Temporal Graph Clustering with Graph Neural Networks

Nelson Aloysio Reis de Almeida Passos NELSON.REISDEALMEIDAPASSOS@ISTI.CNR.IT
National Research Council
Via Giuseppe Moruzzi, 1, 56124 Pisa, Italy

Emanuele Carlini EMANUELE.CARLINI@ISTI.CNR.IT
National Research Council
Via Giuseppe Moruzzi, 1, 56124 Pisa, Italy

Salvatore Trani SALVATORE.TRANI@ISTI.CNR.IT
National Research Council
Via Giuseppe Moruzzi, 1, 56124 Pisa, Italy

Abstract

Clustering temporal graphs with high-dimensional attributes is a challenging task that requires jointly modeling attribute and stricture signals while also accounting for temporal dynamics. Longitudinal modularity is a quality function for evaluating clustering solutions in temporal graphs, but its discrete optimization is NP-hard and existing heuristics are not differentiable, limiting their applicability in end-to-end learning frameworks. In this work, we derive a spectral continuous relaxation of longitudinal modularity that yields an end-to-end differentiable objective for unsupervised node clustering with gradient-based methods. Building on this objective, we introduce a learnable Hawkes-inspired temporal decay embedded in the Laplacian operator of a graph convolutional encoder, allowing it to adapt to event intensity and better capture dynamic community structures. An unpooling layer is (optionally) employed to retrieve discrete cluster assignments from event-level representations, serving for a smoothness term in the objective to encourage persistent structures and temporal consistency in cluster assignments. Across real-world and synthetic graph benchmarks, the proposed model consistently outperforms established baselines on graph-theoretic and clustering metrics, with ablation studies confirming the independent contributions of both the objective and convolution strategy in enhancing performance.

Keywords: graph neural networks, temporal graphs, community detection, modularity

1 Introduction

Measures of cluster quality for graphs commonly assume static topologies: nodes (entities) are connected by edges (relationships) as fixed structures, requiring notions such as persistence, recurrence, and transience of connections to be imposed post-hoc by snapshot discretization over arbitrarily chosen time windows. Among many, modularity remains one of the most widely adopted and well-researched metrics for evaluating or describing a particular graph partitioning, yet reliance on a time-agnostic null model — the configuration model of Chung and Lu (2002) — limits its applicability to temporal settings where community structures may undergo through changes based on, e.g., evolutionary dynamics, temporal heterogeneity, and varying activity levels. These are often observed in real-world complex systems and may be crucial for tasks such as link prediction and graph classification, with applications ranging from recommendation systems and social network analysis to financial transaction monitoring and epidemiological modeling. This limitation is par-

ticularly problematic for systems where evolutionary dynamics are a crucial aspect of the task or system being modeled, such as spreading processes in social and epidemiological networks, financial transactions and fraud detection, among others.

While extensions of modularity have been introduced for temporal graphs, such as in Mucha et al. (2010) for discrete and Brabant et al. (2025) for continuous-time settings, their optimization typically relies on heuristic or approximate strategies that disregard rich attribute information and overlapping (mixed-membership) community structures. Moreover, their integration into modern representation learning pipelines in an end-to-end differentiable manner remains an open problem, although the framework presented in Tsitsulin et al. (2024) for static graphs provides a blueprint that allows for improvements, as discussed by Liu et al. (2024b) and Salehi and Giannacopoulos (2025). Typically, GNN-based models for node classification or clustering in temporal graphs commonly rely instead on proxy objectives (e.g., reconstruction error) and memory-based mechanisms (such as recurrent units or long-short term memory) to capture temporal dependencies, without explicitly modeling community structure or temporal dynamics in a way that is directly interpretable and optimized for the task at hand. Their application for temporal node clustering is also considerably underexplored, as pointed in Longa et al. (2023) — although research in this direction has gained traction, problems with data scarcity, suitable benchmarks, and computational constraints are still under active discussion, as outlined in Liu et al. (2024a). Beyond hindering a model's ability to distinguish long-lasting from short-lived connectivity patterns, conflating persistent structure with transient interactions and resulting in 'topochrone' disconnections, i.e., groups of nodes that are structurally cohesive, but temporally incoherent (disjoint in time).

To resolve this, we derive a spectral relaxation of longitudinal modularity, a variant of the metric for continuous-time temporal graphs that avoids snapshot discretization and incentivizes contiguous community assignments over time. The formulation naturally recovers static modularity in the absence of temporal coupling and allows for a differentiable form suitable for representation learning. Building on this foundation, we instantiate a neural architecture for unsupervised node clustering that directly optimizes the proposed objective, allowing a model to exploit time-aware structural (topological) and attribute (feature) signals. To allow modeling interaction strength, we introduce a Hawkes-inspired temporal decay embedded in the Laplacian operator of a modified graph convolutional layer, enabling the encoder to distinguish persistent structure from transient events. An unpooling layer is optionally employed to retrieve discrete cluster assignments from event-level representations, serving as a smoothness term in the objective to encourage persistent structures and temporal consistency in cluster assignments. Our contributions are summarized as:

- We derive a continuous spectral relaxation of longitudinal modularity, enabling its use as an objective for end-to-end differentiable community detection on temporal graphs.

- We introduce a trainable Hawkes-inspired temporal decay in the Laplacian operator of an encoder (e.g., a GCN), allowing interaction strengths to be parameterized from data.

- We incorporate an event-based temporal pooling mechanism that supports evolving and overlapping (mixed-membership) communities, with optional time discretization.

- We instantiate these components in a neural framework that directly optimizes the proposed objective for temporal community detection with gradient-based optimization.

Our contributions establish a principled and extensible foundation for differentiable community detection on temporal graphs, of particular relevance for networks where high-dimensional attribute features are available and evolutionary dynamics may be observed, such as in social, financial, and epidemiological networks. Empirical results demonstrate that the proposed strategy yields significant improvements over established baselines in several real-world and synthetic benchmark datasets, with ablation studies confirming the independent contributions of the objective and encoder operator. We release the model implementation and code to reproduce our results[1].

(*) this to be confirmed: dynamic pooling performance + model ablation study

2 Related Work

Optimization of modularity and its variants is commonly approached with heuristic strategies, such as greedy algorithms (e.g., Louvain **?**, Leiden **?**), spectral methods, and more recently, machine learning models. Perozzi (2021) proposed a spectral relaxation of modularity for static graphs, which is optimized by a GNN-based encoder. Several extensions of this framework have been proposed, such as in Liu et al. (2024b) for contrastive learning and Salehi and Giannacopoulos (2025) for diversity-aware learning, among others. In the temporal case, extensions of modularity have been introduced, such as in Mucha et al. (2010) for discrete and Brabant et al. (2025) for continuous-time settings, but their optimization typically relies on heuristic or approximate strategies that disregard rich attribute information and overlapping (mixed-membership) community structures.

Graph neural networks (GNNs) have been successfully employed for downstream tasks including node clustering, link prediction, and graph classification, and are currently among the state of the art for modeling networks where high-dimensional node or edge feature vectors are present and must be considered for prediction purposes. Instead, recently proposed models for temporal graph clustering rely on proxy objectives (e.g., reconstruction error, as Liu et al. (2024a)) and memory-based mechanisms (such as recurrent units or long-short term memory, as Pareja et al. (2019)) to capture temporal dependencies, without explicitly modeling community structure or temporal dynamics in a way that is directly interpretable. GNN application for temporal node clustering is considerably underexplored, as pointed in Longa et al. (2023) — although research in this direction has gained traction, problems with data scarcity, suitable benchmarks, and computational constraints remain.

Applicability is arguebly focused on applications where high-dimensional node features are available and may be crucial for uncovering meaningful community structures, such as in recommendation systems, social networks, and financial networks, among others. Generative models for benchmarks in controlled experimental settings usually employ stochastic block models (SBMs) and extensions accounting for temporal dynamics and contextual features, as discussed in the static setting in Perozzi (2021), and serve as an alternative to real-world datasets, which are often scarce and may incur problems with data quality and lack of ground truth labels suitable for the evaluation setting, as noted by Peel et al. (2017).

1. To be updated with link to public repository.

Review this section.

