ECML PKDD 2026 Nectar Track · September 9, 2026

Learning and Clustering on Temporal Graphs: **Principles, Primitives, and Pooling**

Nelson Aloysio Reis de Almeida Passos[1,2],
Emanuele Carlini[2], Salvatore Trani[2]

[1] University of Pisa, Department of Computer Science
[2] National Research Council, Institute of Science & Information Technologies

Outline

1. Introduction
Complex systems, temporal networks

2. Principles
Graph operators, detectability thresholds

3. Primitives
Algorithms, accelerated eigensolvers

4. Pooling
Clustering or detecting communities

5. Final Remarks
Takeaways – and where to go from here?

Outline

1. Introduction
Complex systems, temporal networks

2. Principles
Graph operators, detectability thresholds

3. Primitives
Algorithms, accelerated eigensolvers

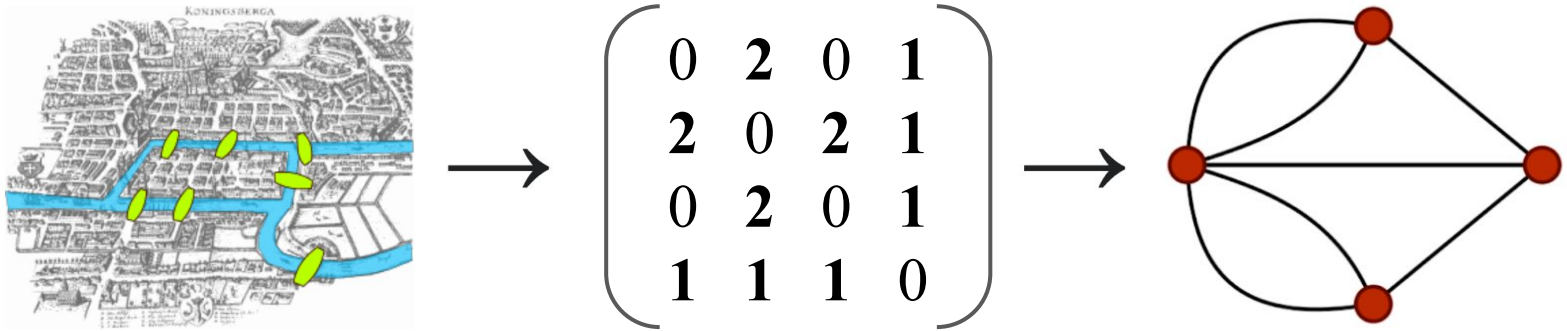
4. Pooling
Clustering or detecting communities

5. Final Remarks
Takeaways – and where to go from here?

1. Introduction

Complex systems, temporal networks

- Many real-world systems can be modeled and studied as **networks** (or **graphs**).
- Elements are **nodes**, connections are **edges** – *often* mathematically constructed as **matrices**.



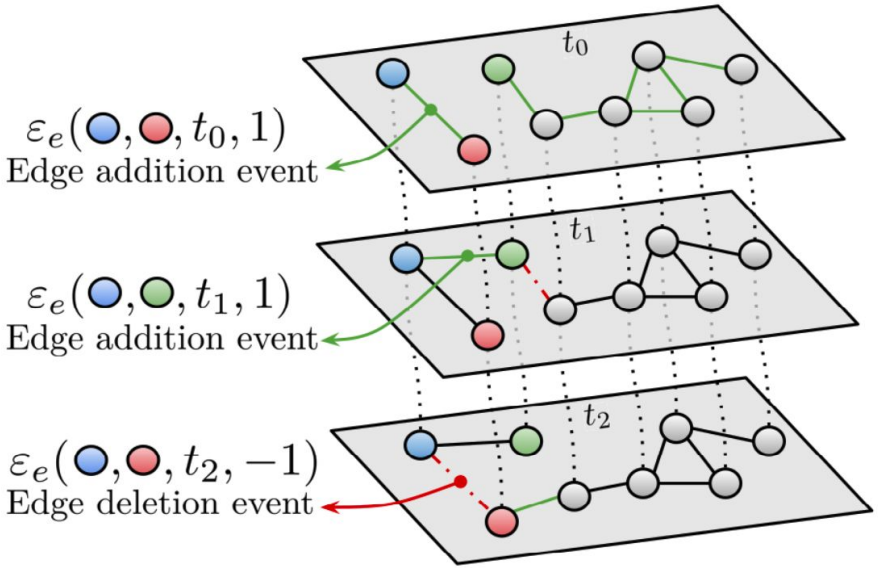
- But the systems we observe are rarely fixed *(static)* in time; instead, they *change* or *evolve.*
- A **temporal network** can be modeled as a **dynamic graph** – with *time-varying* nodes and edges.

Seven Bridges of Königsberg. Wikipedia (2026).

1. Introduction

Complex systems, temporal networks

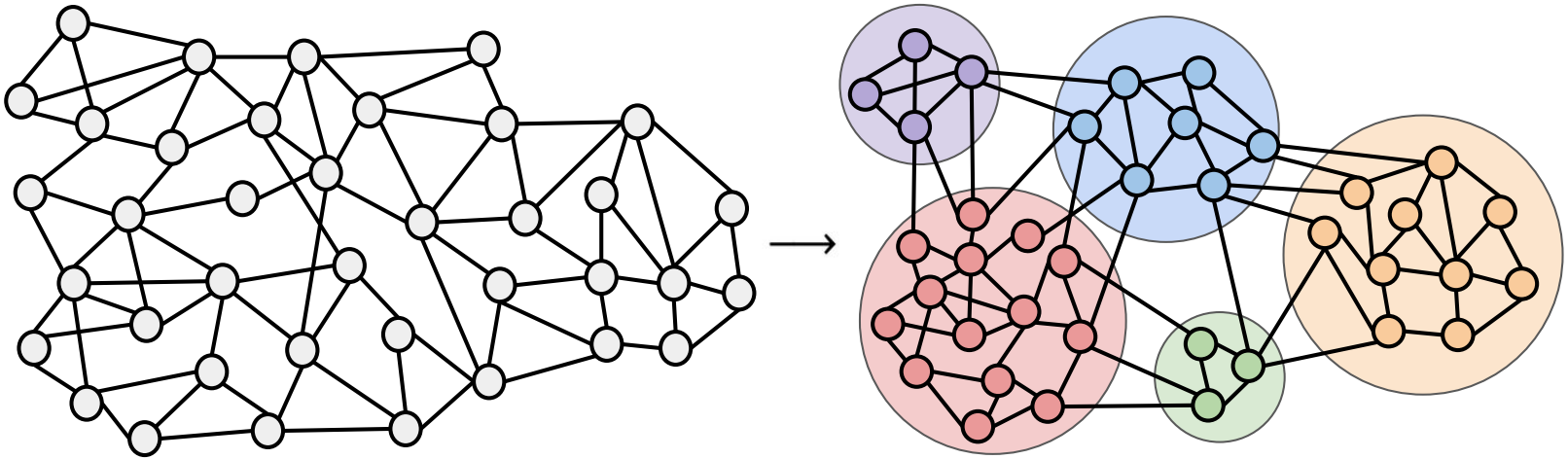
- A dynamic or temporal graph consists of temporal node and edge sets – as *snapshots* or *events*:



