Mining and Learning on Temporal Networks

Tutorial@ITADATA2025

ALESSIA GALDEMAN
MANUEL DILEO
MATTEO ZIGNANI
SABRINA GAITO

ITA DATA 2025

onnets LAB

Computer Science

UNIVERSITÀ DEGLI STUDI DI MILANO

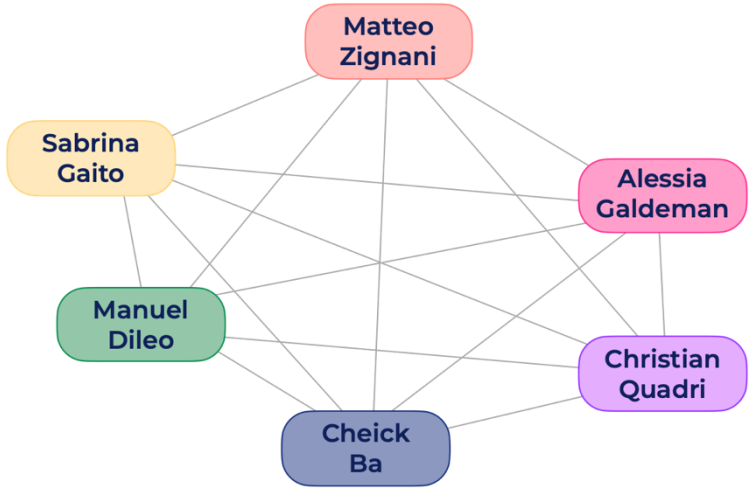
Research Group

Connets Lab @ University of Milan

Network evolution

Graph evolution rules

Change point/Anomaly detection



Graph Machine Learning

Social network analysis using GNN and LLM

Temporal Graph Learning for heterogeneous networks

User behaviour

Multilayer community detection

Influence of hubs in a user migration context

User strategies in a reward-based platform

Web3 platform behavioral and network analysis

Blockchain-based online social network

NFT networks

Cryptocurrency networks (Luna, Steem, Ethereum, Sarafu)

Tutorial Outline

1

Temporal and evolving networks
Definitions, Formalisms, Tools
Hands-on Raphtory

2

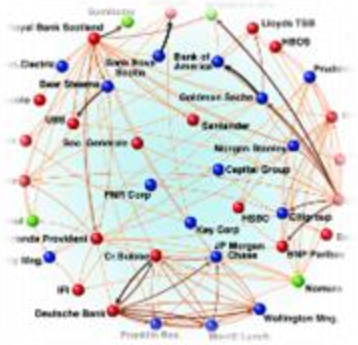
Mining: Graph Evolution Rules
definitions, formalisms, and visualization
Hands-on Geranio

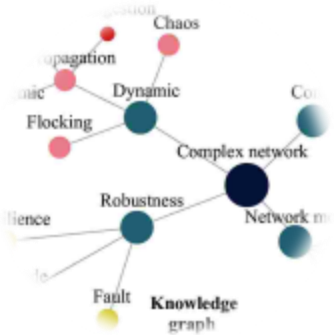
3

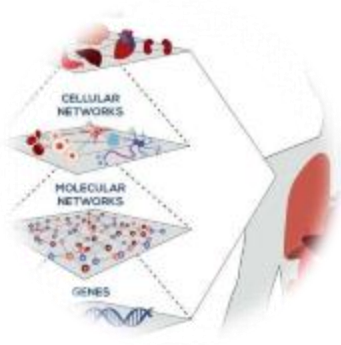
Learning on Temporal Networks
Temporal Graph Learning – continuous and discrete
Hands-on TGL

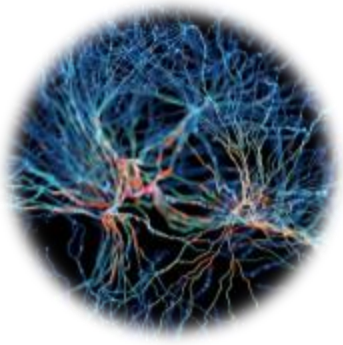
Network Thinking

- Network/Graph model for complex systems
  - System elements – node – and their interactions/relationships – links or edges
- Ubiquitous and powerful model for ...









and much more …

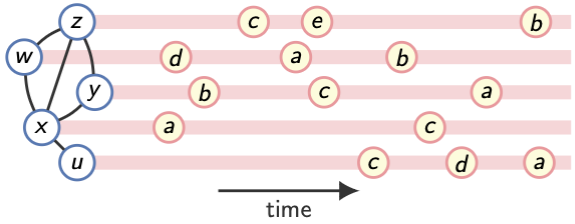
Network Thinking

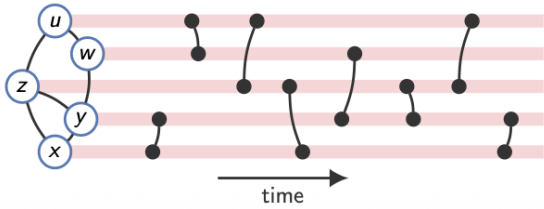
- Network thinking supports:
  - Network *Mining* discoveries and applications:
    - Communities, modules, roles, events
    - Subgraphs, motifs
    - Information diffusion, opinion formation, network generation
  - *Learning* on Networks:
    - Node|link|graph classification|regression
    - Graph representation

But here networks are inherently static

Bring the time into graphs

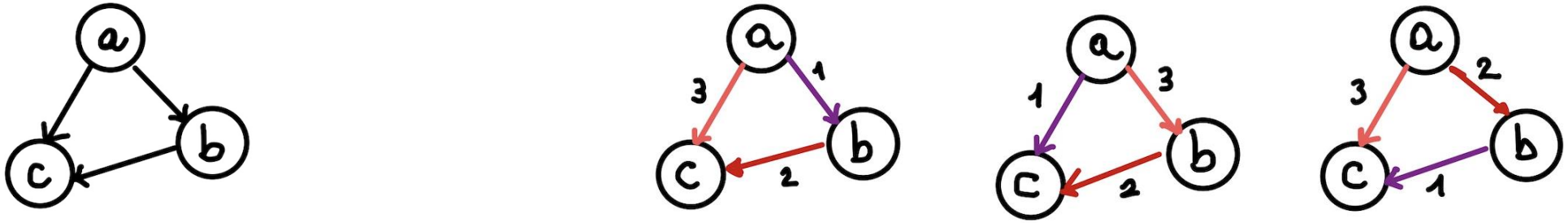
- Data availability of temporal networked data
- Different time resolution from *ms* to years
- $G = (V, E)$
  - Temporal data for V
  - Temporal



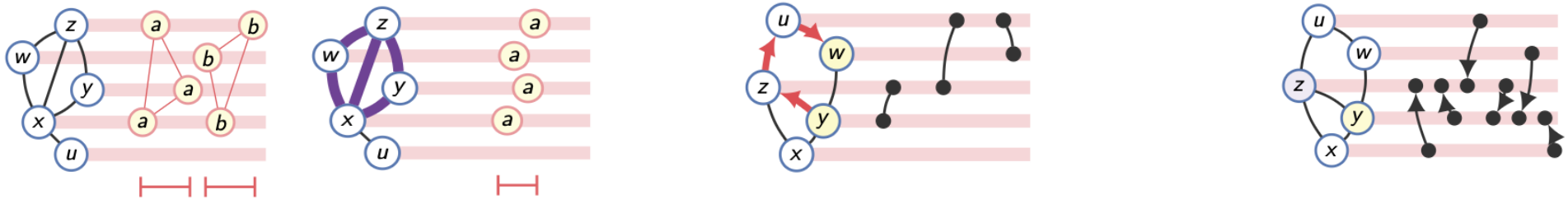


With great power comes great opportunities

- Static graph is not enough for the “arrow of time” ➔ ask for a more powerful representation



- Novel tasks



Synchronization patterns or events

Information flows

Network Evolution

Data sources for temporal networks

- **Digital communication networks**
  - CDR (phone call|text) → $(s, d, t, \Delta t, * data)$
  - Email → $(s, d, t, * data)$
  - Post and social actions in OSNs → $(s, d, t, * data)$
- **Bibliographic networks**
  - Collaborations (scientific) → $(u, v, t, * data)$
  - Citations ($\approx$ acyclic) → $(s, d, t, * data)$
- **Economic networks**
  - Trade networks → $(s, d, t, w, * data)$
  - Transactions networks → $(s, d, t, w, * data)$

Data sources for temporal networks (2)

- **Human and animal proximity networks**
  - Automatic detection: RFID, Bluetooth, WiFi. Direct contacts | co-presence → $(s, d, t, \Delta t, * data)$
  - Manual detection: patient-referral, sexual contact, co-presence same location (e.g. farm) → $(s, d, t, \Delta t, * data)$
- **Travel and transportation networks**
  - Airline connections, public transports, bike-sharing, traffic networks → $(s, d, t, \mathrm{w}, * data)$
- **Biomedical networks**
  - Brain networks: temporal correlations from fRMI signals → $(s, d, t, w, * data)$

Terminology

- No standard terminology for defining networks

Time	Relationships
Temporal	Graph
Dynamic	Network
Evolving	
Time-varying	
Time-dependent	
Evolutionary	

- "… when you see a paper, for example, temporal distance it could mean at least three different things, and look up the definitions paper by paper." P.Holme, 2015

Representation of Temporal Networks

Format common to all the data source = set of contacts

$$\mathbf{I} = \{(\boldsymbol{s}, \boldsymbol{d}, \boldsymbol{t}, \boldsymbol{\Delta t})\}$$

where s and d are elements connected by a relationship starting at time t and lasting for Δt

- Additional information application-dependent
- Node lifespan can be added

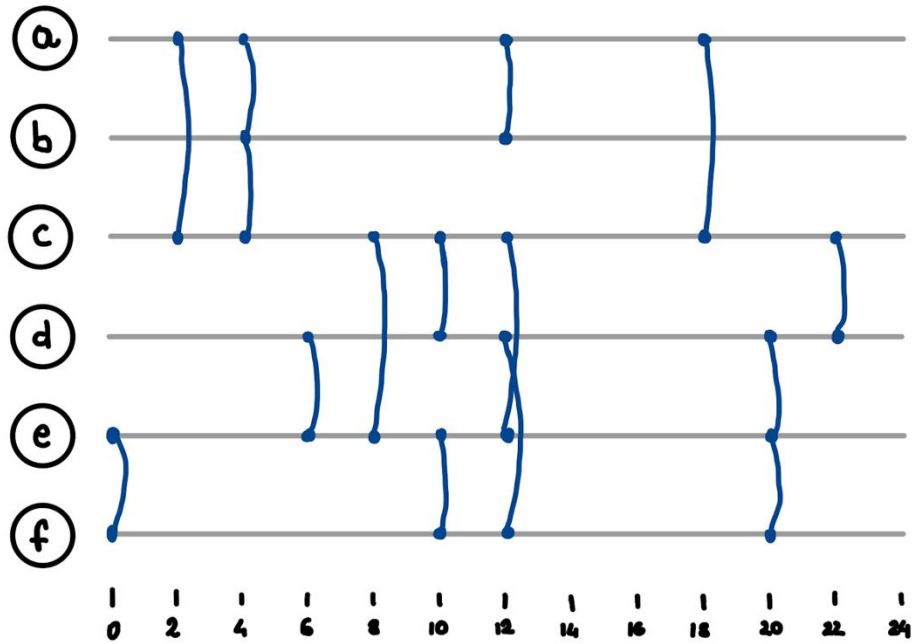
- **Tabular representation**
  - Each row is a contact
  - At least 3 columns