3 Preliminaries

In traditional community detection, communities are often identified based on topological features of the graph, where the relationships between nodes are encoded in non-Euclidean spaces. Methods like spectral clustering or modularity optimization leverage the graph structure to group nodes together based on connectivity patterns, while other approaches rely on probabilistic models for inference. Some of these strategies directly account for associated node and edge features in the graph (e.g., user characteristics in social networks, gene expression levels in biological networks, or transaction details in financial networks). In traditional community detection, communities are often identified based on topological features of the graph, where the relationships between nodes are encoded in non-Euclidean spaces. Methods like spectral clustering or modularity optimization leverage the graph structure to group nodes together based on connectivity patterns, while other approaches rely on probabilistic models for inference. Some of these strategies directly account for associated node and edge features in the graph (e.g., user characteristics in social networks, gene expression levels in biological networks, or transaction details in financial networks).

In contrast, 'neural communities' are to be defined within the context of a real-valued latent space. Rather than relying solely on non-Euclidean graph topology, graph-theoretic objectives defined in the latent space take node representations as points and communities as clusters of these points. This allows advanced techniques such as convolutions on node feature vectors to combine graph structure with semantic information embedded in high-dimensional real data, producing node representations (embeddings) learned from data and optimized for downstream tasks, such as clustering, link prediction, or graph classification. The optimization process is guided by the geometry of the latent space, which can better capture complex, high-dimensional dependencies between nodes and their features, allowing for the discovery of communities that are not only structurally cohesive but also contextually correlated. For example, in recommendation systems, communities could represent groups of users or items that exhibit similar preferences, enabling personalized recommendations based on latent similarities rather than explicit connectivity alone; in social networks, groups of individuals with similar interests or behaviors, even if not directly connected; and in financial networks, clusters of transactions or entities that share similar risk profiles or market behaviors, allowing for more effective fraud detection and risk management, among others. In such settings, high-dimensional node features are often available and may be crucial for uncovering meaningful community structures. This formulation therefore allows for greater flexibility in handling real-world graph-structured data, accounting for both topological and semantic information, and may be extended to temporal dynamics too.

In the context of modularity maximization, for instance, the objective remains to maximize modularity, but the optimization process is guided by the geometry of the latent space, which can better capture complex, high-dimensional dependencies between nodes and their features, allowing for the discovery of communities that are not only structurally cohesive but also semantically meaningful. This formulation therefore allows for greater flexibility in handling real-world graph-structured data, accounting for both topological and semantic information, and may be extended to temporal dynamics too, as shown next.

3.1 Clustering communities in graphs

A common definition of communities in graphs relies on an assortative or homophilous characterization, where groups of nodes are densely connected internally (many within-community edges) and sparsely connected externally (less between-community edges) — allowing for the definition of 'quality' metrics that measure the 'goodness' of its partitions.

In essence, the problem of community detection in graphs is equivalent to that of clustering, but defined on the graph's non-Euclidean domain — finding a mapping $f : \mathrm{X} \to \mathrm{Y}$ where X is the set of nodes in the graph and Y is the set of communities. Among the several methods for uncovering cluster structures, such as spectral techniques and matrix factorization, probabilistic and information-theoretic models, and more recently, machine learning models, a common strategy for evaluating the 'quality' of a partitioning relies on some graph-theoretic objective, such as modularity, conductance, or cut ratio.

Let m be the number of edges in the graph, $(i, j) \in V^2$ denote node pairs, $\mathbf{A}$ the adjacency matrix, d the node in- and out-degrees, c their communities, and $\gamma = 1$ (default) the resolution parameter. Formally, the modularity of a graph partitioning $\mathcal{C}$ is defined as

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

where $\mathbf{A}$ may optionally be weighted, and $\delta(c_i, c_j) = 1$ if nodes i and j belong to the same community, otherwise 0. The optimal partitioning $\mathcal{C}$ is therefore that which yields the highest value of Q. This definition can be reduced to a summation over communities,

$$Q = \sum_{c \in \mathcal{C}} \left[\frac{L_c}{m} - \gamma \left(\frac{d_c}{2m} \right)^2 \right], \tag{2}$$

where L_c is the number of within-community edges and d_c is the sum of degrees of nodes in community c. Setting γ to larger values allow to circumvent the resolution limit, where a maximum of $\sqrt{2m}$ communities may be recovered by modularity maximization approaches. As Q scales with m and the null model couples all node pairs, for any fixed value of γ, small communities may still be merged into larger ones if it increases the global optimum (and no modularity-improving move exists under the regime used). It may also miss on separating communities when it does not increase the global modularity objective, but could later be split into smaller communities that d, which algorithms like Leiden author (2024) partially address this shortcoming by iterative refinement of communities as supernodes.

The metric therefore measures the fraction of edges within communities against the expected fraction in a null graph, that is, a random graph with the same node degree distribution as the observed network. Note that no statistical test is provided against the adopted null model and spurious community structures may still be detected in random graphs **??**. compared to generative approaches such as stochastic block models, where communities are inferred by a model minimizing description length (allowing uncertainty to be quantified and graph reconstruction by sampling from a posterior **?**), heuristics-based optimization provide fast, scalable approximations to the NP-hard problem of modularity maximization, favoring computational efficiency to large-scale networks over exact solutions.

3.1.1 Modularity on graph spectra

Spectral methods cluster vertices via the eigendecomposition of graph operators such as the adjacency matrix or, more typically, the normalized Laplacian $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, where $\mathbf{L} = \mathbf{D} - \mathbf{A}$, $\mathbf{D}$ is the degree matrix, and $\mathbf{A}$ is the adjacency matrix. Let $\mathbf{U} \in \mathbb{R}^{|V| \times k}$ denote the matrix of eigenvectors corresponding to the k smallest eigenvalues of $\tilde{\mathbf{L}}$. The rows of $\mathbf{U}$ yield a k-dimensional representation of the graph's topological structure, and Minimizing the normalized cut can be relaxed to this spectral problem, with discrete partitions typically obtained via a subsequent clustering step (e.g., k-means). The detectability of community structure is also characterized in spectral terms: above a threshold, informative eigenvalues separate from the spectral bulk, whereas below it no such separation occurs and no algorithms performs better than chance in the asymptotic limit **?**, unless additional information is available, such as (node/edge) attribute atributes or evolutionary temporal dynamics.

Modularity allows for a spectral formulation where the objective is expressed in matricial form, as shown by Newman (2006). Under the configuration model, the modularity matrix $\mathbf{B}$ is similarly defined as in Equation 1 by the expected number of edges between each pair,

$$\mathbf{B} = \mathbf{A} - \gamma \frac{\mathbf{d}_{\text{out}} \mathbf{d}_{\text{in}}^{\top}}{2m}, \tag{3}$$

where $\mathbf{A}$ is the adjacency matrix, $\mathbf{d}_{\text{in}}$ and $\mathbf{d}_{\text{out}}$ are degree vectors, and γ is the resolution. In this case, eigenvectors are obtained from $\mathbf{B}$ instead of $\mathbf{L}$, and the leading eigenvectors of $\mathbf{B}$ are used to cluster nodes and the trivial solution (all nodes collapse to a single community) correspond to an eigenvector of all ones with an eigenvalue of zero **?**.

Let $\mathbf{C} \in \mathbb{R}^{n \times k}$ be a community assignment matrix, with possibly overlapping (mixed) membership assignments. The modularity value of a graph then corresponds to

$$Q = \frac{1}{2m} \text{Tr}(\mathbf{C}^{\top} \mathbf{B} \mathbf{C}), \tag{4}$$

that is, the trace (sum of diagonal elements) of the matrix product $\mathbf{C}^{\top}\mathbf{B}\mathbf{C}$, scaled by the total number of edges in the graph (doubled in the case of undirected graphs). Note that this formulation naturally allows for mixed memberships (overlapping or 'soft' clusters), where nodes may belong to many groups with different weights and $C_{i,c}$ indicates the strength of node i membership to community c (e.g., a probability or a fuzzy assignment).

Spectral modularity methods therefore approximate the discrete optimization problem by projecting nodes onto the leading eigenvectors of $\mathbf{B}$, yielding a low-dimensional representation that is subsequently clustered. The trivial solution, where all vertices 'collapse' in one group, correspond to an eigenvector $[1, 1, 1, ...]$ with an eigenvalue of zero. Meanwhile, recursive application to induced subgraphs enables multi-layer spectral modularity optimization, such as in the temporal case, among extensions to continuous-time domains.