1. Introduction

Complex systems, temporal networks

- But complex networks are often *too large* to analyze; we need to 'aggregate' them first; by *clustering (computer science), community detection* (network *science),* or ***pooling*** *(graph ML).*

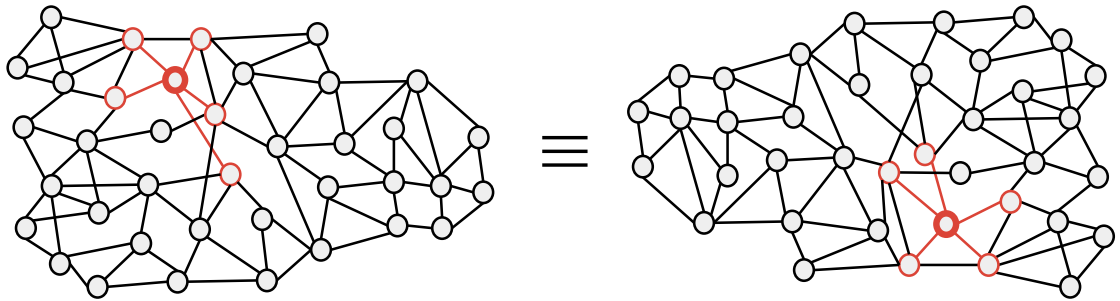


- **But *how*, when the graph changes over time and is associated with additional metadata?**

1. Introduction

Complex systems, temporal networks

- Machine learning models are very good in approximating (*learning* from) lots of data.
- **But** classical ML models can not learn from relational (*non-Euclidean*) datasets directly...



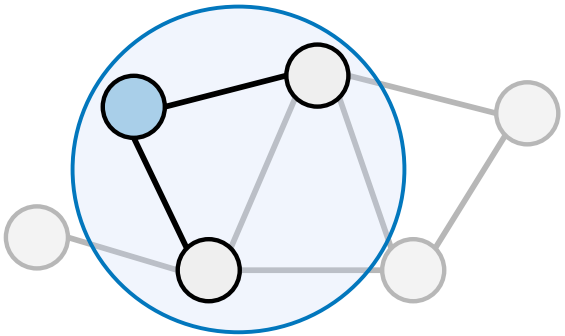
...unless first *mapped* to a real-dimensional (*Euclidean*) space – the sort of space *we live in*.

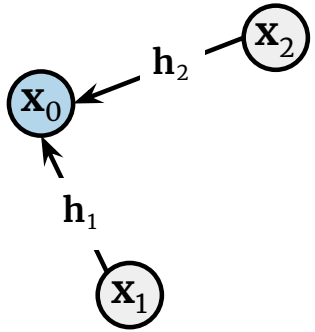
1. Introduction

Complex systems, temporal networks

- **State of the art**: community detection is done with **algorithms** or **neural** models *for graphs*.
  - **Algorithms**: *spectral clustering, heuristic optimization, statistical inference...*
  - **Neural models**: *graph neural networks (GNNs)*, which can learn on graphs by ***message passing.***

Methods from both approaches have been adapted **from static to dynamic** settings.





$$\mathbf{m}_i^{(\ell)} = \bigoplus_{j \in \mathcal{N}(i)} \phi\left(\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)}, \mathbf{e}_{ij}\right)$$

$$\mathbf{h}_i^{(\ell+1)} = \psi\left(\mathbf{h}_i^{(\ell)}, \mathbf{m}_i^{(\ell)}\right)$$

Outline

1. Introduction
Complex systems, temporal networks

2. Principles
Graph operators, detectability thresholds

3. Primitives
Algorithms, accelerated eigensolvers

4. Pooling
Clustering or detecting communities

5. Final Remarks
Takeaways – and where to go from here?

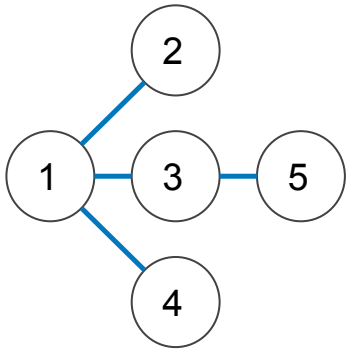
2. Principles

Graph operators, detectability thresholds

- Many methods for node clustering exist – usually split in *descriptive* or *inferential* ones.
- But there is 'no free lunch' for node clustering – as is true for other optimization tasks.
- There is a fundamental limit for detection – the **detectability threshold** , λ being the *community signal*, *c* the *mean community degree*, and *k* the number of communities.
- In SBM graphs, it is known that at $c\lambda^2 = 1$, a *phase transition* (Almeida-Thouless line) occurs, and no algorithm can recover community structure better than chance (Kesten-Stigum bound).
- **But** if there are $k \geq 4$ communities, **those two thresholds diverge**; a new bound appears where algorithms ***may*** recover them better than chance, but is not yet known how to do it ***efficiently***.
- Our work targets it by means of neural approximation of spatio-temporal and metadata signals.

3. Primitives

Algorithms, accelerated eigensolvers



G

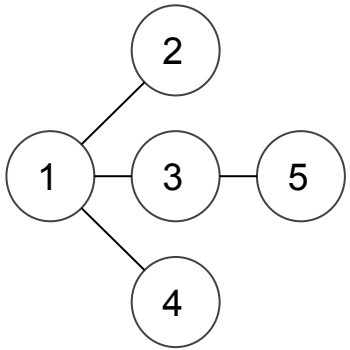
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\rightarrow \mathbf{D} - \mathbf{A} = \mathbf{L} = \begin{bmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

3. Primitives

Algorithms, accelerated eigensolvers



G

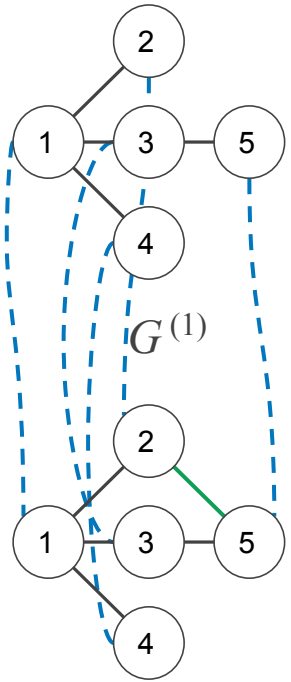
$$\mathbf{L}_{\text{norm}} := \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0 \\ -\dfrac{1}{\sqrt{\deg(v_i)\deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{3} & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & 0 \\ \mathbf{-1} & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{-1} & 0 & \mathbf{2} & 0 & \mathbf{-1} \\ \mathbf{-1} & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{-1} & 0 & \mathbf{1} \end{bmatrix} \rightarrow \mathbf{L}_{\text{norm}} = \begin{bmatrix} 1 & -1/\sqrt{3} & -1/\sqrt{6} & -1/\sqrt{3} & 0 \\ -1/\sqrt{3} & 1 & 0 & 0 & 0 \\ -1/\sqrt{6} & 0 & 1 & 0 & -1/\sqrt{2} \\ -1/\sqrt{3} & 0 & 0 & 1 & 0 \\ 0 & 0 & -1/\sqrt{2} & 0 & 1 \end{bmatrix}$$

