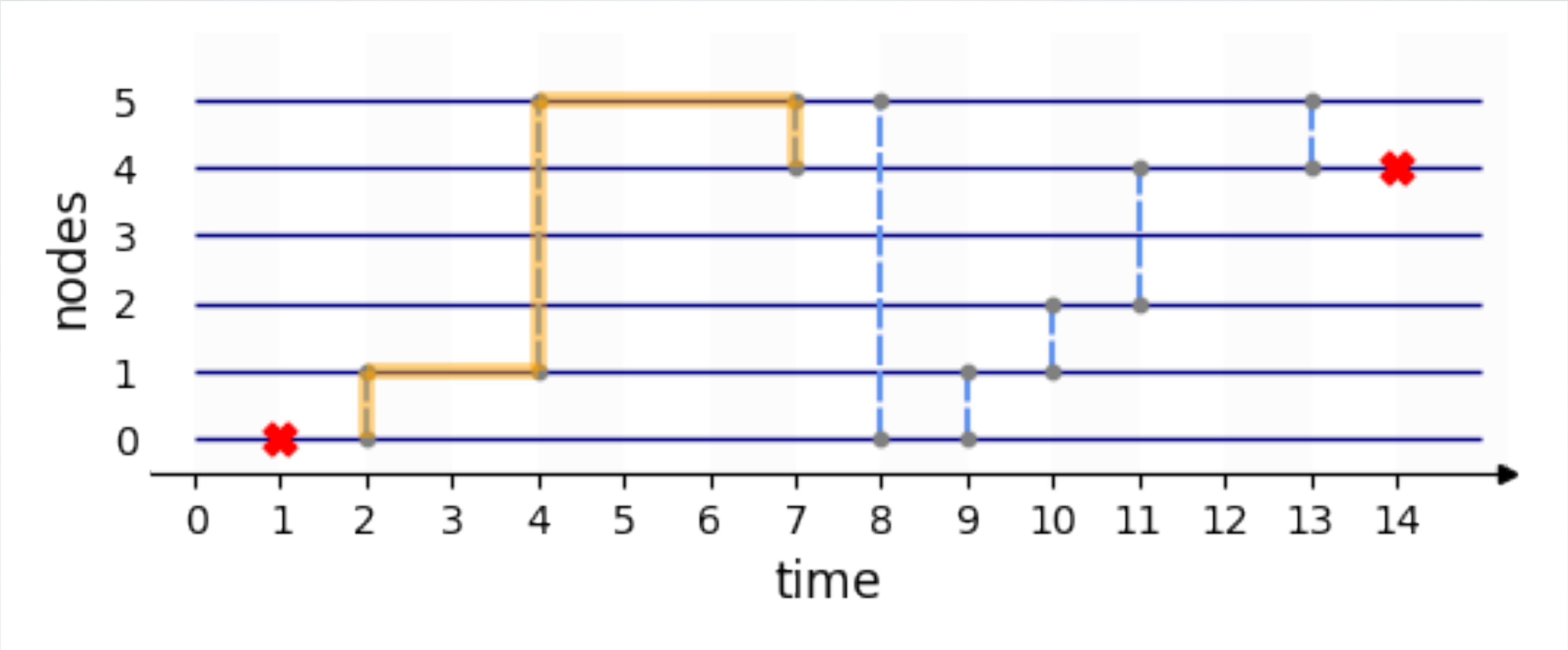
Shortest

PATHS



Foremost ?

PATHS

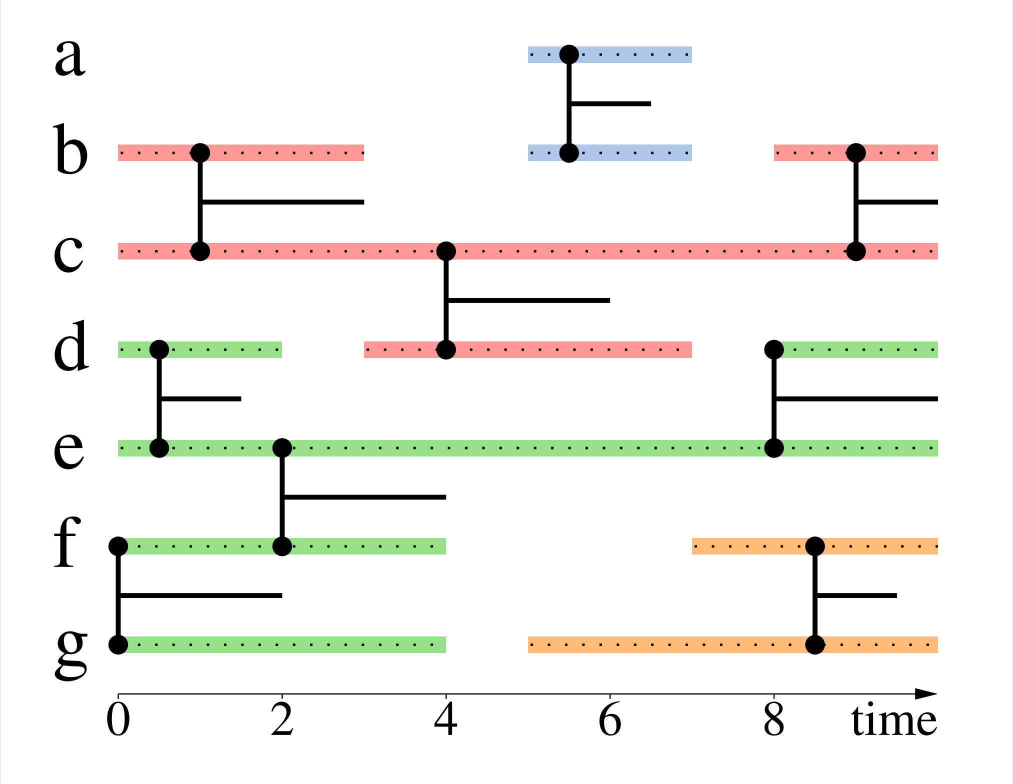


Foremost

PATHS

SG - Connected Components

Various definitions for connected components have been proposed for temporal networks, see (Latapy, Viard, and Magnien 2018) for details. One of the simplest one is the **weakly connected component**, defined such as two node-times belong to the same connected component if and only if there is a path from one to the other, *ignoring time*.



Example of a Stream Graph decomposed in 4 weakly connected components.

RANDOM MODELS FOR DYNAMIC NETWORKS

Laetitia Gauvin et al. “Randomized reference models for temporal networks”. In: *SIAM Review* 64.4 (Nov. 2022)

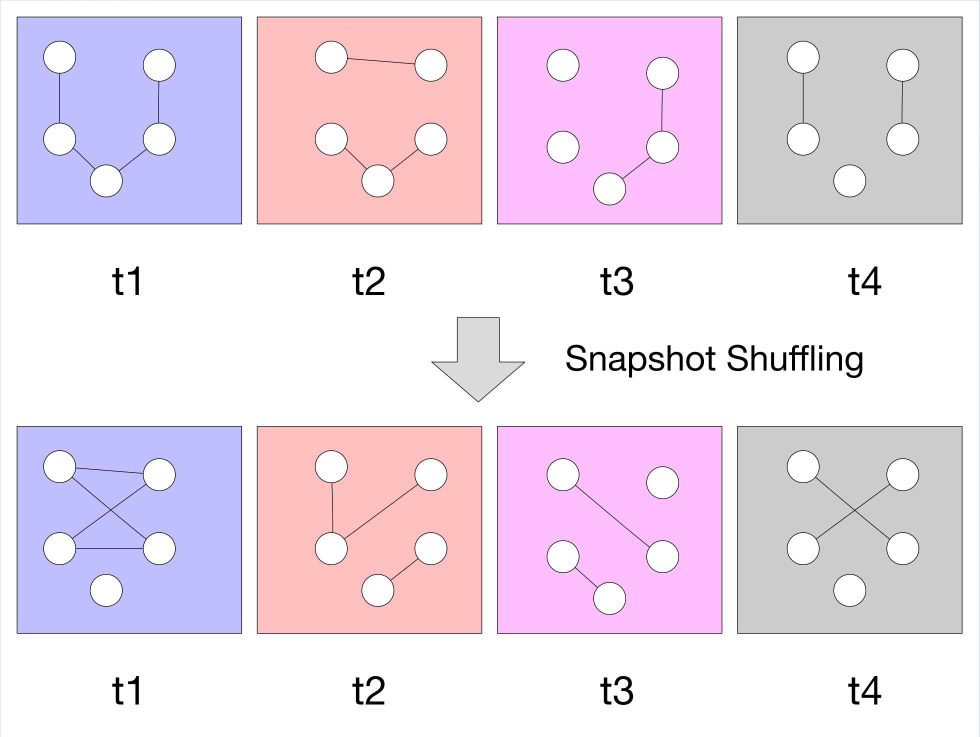
RANDOM MODELS

- In many cases, in network analysis, useful to compare a network to a randomized version of it
  - Clustering coefficient, assortativity, modularity, …
- In a static graph, 2 main choices:
  - Keep only the number of edges (ER model)
  - Keep the number of edges and the degree of nodes (Configuration model)
- In dynamic networks, it is more complex…