3.1.2 Extensions to temporal graphs

Temporal variants of modularity extend it to incorporate evolutionary dynamics, allowing for both discrete (snapshot-based) and continuous (event-based) graph representations.

For instance, multislice modularity **?** considers temporal dynamics by adding inter-layer edges connecting corresponding nodes across time slices. Both intra-layer and inter-layer

connections are accounted for, with a coupling parameter controlling their weight, allowing it to capture temporal dependencies and community evolution across snapshots. Formally,

$$Q_{\text{MS}} = \frac{1}{2\mu} \sum_{G \in \mathcal{G}} \left[\sum_{i,j} \left(A_{ij} - \gamma \frac{d_i^{\text{out}} \, d_j^{\text{in}}}{2m} \delta(c_i, c_j) \right) + \sum_{i \in V} \omega \, \delta(c_i^{(t)}, c_i^{(t+1)}) \right], \tag{5}$$

where μ is the number of intra- and inter-layer edges in the temporal graph, $G \in \mathcal{G}$ are graph snapshots (slices/layers), t is the time or snapshot index, and additional terms γ and ω are the resolution and inter-layer coupling parameters, with both defaulted to 1.

For completeness, multislice modularity may too be expressed in matricial form as

$$Q_{\text{MS}} = \frac{1}{2\mu} \text{Tr}(\mathbf{C}^\top \hat{\mathbf{B}} \mathbf{C}), \tag{6}$$

where $\hat{\mathbf{B}}$ is the modularity matrix obtained from a supra-adjacency matrix $\hat{\mathbf{A}}$ whose sum matches the number μ of edges in the graph, with inter-layer edges weighted by ω, that is,

$$\hat{\mathbf{B}} = \hat{\mathbf{A}} - \gamma \frac{\mathbf{d}_{\text{out}} \mathbf{d}_{\text{in}}^\top}{2\mu}, \tag{7}$$

and $\hat{\mathbf{A}}$ is obtained either by stacking adjacency matrices for each graph snapshot, or element-wise sum so edge weights represent total interactions, with self-loops $\hat{A}_{ii} = \sum_{i \in C} (\omega = 1)$.

This definition allows for spectral optimization of a temporal graph and time-varying, mixed-membership community assignments depending on the adopted graph representation. While sparse-matrix implementations allow efficiently building $\hat{\mathbf{A}}$, which otherwise would quickly become prohibitive in case of large graphs with several snapshots, the need for temporal discretization introduces additional complexity for e.g. networks with continuous-time domains and varying activity levels, which are often approached heuristically.

3.1.3 Longitudinal modularity

A more recent extension of the metric proposes instead to compute time-aware modularity following a 'longitudinal' paradigm **?**. The need to select an arbitrary window length to bin or slice data into discrete snapshots is removed in place of a continuous-time temporal graph representation, treating edges as time-stamped events, or a 'link stream'. The null model in Eq.1 is adapted to include temporal dynamics based on membership expectation to incentivize temporally stable communities, and an additional parameter is added to penalize solutions with frequent node transitions and prioritize 'smooth' community assignments.

With mean membership link expectation, which assumes the expected number of links between two nodes is proportional to the geometric mean of their lifetime within communities (reflecting the intuition that nodes active for longer periods within a community are more likely to interact), the equation for longitudinal modularity may be written as

$$Q_{\text{L}} = \frac{1}{2m} \sum_{\mathcal{C} \in \mathcal{G}} \left[\sum_{i,j} A_{ij} \, \delta(c_i, c_j) - \gamma \frac{d_i^{out} \, d_j^{in}}{2m} \frac{\sqrt{|\text{T}_{i \in \mathcal{C}}||\text{T}_{j \in \mathcal{C}}|}}{|\text{T}|} \right] - \frac{\omega}{2m} \sum_{i \in V} \eta_i, \tag{8}$$

where $C \in \mathcal{G}$ are all communities in the graph, $|\mathrm{T}|$ is its total time span, $|\mathrm{T}_{i \in V_C}|$ is the lifetime of all nodes i within community C, $\omega = 1$ (default) is a scaling parameter, and η_i is the community switch count of a node, defined as the number of times a node transitions between communities across time, that is, $\eta_i = \sum_{t_i \in \mathrm{T}_i} 1 \,\text{if}\, (t_{i+1} - t_i > \Delta)$, where $\Delta = 1$ (default) is a tunable parameter representing the minimum interval that configures a switch.

The null model in Equation 8 therefore accounts for temporal dynamics by modeling interaction expectations based on node presence within communities, allowing to distinguish persistent from transient affiliations, and incentivizing contiguous community assignments over time by penalizing frequent node transitions. In case of fixed membership assignments (*static* community detection on *dynamic* graphs), the penalty becomes a constant depending only on the number of disjoint time intervals in which a node is present, and may be ignored for optimization. This formulation, however, is not directly applicable to continuous-time domains or overlapping communities, where pairwise interactions are not necessarily observed at discrete interval and the notion of 'switch' is less well-defined. The null model in Equation 8 therefore accounts for temporal dynamics by modeling interaction expectations based on node presence within communities, allowing to distinguish persistent from transient affiliations, and incentivizing contiguous community assignments over time by penalizing frequent node transitions. In case of fixed membership assignments (*static* community detection on *dynamic* graphs), the penalty becomes a constant depending only on the number of disjoint time intervals in which a node is present, and may be ignored for optimization. This formulation, however, is not directly applicable to continuous-time domains or overlapping communities, where pairwise interactions are not necessarily observed at discrete interval and the notion of 'switch' is less well-defined.

Rewriting in matricial form, a spectral formulation of longitudinal modularity may be given by a similar expression to the static case (Eq. 4) with an additional penalty term,

$$Q_{\mathrm{L}} = \frac{1}{2m}\mathrm{Tr}(\mathbf{C}^\top \tilde{\mathbf{B}}\mathbf{C}) - \frac{\omega}{2m}\sum_{i \in V}\sum_{t \in T_i} \left\| \mathbf{c}_i^{(t+1)} - \mathbf{c}_i^{(t)} \right\|_2^2, \tag{9}$$

where $\|\cdot\|_2$ is the Euclidean (L2) norm, $\tilde{\mathbf{B}}$ is the (temporal) modularity matrix, and other terms follow previous definitions. Note that the second term corresponds to the sum of squared differences between consecutive community assignments for each node across time, which has the same bounds as the original formulation (see Section 4 for a proof).

In this case, temporal dynamics are incorporated by defining $\tilde{\mathbf{B}} = \tilde{\mathbf{A}} - \boldsymbol{\Theta}$, where $\tilde{\mathbf{A}}$ is the (weighted, directed) adjacency matrix and $\boldsymbol{\Theta}$ is a null model matrix accounting for temporal node activity. With the mean-membership link expectation model as the null graph (Eq. 8), we can derive a spectral formulation of the longitudinal modularity matrix

$$\tilde{\mathbf{B}} = \tilde{\mathbf{A}} - \gamma \frac{(\mathbf{d}_{\mathrm{out}} \odot \boldsymbol{\theta})(\mathbf{d}_{\mathrm{in}} \odot \boldsymbol{\theta})^\top}{2m}, \tag{10}$$

where $\mathbf{d}$ is the degree vector computed from the adjacency matrix, and m is the total edge weight (or total intensity mass in the temporal case), i.e., $m = \frac{1}{2}\sum_{(i,j)} A_{ij}$.

The temporal presence vector $\boldsymbol{\theta}$ is computed from node lifetimes using the observed event timestamps, with $\theta_i = (t_i - t_i')/(t - t^0 + \varepsilon)^{1/2}$, so transient nodes receive smaller values than persistent ones. The normalized assignment matrix $\bar{\mathbf{C}}$ is used to stabilize the

pooled communities by dividing each assignment column by its cluster size, matching the implementation used before the pooled graph and spectral loss are evaluated. In effect, this allows efficient computation of community structure in the pooled latent space and optimization with gradient methods, enabling the detection of temporal communities without snapshot discretization while remaining compatible with overlapping assignments.