3. Primitives

Algorithms, accelerated eigensolvers

— Static edge — Temporal edge ··· Inter-slice coupling



$$\mathbf{A}^{(1)} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A}^{(2)} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}^{\text{supra}} = \begin{pmatrix} \mathbf{A}^{(1)} & \omega\mathbf{I} & \cdots & \mathbf{0} \\ \omega\mathbf{I} & \mathbf{A}^{(2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}^{(T)} \end{pmatrix}$$

$$\rightarrow \mathbf{L}_{\text{norm}} = \ldots$$

2. Principles

Graph operators, detectability thresholds

- The most common method for node clustering across domains is *modularity optimization* [3,4].

- Modularity is a ‘quality’ function of the ‘goodness’ of a graph partitioning, which measures the proportion of edges within communities against the expected amount in a null (random) graph:

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \gamma \frac{d_i^{\text{out}} \, d_j^{\text{in}}}{2m} \right) \delta(c_i, c_j)$$

i, j: nodes *d*: degree
m: total edges in graph
c: community/cluster
γ: resolution param

- It has since been adapted to **temporal** graphs as well, such as the **multislice modularity** variant:

$$Q_{\text{MS}} = \frac{1}{2\mu} \sum_{s,r} \sum_{i,j} \left[\left(A_{ij}^{(s)} - \gamma_s \frac{d_{is}^{\text{out}} \, d_{js}^{\text{in}}}{2m_s} \right) \delta_{sr} + \delta_{ij} \omega_{jsr} \right] \delta\left(c_i^{(s)}, c_j^{(r)} \right)$$

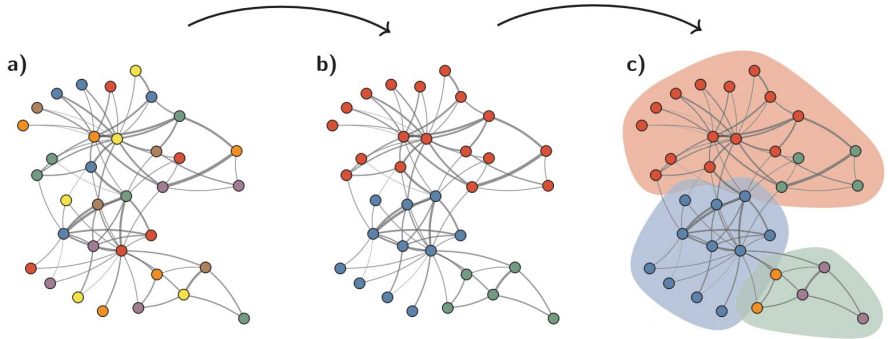
s, r: snapshots (slices)
μ: intra + inter-slice edges
ω: inter-slice weight param

[1] TADC-SBM: a Time-varying, Attributed, Degree-Corrected Stochastic Block Model. Passos, N.A.R.A. et al. (2026). In Proceedings: IEEE ISCC 2025.
[2] Graph Clustering with Graph Neural Networks. Tsitsulin, A. et al. (2023). Journal of Machine Learning Research (JMLR).

2. Principles

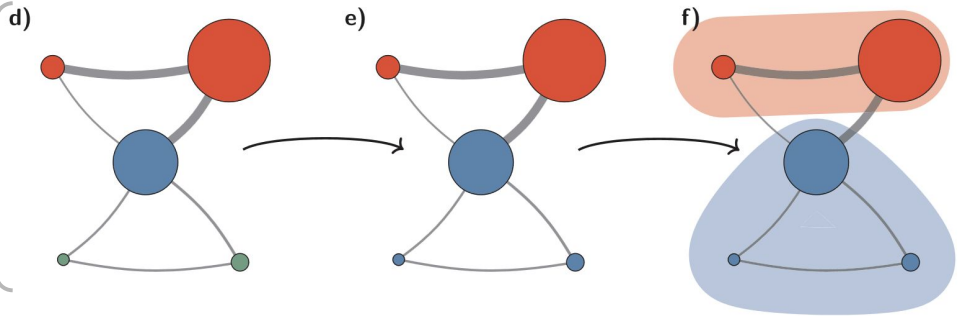
Graph operators, detectability thresholds

- **Leiden** is a refinement over **Louvain** for 2-step greedy optimization of quality functions [5,6].



Level 1 (*node aggregation and refinement*)

Level 2 (*supra-node aggregation and refinement*)



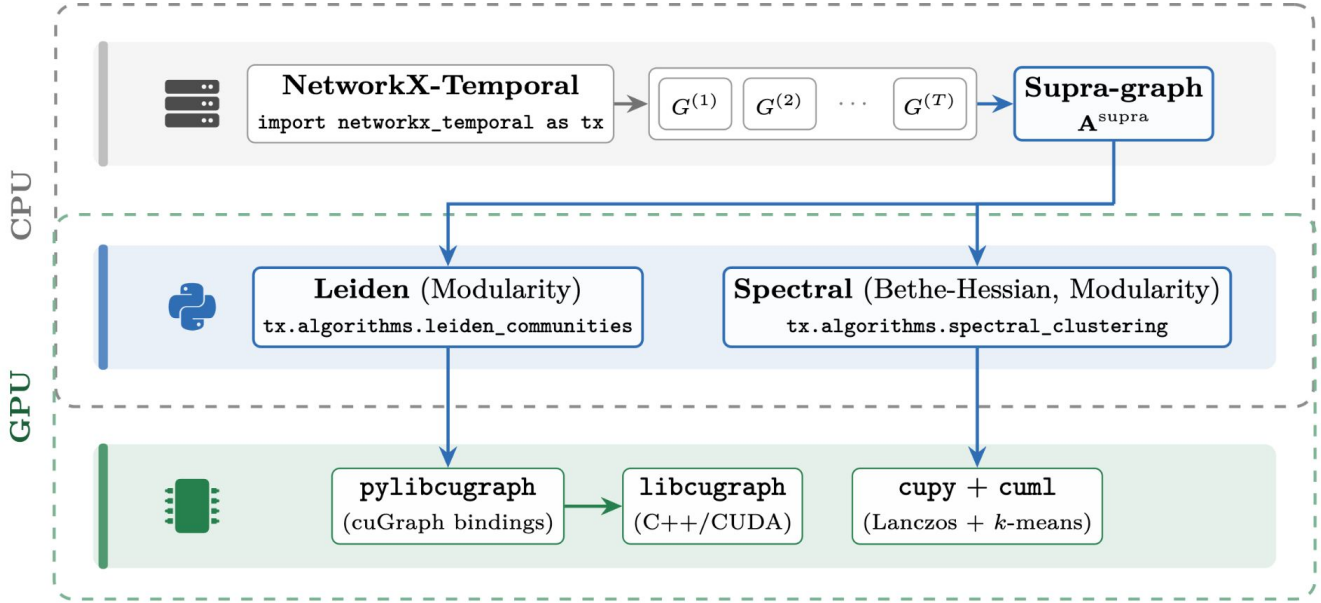
[5] Traag, V. et al. (2019). From Louvain to Leiden: guaranteeing well-connected community. Scientific Reports.