RANDOM MODELS

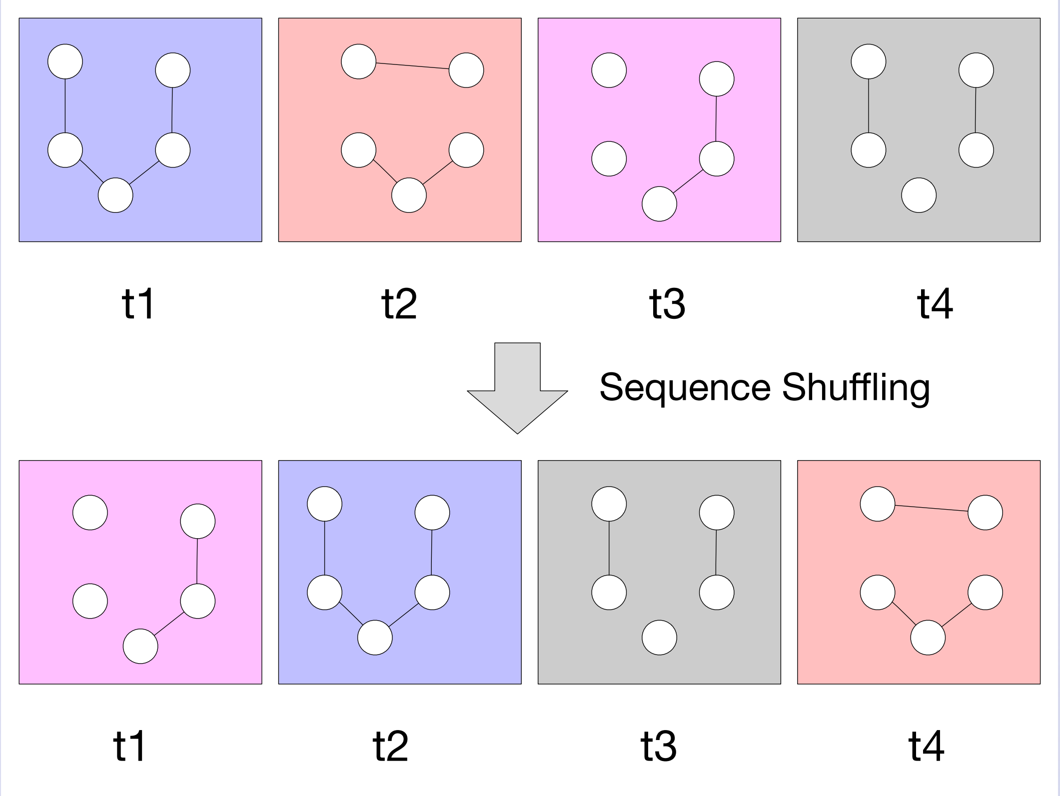
Snapshot Shuffling

Snapshot Shuffling keeps the order of snapshots, randomize edges inside snapshots. Any random model for static network can be used, such as ER random graphs or a degree preserving randomization.



Sequence Shuffling

Sequence Shuffling keeps each snapshot identical, switch randomly their order.



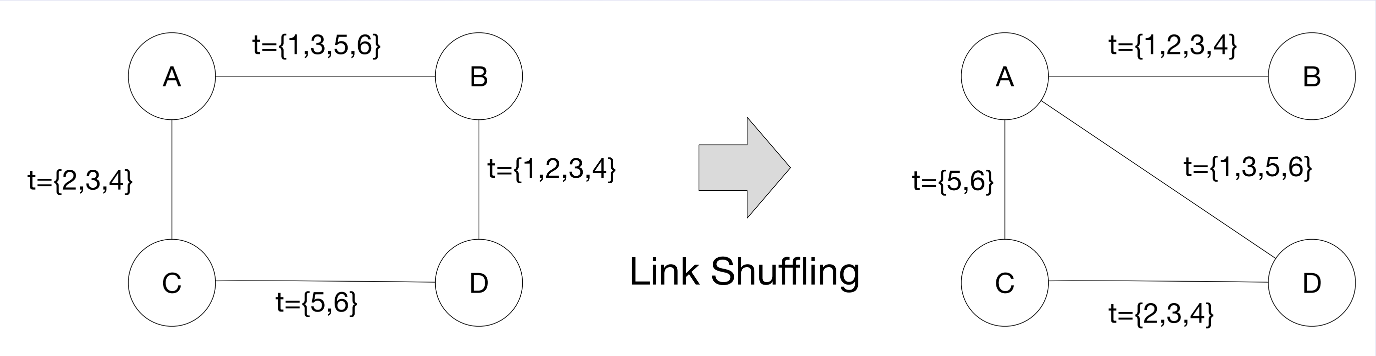
Properties preserved?

Properties lost?

RANDOM MODELS

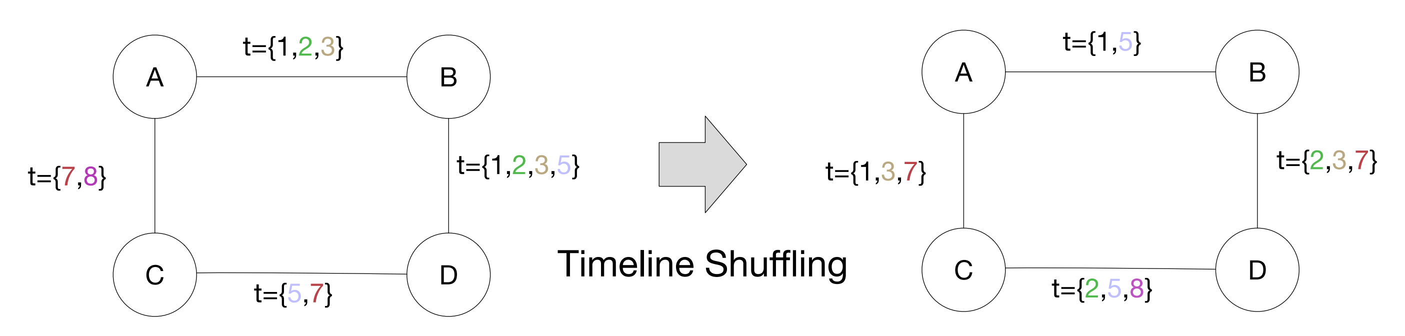
Link Shuffling

Link Shuffling keeps activation time per node pairs, randomize the aggregated graph. For instance, a simple way to achieve this is to pick two node pairs at random (connected or not) of the aggregated graph, and to exchange activation time of these node pairs, e.g.:



Timeline Shuffling

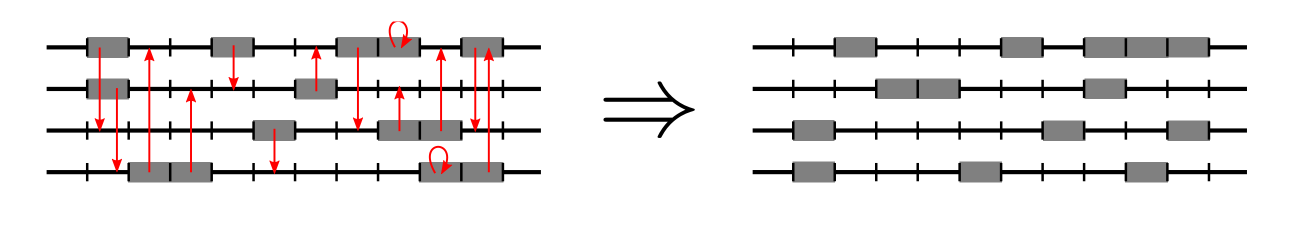
Timeline Shuffling keeps the aggregated graph, randomize edges activation time. For instance, a simple way to achieve this is to redistribute randomly activation period among all edges, e.g.:

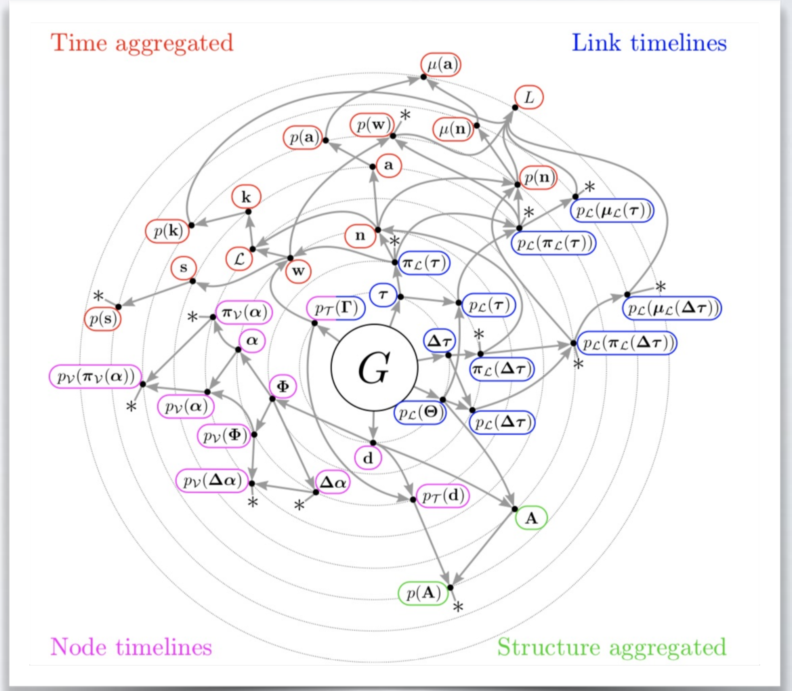


RANDOM MODELS

More constrained Shuffling

Variants of these shufflings with additional constraints have been proposed, for instance the **Local timeline shuffling**, randomizing events time edge by edge, or the **Weight constrained timeline shuffling**, randomizing events while conserving the number of observations for each edge. See (Gauvin et al. 2018) for details.





Laetitia Gauvin et al. "Randomized reference models for temporal networks". In: *SIAM Review* 64.4 (Nov. 2022)

DYNAMIC COMMUNITY DETECTION

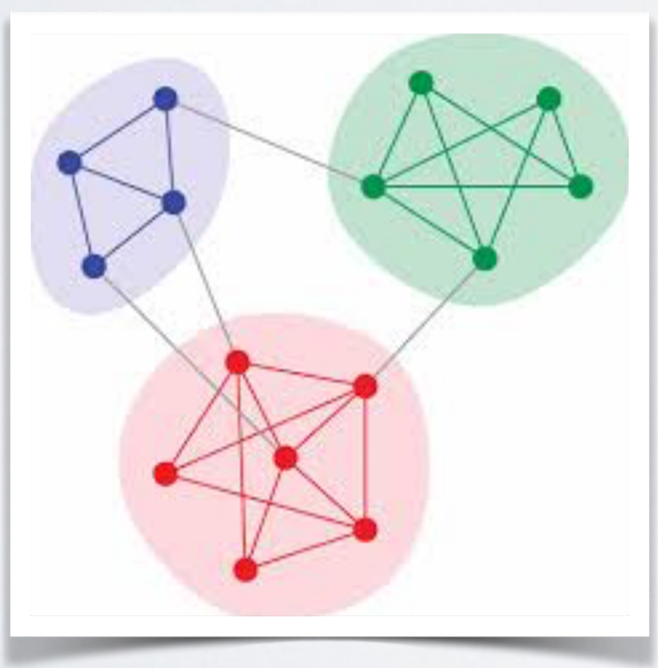
Rossetti, G., & Cazabet, R. (2018). Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)*, *51*(2), 1-37.