3.2 Modeling temporal adjacency with Hawkes processes

Applying a temporal weighting mechanism on the edges before normalization allows to model interaction strength as a function of time, where the influence of past interactions decays over time, and more recent interactions have a stronger influence on the current state of the graph or an interaction. One key challenge in applying Hawkes processes to real-world applications is efficiently learning the parameters of the interaction function from data, with traditional methods often built on non-differentiable optimization techniques or inference methods such as maximum likelihood estimation (MLE), with more recent work on developing end-to-end learnable neural networks for this purpose author (2024).

Integrating it into a modularity-based loss function enable modeling of interactions in continuous-time graph data, guiding the learning process to capture the underlying dynamics of the system and improving the performance of community detection algorithms.

3.2.1 Hawkes process model

In general, Hawkes processes are point processes where the interaction strength between variables (nodes) is conditioned by a function taking into account the previous interactions, It is a self-exciting process, meaning the occurrence of an event increases the likelihood of future events, and the influence of past events decays over time following a kernel function,

$$\lambda(t) = \mu + \sum_{t_k < t} \phi(t - t_k), \tag{11}$$

where $\lambda(t)$ is the intensity function at time t, μ is the base intensity, and $\phi(t - t_k)$ is the kernel function that models the influence of past events $t_k < t$ on the current intensity.

The kernel typically takes the form of an exponential function, $\phi(\Delta t) = \alpha e^{-\beta \Delta t}$, where α is the magnitude of the contribution of past events to current intensity, and β is a decay parameter determining how quickly their influence diminishes over time, where higher values indicate faster decays and shorter-lasting influence, while lower values a slower decay and longer-lasting influence. The parameter β may be parameterized as $\beta = -\log(\rho)$, where $\rho \in (0, 1)$, allowing to directly learn a retention rate that captures the proportion of influence retained from past interactions, with ρ close to 1 indicating a longer-lasting influence and ρ close to 0 indicating a shorter-lasting influence of past interactions on current intensity. β as $\beta = -\log(\rho)$, where $\rho \in (0, 1)$ is a retention rate that captures the proportion of influence retained from past interactions, with ρ close to 1 indicating a longer-lasting influence and ρ close to 0 indicating a shorter-lasting influence of past interactions on current intensity.

Standard spectral methods rely on the normalized Laplacian $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ derived from a static adjacency matrix $\mathbf{A}$ (with optional edge weights). In temporal graphs, edges represent interactions with varying degrees of recency and frequency, allowing for the modeling of interaction strength as a function of time by adopting a simple edge weighting

mechanism modeled by a kernel function. We define a time-weighted adjacency matrix $\tilde{\mathbf{A}}(t) = \mathbf{A} \odot \mathbf{\Lambda}(t)$, where $\mathbf{\Lambda}$ is a matrix of intensities derived from a Hawkes kernel λ, as in

$$\lambda_{ij}(t) = \mu_{ij} + \alpha \sum_{t_k < t} w_k e^{-\beta(t-t_k)}, \tag{12}$$

where μ, α, β are parameters of the Hawkes process, t_k are the timestamps of past interactions between nodes i and j, and w_k are the weights of past interactions, which may be set to 1 for binary adjacencies or learned from data to account for interaction type or strength.

This formulation follows a standard modeling approach where the intensity of interactions between nodes i and j at time t is given by a baseline intensity μ_{ij} plus a sum of (time-decayed) contributions from past interactions. Considering the exponential kernel, α is the magnitude of the contribution of past events to current intensity, and β is a decay parameter determining how quickly their influence diminishes over time. From a modeling perspective, an exponential kernel of the presented form is often chosen for its tractability with closed-form solutions for likelihood and efficient simulation algorithms, as well as its ability to capture short-term dependencies while ensuring that the influence of past events diminishes over time in a simple and interpretable manner. Others include (shifted) power-law $\phi(\Delta t) = \alpha(\Delta t + \tau)^{-\rho}$ (with $\rho, \tau > 0$) and Rayleigh kernels $\phi(\Delta t) = \alpha \Delta t e^{(\beta/2)\Delta t^2}$, which respectively model long-term dependencies and delayed influence (peak at $\Delta t = 1/\sqrt{\beta}$) and may be more suitable for specific applications, such as social media interactions or financial transactions, where the influence of past events may not decay exponentially but rather follow a different temporal pattern that is better captured by these alternative models.

The time-weighted adjacency matrix $\tilde{\mathbf{A}}(t)$ is normalized following Equation 18,

$$\tilde{\mathbf{A}}(t) = \mathbf{D}^{-1/2} \left(\mathbf{A} \odot \mathbf{\Lambda}(t) + \mathbf{I} \right) \mathbf{D}^{-1/2}, \tag{13}$$

where $\odot$ is the Hadamard product, $\mathbf{\Lambda}$ contains interaction intensities λ_{ij} (Eq. 12), and $\mathbf{D}$ is the time-weighted degree matrix. The identity $\mathbf{I}$ adds self-loops before normalization.

An instantiation of this approach is introduced in the Longitudinal Modularity Network (LMoN) model in the following section, where the (optionally multilayer) graph convolutional encoder incorporates a (trainable) Hawkes temporal decay embedded in the Laplacian operator of a GCN, allowing to parameterize interaction strength and capture evolving community structures, while preserving the semantics of modularity in the learned (latent) space. Similar strategies have been proved useful for optimization with neural networks **?**, although not in an end-to-end differentiable fashion with graph-theoretic objectives. Experimental results demonstrate that LMoN improves performance on temporal graphs with varying degrees of community stability and signal strength. Next section presents implementation details, followed by model evaluation on both real-world and synthetic temporal graphs, where its performance is compared against state-of-the-art baselines.

4 Theoretical insights

We offer theoretical insights into the proposed formulation, including a formal definition of temporal graphs and neural communities, bounds on the smoothness penalty of longitudinal modularity, and an additional proof that the model recovers DMoN as a special (static) case.

Definition 1. *A temporal graph is a set of discrete-time snapshots* $\mathcal{G} = \{(V, E)\}_{t=0}^{T}$*, where* V *and* E *are the node and edge sets of a network observed at time* t*; or a set of continuous-time events* $\mathcal{G} = \{(u, v, t)\}_{t=0}^{T}$*, where* t *is the time of the interaction between nodes* u *and* v*.*

Both discrete-time (snapshot-based) and continuous-time (event-based) representations allow for the modeling of temporal networks, although the ways in which they encode temporal information, the granularity at which they capture changes in the graph structure, data structure costs, and the algorithmic solutions arising from each differ among themselves. Discrete-time representations divide the network into a sequence of static graphs (snapshots) at specific time intervals, allowing for extensions that incorporate inter-snapshot dependencies considering the network as a multilayer graph. Meanwhile, continuous-time representations model the graph as a stream of time-stamped interactions (events), enabling a more fine-grained analysis of temporal dynamics that is often more suitable for networks with high temporal resolution or varying activity levels. However, although it enables a more natural representation of temporal dynamics, it also introduces challenges in terms of computational complexity and the need for efficient algorithms to handle the continuous nature of the data, while discrete-time representations may be more suitable for applications where specific intervals of interest (e.g., daily or weekly patterns) are relevant instead.

In general, while discrete-time representations expand the graph formalism to multilayer graphs and are more commonly found in the literature, continuous-time representations instead expand it to multiplex graphs and are gaining traction due to their ability to capture more nuanced temporal dynamics and avoid issues related to snapshot discretization, such as information loss or the need for arbitrary window lengths. Variations exist for continuous-time events where an additional element may correspond to, e.g., a float value representing the duration of the event, or an integer value representing an edge addition (1) or removal (-1), among others. Note that node and edge attribute features were omitted for simplicity, but may be included in both representations as needed.

Definition 2. *Neural communities are groups formed by node representations in a real-dimensional (latent) space, obtained by a mapping function* $f : \mathcal{G} \to \mathbb{R}^{d \times k}$ *that accounts for the graph's structural properties and (optionally) available node and/or edge attributes.*

Traditional community detection methods typically rely on the graph's non-Euclidean topology to identify node groupings, largely based on their connectivity patterns. Modern extensions and general clustering techniques usually operate instead in real-dimensional spaces, where data points (nodes) represented as vectors are grouped based on their proximity, frequently learned from data and optimized for specific objectives. Although their complete mapping is possible Guidotti and Coscia (2018), this arguably introduces a theoretical mismatch between paradigms, commonly approached as separate problems in the literature due to differing mathematical foundations, underlying assumptions, and overall motivations.