[6] Blondel, V. et al. (2008). Fast unfolding of community in large networks. Journal of Statistical Mechanics: Theory and Experiment.

2. Principles

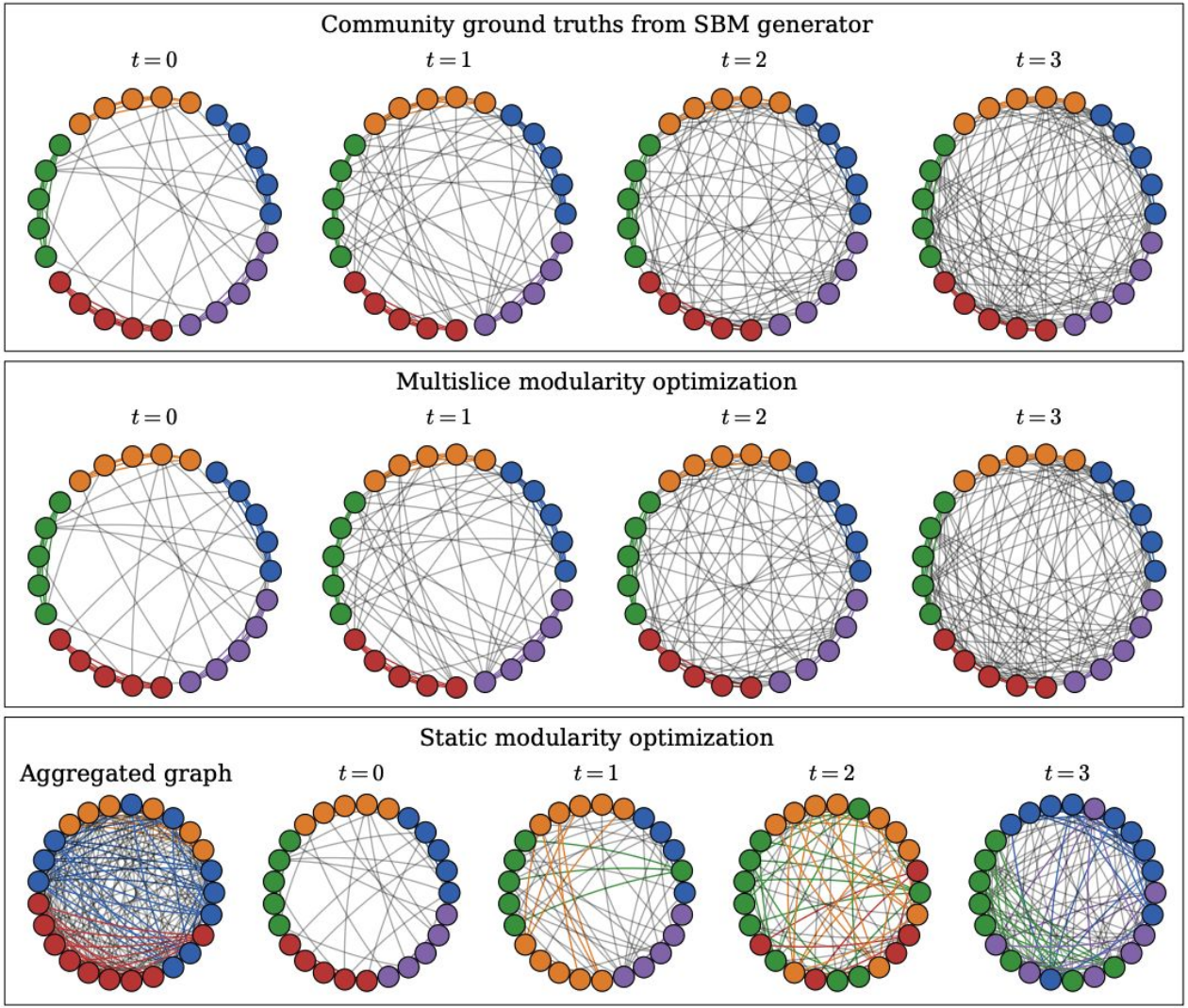
Graph operators, detectability thresholds

- NetworkX-Temporal: implemented on CPU and GPU (NVIDIA CUDA and AMD ROCm) devices, with a single environment variable **`NX_GPU_AUTOCONFIG=1`** enabling acceleration by default.



2. Princ

Graph operators



2. Principles

Graph operators, detectability thresholds

- However, heuristic optimization of functions like modularity is **not optimal**.

  → It brings *no guarantees* that the results (the *detected* communities) are the ‘best’ we may find.

- Another way of using modularity for community detection is employing its **spectral** version.

- **The spectrum of a graph** is obtained by a **matricial operation** (*eigendecomposition*) and preserves its most fundamental properties; so **spectral modularity** is therefore an alternative:

$$\mathbf{M} = \mathbf{A} - \gamma \frac{\mathbf{d}_{\text{out}} \mathbf{d}_{\text{in}}^{\top}}{m}, \quad Q = \frac{1}{2m} \text{Tr}(\mathbf{C}^{\top} \mathbf{M} \mathbf{C})$$

M: modularity matrix
C: community matrix
Tr: trace (sum of diags)

- But modularity is **not needed** for spectral decomposition – other graph operators may be used.