Cazabet, R., Boudebza, S., & Rossetti, G. (2020). Evaluating community detection algorithms for progressively evolving graphs. *Journal Of Complex Networks*

COMMUNITY DETECTION

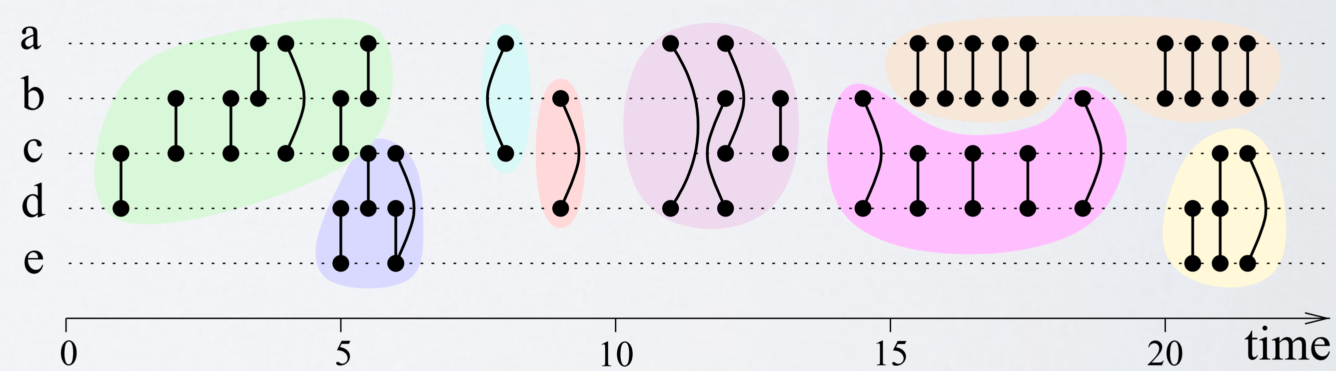
Static networks

Clusters: Sets of nodes



Dynamic Networks

Clusters: Sets of time-nodes, i.e., pairs (node,time)

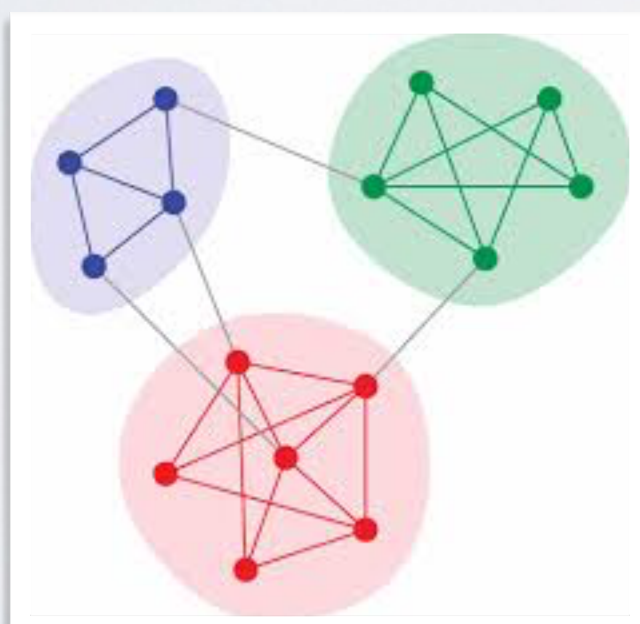


Gaumont, N., Viard, T., Fournier-S'Niehotta, R., Wang, Q., & Latapy, M. (2016). Analysis of the temporal and structural features of threads in a mailing-list. In *Complex Networks VII*

COMMUNITY DETECTION

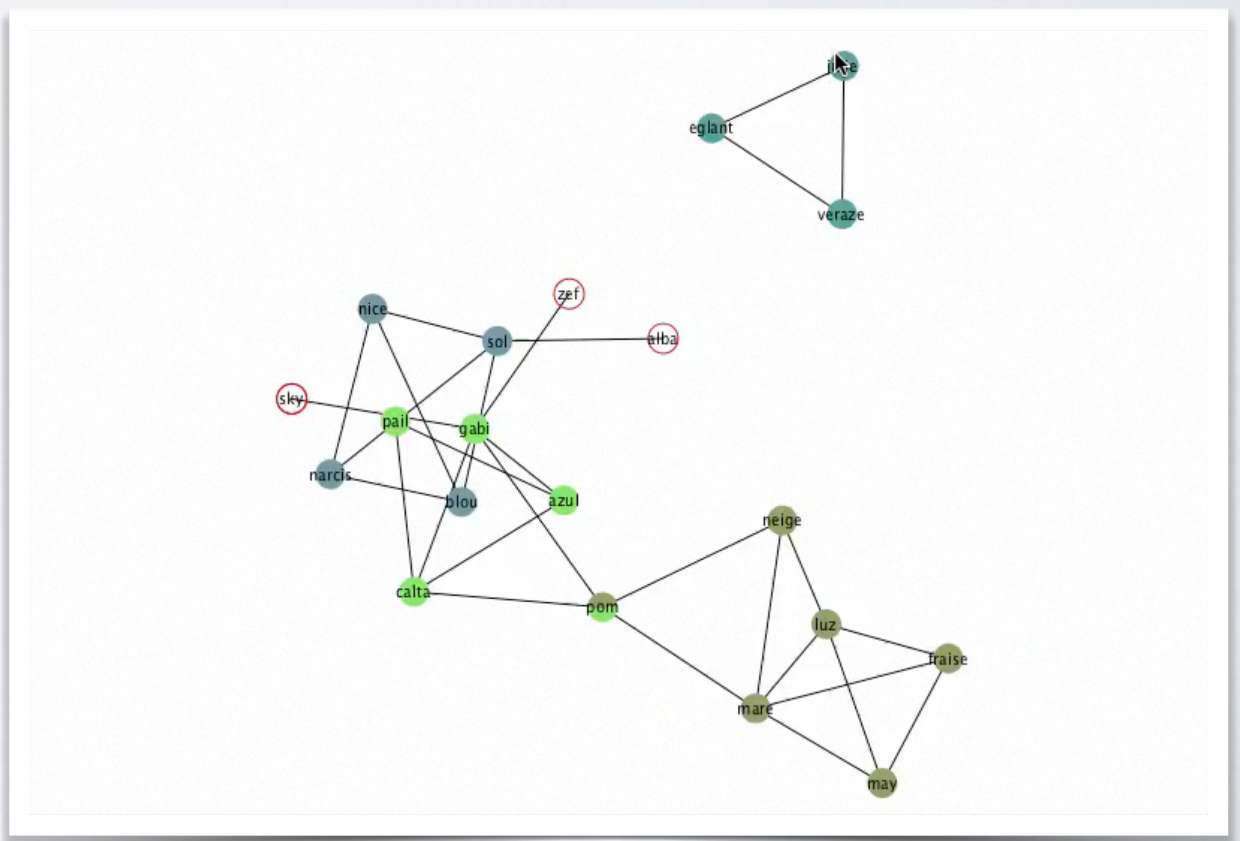
Static networks

Clusters: Sets of nodes



Dynamic Networks

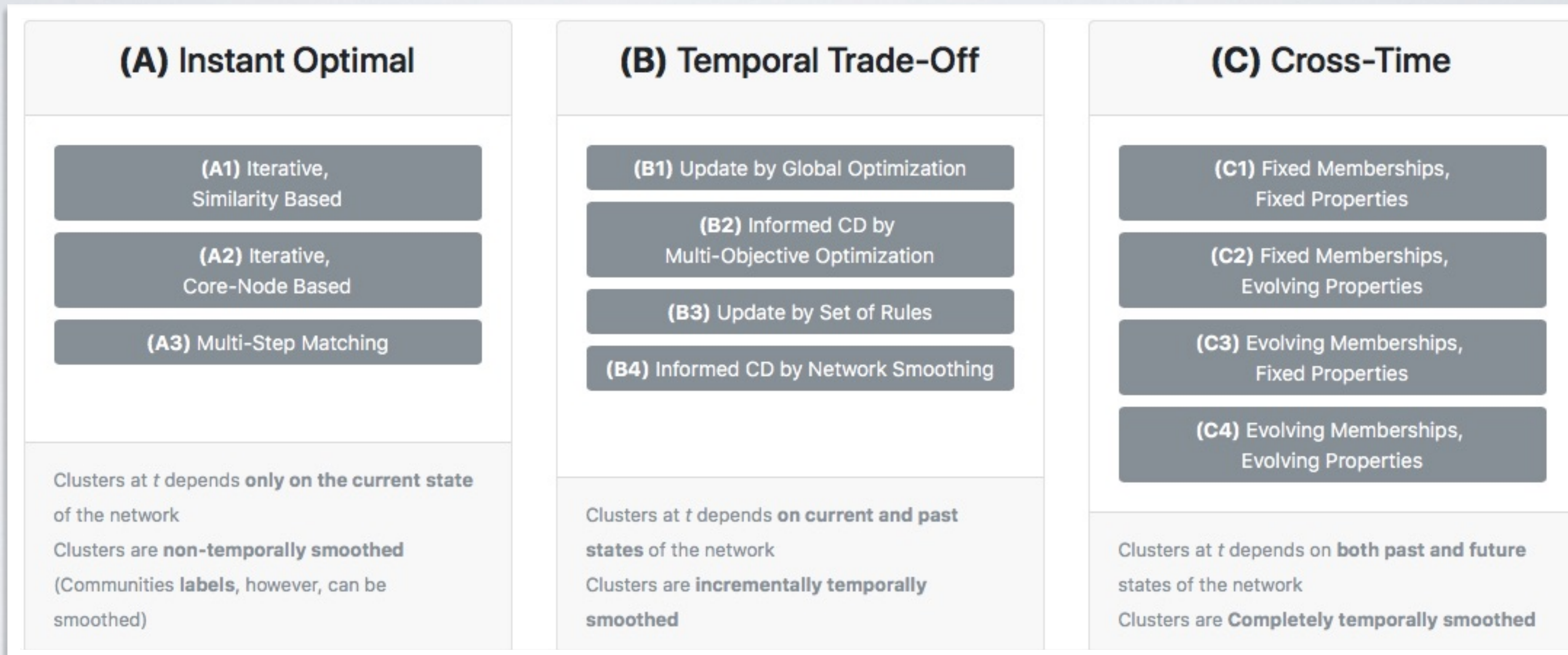
Clusters: Sets of time-nodes, i.e., pairs (node,time)



APPROACHES TO DCD

DYNAMIC COMMUNITIES ?