A	B	1
A	B	2
B	C	3
C	D	4
C	D	5
B	C	5
B	D	6
C	D	7

Representation of Temporal Networks

- **Visualization of the interactions over time**
  - Drop the fourth column, i.e. interaction duration
    - Many data sources with instantaneous interactions

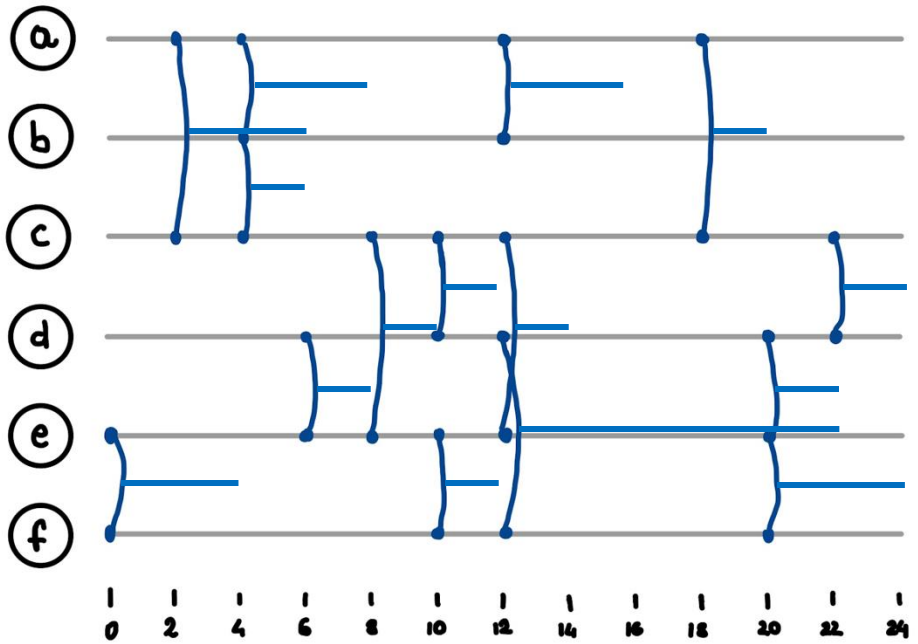
e	f	0	~~4~~
a	c	2	~~4~~
a	b	4	~~4~~
b	c	4	~~2~~
d	e	6	~~2~~
c	e	8	~~2~~
c	d	10	~~2~~
e	f	10	~~2~~
a	b	12	~~4~~
c	e	12	~~2~~
d	f	12	~~10~~
a	c	18	~~2~~
d	e	20	~~2~~
e	f	20	~~4~~
c	d	22	~~2~~



Representation of Temporal Networks

- **Visualization of the interactions over time**

e	f	0	4
a	c	2	4
a	b	4	4
b	c	4	2
d	e	6	2
c	e	8	2
c	d	10	2
e	f	10	2
a	b	12	4
c	e	12	2
d	f	12	10
a	c	18	2
d	e	20	2
e	f	20	4
c	d	22	2



Representation of Temporal Networks

- **Stream graph and link streams**
  - A ***stream graph*** is defined as $G = (T, V, W, E)$ where
    - T is the time domain, V is a set nodes , $W \subseteq T \times V$ and $E \subseteq T \times V \otimes V$
    - $(t, u) \in W$ → node u is present at time t
    - $(t, uv) \in E$ → u and v are linked at time t
  - T_u = set of instant t at which u is present
  - T_{uv} = set of instants t at which u and v are linked
  - In a ***link stream,*** W is missing since $\mathrm{T}_u = T$ for all u

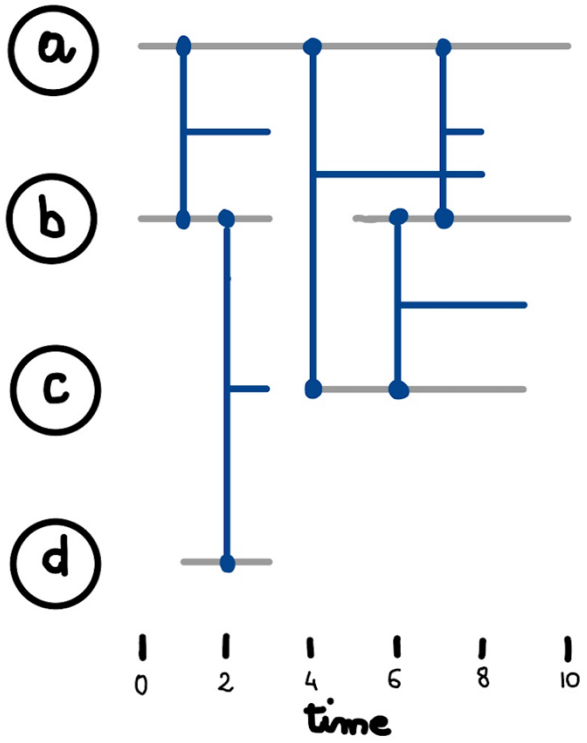
Example Graph Stream

a	b	1	2
b	d	2	1
a	c	4	4
b	c	6	3
a	b	7	1

Interactions

a	0	10
b	0	3
b	5	5
c	4	5
d	1	2

Node life



$$W = [0,10] \times \{a\} \cup ([0,3] \cup [5,10]) \times \{b\} \cup [4,9] \times \{c\} \cup [1,3] \times \{d\}$$

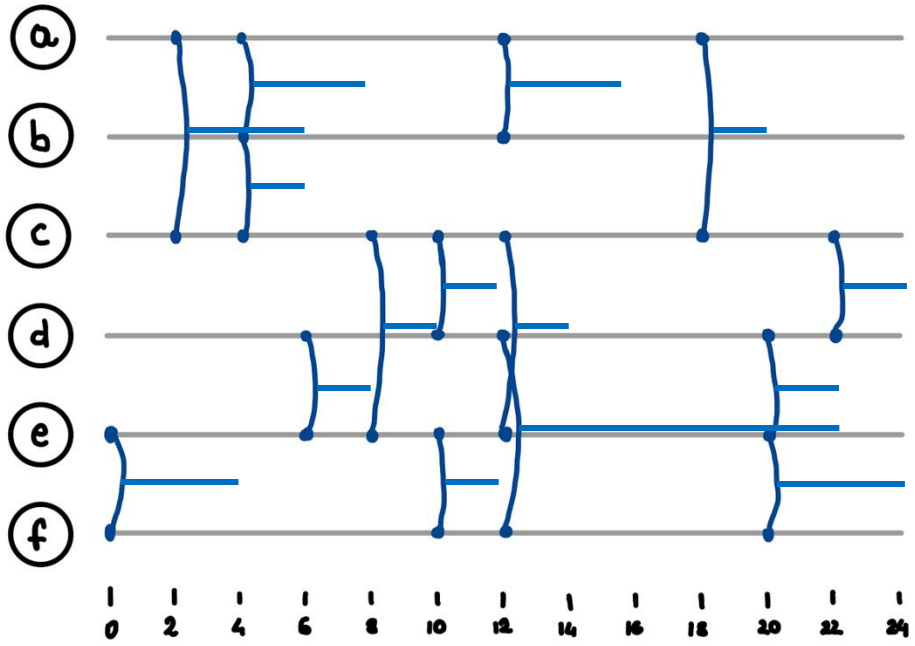
$$E = ([1,3] \cup [7,8]) \times \{ab\} \cup [2,3] \times \{bd\} \cup [4,8] \times \{ac\} \cup [6,9] \times \{bc\}$$

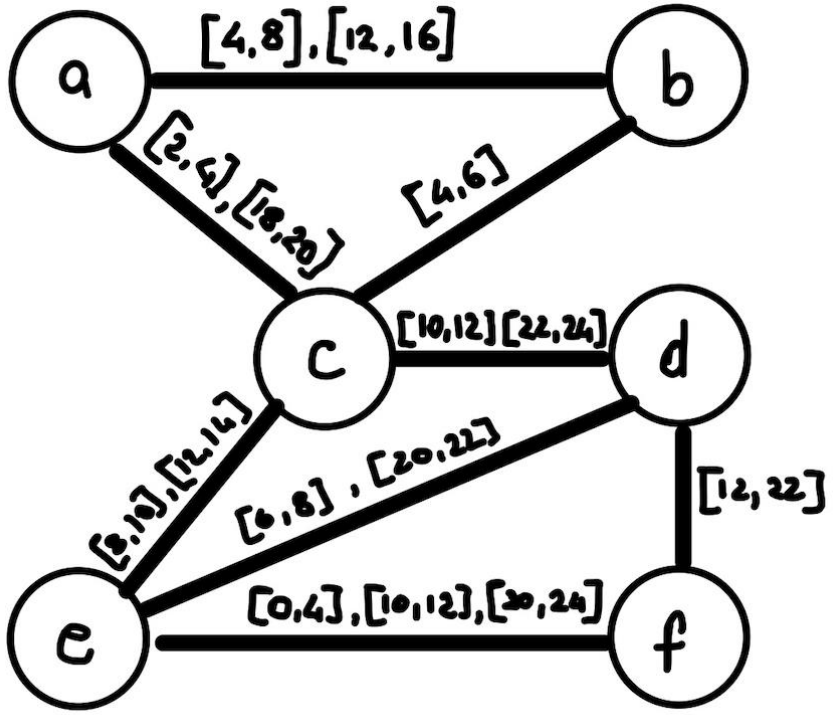
$$T_a = [0,1,2,3,4,5,6,7,8,9,10]$$
$$T_b = [0,1,2,3,5,6,7,8,9,10]$$
$$T_c = [5,6,7,8,9,10]$$
$$T_d = [1,2,3\ ]$$

$$T_{ab} = [1,2,3] \cup [7,8]$$
$$T_{bd} = [2,3]$$
$$T_{ac} = [4,5,6,7,8]$$
$$T_{bc} = [6,7,8,9]$$

Graph representation of Link Stream

e	f	0	4
a	c	2	4
a	b	4	4
b	c	4	2
d	e	6	2
c	e	8	2
c	d	10	2
e	f	10	2
a	b	12	4
c	e	12	2
d	f	12	10
a	c	18	2
d	e	20	2
e	f	20	4
c	d	22	2