2. Principles

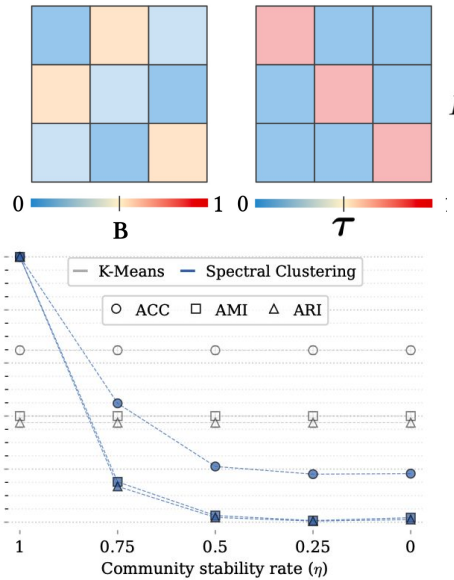
Graph operators, detectability thresholds

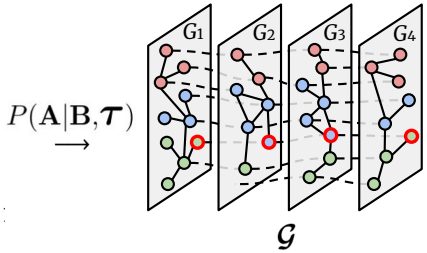
- In another vein, inference with stochastic block models (**SBMs**) also allows *reconstructing* a graph.
- **The *inverse* problem –** given a network, how was it generated? – allows a *clustering* form, i.e., communities, where the probabilities of nodes connecting is based on their (*prior*) affiliations.
- **SBMs** may also be used for *principled* model benchmarking with *attributed temporal graphs*.
- Artificially, we are able to escape two limitations of other model evaluation frameworks:
    - *Real-world data is messy:* **no guarantees** of community alignment with topology/attributes.
    - *Real-world data is scarce:* most datasets come from few application **domains** (e.g., citations).
- Result: controlling spatio-temporal (structural/time) and attribute (metadata) signal alignment.

2. Principles

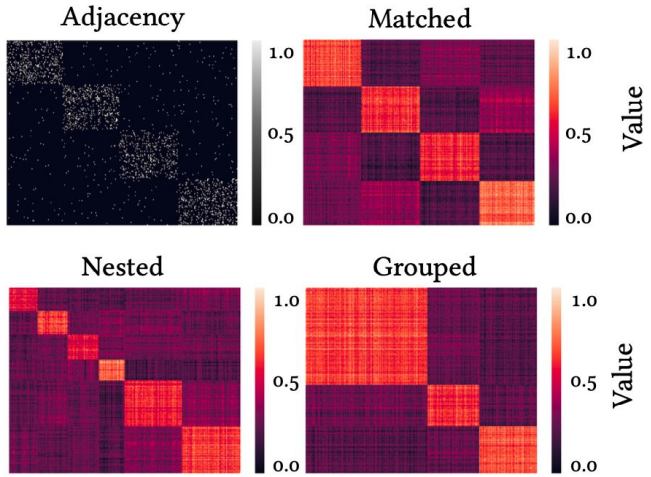
Graph operators, detectability thresholds

Benchmarking with TADC-SBM, a time-varying, attributed, degree-corrected SBM generator, allows to control and reproduce the exact known conditions for evaluating models under well-known settings.





Model	η	Accuracy
k-means	$*$	0.648 ± 0.016
Spectral	1	1.000 ± 0.000
	0.75	0.448 ± 0.000
	0.50	0.210 ± 0.000
	0.25	0.181 ± 0.000
	0	0.183 ± 0.000

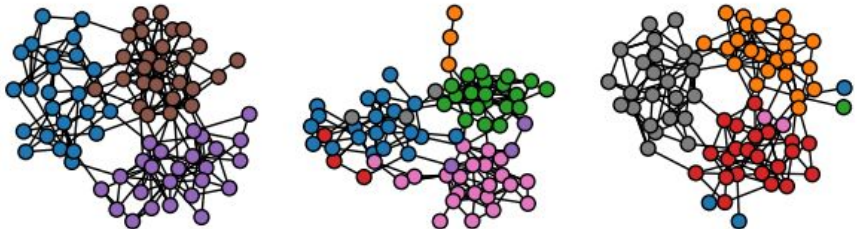


[1] TADC-SBM: a Time-varying, Attributed, Degree-Corrected Stochastic Block Model. Passos, N.A.R.A. et al. (2026). In Proceedings: IEEE ISCC 2025.

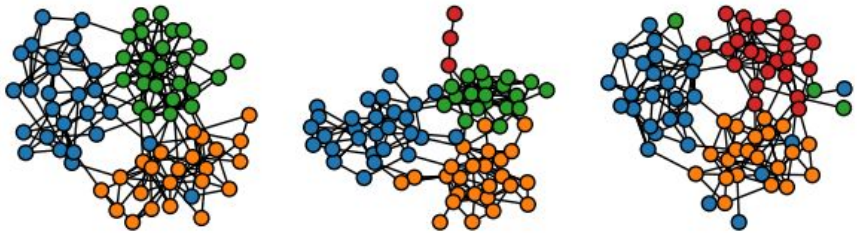
[2] Graph Clustering with Graph Neural Networks. Tsitsulin, A. et al. (2023). Journal of Machine Learning Research (JMLR).

$$Q = \frac{1}{2m} \sum_{(i,j) \in V^2} \left[A_{ij} - \gamma \frac{d_i^{out} \, d_j^{in}}{2m} \right] \delta(c_i, c_j)$$

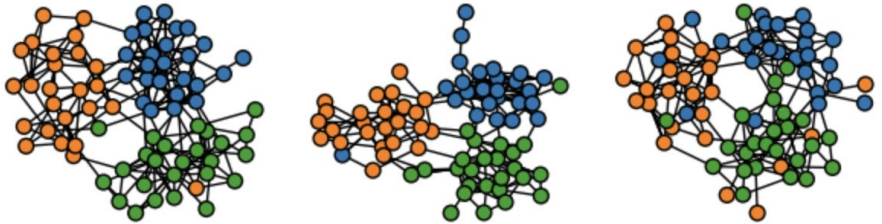
Static Modularity Optimization



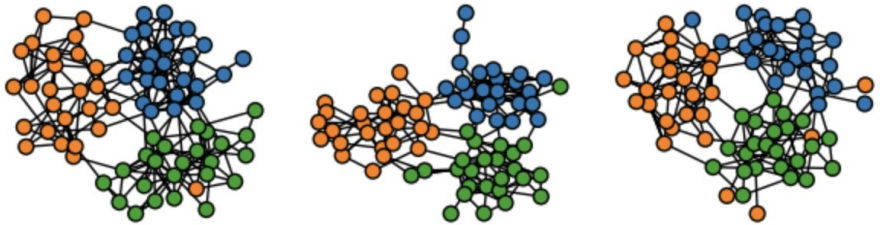
Static Modularity Optimization (Re-Weighted)



Temporal Modularity Optimization

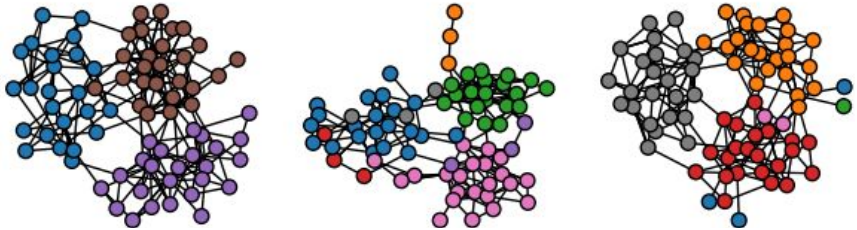


Temporal Modularity Optimization (parallelized)