More than 50 methods published, broad categories



Rossetti, G., & Cazabet, R. (2018). Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)*, *51*(2), 1-37.

CATEGORIES

- Instant optimal:
  - Allows reusing static algorithms
  - No partition smoothing
  - Labels can be smoothed
  - Simple to parallelize

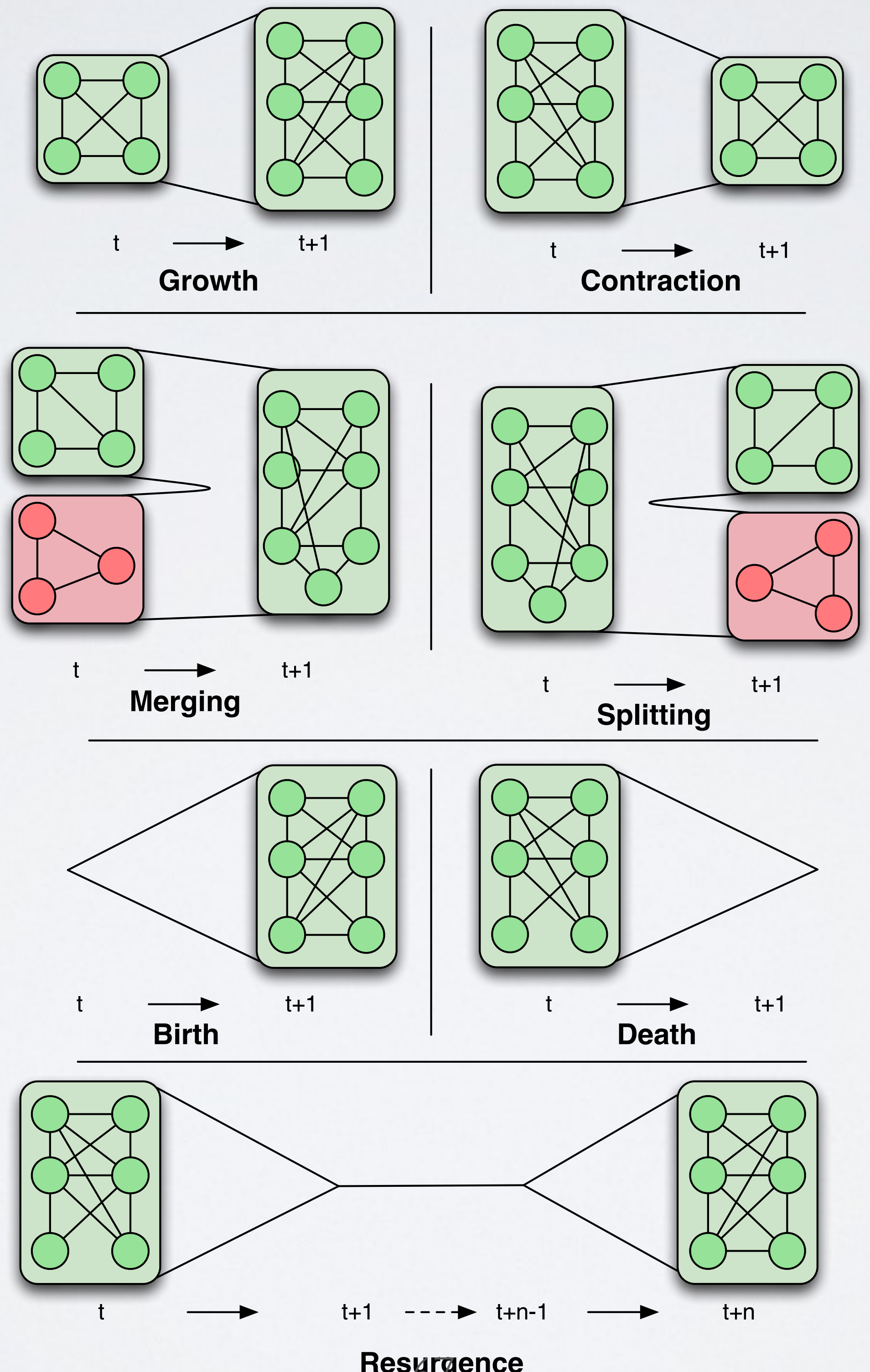
CATEGORIES

- Temporal trade-off
  - Cannot be parallelized (iterative)
  - => Best suited for real-time analysis / tasks
- Cross-Time
  - Requires to know the whole evolution in advance
  - => Not suited for real-time analysis, potentially the best smoothed (a posteriori interpretation)

WHAT MAKES DCD INTERESTING

NARRATIVES ?

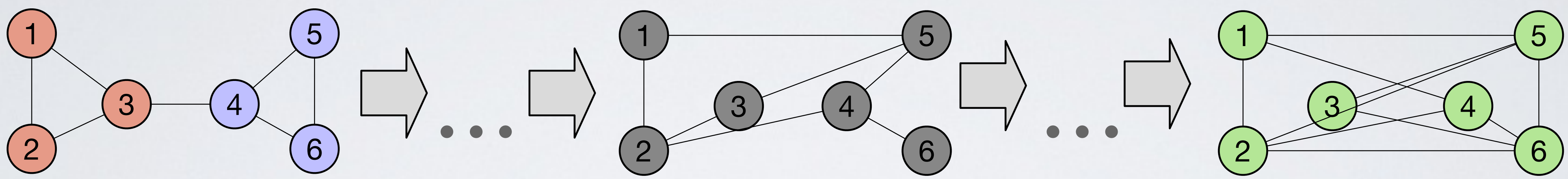
COMMUNITY EVENTS



SMOOTHNESS / STABILITY

- No Smoothness: Partition at **t** should be the same as found by a static algorithm.
- Smoothness: Partition at **t** is a trade-off between “good” communities for the graph at **t** and similarity with partitions at different times
  - Good story, Occam’s razor…

PROGRESSIVE EVOLUTION



2 communities

??

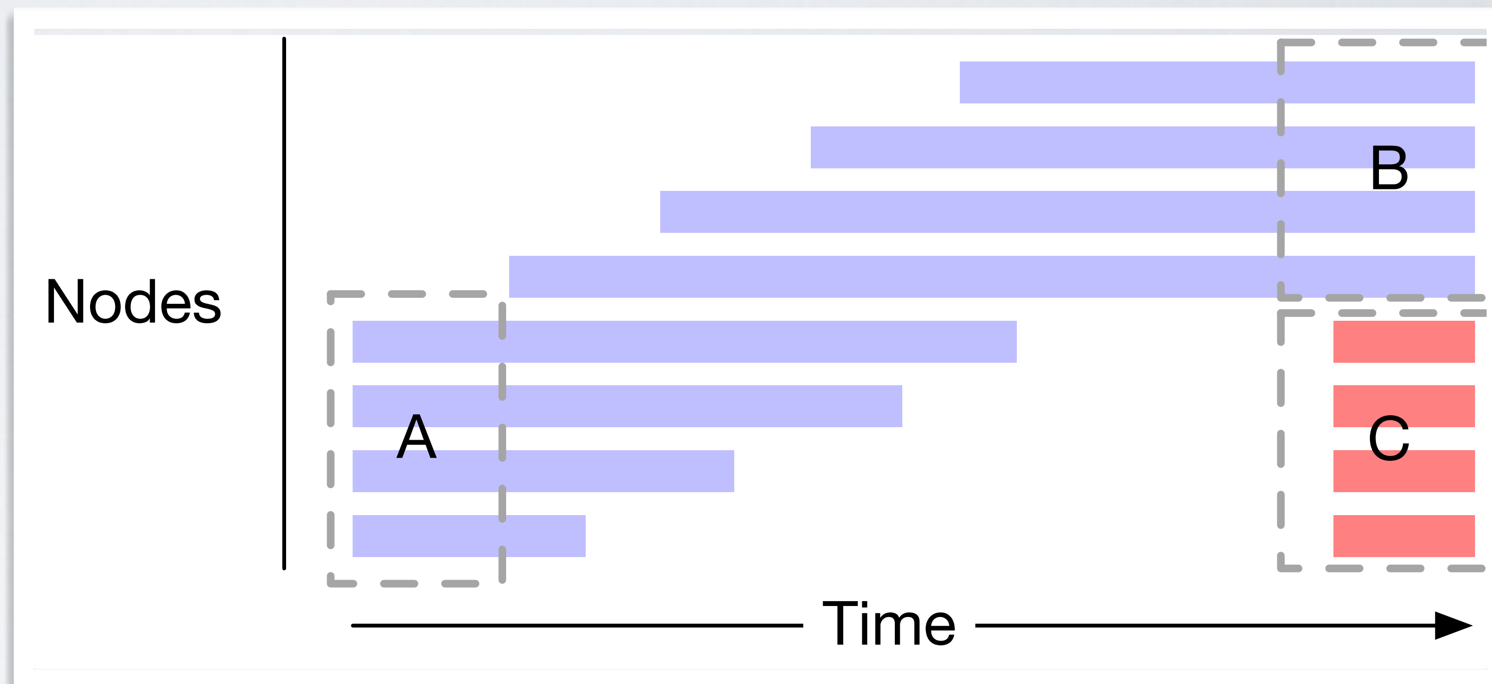
Intermediate state

1 community

How to *track* communities, giving a *coherent* dynamic structure ?

IDENTITY PRESERVATION

Ship of Theseus [Plutarch., 75]



2 problems:

1)Find node clusters at each t

2)Assign labels between same communities at ≠ t

Cazabet, R., & Rossetti, G. (2019). Challenges in community discovery on temporal networks. In *Temporal Network Theory* (pp. 181-197). Springer, Cham.