Interval Graph

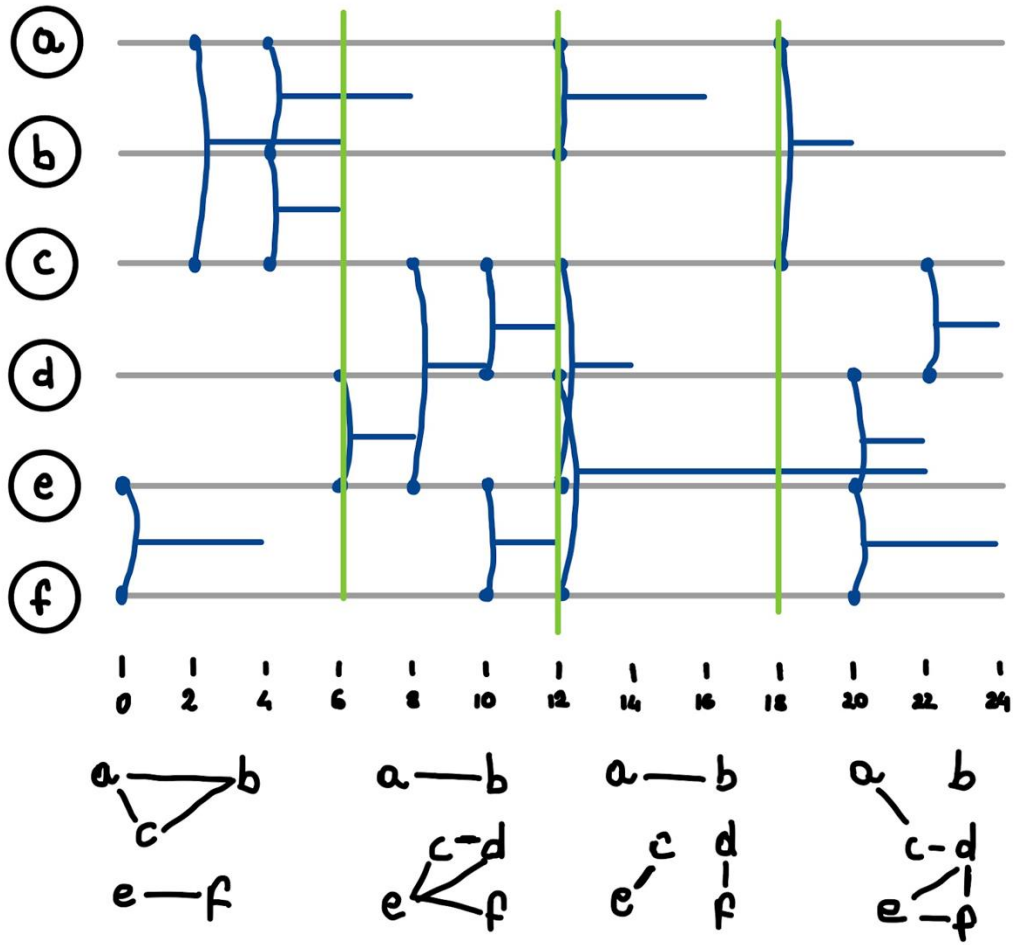
Representation of Temporal Networks

- **Time-varying graphs**
  - A ***time-varying graph*** is defined as $G = (V, E, T, p, \lambda)$ where
    - T is the time domain, V is a set nodes , E$\subseteq V \times V$ is the link set
    - $p: E \times T \rightarrow \{0,1\}$ is a presence function
    - $\lambda: E \times T \rightarrow \mathbb{R}$ is a duration function
  - Equivalent to link stream

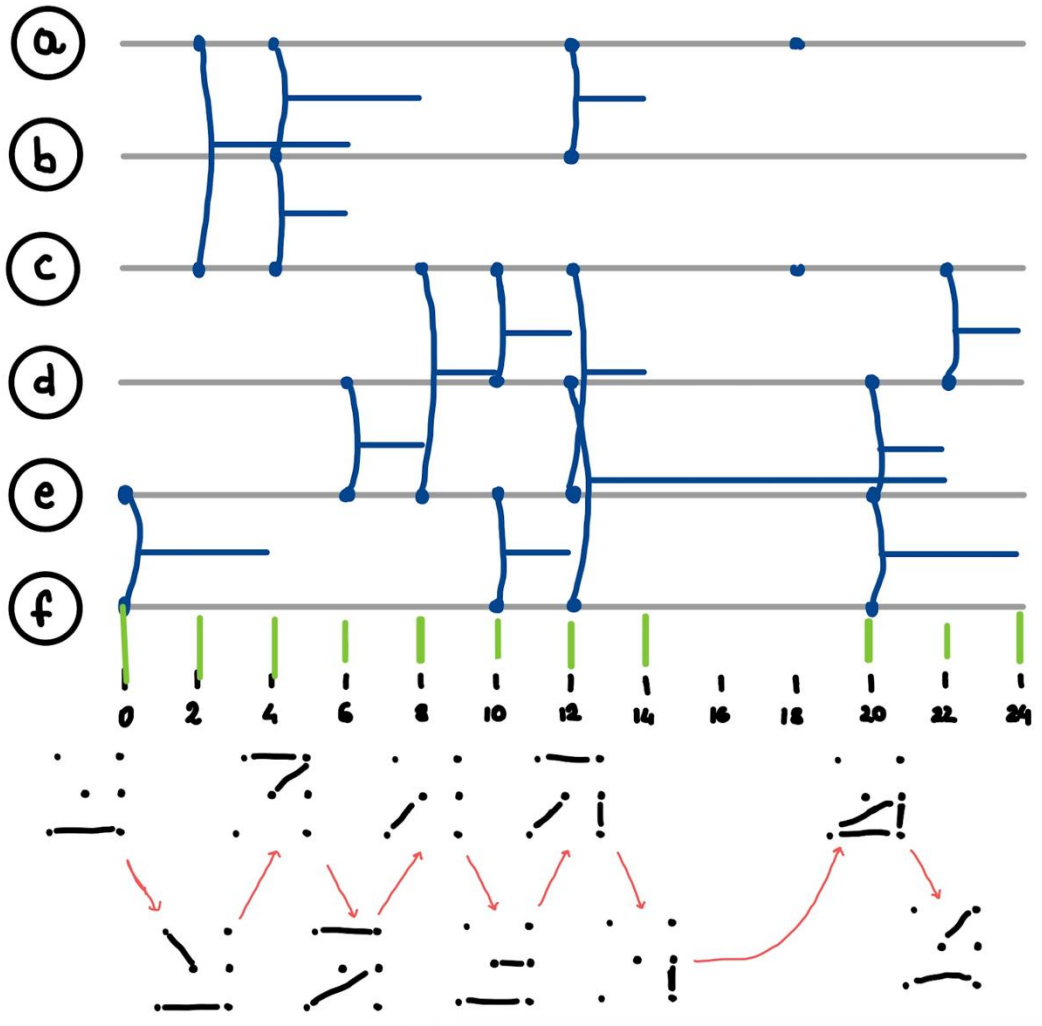
Network Science oriented representation

- **Sequence of static graphs**
  - Sequence of static graphs $G_1, \dots G_{T_s}$ where
    - $G_s = (V_s, E_s)$ is a static graph with V_s the set nodes active in the time slice s, E_s the links active in s
  - Usually $V_s = V$, i.e. nodes are fixed
  - Slicing/windowing fundamental
    - Lossless representation is $s \approx t$ → fine resolution (sparsity issue)
    - Lossy otherwise → coarse resolution

Example of slicing



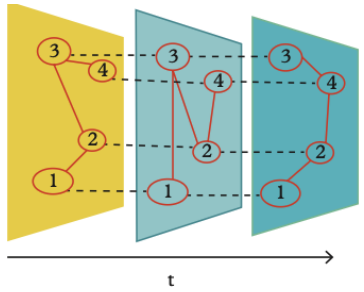
Coarse Resolution



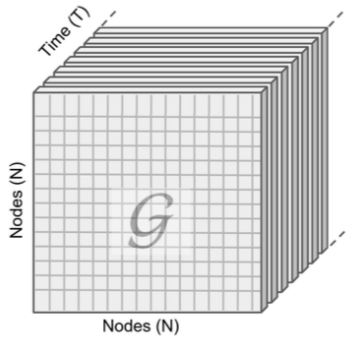
Variable Resolution

Network Science oriented representation

- **Sequence of static graphs**
  - Multilayer network or multi-slice network where inter-layer links connect each node in one layer only to the same node in the next layer.



  - Tensor representation
    - $V \times V \times T$ tensor → tensor operations may break the time ordering



Lossy representation

- **Time window or projected graph**
  - Static graph $G_I = (V_I, E_I)$ for a given time interval $I = [a, b]$ where
    - $E_I = \{(u, v) | (u, v, t) \in E, t \in I\}$
    - $V_I = \{u, v | (u, v) \in E_I\}$
  - I can correspond to T (graph lifespan)
  - Add weights to links reflecting temporal properties, i.e. number of interactions, sum or average duration

Tools for Temporal Networks

RAPHTORY

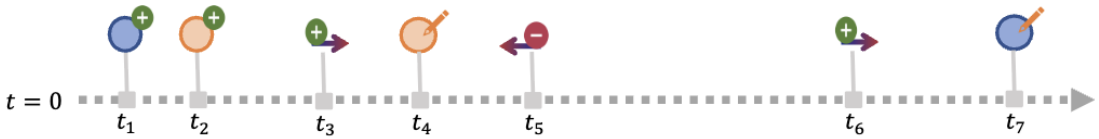
Libraries

- Many tools in Python (backend in C/C++ or Rust)

Library	Characteristic
DyNetX	Interaction|Snapshot|Interval graph, basic functions, few temporal properties
tnetwork	Interaction|Snapshot|Interval graph, dynamic com detection
Teneto	Interaction|Snapshot graph, net measures, dynamic com detection
Tacoma	Interaction|Snapshot|Interval graph, basic network properties, spreading on temp net
Tglib	Snapshot|SpecialCases TN representation, temporal centralities, temporal paths
Reticula	Temporal paths (a couple of algo)
Raphtory	Interaction|Snapshot|Interval graph, temporal node and link properties, temporal paths, temporal network motifs, temporal triangles, network algorithms for static nets

Raphtory Graph Model

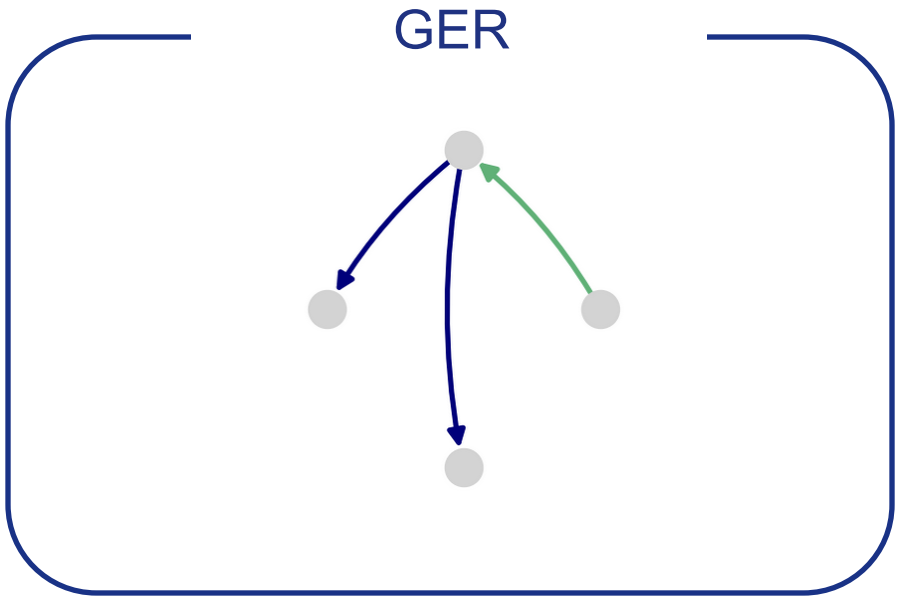
- Storage: graph history = sequence of events involving graph elements