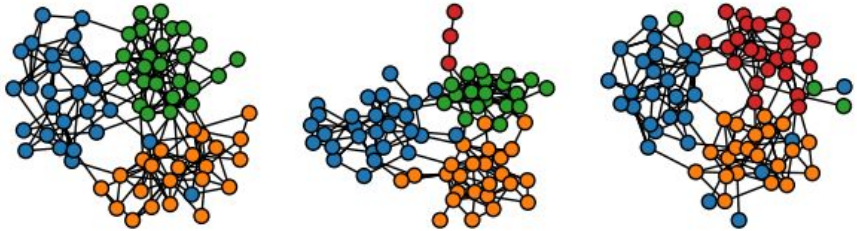


$$Q_{\mathrm{MS}} = \frac{1}{2\mu} \sum_{s,r} \sum_{i,j} \left[\left(A_{ij}^{(s)} - \gamma_s \frac{d_{is}^{\mathrm{out}} \, d_{js}^{\mathrm{in}}}{2m_s} \right) \delta_{sr} + \delta_{ij} \omega_{jsr} \right] \delta\left(c_i^{(s)}, c_j^{(r)} \right)$$

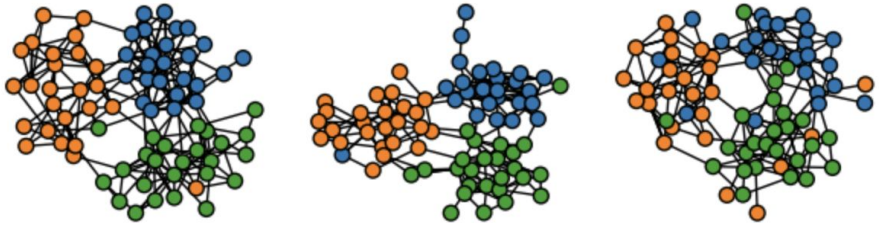
Static Modularity Optimization



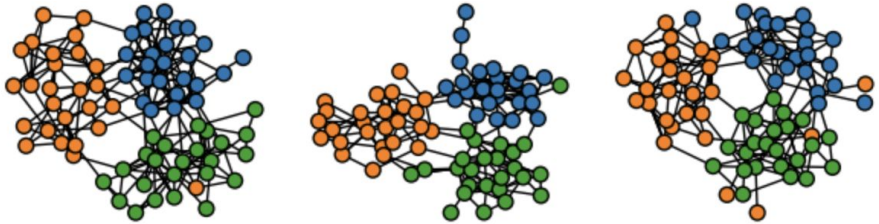
Static Modularity Optimization (Re-Weighted)



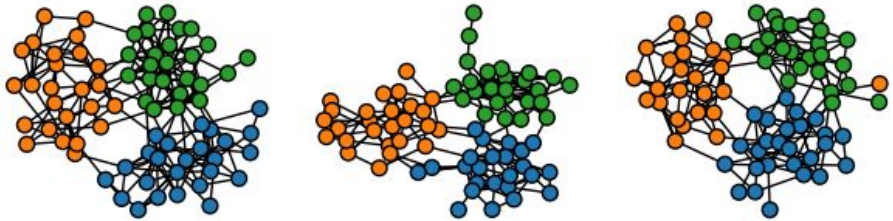
Temporal Modularity Optimization



Temporal Modularity Optimization (parallelized)



Spectral Clustering (Laplacian eigendecomposition + k-means)



Outline

1. Introduction
Complex systems, temporal networks

2. Principles
Graph operators, detectability thresholds

3. Primitives
Algorithms, accelerated eigensolvers

4. Pooling
Clustering or detecting communities

5. Final Remarks
Takeaways – and where to go from here?

3. Primitives

Algorithms, accelerated eigensolvers

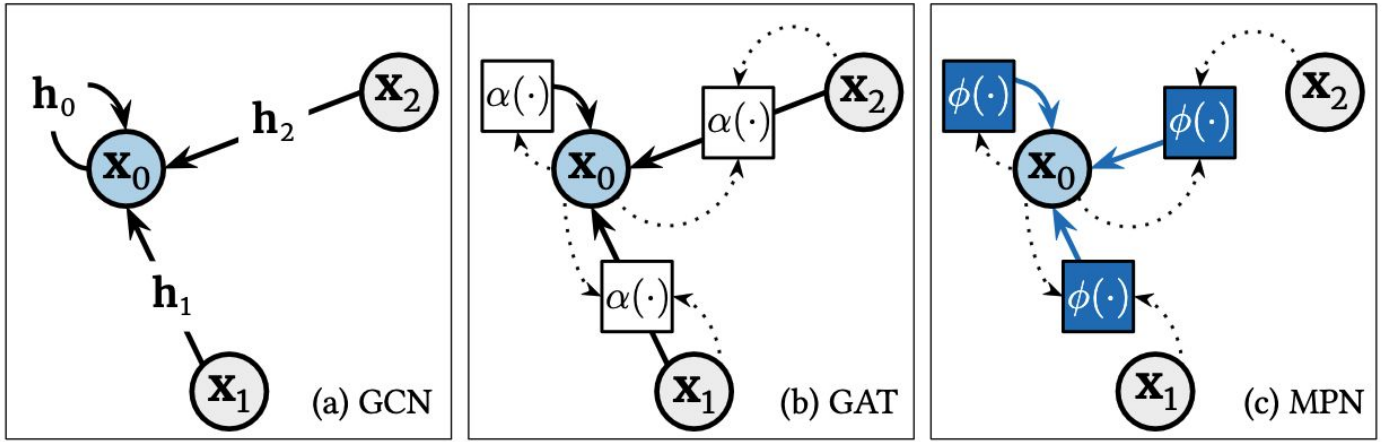
- This is where the state of the art for **community detection** with **spectral** algorithms fall short.
- It is, therefore, currently limited by both **theory** and **practice**:
  - **Theory:** unknown symmetrization without spectrum shift (e.g., the Bethe-Hessian)
  - **Practice:** missing GPU primitives of sparse asymmetric eigensolvers (for the Hashimoto)
- But spectral graph theory also has important implications for ***learning*** *on graphs*.
- Graph convolutional operations (e.g., in GCNs) can be understood as a node filtering operation where the graph signal is represented in a Fourier domain. Such graph *operations* require a graph *operator* – a matrix transformation – which usually considers the **Laplacian**.

3. Primitives

Algorithms, accelerated eigensolvers

- The **Laplacian** operator is the one most graph neural networks perform message passing on: node signal **x** is projected to the spectral domain as $\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$, with **U** the Laplacian eigenbasis.