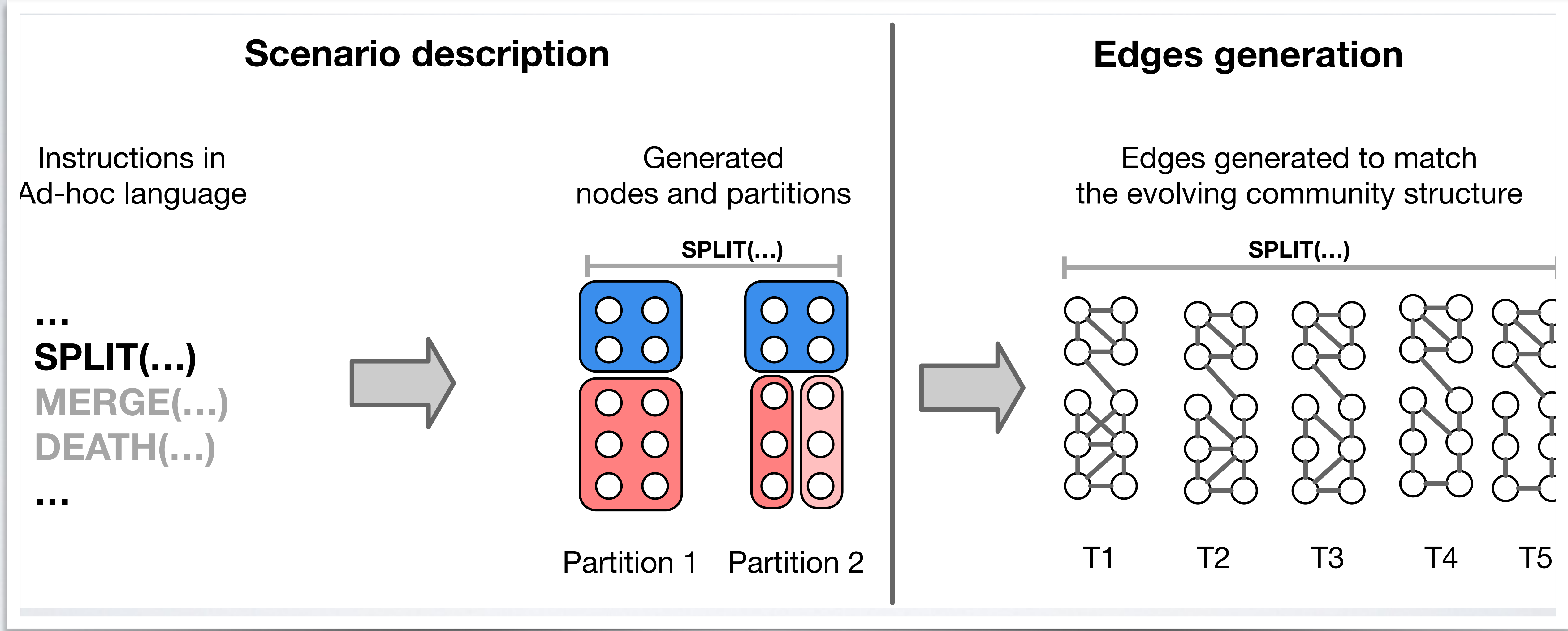
EMPIRICAL EVALUATION

Cazabet, R., Boudebza, S., & Rossetti, G. (2020). Evaluating community detection algorithms for progressively evolving graphs. *Journal Of Complex Networks*

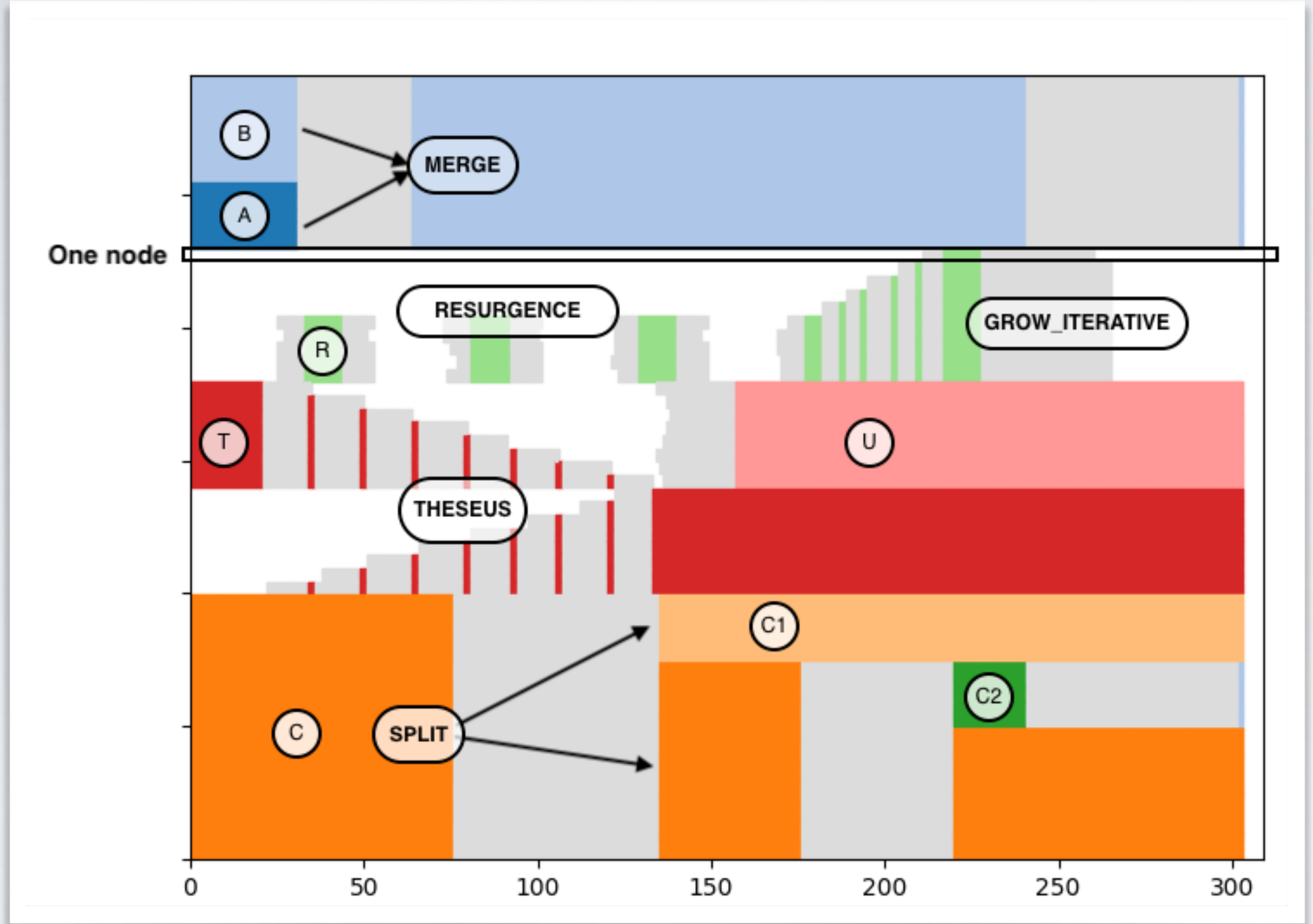
SETTING

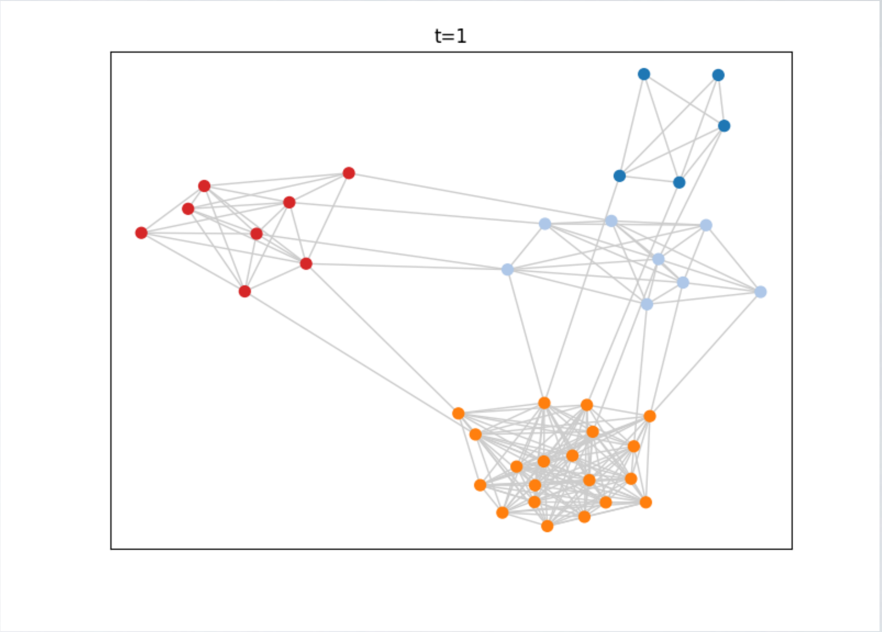
- Choose methods based on the same definition of a static community: Modularity (most widespread), but different approaches to dynamics
- Generate dynamic networks with planted dynamic community structure