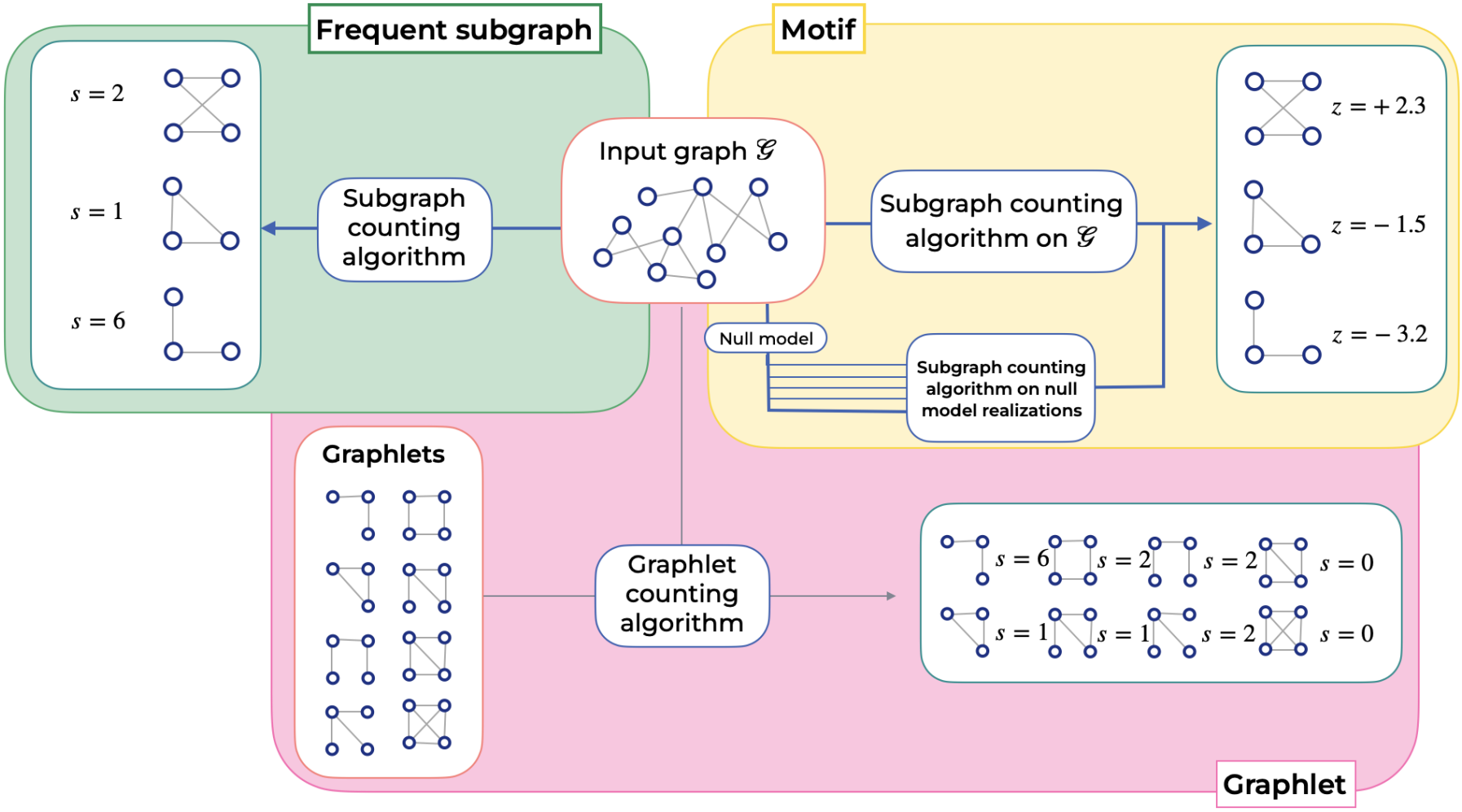
  - History of node properties
  - History of link properties

- Supported representation:
  - Snapshot: sequence of graphs
  - Link Stream: instantaneous interactions
  - Interval Graph: graph stream
  - Project Graph

Part II: Mining Graph Evolution Rules

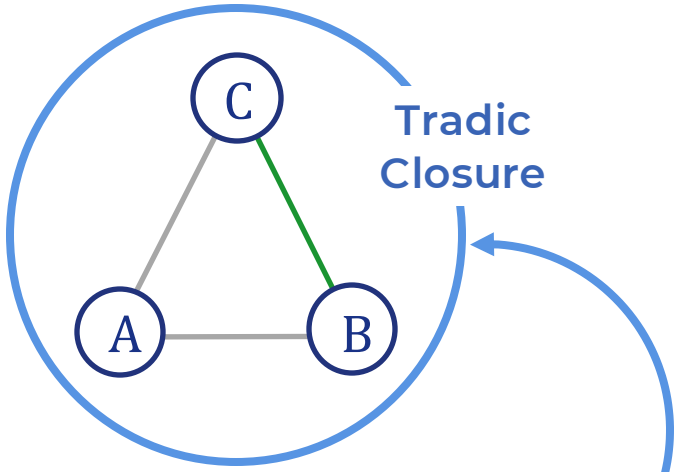


Frequent subgraph based methods



Graph evolution rules

Motivations



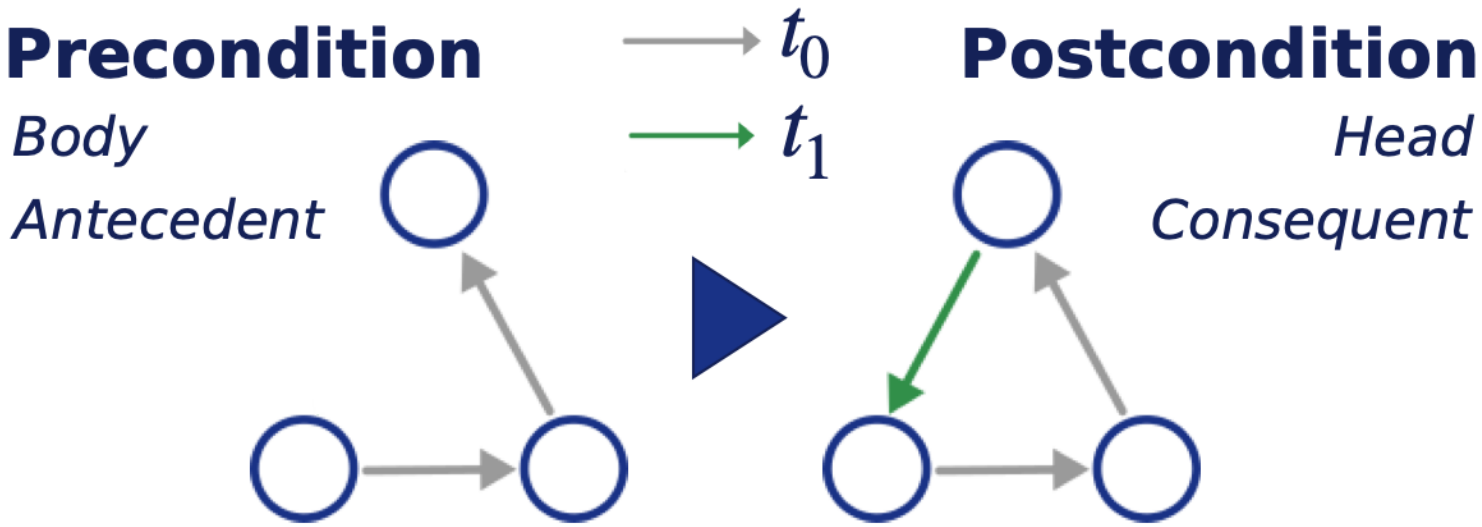
Several models, mechanisms and measures have been proposed to describe the network growth

BUT

- They assume that the growth is guided by a single parameterized mechanism
- Identifying which mechanism plays a more important role is challenging

Graph evolution rules mining can detect evolutionary behaviors, while avoiding any a-priori mechanism

Rules: Composition and meaning

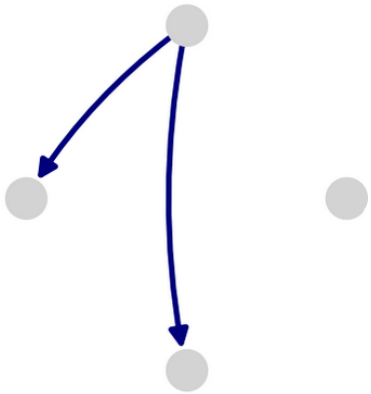


A rule matching (being isomorphic) to the precondition

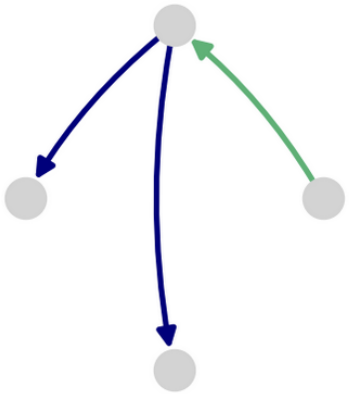
will probably (frequently) evolve into one matching the postcondition

GER visualizations

Precondition



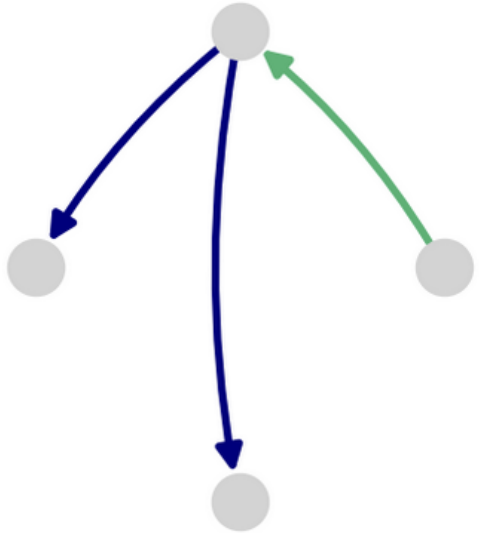
Postcondition



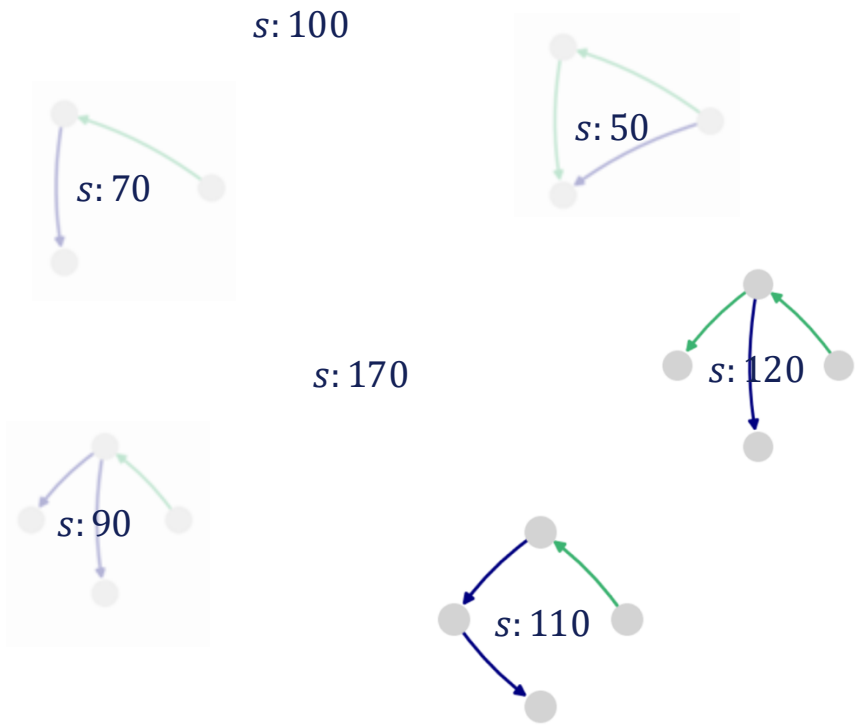
Thanks to the anti-monotonicity property, there's one pre-condition for each post-condition

visualize the post condition (with colored edges)

Compact version



Pattern Support



Support threshold $\sigma \geq 100$

- The **support** is a fundamental parameter in ger mining algorithms because it filters the patterns to **determine which are frequent**, and so can be considered as rules.
- In the data mining field, it correspond to the frequency of the pattern, but ..
- In graphs, it can't be simply the number of occurrences of the pattern because it should satisfy the anti-monotonicity property

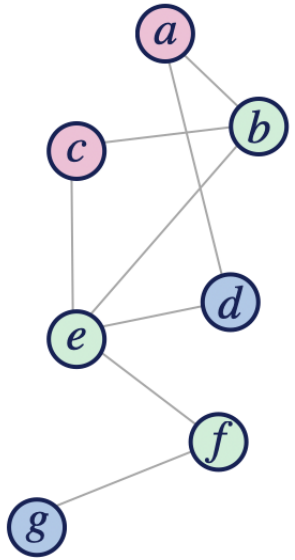
$$q \subset p$$

$$q \qquad \sigma(q) \geq \sigma(p) \qquad p$$

For each pattern p, there is also q because q is a subset of p, so q's support should be larger

Pattern Support – Minimum Image Base (MIB)

- Common definition for pattern support is Minimum Image Based Support: minimum number of unique nodes of the input graph G that each node of the subgraph $p = (V_p, E_p)$ is mapped to.