  So a learned filter acts as $\mathbf{y} = \mathbf{U} g_\theta(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{x}$; graph convolutions become localized polynomial approximations of K-hop neighborhood aggregation without explicit eigendecomposition.



Outline

1. Introduction
Complex systems, temporal networks

2. Principles
Graph operators, detectability thresholds

3. Primitives
Algorithms, accelerated eigensolvers

4. Pooling
Clustering or detecting communities

5. Final Remarks
Takeaways – and where to go from here?

4. Pooling

Clustering or detecting communities

- So a *practical* alternative is employing machine learning methods for *pooling* a graph.
    - **Advantage:** jointly modeling **structure** (topology), **attributes** (metadata), and **time**.
    - Disadvantage: expensive to compute, and *usually* not consistent (that is, not *optimal*).
- It has been empirically shown that considering metadata and time can **improve** detection.
- If a GNN learns a *graph-theoretical objective* that is *consistent* with detectability theory, and if **structure, metadata, and time are aligned**, additional signals → stronger detectability.
- For graphs with high-dimensional attributes, this naturally opens an interesting venue of exploration with machine-learning based methods such as **graph neural networks** (GNNs).

4. Pooling

Clustering or detecting communities

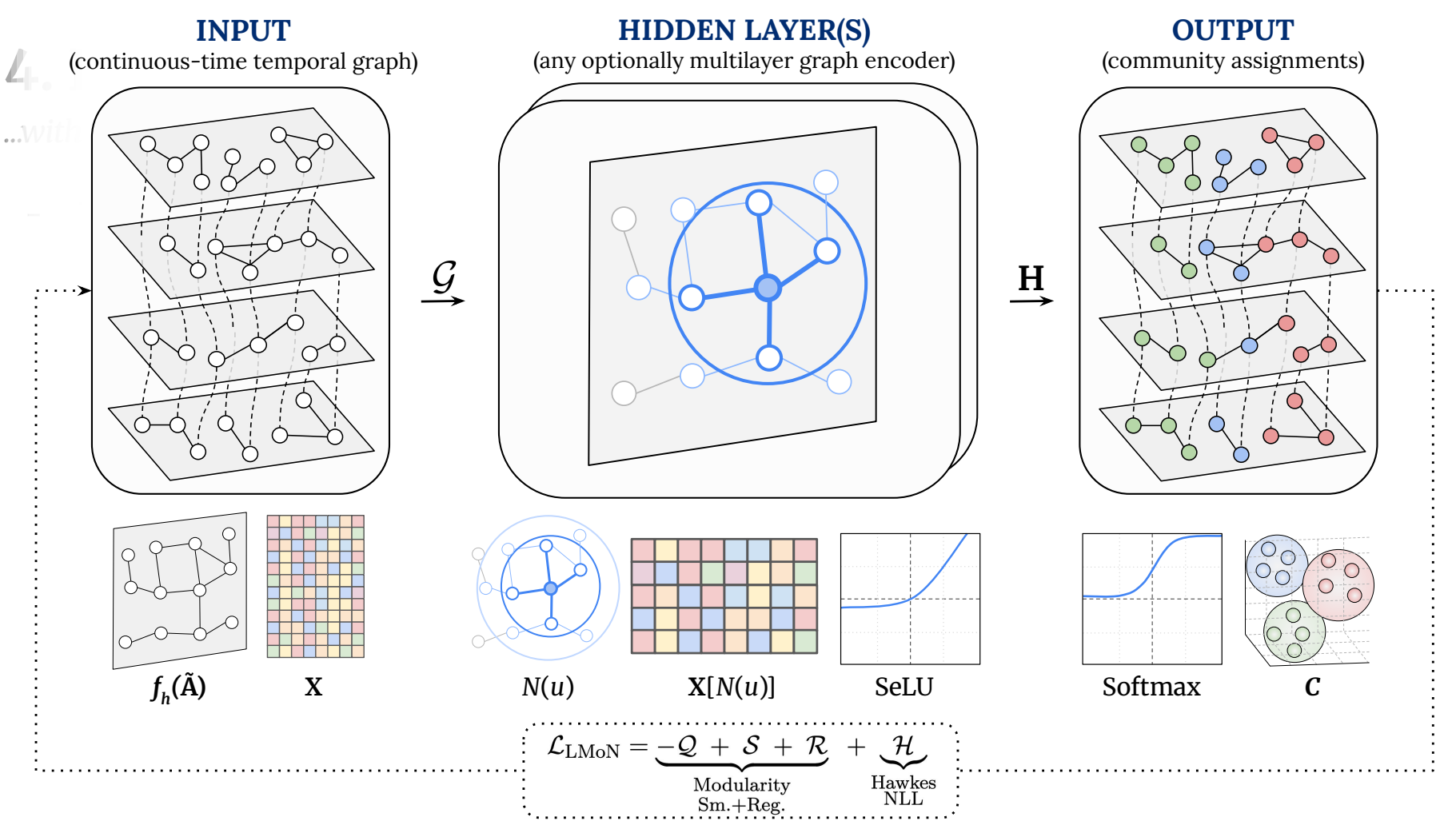
- **However**, few GNNs are *designed* for community detection, especially on *temporal* graphs.
  - *Not graph-theoretic:* they do not learn to approximate a *consistent* community objective.
  - *Not end-to-end differentiable*: additional post-processing steps are required for clustering.
  - *Not interpretable:* temporality is introduced by gating and memory encoder mechanisms.
- So a question appears: *can we employ GNNs for this task in a consistent and efficient way?*
- We introduce 🍋 **LMoN** – an *encoder-agnostic* Longitudinal Modularity optimization Network.

4. Pooling

Clustering or detecting communities

- **LMoN** builds upon a SOTA model (**DMoN**) and maintains its best-in-class scalability $O(d^2n+m)$.
    - *Graph-theoretic:* the objective is built specifically for community detection in graphs.
    - *End-to-end differentiable*: no additional post-processing steps are required for clustering.
    - *Interpretable:* temporality introduced by a *learned* Hawkes function (time-decaying edges).
- Result: we know the adjacency matrix (*re-weighted Laplacian*) and can *visualize it* after train.
- In addition, our architecture is **encoder-agnostic** – advancements in the state of the art for GNNs are transferable to LMoN's architecture by simply *swapping the encoder* (hidden layers).

  The two main components LMoN introduces – Hawkes-Laplacian operator and spectral optimization of longitudinal modularity – are independent of the encoder's expressiveness.