SETTING

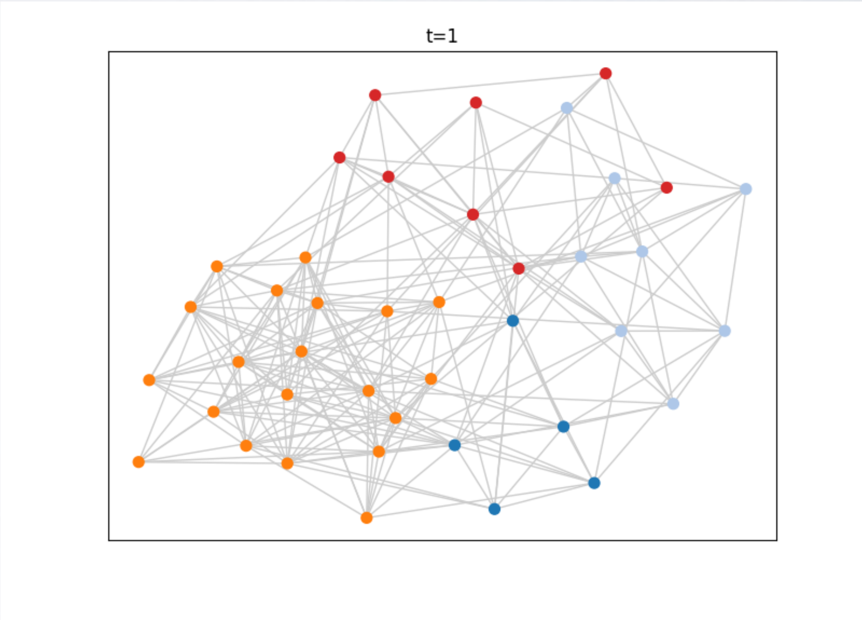


Scenario description
Edges generation
Instructions in
Ad-hoc language
...
SPLIT(...)
MERGE(...)
DEATH(...)
...
Generated
nodes and partitions
SPLIT(...)
Partition 1
Partition 2
Edges generated to match
the evolving community structure
SPLIT(...)
T1
T2
T3
T4
T5





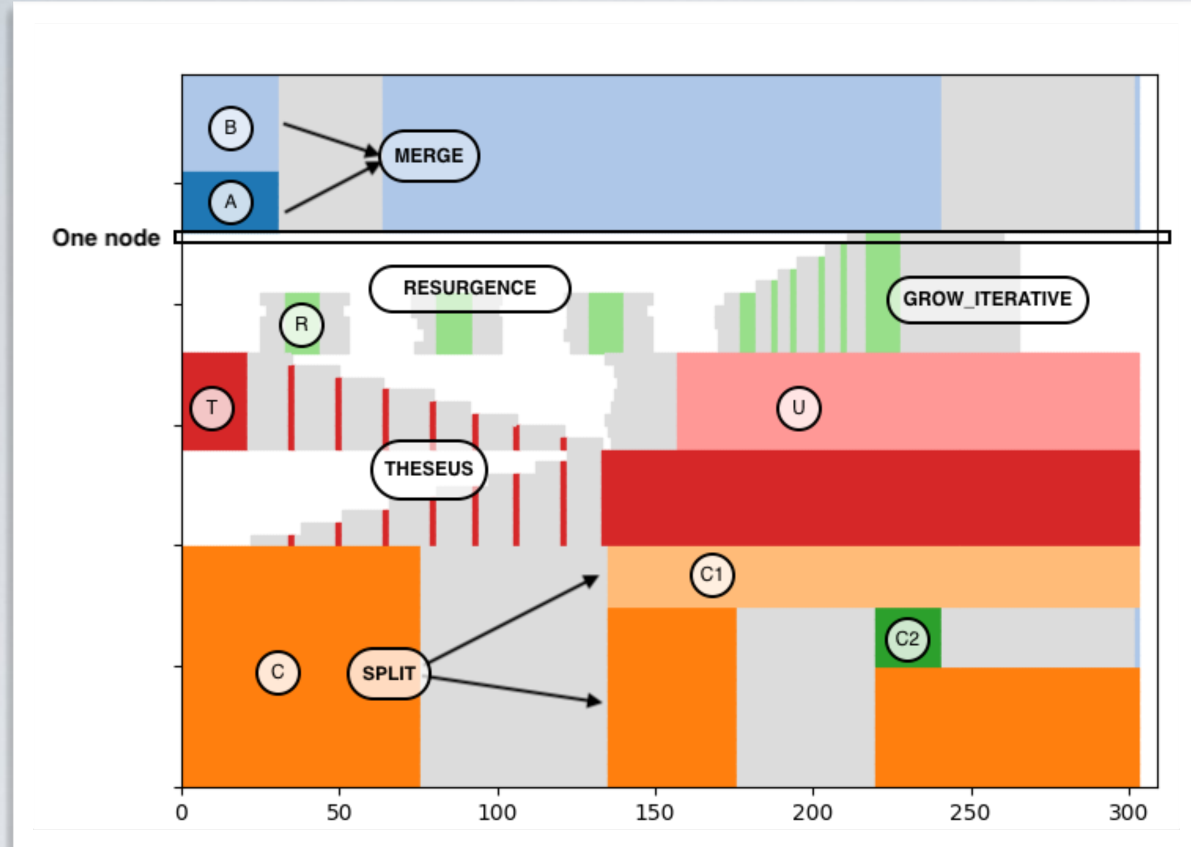
(b) The static graph at time t=0, version *sharp*
($\alpha = 0.9, \beta = 0.05, \beta_r = 0.01$)



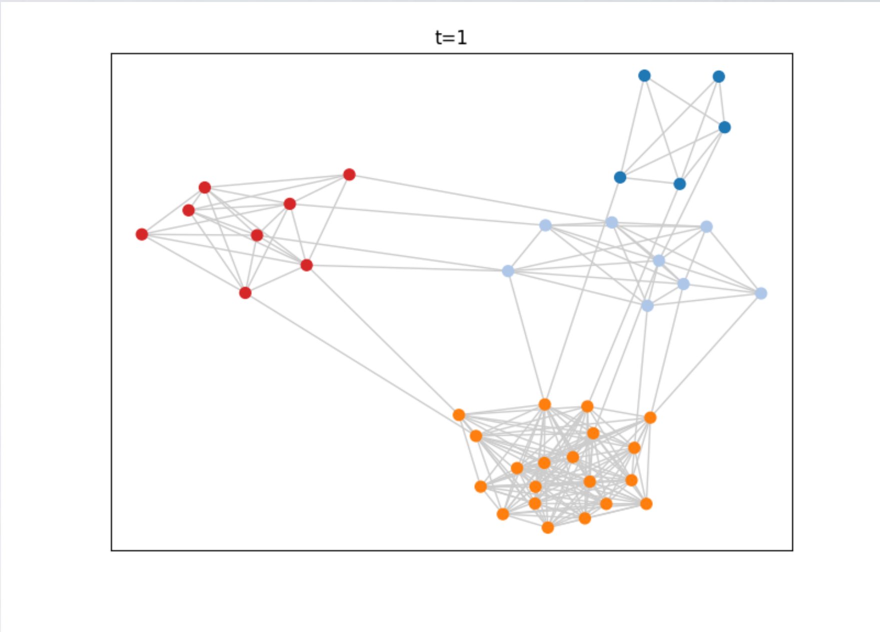
(c) The static graph at time t=0, version *blurred*
($\alpha = 0.8, \beta = 0.25, \beta_r = 0.01$)

METHODS

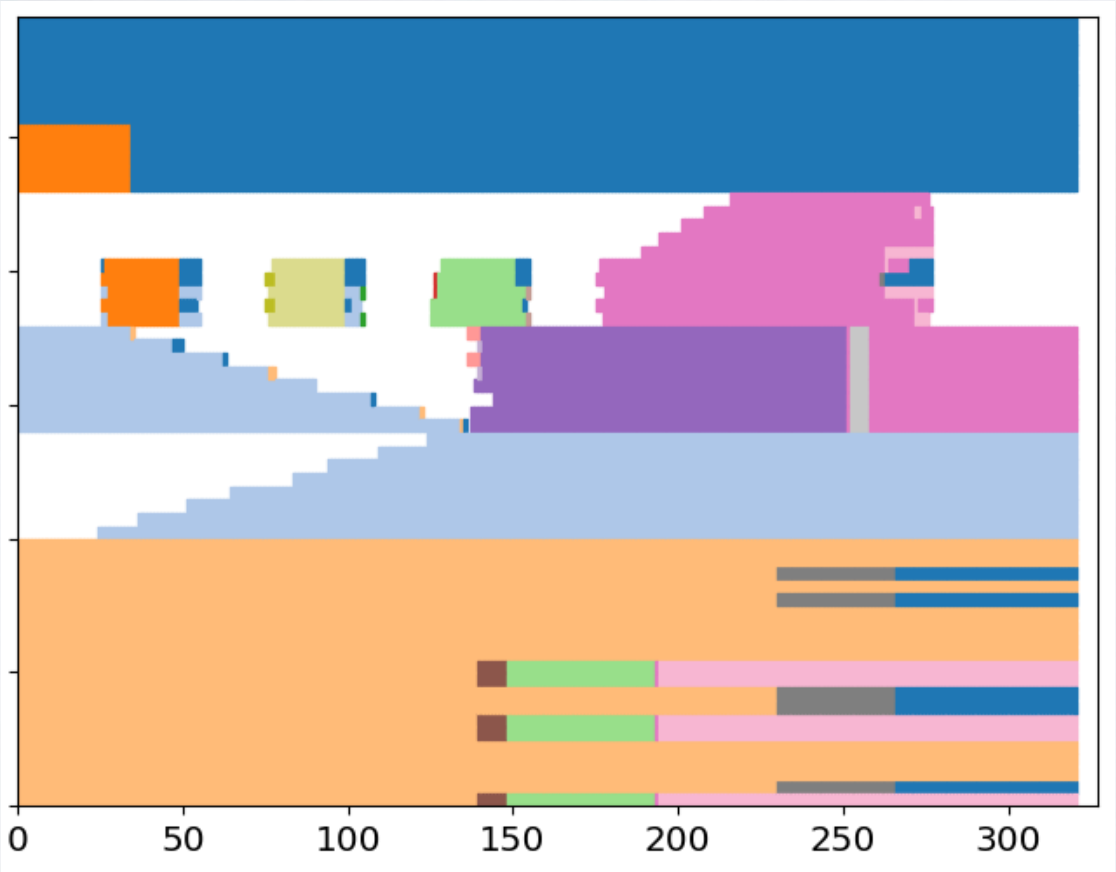
- Instant Optimal
  - No smoothing
    - Louvain at each step, match with Jaccard
- Temporal trade-off
  - Implicit Global
    - Louvain at each step in initialized by the previous partition (same local maximum), +Jaccard
  - DYNAMO
    - Update partition only based on edge changes to keep modularity high
  - Smoothed-graph
    - Each snapshot is modified to artificially raise the probability to obtain similar partition as previous step, then Louvain+Jaccard
- Cross-Time
  - Transversal Network
    - Create a single graph by adding edges between same nodes in successive snapshots (Mucha et al.), then (modified) Modularity optimization
  - Label-Smoothing
    - Create a "Community Survival graphs": nodes are static communities(Louvain), edges weighted by Jaccard Similarity. Apply Louvain on it.