The 'neural' community definition extends the former notion by explicitly setting them in a latent (real-valued) space. In a representational paradigm, the mapping function $f : \mathcal{G} \to \mathbb{R}^{d \times k}$ can be approximated by a learnable function (e.g., a graph neural network) that captures the underlying structure of the graph and optimizes for a specific community detection objective. Overlapping (mixed-membership) communities therefore correspond to

projections in a high-dimensional space, and the strength of node affiliation to each community is given by the value of $C_{i,k}$, interpreted as a probability or a fuzzy assignment. This definition is not prescriptive, but rather a conceptual framing of community detection, bridging literature on community detection in graphs with that of machine learning.

Lemma 1. *If a graph generated from a* DC-SBM *with equal cluster sizes exhibits positive modularity, then the* LMoN *model is consistent both when $d_n \to \infty$ (weak consistency) and $d/\log(n) \to \infty$ (strong consistency), where d is the average degree of the graph.*

Proof. As the trivial solution of spectral modularity (Eq 4) correspond to the eigenvector of all ones with an eigenvalue of zero, the value Q of such a partition is 0. For any partition where $Q > 0$, the first term becomes negative, thus reducing the total loss relative to the trivial case, and for a fixed L_1 norm (total number of nodes n), the L_2 norm is maximized for the term $\mathcal{R}$ if and only if only one element is non-zero (i.e., all nodes in one cluster).

Therefore, any non-trivial partition yields a smaller value for the regularization term, ensuring the model fits the requirement of avoiding the trivial solution and maximizing the modularity term. As the collapse regularization term is also minimized when cluster sizes are equal, the global maximum solution has (in-probability) vanishing error rate in the limit of large graphs, and so LMoN remains asymptotically consistent under the conditions of the symmetric SBM dynamic setting (equal-size groups), as seen in **?** and **?**. Note that in graphs where communities are not fixed over time, the detectability threshold of the communities varies with the degree of change in the graph structure, as shown in Ghasemian et al. (2016).

Lemma 2. *Let n and m be the number of nodes and interactions in a temporal (undirected) graph. The smoothness term of longitudinal modularity is bounded by $[0, 2m-n]$, equal to 0 if nodes never change community and $2m-n$ if they undergo maximum change at every event.*

Proof. By construction, the squared difference $s_i^{(t)} = \frac{1}{2}\|\mathbf{c}_i^{(t+1)} - \mathbf{c}_i^{(t)}\|_2^2$ for each node and event is bounded in $[0, 1]$, with maximum value 1 when $\mathbf{c}_i^{(t)}$ and $\mathbf{c}_i^{(t+1)}$ are orthogonal one-hot vectors on the probability simplex. Since the first appearance of each of the n nodes does not constitute a transition, there are at most $2m - n$ community updates. The lower bound is trivial: if no node transitions community over time, $s_i^{(t)} = 0$ for all i, t, hence $\mathcal{S} = 0$.

A desirable property of the smoothness term introduced by longitudinal modularity are bounds that allow to interpret the modularity value Q in terms of both the underlying graph structure and the stability of (node-level) community assignments across time.

The original formulation normalizes the smoothness S (Eq. 8) by the number of edges m in the graph, ensuring it is bounded by $[0, 1]$ when $\omega = 1$, so properly scaling with Q. However, it does not directly translate to overlapping communities, where the notion of a 'switch' is not as well-defined and nodes may belong to multiple communities with varying strengths. Moreover, it is also based on the assumption of discrete time steps, and includes the number of times a node leaves and rejoins the network in the applied penalty term, introducing additional hurdles for its application to continuous-time graph domains;

We may consider instead the magnitude of change in community assignments over consecutive time steps, i.e., $\|\mathbf{c}_i^{(t+1)} - \mathbf{c}_i^{(t)}\|$, where $\mathbf{c}_i^{(t)}$ is the k-dimensional community assignment vector of node i at time t. If community assignment vectors lie on the probability

simplex, then the maximum possible change in community assignment for a node in an event equals $\sqrt{2}$, attained when $\mathbf{c}_i^{(t)}$ and $\mathbf{c}_i^{(t+1)}$ are different vertices of the simplex (i.e., orthogonal one-hot encodings). In such case, node i is completely reassigned to a different community, therefore corresponding to the notion of a 'switch' in the original formulation. Assuming fixed node ordering across time and label-aligned communities, we may rewrite $\mathcal{S}$ as the (normalized) Riemann sum of the integral of the squared magnitude of changes,

$$\mathcal{S} \approx \frac{1}{2m} \int_0^T \left\| \frac{d\mathbf{C}(t)}{dt} \right\|_F^2 dt, \tag{14}$$

where $\|\cdot\|_F$ is the Frobenius norm, up to a scaling factor depending on temporal discretization, e.g., a minimum interval Δt for continuous-time or 1 for discrete-time interactions.

This formulation allows to interpret $\mathcal{S}$ as a measure of the total 'energy' in temporal community dynamics, and provides a useful reference for understanding the trade-offs between maximizing modularity and temporal smoothness in the optimization process. Given that each timestamps edge event (u, v, t) contributes with two node updates, the maximum total change across all nodes and events is therefore equal to $2m$ (when every node undergoes maximal change in its community assignment at every interaction), while the minimum total change is equal to 0 (when no node changes community across time), i -ndicating perfectly stable community structures and retrieving static modularity (Eq. 1).

Lemma 3. *If the graph is static,* $\forall t : t = 0$*, then* LMoN *recovers* DMoN *as a special case.*

Proof. If $\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)}$ for all i and t, then $\mathcal{S} = 0$, $\tilde{\mathbf{B}} = \mathbf{B}$ and $Q_L = Q$ (Equations 4 and 9). Hence, the LMoN objective reduces to static modularity, which is maximized by DMoN.

If $t = 0$ for all interactions, then $\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)}$ for all i and t, resulting in $\mathcal{S} = 0$. The temporal modularity matrix $\tilde{\mathbf{B}}$ collapses to the static modularity matrix $\mathbf{B}$ as the temporal presence vector $\boldsymbol{\theta}$ becomes a constant and $\tilde{\mathbf{A}} = \mathbf{A}$ due to the lack of temporal decay. If $t = 0$ for all interactions, then $\mathbf{c}_i^{(t+1)} = \mathbf{c}_i^{(t)}$ for all i and t, resulting in $\mathcal{S} = 0$. The temporal modularity matrix $\tilde{\mathbf{B}}$ collapses to the static modularity matrix $\mathbf{B}$ as the temporal presence vector $\boldsymbol{\theta}$ becomes a constant and $\tilde{\mathbf{A}} = \mathbf{A}$ due to the lack of temporal decay. The null model $\boldsymbol{\Theta}$ also accounts for static node degree distributions without temporal dynamics (Eq. 10). Under such conditions, Q_L also reduces to Q, which is the objective optimized by DMoN, effectively recovering it as a special case of the more general modularity framework.

This property ensures that LMoN is applied to both static and temporal graphs without loss of generality, providing a unified approach for community detection across different network types and allowing direct comparability with existing methods in the static case.

5 Model architecture

Longitudinal modularity introduces a continuous-time quality function whose optimization is combinatorial and incompatible with end-to-end representation learning. A continuous relaxation of the problem allows it to be expressed in a differentiable form, yielding a tractable surrogate objective that can be optimized with gradient-based methods, naturally

suggesting a neural parameterization that enables its integration into modern architectures for representation learning on temporal graphs with high-dimensional attribute features.

Based on the presented preliminaries, we formulate the Longitudinal Modularity Network (LMoN), a neural instantiation of the proposed objective designed for end-to-end differentiable community detection in temporal graphs. On a high level, the model learns a differentiable function from a dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to a community assignment matrix $\mathbf{C}_{(n \times k)}$, where n is the number of nodes and k is the number of communities, optimized following Equations 8 and 9. We additionally leverage a modified graph convolutional encoder strategy that models interaction strength following a Hawkes process, where the influence of past interactions decays over time, and more recent interactions have a stronger influence on the current state of the graph or an interaction. This strategy allows a time-agnostic encoder to consider interaction strength by edge weighting and capture evolving community structures, while preserving the semantics of modularity in the real-dimensional space. For model experimentation, we employ modified graph convolutional and attentional encoders with an added temporal adjacency weighting scheme (Hawkes-Laplacian operator) to promote learnning on dynamic graphs in a scalable and end-to-end differentiable manner.