INPUT
(continuous-time temporal graph)
HIDDEN LAYER(S)
(any optionally multilayer graph encoder)
OUTPUT
(community assignments)
$\mathcal{G}$
H
$f_h(\tilde{\mathbf{A}})$
X
$N(u)$
X[$N(u)$]
SeLU
Softmax
C
$\mathcal{L}_{\text{LMoN}} = \underbrace{-\mathcal{Q} + \mathcal{S} + \mathcal{R}}_{\text{Modularity Sm.+Reg.}} + \underbrace{\mathcal{H}}_{\text{Hawkes NLL}}$

4. Pooling

Clustering or detecting communities

- Longitudinal modularity abstract away *from snapshot* slicing, reducing modeling complexity.
- Moreover, if the graph is static, then our model **LMoN** *recovers* **DMoN** as a special case.
- At each epoch, the adjacency matrix of the graph is re-weighted...

$$\hat{A}_{ij}(t) = A_{ij}(t)\left(\mu_{ij} + \alpha \sum_{t_k \in T_{ij}:\, t_k < t} w_k\, \kappa(t - t_k)\right)$$

μ: baseline intensity
α: decay amplitude
κ: Hawkes kernel (e.g., exp.)
w: original edge weight

...so **pairwise interaction strength** is modeled as a function of time, e.g., exponentially decaying.

4. Pooling

Clustering or detecting communities

- But this is **not enough**: *principled* operators (e.g., the Bethe-Hessian) may still outperform GNNs.
- This has been empirically observed in benchmarks with both **real-world** and synthetic graphs.
- Moreover, several real-world systems (e.g., energy grids) are too **critical** to be left for heuristics.
- Considering *structure*, *attributes*, and ***time*** may allow us to improve on SOTA modeling/prediction.
- Graph eigendecomposition does not take into consideration additional attribute metadata — *enters machine learning* — but we know graph operators that are detectability-optimal.
- The question then becomes: *how to introduce **principled pooling primitives** for learning on graphs?*

Dataset	Model	Accuracy ↑	AMI ↑	ARI ↑	Cov. ↑	Cond. ↓
$SBM_{\eta=1.00}$	LMoN	**.894 ± .004**	**.766 ± .007**	**.776 ± .009**	**.471 ± .003**	**.530 ± .003**
	DMoN	.888 ± .015	.755 ± .022	.765 ± .027	.467 ± .010	.535 ± .010
$SBM_{\eta=.75}$	LMoN	**.249 ± .014**	**.056 ± .008**	**.038 ± .006**	.192 ± .002	.810 ± .003
	DMoN	.216 ± .008	.037 ± .004	.024 ± .003	**.193 ± .002**	**.808 ± .002**
$SBM_{\eta=.5}$	LMoN	**.194 ± .005**	**.020 ± .004**	**.012 ± .002**	.183 ± .002	.818 ± .002
	DMoN	.186 ± .005	.013 ± .002	.007 ± .001	**.185 ± .001**	**.816 ± .001**
$SBM_{\eta=.25}$	LMoN	**.177 ± .005**	**.009 ± .002**	**.005 ± .001**	.180 ± .001	.821 ± .001
	DMoN	**.177 ± .002**	.004 ± .002	.003 ± .001	**.182 ± .001**	**.819 ± .001**
$SBM_{\eta=0}$	LMoN	**.182 ± .010**	**.013 ± .007**	**.007 ± .004**	.180 ± .001	.820 ± .001
	DMoN	.169 ± .004	.003 ± .001	.002 ± .001	**.183 ± .001**	**.818 ± .001**

Evaluation on synthetic graphs: performance improves significantly as the temporal signal η gets stronger.

Dataset	Model	Accuracy ↑	AMI ↑	ARI ↑	Cov. ↑	Cond. ↓
$\mathrm{SBM}_{\eta=1.00}$	S(**H**)	**1.000 ± .000**	**1.000 ± .000**	**1.000 ± .000**	**.558 ± .000**	**.442 ± .000**
	LMoN	.894 ± .004	.766 ± .007	.776 ± .009	.471 ± .003	.530 ± .003
	DMoN	.888 ± .015	.755 ± .022	.765 ± .027	.467 ± .010	.535 ± .010
$\mathrm{SBM}_{\eta=.75}$	S(**H**)	**.454 ± .001**	**.158 ± .001**	**.141 ± .001**	**.295 ± .000**	**.705 ± .000**
	LMoN	.249 ± .014	.056 ± .008	.038 ± .006	.192 ± .002	.810 ± .003
	DMoN	.216 ± .008	.037 ± .004	.024 ± .003	.193 ± .002	.808 ± .002
$\mathrm{SBM}_{\eta=.5}$	S(**H**)	**.202 ± .003**	**.026 ± .002**	**.016 ± .001**	**.212 ± .001**	**.788 ± .001**
	LMoN	.194 ± .005	.020 ± .004	.012 ± .002	.183 ± .002	.818 ± .002
	DMoN	.186 ± .005	.013 ± .002	.007 ± .001	.185 ± .001	.816 ± .001
$\mathrm{SBM}_{\eta=.25}$	S(**H**)	**.192 ± .009**	**.015 ± .003**	**.010 ± .002**	**.205 ± .001**	**.796 ± .000**
	LMoN	.177 ± .005	.009 ± .002	.005 ± .001	.180 ± .001	.821 ± .001
	DMoN	.177 ± .002	.004 ± .002	.003 ± .001	.182 ± .001	.819 ± .001
$\mathrm{SBM}_{\eta=0}$	S(**H**)	**.190 ± .011**	**.020 ± .003**	**.012 ± .002**	**.203 ± .001**	**.798 ± .001**
	LMoN	.182 ± .010	.013 ± .007	.007 ± .004	.180 ± .001	.820 ± .001
	DMoN	.169 ± .004	.003 ± .001	.002 ± .001	.183 ± .001	.818 ± .001

Evaluation on synthetic graphs: spectral clustering of the Bethe-Hessian S(**H**) wins over LMoN as η gets stronger.

Outline

1. Introduction
Complex systems, temporal networks

2. Principles
Graph operators, detectability thresholds

3. Primitives
Algorithms, accelerated eigensolvers

4. Pooling
Clustering or detecting communities

5. Final Remarks
Takeaways – and where to go from here?

5. Final Remarks

Takeaways – and where do we go from here?

- **Conclusion:** model performance only **improves** if structure/attribute/time signals align.
- **Limitation:** when those signals do **not** align, then *principled* algorithms may outperform GNNs.
- Two **future directions** for research departing from this thesis:
  - ***Improving message passing:*** the Laplacian operator of GNNs is a bottleneck for clustering.
  - ***Principled pooling primitives:*** enabling applications such as *causal* learning *at scale*.
- Network science is an interdisciplinary field by excellence, and field hybridization contributes to the compendium of techniques it has accumulated – and much like statistics and machine learning in the past, notable advancements may be made possible through transdisciplinarity.

Thank you for your attention!

Learning and Clustering on Temporal Graphs: **Principles, Primitives, and Pooling**

arXiv: 2608.03696 [cs.LG]

nelsonaloysio.github.io

@nelsonaloysio.bsky.social

nelson.reis@phd.unipi.it