(a)Input graphG

(b)Subgraphp

	ϕ_1	ϕ_2	ϕ_3	ϕ_4	$\lvert\Phi(v_i)\rvert$
v_1	a	c	c	c	2
v_2	b	b	e	e	2
v_3	e	e	b	g	3

(c)Four isomorphisms (columns) and unique mappings (rows)

MIB SUPPORT:

$$\sigma(p, G) = 2$$

minimum of the number of unique mappings for the nodes in the pattern

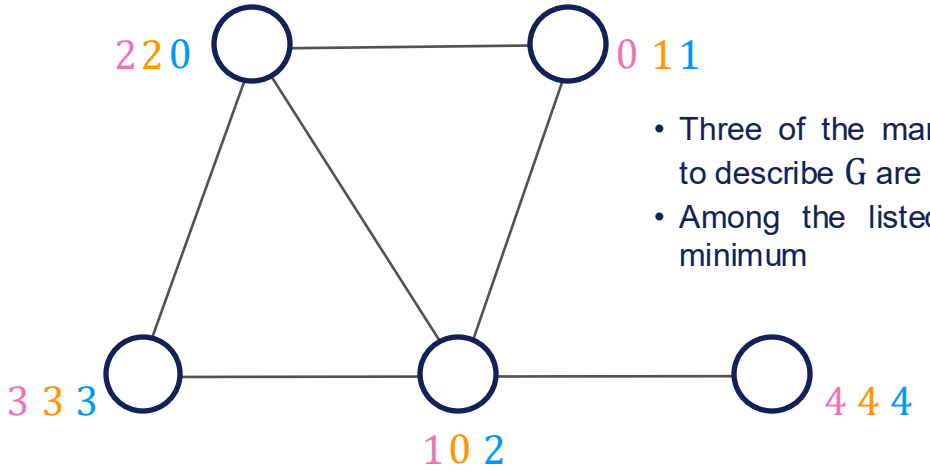
GSPAN – Minimum DFS Code

A graph (or subgraph) can be described through a list of 5-tuple, called DFS code:

$$(i, j, l(i), l(i,j), l(j))$$

- i: source node
- j: destination node
- $l(i)$: label of node i
- $l(j)$: label of node j
- $l(i,j)$: label of edge between i and j

The multiple DFS code for a graph can be lexicographically ordered to obtain the minimum DFS code



- Three of the many possible DFScode to describe G are listed below
- Among the listed, DFScode_2 is the minimum

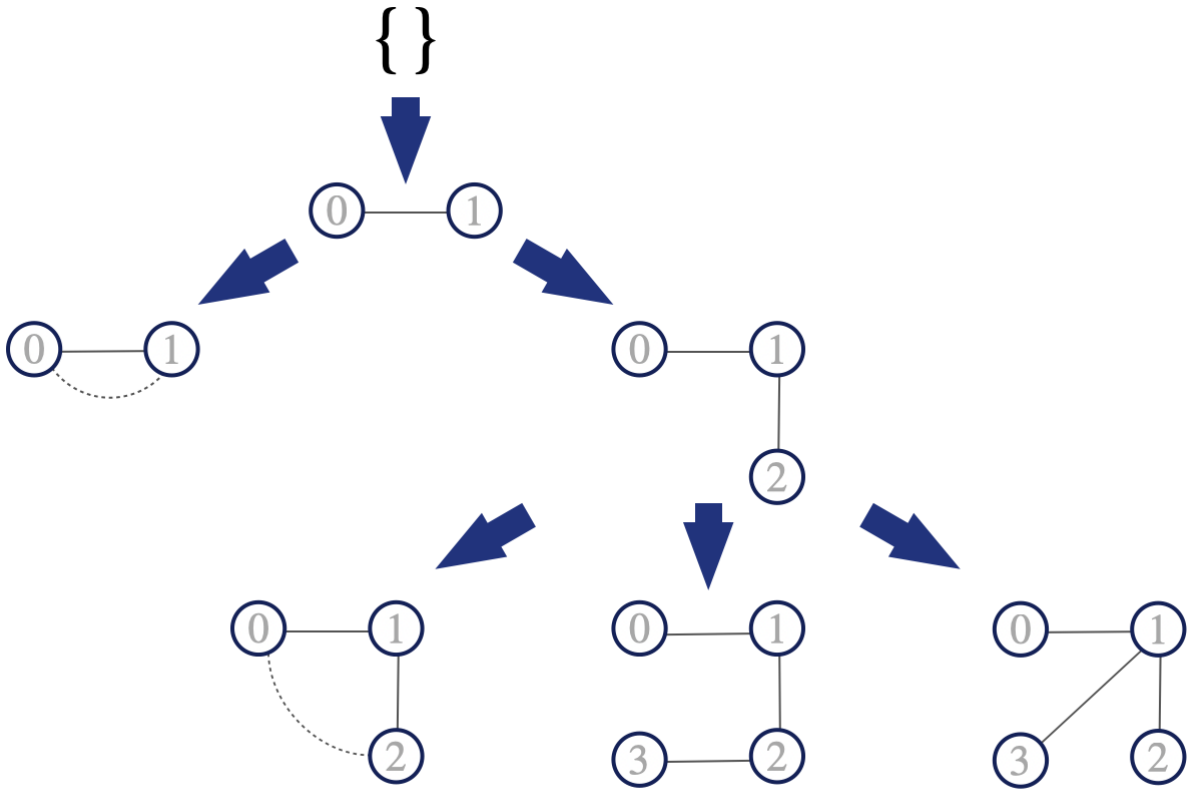
MIN

$$DFScode_1 = (0,1),(1,2),(2,0),(2,3),(3,1),(1,4)$$

$$DFScode_2 = (0,1),(1,2),(2,0),(2,3),(3,0),(0,4)$$

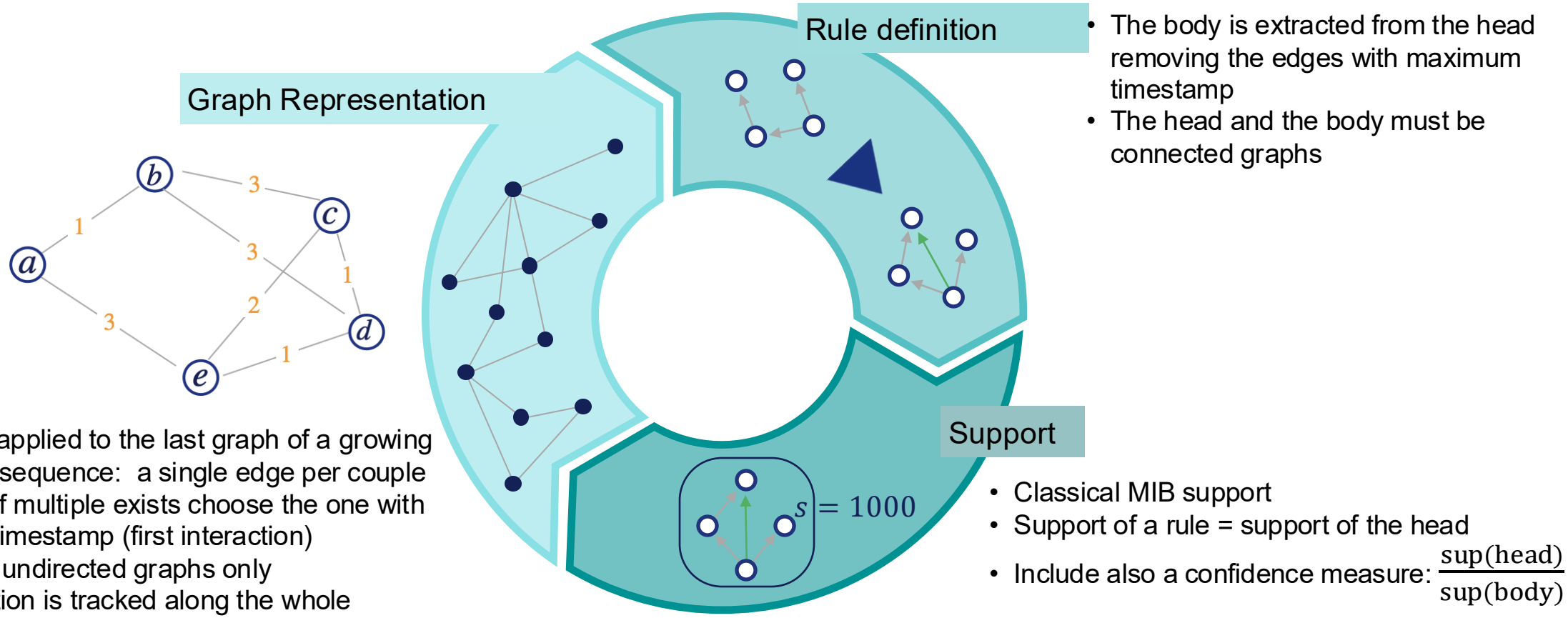
$$DFScode_3 = (0,1),(1,2),(2,0),(2,3),(3,0),(2,4)$$

GSPAN – DFS Tree



- Each node of the DFS Tree is a DFScode;
- The n^{th} level contains DFS codes for graphs with $n - 1$ edges
- The n^{th} level is obtained through rightmost-extension of the parent node
- If a DFScode is not minimum or not frequent, the tree is pruned on that node (nothing will be frequent coming from that branch)
- Setting a maximum of edges (levels of the tree), the DFS tree is expanded up to the specified level and all the subgraphs in the tree are frequent

GERM [1]