We highlight the main components of the architecture in Figure 1, including the encoder strategy to model evolutionary dynamics (Section 5.1), the optimization objective of (spec-

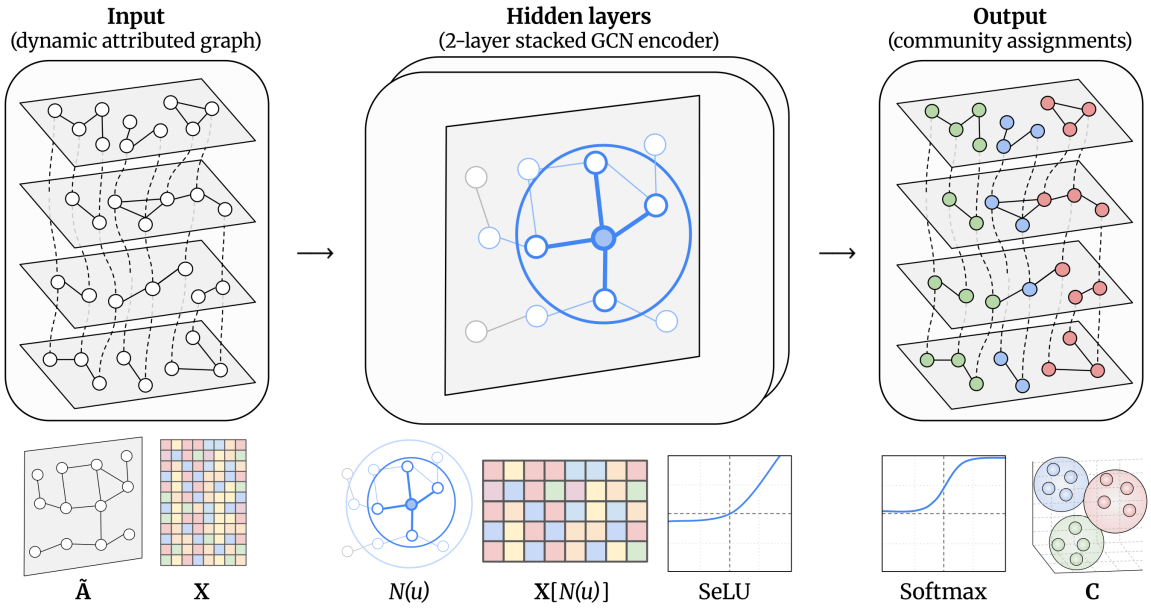

Figure 1: Illustrated architecture of the LMoN model for temporal community detection. A temporal graph $\mathcal{G}$ is first processed to compute a weighted normalized adjacency matrix $\tilde{\mathbf{A}}$ and a node attribute matrix $\mathbf{X}$. The inputs are fed into a 2-layer graph convolutional network (GCN) encoder to produce low-dimensional node embeddings that capture both structural and attribute information, with a Hawkes-inspired temporal decay embedded in the convolution operation to model interaction strength. Community assignments $\mathbf{C}$ are then obtained by applying a normalized exponential function to the encoder's output.

tral) longitudinal modularity (Section 5.2), and the pooling layer for obtaining dynamic community assignments (Section 5.3). Note that the clustering layer is agnostic to the specific encoder employed, allowing for different approaches appropriate, e.g., for inductive learning, as in Hamilton et al. (2017). We describe our contributions to each component in detail in the following sections, followed by an evaluation of the model on several graph benchmarks.

5.1 Encoder strategy

Employing a time-aware loss function such as longitudinal modularity allows optimization of model parameters and subsequent node assignments for dynamic community detection. Encoder strategies that are unaware of temporal dynamics (e.g., GCN) will produce node embeddings resulting from message-passing on the static graph structure, unable to capture the influence of interaction strength and recency on the current state of the graph.

To incorporate temporal dynamics into the spectral domain, we employ the Hawkes-Laplacian operator described in Section **??**. In this formulation, the time-weighted adjacency matrix $\tilde{\mathbf{A}}$ is derived from a Hawkes process model of interaction strength, enabling the model to capture the influence of past interactions on the current graph structure and community assignments. Following a standard process formulation (Eq. 12), the kernel λ models the intensity of interactions between nodes i and j at time t following an additive self-exciting process. Past interaction influence decays exponentially over time, as in

$$\lambda_{ij}(t) = \mu_{ij} + \alpha \sum_{t_k<t} w_k e^{-\beta(t-t_k)}, \tag{15}$$

where $\alpha \geq 0$ is the amplitude or excitation, a learnable parameter scaling the influence of past events, and $\beta > 0$ is the exponential decay rate, determining the temporal dissipation of historical influence. We reparameterize $\beta = -\log(\rho)$ as the decay factor, resulting in an exponential decay of the form $e^{-(-\log\rho)\Delta t} = \rho^{\Delta t}$, and we allow ρ to be learned on train.

We define the baseline intensity conditioned on node features $\mathbf{x}$ and degree-based bias:

$$\mu_{ij} = \sqrt{\mu_i \mu_j}, \quad \mu_i = \text{softplus}(\mathbf{w}_\mu^\top \mathbf{x}_i + b_i), \quad b_i = \frac{\log(1+d_i)}{\max_u \log(1+d_u) + \varepsilon}, \tag{16}$$

where $d_i = \sum_j A_{ij}$. is the degree of node i, $\mathbf{w}_\mu$ is a (learnable) weight vector, b_i is a bias, and w_k is the weight of the k-th historical interaction occurring at time t_k, which may be set to a constant or learned as a function of the interaction's features, such as edge attributes. We ensure positivity of the baseline intensity μ_{ij}, and add the degree-based bias b_i to account for node activity levels and prevent overfitting to high-degree nodes (confirm if optimal).

To introduce nonlinearity and ensure appropriate training ranges, we parameterize the learnable operator parameters α and ρ using smooth, monotonic functions that map from the real line to the desired ranges, allowing for unconstrained optimization and defined as:

$$\alpha = \text{softplus}(\hat{\alpha}) + \varepsilon, \quad \beta = -\log(\rho), \quad \rho = (1-\varepsilon)\,\sigma(\hat{\rho}) + \varepsilon, \tag{17}$$

where $\hat{\alpha}$ and $\hat{\rho}$ are the raw trainable parameters, $\text{softplus}(\hat{\alpha}) = \log(1+e^{\hat{\alpha}})$ is the smooth rectified linear unit function, $\sigma(\hat{\rho}) = 1/(1+e^{-\hat{\rho}})$ is the sigmoid, and $\varepsilon = 10^{-6}$ is a small constant added to prevent vanishing gradients from numerical instability in log/exponential

calculations. Note that ρ is bounded in $(0, 1)$ range and α is strictly positive, allowing the model to learn a wide range of decay behaviors from no decay ($\rho \approx 1$) to rapid decay ($\rho \approx 0$), and from strong influence of past interactions ($\alpha \gg 0$) to zero to no influence ($\alpha \approx 0$).

Message passing is performed on the Laplacian of the resulting (temporal) adjacency matrix $\tilde{\mathbf{A}}_\Lambda$, normalized by the degree matrix $\mathbf{D}$ to obtain a symmetric adjacency matrix

$$\tilde{\mathbf{A}}_\Lambda = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} \odot \mathbf{\Lambda})\mathbf{D}^{-\frac{1}{2}}, \tag{18}$$

where $\odot$ is the Hadamard product and $\mathbf{\Lambda}$ is the matrix of interaction strengths λ_{ij} computed for each edge (i, j) considering interaction history and recency as in Equation 12. As observed by Kipf and Welling (2017), a bipartite projection can be applied using separate in- and out-degree to learn asymmetric relationships, so a left normalization $\mathbf{D}_{\text{in}}^{-1}\mathbf{A}$ or a random walk matrix $\mathbf{D}_{\text{out}}^{-1}\mathbf{A}$ can be used instead, depending on the specific application.