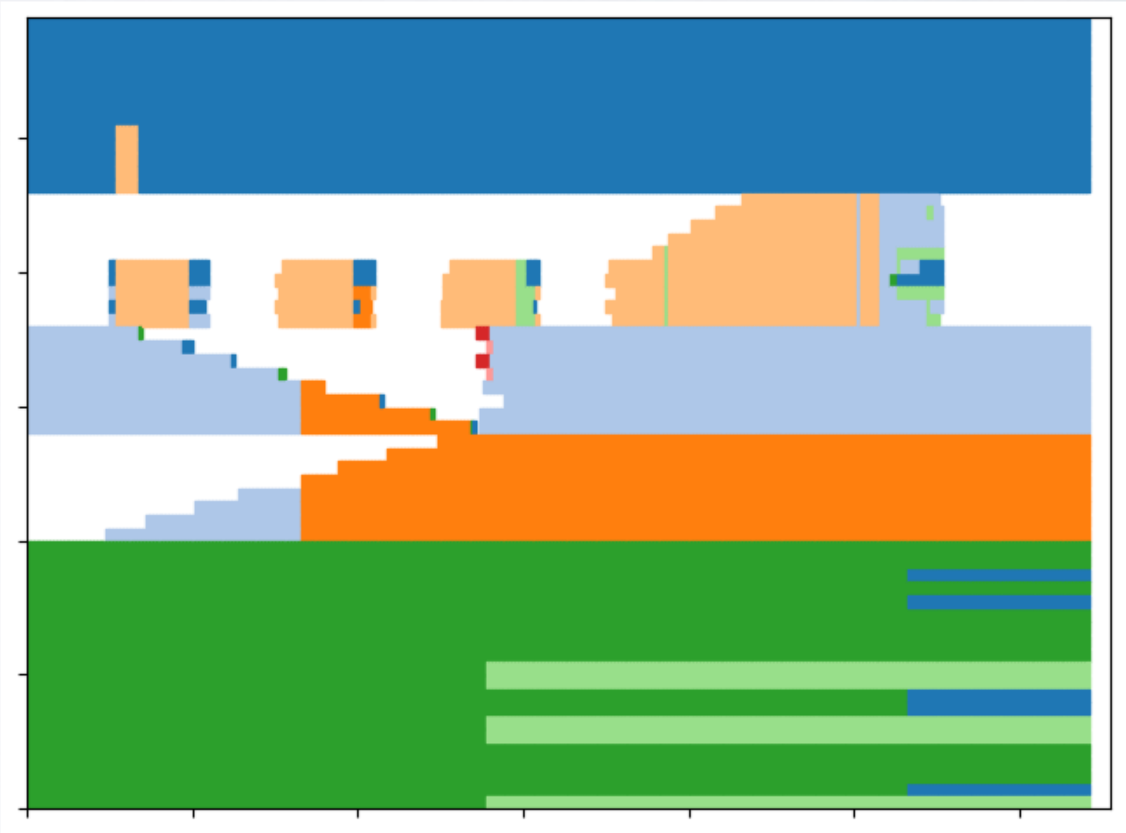
Ground truth



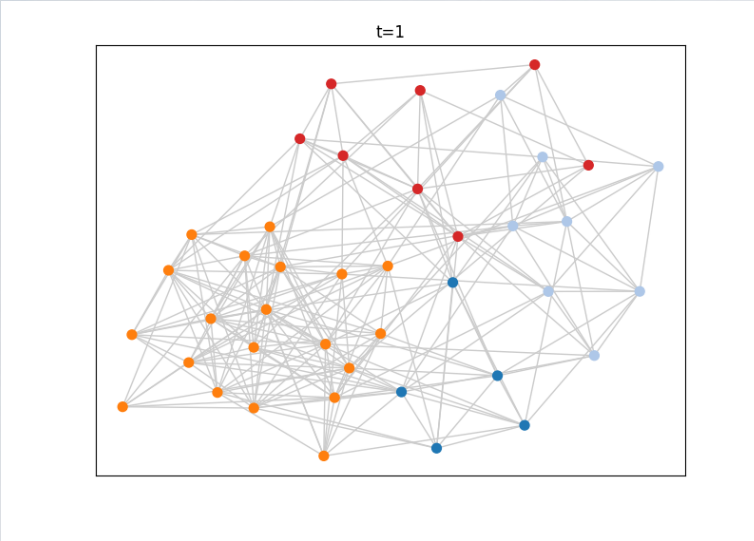
(b) The static graph at time t=0, version *sharp*
($\alpha = 0.9, \beta = 0.05, \beta_r = 0.01$)



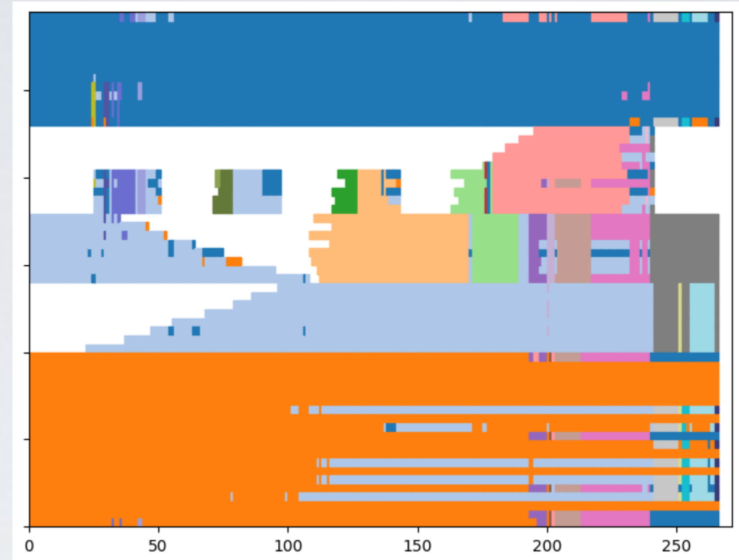
(a) No-Smoothing



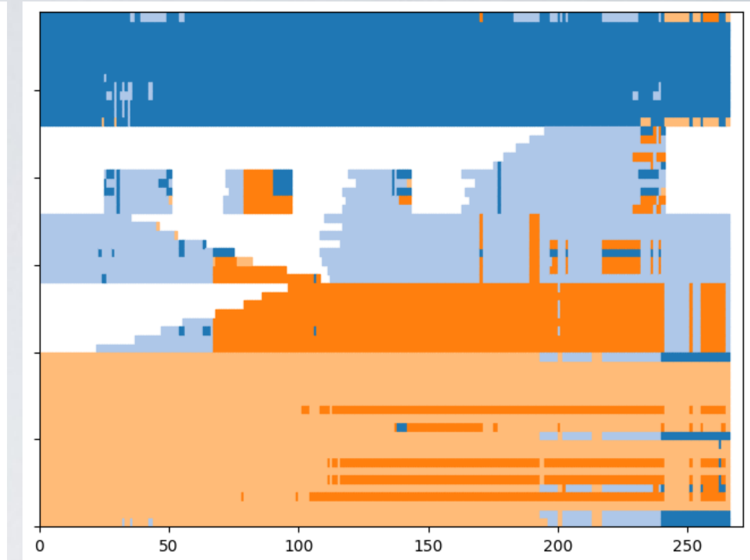
(b) Label-Smoothing



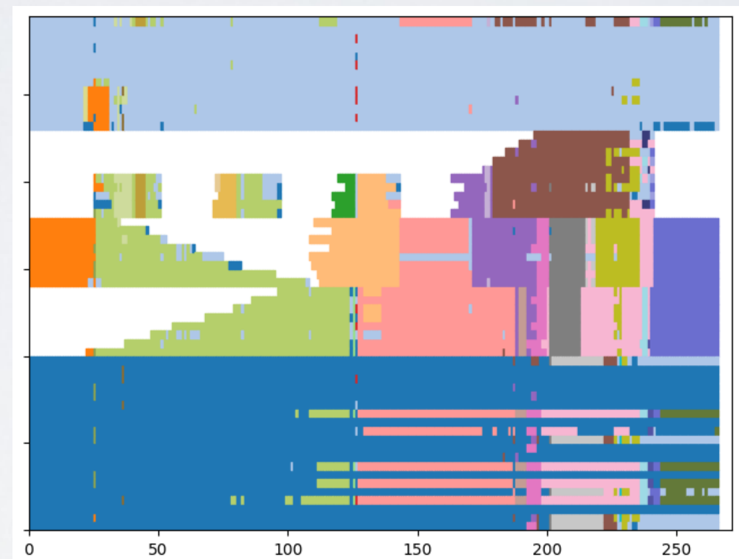
(c) The static graph at time t=0, version *blurred*
($\alpha = 0.8, \beta = 0.25, \beta_r = 0.01$)