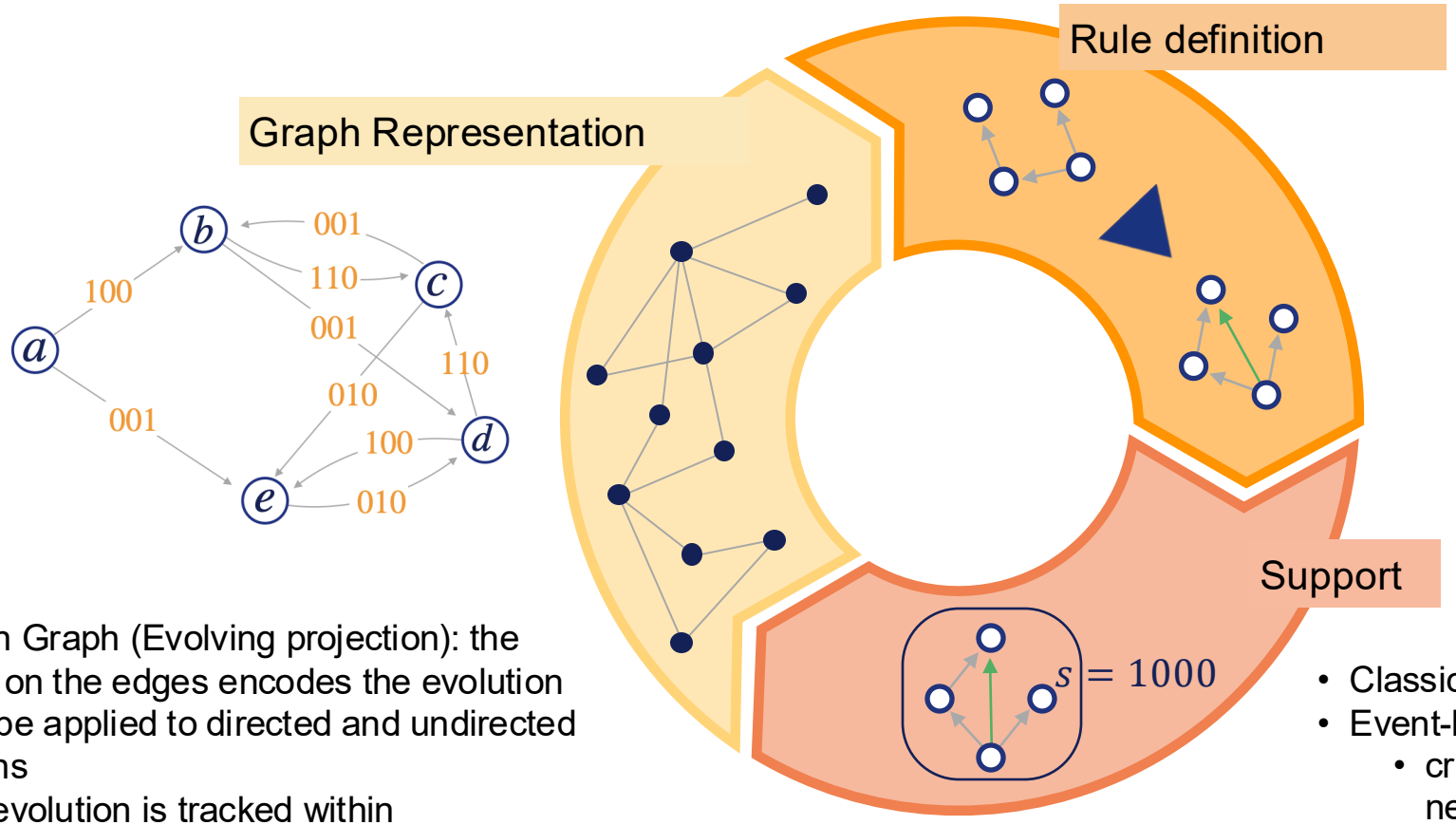
- The body is extracted from the head removing the edges with maximum timestamp
- The head and the body must be connected graphs

- Algorithm applied to the last graph of a growing projection sequence: a single edge per couple of nodes, if multiple exists choose the one with minimum timestamp (first interaction)
- Applied to undirected graphs only
- The evolution is tracked along the whole timespan

- Classical MIB support
- Support of a rule = support of the head
- Include also a confidence measure: $\frac{\sup(\text{head})}{\sup(\text{body})}$

[1] Berlingerio, M., Bonchi, F., Bringmann, B., and Gionis, A. Mining graph evolution rules. In joint European conference on machine learning and knowledge discovery in databases (2009), Springer, pp. 115–130.

EvoMine [2]



- Union Graph (Evolving projection): the label on the edges encodes the evolution
- Can be applied to directed and undirected graphs
- The evolution is tracked within consecutive timestamps

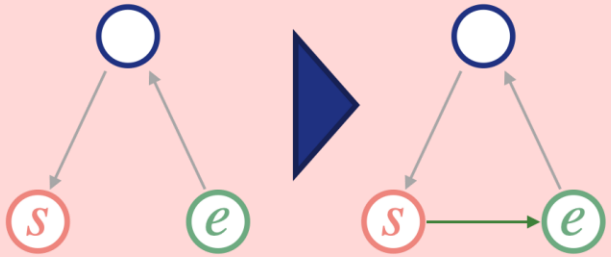
- The only timestamps on the edges are t_0, t_1
- The body (pattern without the edges with t_1 timestamp) must have the same nodes as the head
- From body to head something must change, labels or edges
- The union graph of the rule must be connected

- Classical MIB support
- Event-based support:
  - creates event graphs: subgraphs including the neighborhood of each event (edge insertion, node relabeling and so on)
  - count the event graphs in which a rule appears

[2] Scharwächter, E., Müller, E., Donges, J., Hassani, M., and Seidl, T. Detecting change processes in dynamic networks by frequent graph evolution rule mining. In 2016 IEEE 16th International Conference on Data Mining (ICDM) (2016), IEEE, pp. 1191–1196.

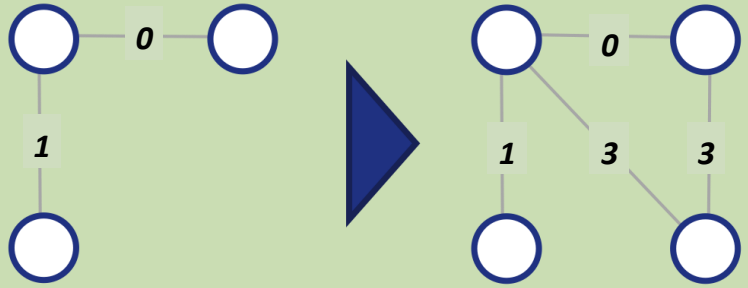
LFR [3]

- The focus is on the process that drives single links formation;
- For this reason, LF rules are more restrictive with respect to the others, but the mining time decreases;
- A null model is integrated to extract meaningful rules;
- They have a tailored support measure and also consider a confidence measure



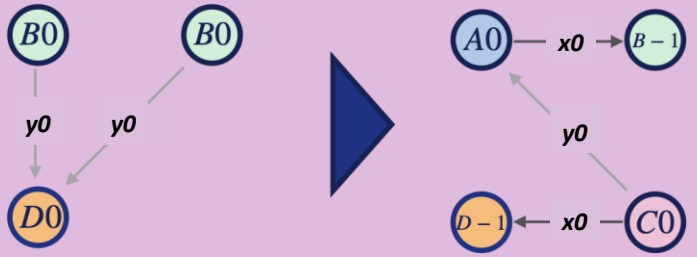
TP-MINER [4]

- It proposes the idea of representative time pattern;
- The algorithm extract the body from the head in the same way as the other ones;
- Builds a DAG from graph evolution rules
- The confidence measure takes into consideration the evolution from body to head



DGR-MINER [5]

- It is designed for labeled multigraph, both directed and undirected
- Proposes its own graph representations and support measures



[3] Leung, C., Lim, E.-P., Lo, D., and Weng, J. Mining interesting link formation rules in social networks. pp. 209–218.
[4] Yuuki, M., Ozaki, T., and Takenao, O. Mining interesting patterns and rules in a time-evolving graph. Lecture Notes in Engineering and Computer Science 2188 (03 2011)
[5] Vaculik, K. A versatile algorithm for predictive graph rule mining. In ITAT (2015), pp. 51–58

Rules significance

Problem

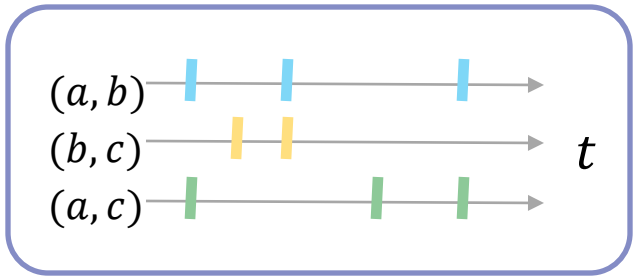
The support alone is not enough to measure if a pattern (rule) is representative of the evolution of the graph:
A pattern can be frequent as a consequence of a general process of a dynamic network, not telling anything on how the network we're studying is evolving

Solution

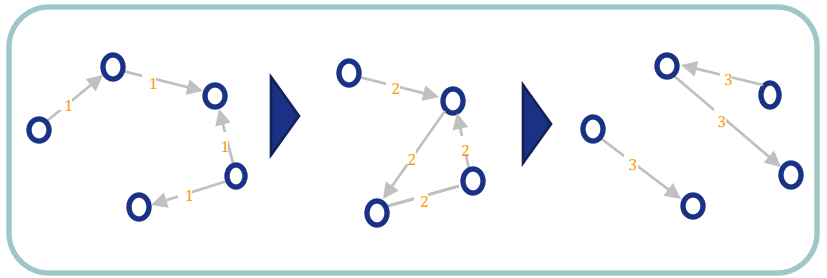
Apply a null model on the graph evolution rules algorithm

- Apply the graph evolution rules algorithm on the real graph
- Apply the graph evolution rules algorithm on a randomized version of the graph
- The rules whose support is higher in the real graph are significative

Microcanonical Randomized Reference Models [6]



MRRM



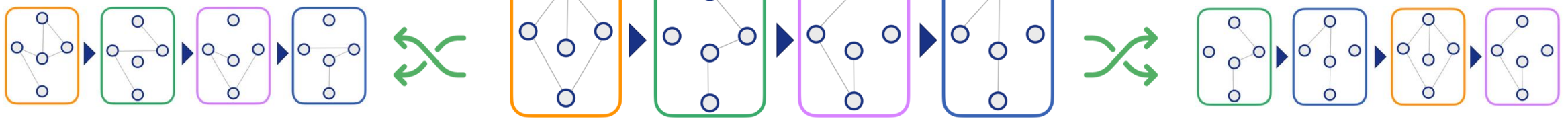
Graph representation

MRMMs categories	Timeline representation	Snapshot representation
Preserve Topology	Timeline shuffling	Sequence shuffling
Preserve Temporal Distro	Link shuffling	Snapshot shuffling

Preserving Temp. Distribution

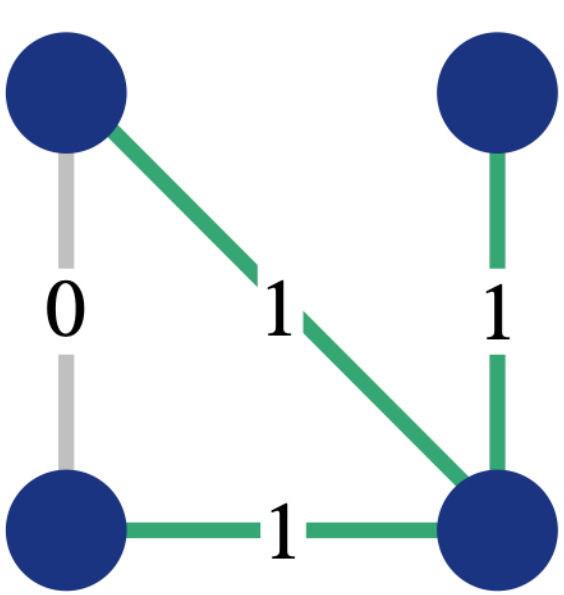
Original graph

Preserving Topology



Example from DBLP

Significative rules based the GERM algorithm



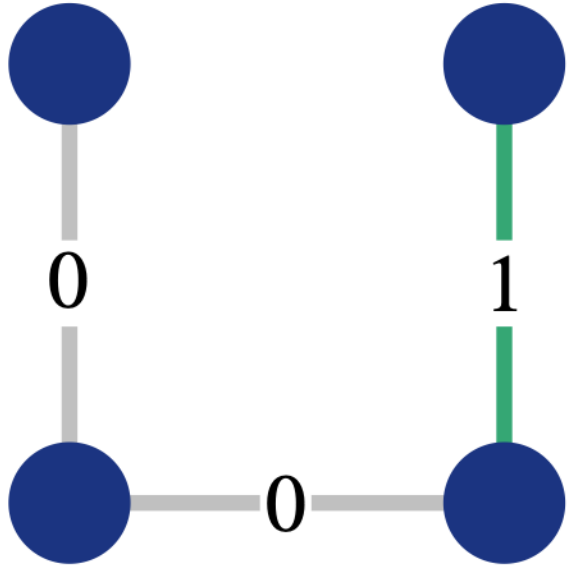
Low support
(over 100th)