This enables the encoder to learn time-sensitive node embeddings, where more recent interactions have a stronger influence on the current state of the graph if it allows better modularity optimization, while still allowing the model to capture long-term dependencies when appropriate by learning optimal decay parameters. Employing temporal unpooling strategies allows to further enhance the model's ability to capture evolving community structures by aggregating node representations over time, without fixed constraints. This is shown to improve model convergence and performance in temporal graphs with varying degrees of community stability and signal strength, as described in the next section. (we have to show this experimentally, but it is a natural extension of longitudinal modularity)

5.2 Optimization objective

The objective of LMoN is defined by a multi-component loss function $\mathcal{L}_{\text{LMoN}}$, which is minimized in the graph spectral domain and directly optimizes a continuous relaxation of longitudinal modularity. It consists of a modularity term with smoothness and regularization penalties to promote stable community assignments and avoid trivial solutions, and a Hawkes process negative log-likelihood term to regularize the temporal decay parameters based on the observed interaction data. It might be expressed in a general form as

$$\mathcal{L}_{\text{LMoN}} = \underbrace{-\mathcal{Q} + \mathcal{S} + \mathcal{R}}_{\substack{\text{Modularity} \\ \text{Sm.+Reg.}}} + \underbrace{\mathcal{H}}_{\substack{\text{Hawkes} \\ \text{NLL}}}, \tag{19}$$

where $\mathcal{Q}$ is (temporal) modularity, $\mathcal{S}$ is a smoothness penalty, $\mathcal{R}$ is a collapse regularization term, and $\mathcal{H}$ is the negative log-likelihood of the Hawkes process. A negative sign is applied to $\mathcal{Q}$ since the loss is minimized, while other terms are added as penalties to encourage desirable properties in the learned community assignments, avoiding trivial solutions and topochrone disconnections, and promoting the interpretability of the temporal dynamics.

Expanding on each of the aforementioned components, modularity $\mathcal{Q}$ is obtained by

$$\mathcal{Q} = \frac{1}{2m}\text{Tr}(\mathbf{C}^\top \tilde{\mathbf{B}}\mathbf{C}), \tag{20}$$

where $\tilde{\mathbf{B}} = \tilde{\mathbf{A}} - \mathbf{\Theta}$ is the modularity matrix resulting from Equation 10, with $\tilde{\mathbf{A}}$ as the weighted normalized adjacency matrix incorporating temporal decay based on interaction recency, and $\mathbf{\Theta}$ as the null graph matrix that accounts for temporal node activity.

The temporal smoothness term $\mathcal{S}$ is added to encourage stable community assignments over time, penalizing solutions where nodes frequently switch communities over time and favoring partitions that are stable across the temporal dimension. It may be expressed as

$$\mathcal{S} = \frac{1}{2m} \sum_{t=1}^{T-1} \left\| \mathbf{C}^{(t+1)} - \mathbf{C}^{(t)} \right\|_F^2, \tag{21}$$

where $\mathbf{C}^{(t)} \in \mathbb{R}^{n \times k}$ is the community-assignment matrix at time (or event index) t, and $\| \cdot \|_F$ is the Frobenius norm. This is equivalent to the node-wise expression and therefore matches the second term in Equations 8 and 9, as shown in Lemma 1 (Section 4). This term effectively measures the total change in community assignments for each node across time, providing a continuous relaxation of the original discrete penalty for community switches, and allowing for its application to both discrete- and continuous-time domains. Note that, in case community assignments are fixed across time (community stationary on dynamic graphs), it becomes a constant and may be ignored during optimization, as it does not influence the relative ordering of solutions in terms of their modularity score.

To avoid the trivial solution with zero-assignments, a collapse regularization term $\mathcal{R}$ introduced by Tsitsulin et al. (2024) is integrated into the objective function, defined by

$$\mathcal{R} = \frac{\sqrt{k}}{n} \left\| \sum_i \mathbf{C}_i \right\|_2 - 1, \tag{22}$$

where $\sum_i \mathbf{C}_i$ is the vector of cluster sizes, $\| \cdot \|_2$ is the Euclidean norm, normalized so that larger deviations from balanced cluster masses are penalized more strongly. The regularizer is designed to be small enough to not dominate the optimization process, but provide a useful signal to steer the model away from trivial solutions, as seen in Lemma 1 (Section 4).

To ensure that the learned decay parameters are grounded in the actual temporal distribution of the data, the module computes an auxiliary temporal loss term given by

$$\mathcal{H} = -\sum_k \ln(\lambda(t_k)) + \int_0^T \lambda(t) dt, \tag{23}$$

where the first term is the negative log-likelihood (NLL) of the observed event timestamps under the Hawkes process, and the second term is the integral of the intensity function over the observation period, acting as a normalizing constant to ensure that the intensity function is properly scaled. This term encourages the model to learn decay parameters that accurately reflect the dynamics of the interactions, ensuring that the temporal weighting of edges in the adjacency matrix is informed by the actual distribution of events in the data.

The composite function $\mathcal{L}_{\text{LMoN}}$ therefore combines the primary objective of maximizing modularity with regularization terms that promote stable community assignments and prevent trivial solutions, as well as an auxiliary loss to promote learning of meaningful temporal decay parameters, guiding the optimization process towards solutions that capture both the structural and temporal dynamics of the graph, modeled by the Hawkes process. Gradient descent strategies allow end-to-end differentiability through a continuous relaxation of the NP-hard problem of (discrete) modularity maximization in the $\mathbb{R}^+$ domain, which may

employ a number of encoder strategies to obtain expressive node representations. The presented loss function may be employed as the objective for training a graph neural network (GNN) architecture for dynamic community detection, as described in the next section.

5.2.1 Computational complexity

The modularity matrix $\tilde{\mathbf{B}}_\Lambda$ is defined as the difference between the temporally weighted normalized adjacency matrix $\tilde{\mathbf{A}}_\Lambda$ and a null model $\boldsymbol{\Theta}$ that accounts for temporal node activity. We leverage sparse-matrix operations to efficiently compute the temporal modularity matrix $\tilde{\mathbf{B}}$ and the modularity term $\mathcal{Q}$, which are key components of the loss function, as in

$$\tilde{\mathbf{B}} = \tilde{\mathbf{A}} - \boldsymbol{\Theta} = \mathbf{D}^{-1/2}\left(\mathbf{A} \odot \boldsymbol{\Lambda} + \mathbf{I}\right)\mathbf{D}^{-1/2} - \gamma\frac{(\mathbf{d}_{\text{out}} \odot \boldsymbol{\theta})(\mathbf{d}_{\text{in}} \odot \boldsymbol{\theta})^\top}{2m}, \tag{24}$$

where $\odot$ is the Hadamard product, $\boldsymbol{\Lambda}$ contains interaction intensities λ_{ij} (Eq. 12), $\boldsymbol{\theta}$ is the temporal presence vector, $\mathbf{d}$ is the degree vector, and $\mathbf{D}_\Lambda$ is the node degree matrix.

The Hawkes kernel $\boldsymbol{\Lambda}$ can be efficiently computed in vectorized form, where interactions are sorted by timestamps and cumulative influence is obtained for each edge using a prefix-sum operation. For a sorted sequence of events, the cumulative influence is calculated in $O(m \log m)$ time, making it scalable for large event streams, while the adjacency matrix $\mathbf{A}$ is typically sparse in real-world graphs, allowing for efficient storage and computation.

The trace of the modularity matrix to compute $\mathcal{Q}$ (Eq. 20) can be decomposed as a sum of sparse-matrix multiplications and rank-one (weighted) degree normalization. Formally,

$$\operatorname{Tr}(\mathbf{C}^\top \tilde{\mathbf{B}}_\Lambda \mathbf{C}) = \operatorname{Tr}(\mathbf{C}^\top \tilde{\mathbf{A}}_\Lambda \mathbf{C}) - \gamma\frac{1}{2m}\operatorname{Tr}\Big(\mathbf{C}^\top(\mathbf{d}_{\text{out}} \odot \boldsymbol{\theta})(\mathbf{d}_{\text{in}} \odot \boldsymbol{\theta})^\top \mathbf{C}\Big), \tag{25}$$

where $\mathbf{d}$ is the degree vector and $\boldsymbol{\theta}$ is the presence vector accounting for node lifetimes. This decomposition reduces the cost of evaluating the longitudinal modularity term from $O(n^2)$ to $O(d^2 m)$ per update, yielding an overall computational complexity of $O(d^2 n + m)$, where d, n, and m denote the number of features, nodes, and edges, respectively.