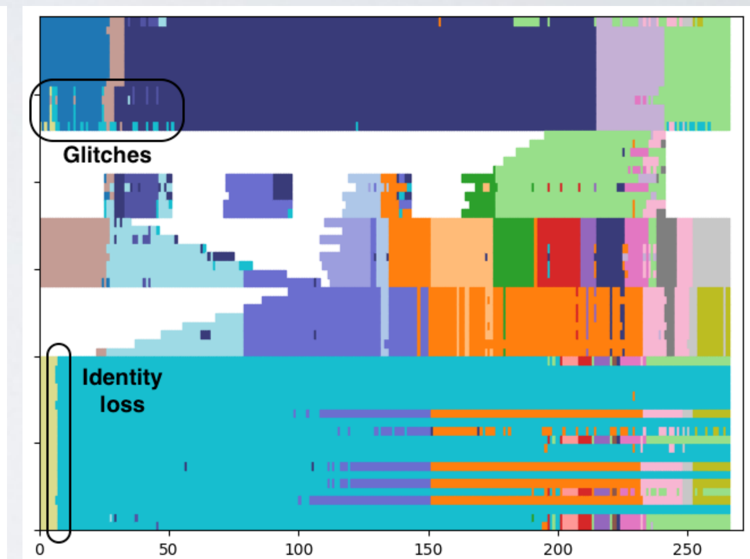
(a) No-Smoothing



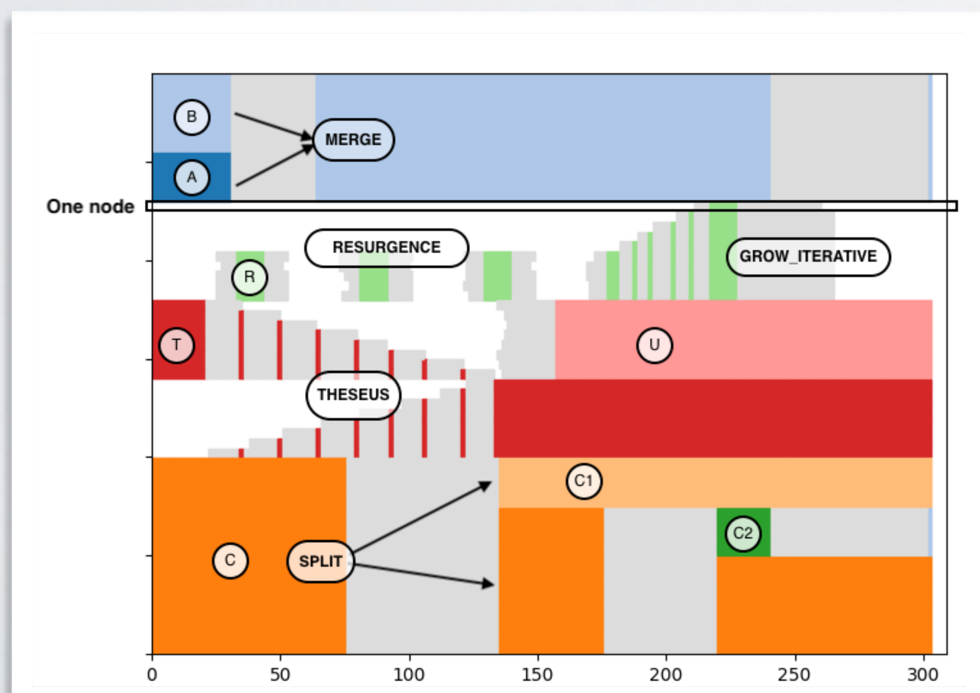
(b) Label-Smoothing



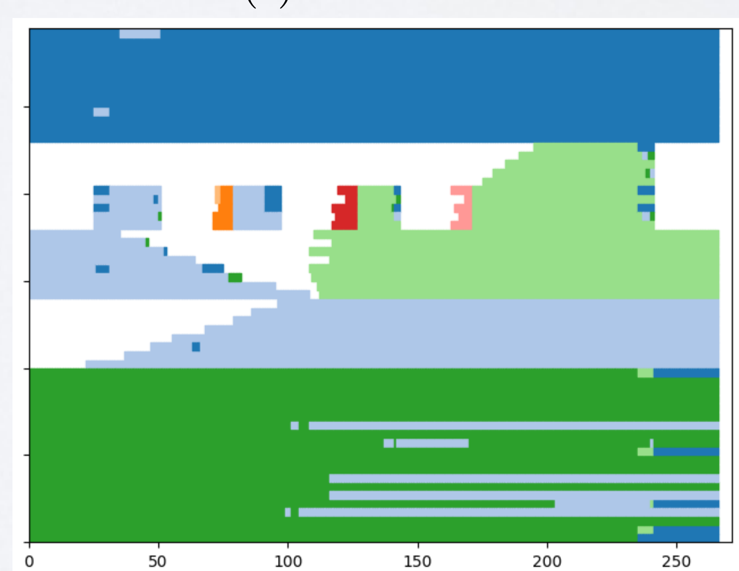
(c) DYNAMO



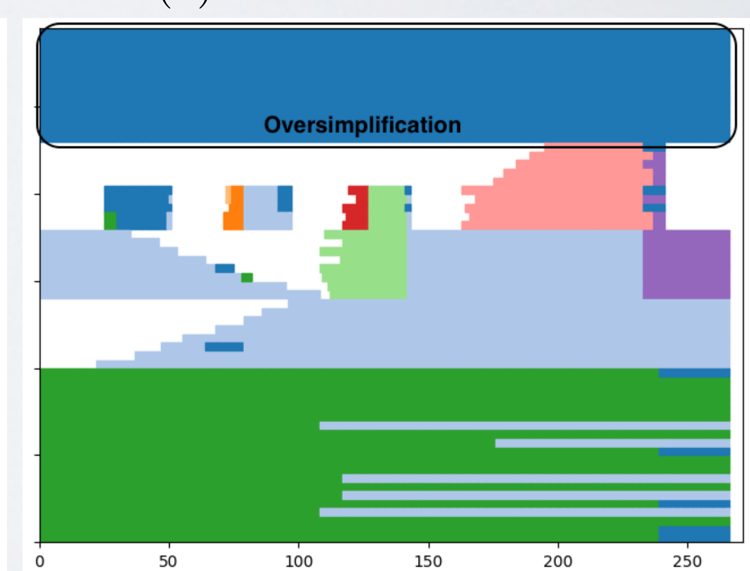
(d) Transversal-Network



Ground truth



(e) Implicit-Global



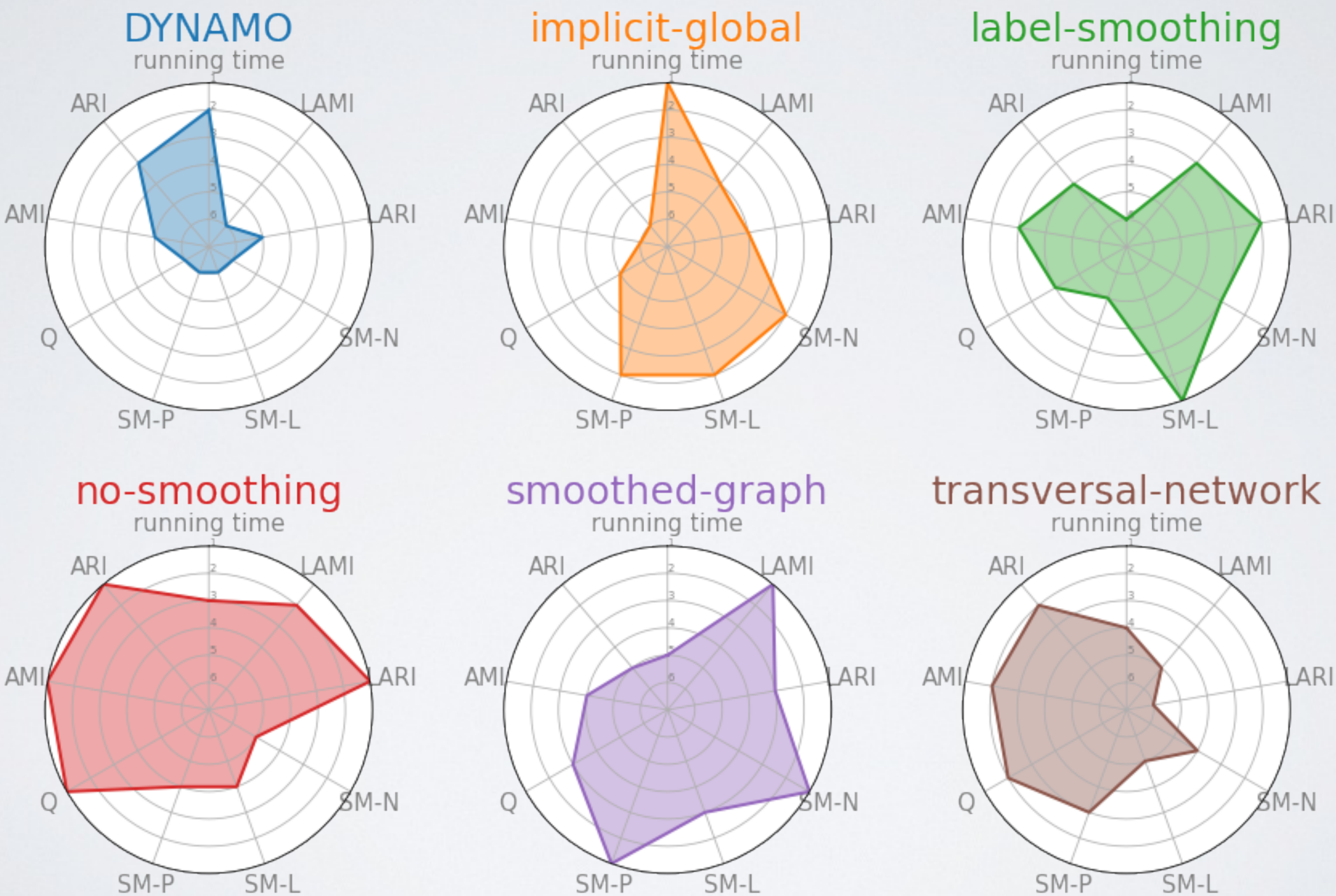
(f) Smoothed-Graph

MEASURING DC QUALITY?

- Evaluation at each step (No smoothness)
  - Average Mutual Information (similarity at each step)
  - Average Modularity

- Evaluation of Smoothness
  - **SM-P**artitions: Average Mutual Information between successive partitions
    - Label independent, insensitive to glitches, Identity loss
  - **SM-N**odes: Inverse of number of affiliation change
    - Sensitive to glitches
  - **SM-L**abels: Inverse of Shannon entropy of nodes labels
    - Sensitive to Identity Loss
- Longitudinal Score
  - Modified mutual information of time-node (u,t)

MEASURING DC QUALITY?



TO SUM UP ON DYNAMIC GRAPHS

TO SUM UP

- Currently, most practitioners still use the snapshot approaches
  - No widespread framework
  - No widespread coding libraries (pathpy, tnetwork, tacoma=>limited usage)
  - Datasets still relatively limited

- But considered an important topic to work on
  - Dynamic is everywhere
  - Dynamic changes many things in many cases