High z-score

Rule apparently NOT important but actually worthy of attention

High support
(14th)

Low z-score



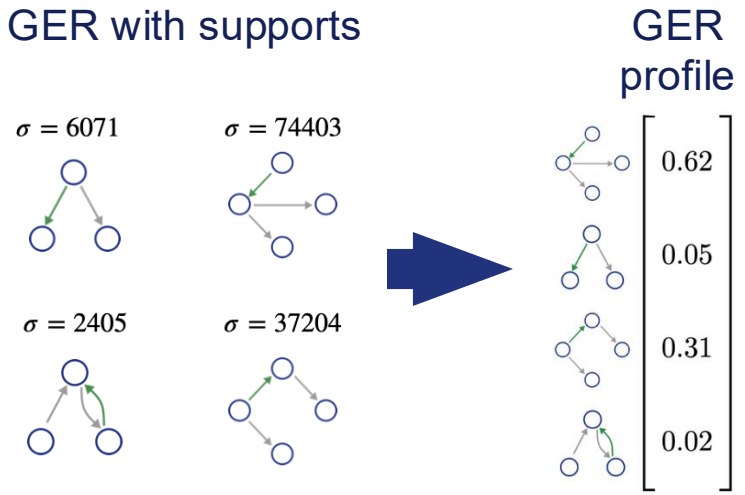
Rule apparently important but actually NOT worthy of attention

Example from Web3 Platforms

1 **Web3 data modeled as temporal networks**

2 **Graph evolution rule mining with EvoMine**

3 GER PROFILE



- GER profiles show the distribution over types of evolution rules for a given dynamic graph
- The comparison of the GER profiles for different graphs makes possible to find similar evolutionary behaviors

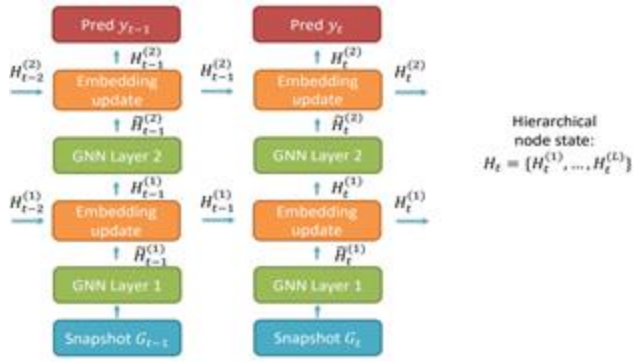
Specifically we worked on

- two networks extracted from operations (transfer and follow) on Steemit, that is a blockchain-based online social network
- two networks from NFT exchanged on two different markets (Cryptokitties and OpenSea)

[8] Galdeman, Alessia, Matteo Zignani, and Sabrina Gaito. "Disentangling the Growth of Blockchain-based Networks by Graph Evolution Rule Mining." 2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA). IEEE, 2022.

Part III: Machine Learning on temporal Networks

TGL



ML on Temporal Networks

Temporal Graph Learning

Temporal Graph Learning (TGL) is a rapidly evolving area of machine learning that focuses on extracting, learning, and predicting graph-structured data that evolves over time.

Get start:

- Temporal Graph Learning in 2023: the story so far
- Temporal Graph Learning in 2024: continue the journey for evolving networks

Keep in touch:

- The Temporal Graph Learning reading group
- The Temporal Graph Learning workshop

Learning task

Temporal Graph Learning

In the graph machine learning literature, tasks are typically categorized according to the granularity of prediction: node, link, and (sub)graph-level tasks.

When considering these tasks to temporal graphs, a crucial distinction arises between dynamic and static temporal tasks.

- Dynamic tasks require predictions that evolve over time, generating distinct outputs for each timestamp (e.g. user interest).
- Static temporal tasks produce a single prediction summarizing the entire time series (e.g. bot detection).

Link prediction (LP)

Learning task

Link prediction is the most intensive studied task in the temporal graph learning literature.

It has a fundamental role in shaping the dynamics of temporal networks.

We will focus on link forecasting: given a temporal graph up to time t, we want to predict links at future time interval t' > t.

Models

A wide range of models have been proposed to address learning on temporal graphs [1].

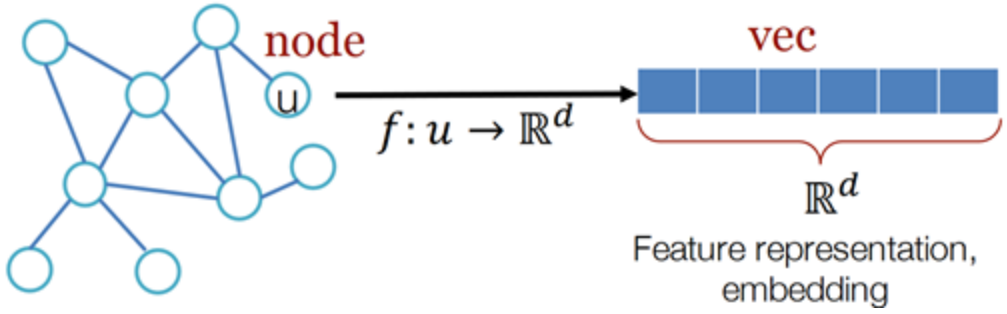
These architectures differ in how they incorporate the temporal dimension:

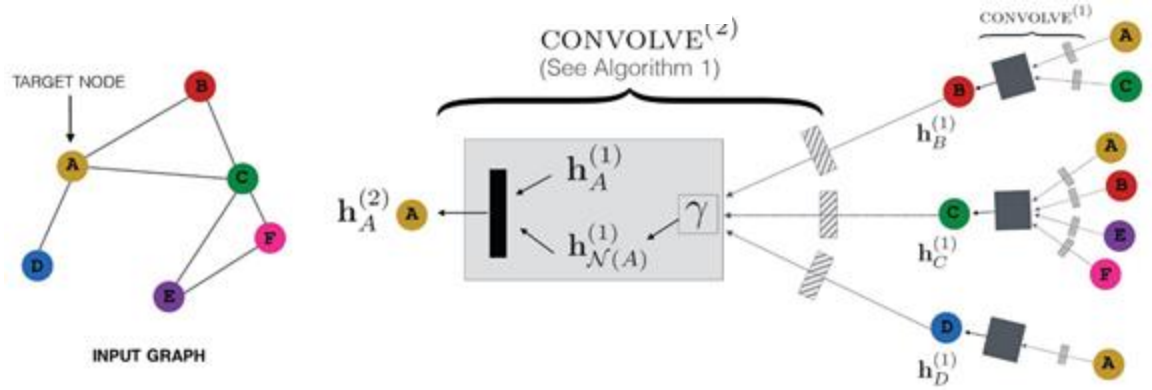
- Attention mechanisms over events and/or timestamps
- Sequence modeling through recurrent architectures
- Timestamp embeddings with positional encoding
- Time-aware regularization
- Temporal random walks
- Temporal graph neural networks

[1] Kazemi et al. (2020). Representation Learning for Dynamic Graphs: A Survey. JMLR

Graph Neural Networks (GNNs)

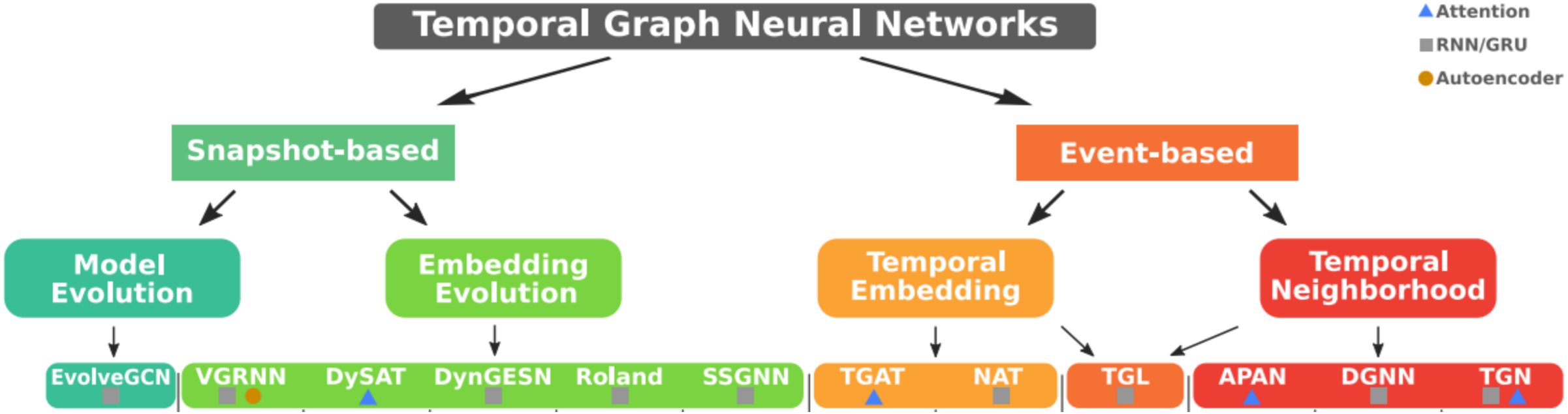
- Neural Networks that works naturally on graph-structured data.
- Automatic feature learning on graph with node attributes
- Node features combined with neighbour information





Representation Learning on Networks, snap.stanford.edu/proj/embeddings-www, WWW 2018

Temporal GNNs



[2] Longa et al. (2023). Graph Neural Networks for temporal graphs: State of the art, open challenges, and opportunities. TMLR

Snapshot-based TGNN models

Practical notes

This modeling framework is well-suited when data can be collected in regular periodic batches and/or nodes act very similar in close timestamps.

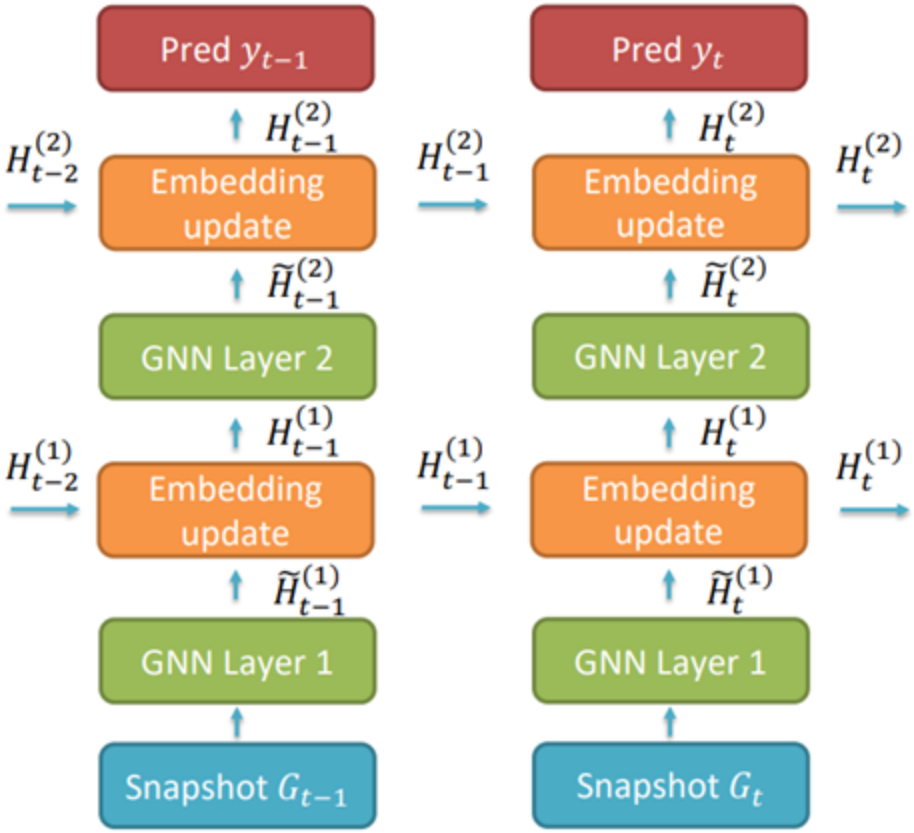
Examples include online social networks and event networks extracted from newspapers.

Implementations are often found in libraries that are no longer maintained.

They receive little attention; however, these models remain fast, interesting for the use cases, and achieve performance comparable to event-based approaches [3].

[3] Huang et al. (2024). UTG: Towards a Unified View of Snapshot and Event Based Models for Temporal Graphs. LOG

ROLAND



Hierarchical node state:

$H_t = \{H_t^{(1)}, \ldots, H_t^{(L)}\}$

[4] You et al. (2022). ROLAND: Graph Learning Framework for Dynamic Graphs. KDD

Event-based TGNN models

Practical notes

This fine-grained temporal resolution is essential when high-precision timing is available and relevant, such as sensor networks or critical systems, or when temporal data are irregular over time, such as biomedical records.

More expressive models, usually slower than snapshot-based models.

The ecosystem for these methods is stronger than for discrete-time ones: public libraries usually provide better benchmark datasets, training routines, and evaluation settings.

Most models exhibit high variability in the results across different datasets and there is no clear SOTA method.

https://tgb.complexdatalab.com/

GraphMixer

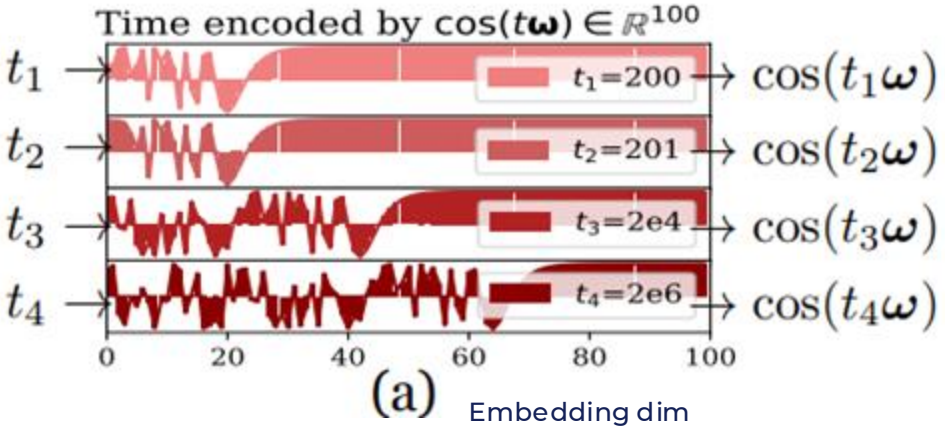
It leverages three modules:

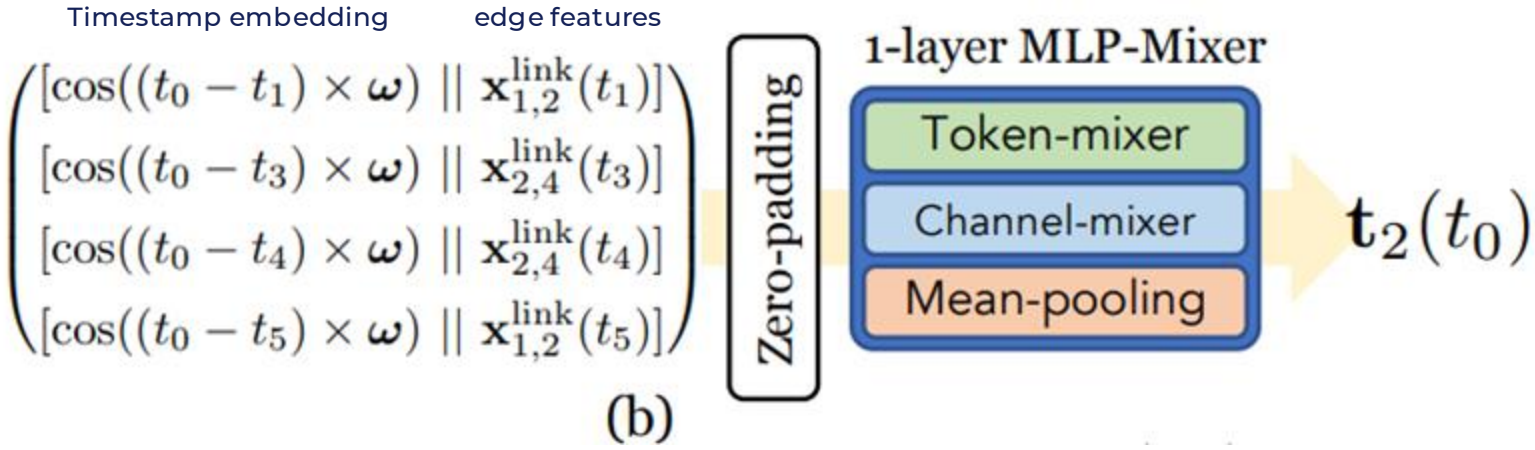
- A link encoder that summarizes the information from temporal links (both features and timestamps)
- A node encoder that summarizes the information from nodes (features and neighbourhood)
- A link classifier that assigns a link prediction score to a candidate pair of nodes.

GraphMixer

Link encoder

Encoding links around node 2.
All timestamps are first passed to a positional encoder.
Then, timestamp embeddings are concatenated with edge features.
Finally, edge embeddings are processed through a MLP layer followed by a mean pooling.





GraphMixer

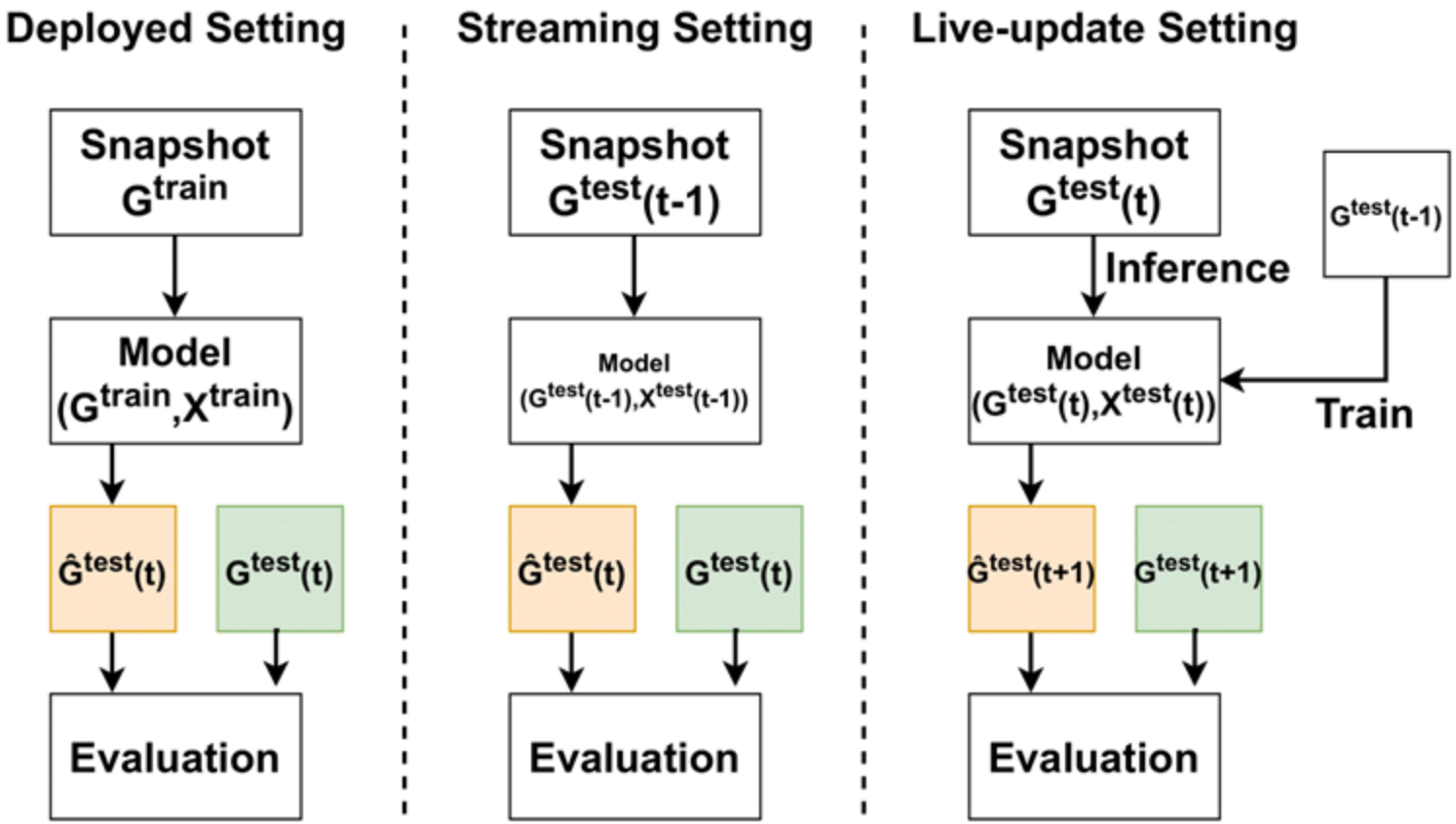
Node encoder and link clf

The node encoder is designed to capture the node feature information via neighbor mean-pooling.

Given a candidate pair (u,v), the link classifier is a 2-layer MLP fed with node_emb(u) | edge_emb(u) | node_emb(v) | edge_emb(v)

[5] Cong et al. (2023). Do We Really Need Complicated Model Architectures For Temporal Networks? ICLR

Training and evaluation setting



Adopted by discrete-time | event-based | New, interesting for both

Link prediction metrics

Link prediction consist of assigning a score to each candidate pair of nodes; the higher the score; the higher the likelihood that the link will exist.

It can be treated as a:

- Binary classification problem: F1-score, AP, AUROC and AUPRC are tipically adopted as evaluation metrics.

- Ranking problem: the score of each future edge should be higher than no existing edges. In this case, Mean Reciprocal Rank (MRR) is adopted as the main evaluation metric.

LP Negative sampling

Compute and rank the score of all the negative edges could be really expensive.

Negative sampling is typically adopted in both binary and ranking scenarios and is computed by corrupting the dst of existing future edges.

- Random negative sampling
- Historical negative sampling: pair of nodes that interacted in the past but not at the query time
- Inductive negative sampling: pair of nodes that will interact in the future but not at the query time

Thanks for your attention