With the aforementioned optimizations, LMoN maintains a total runtime bound of $O(d^2 n + m \log m)$ per update, scaling favorably in graphs with sparse connectivity ($m < n^2$) and allowing for efficient computation of the Hawkes kernel for large event streams. Note that time complexity per update remains $O(d^2 n + m)$ for static graphs, as the Hawkes NNL term reduces to a constant (Lemma 2) and $\mathbf{A} \odot \boldsymbol{\Lambda} = \mathbf{A}$ due to the absence of temporal decay, resulting in $\tilde{\mathbf{A}} = \mathbf{A}$ and $\boldsymbol{\Theta}$ accounting for static node degree distributions (Eq. 10).

5.3 Temporal pooling

We experiment with several graph that receives a (temporal) normalized adjacency matrix $\tilde{\mathbf{A}}$ and node features $\mathbf{X}$ as input. In comparison to standard GCN layers, we employ trainable skip connections $\mathbf{W}_{\text{skip}}$ in place of standard self-loops, and the scaled exponential linear unit **?** activation function to introduce nonlinearity. This allows self-normalization to ensure stable variance and mean throughout deeper architectures **?**.

Each layer l computes node embeddings using a modified graph convolution operation

$$\mathbf{x}_i^{(l+1)} = \text{SeLU}\left(\sum_{j \in \mathcal{N}(i)} \tilde{A_{ij}} \mathbf{x}_j^{(l)} \mathbf{W} + \mathbf{x}_i \mathbf{W}_{\text{skip}}\right), \tag{26}$$

where $\mathcal{N}(i)$ is the neighborhood of node i in the graph, with the receptive field of the encoder increasing with the number of layers L for a maximum of L-hop neighborhoods. Last, dropout is applied between layers author (2024), to prevent overfitting and improve generalization, and the final layer outputs low-dimensional node embeddings $\mathbf{X}^{(L)}$ that are then fed into a normalized exponential function to obtain community assignments $\mathbf{C}$.

The model outputs final representations from a normalized exponential function, relaxing the discrete modularity maximization problem into a continuous, real-valued space, as in

$$C_{i,k} = \text{softmax}(\mathbf{X}_i^{(L)}\mathbf{W})_k = \frac{\exp(\mathbf{X}_i^{(L)}\mathbf{W}_k)}{\sum_{k'=1}^{K}\exp(\mathbf{X}_i^{(L)}\mathbf{W}_{k'})}, \tag{27}$$

where $C_{i,k}$ is the community assignment strength of node i to community k, and $\mathbf{W}_k$ is the weight vector corresponding to community k in the final layer, representing soft community memberships or overlapping (mixed-membership) community structures. An optional unpooling layer allows to obtain $\tilde{\mathbf{A}}(t)$ at the event level, where each (temporal) node embedding corresponds to a single message passing by the node pair, with interaction strength modeled by $\lambda_{ij}(t)$, and the resulting pooled representations used to compute a smoothness penalty in the loss function (Eq. 21), matching the second term in Equations 8 and 9 and thereby allowing for optimization' without fixed time window constraints.

To enhance the model's ability to capture evolving community structures, we introduce an event-based temporal pooling strategy that aggregates node representations over time, allowing for the detection of communities that evolve without being constrained by fixed time windows. This strategy involves pooling node embeddings based on the occurrence of events (interactions) rather than fixed time intervals, enabling the model to adapt to varying temporal dynamics and capture transient community structures that may not be detected with traditional snapshot-based approaches. By pooling node representations at the event level, the model can effectively capture the temporal evolution of communities and improve performance on dynamic graphs with varying degrees of community stability and signal strength, and allow optimization of the longitudinal modularity objective of LMoN.

At the final layer, node embeddings are pooled for each node i across its lifetime, considering the timestamps of its interactions and the corresponding community assignments, and the resulting pooled representations are then used to compute a smoothness penalty in the loss function, allowing for end-to-end optimization of community detection in temporal graphs. It is based on the variance of community assignments across time for each node,

$$h_i^{\text{pooled}} = \frac{1}{|\mathrm{T}_i|}\sum_{t\in\mathrm{T}_i} C_i(t), \tag{28}$$

where $C_i(t)$ is the community assignment of node i at time t, and T_i is the set of timestamps corresponding to node i's interactions. This pooled representation captures the temporal consistency of community assignments for each node, allowing the model to learn temporal dynamics and improve the detection of evolving community structures in temporal graphs.

In this setting, each temporal node embedding at event level (i, j, t) corresponds to a single message passing by the node pair, weighted by the interaction strength $\lambda_{ij}(t)$, and the resulting pooled representations are used to compute the smoothness penalty in the loss function (Equation 21). Although in most real-world datasets community assignments are

fixed, this strategy allows the model to capture temporal dynamics and improve performance on dynamic graphs with varying degrees of community stability and signal strength, as we show next, simply by optimizing the longitudinal modularity objective of LMoN.

6 Experimental results

Experiments were conducted on both real-world and synthetic graphs, allowing performance evaluation considering different domains and controlled scenarios. Table **??** summarizes the selected models for comparison, including static and temporal graph clustering models, parameterized as follows. When omitted, parameters follow the original paper's description.

- **LMoN** is trained with a learning rate of 0.001 (0.03 for decay parameters) and dropout rate of 0.3 for all datasets. Smoothness and collapse regularization are set to 1.0. Dimensionality is set to $L = [512]$ for real-world and $L = [256, 128]$ for synthetic datasets. Maximum number of epochs is set to 1000 and we consider the last epoch for evaluation.
- **Node2Vec** performs random walk sampling with in/out parameters $p = q = 1$ and walk length to 80, with 10 walks per node and a window size of 10 for the skip-gram model. Extensions accounting for feature signals (**Attri2Vec**) and temporal dynamics (**DynNode2Vec** and **TNodeEmbed**). Dimensionality is set to $d = 128$ and the adjacency matrix is symmetrically normalized (undirected graph), as in a standard GCN.
- **DAEGC** initializes centroids with K-Means and employs a 2-layer graph attentional encoder and the model optimizes binary cross-entropy of the reconstructed (static) adjacency matrix and the Kullback-Leibler divergence of cluster assignments.
- **TGC** also initializes centroids with K-Means and employs a Hawkes function to model interaction strength over time, where base intensity is parameterized by node similarity. A clustering loss is optimized at each epoch by negative sampling on edge-level batches, following a different algorithmic approach for objective optimization than LMoN.
- For comparison with traditional approaches, we include clustering with ***k*-means** and **spectral** decomposition on the original adjacency matrix and node feature sets.

Training occurred on a fully unsupervised scenario common for the task, where a validation set is not available and model selection is performed by observing the modularity score instead. Initialization of model parameters for LMoN followed a Glorot and Bengio (2010) distribution (normal) and separate optimizers were used for the Hawkes process parameters α and ρ to allow for different learning rates, as they operate on different scales and the former are more sensitive to changes in the loss landscape, while the latter was shown to benefit from a more stable learning rate to ensure convergence. The strategy outlined by **?** was followed to update the parameters based on the first and second moments of the gradients, allowing for adaptive learning rates that can improve training performance. Implementation relies on the Tensorflow and PyTorch Geometric frameworks.

We evaluate the performance of LMoN and the selected baselines on both real-world and synthetic datasets, analyzing the results in terms of clustering accuracy and community quality, and discuss its implications for neural community detection on temporal graphs.

Each experiment was repeated across 15 runs with preset initialization seeds, and the mean and standard deviation of the metrics are reported to account for variability in the optimization process and ensure reliability of the results. We select Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI) as clustering quality metrics to provide a more robust measure of performance accounting for chance agreement between predicted and true cluster assignments and imbalanced cluster sizes. In addition, conductance $\mathcal{C}$ (Eq. **??**) and modularity Q (Eq. **??**) were included to evaluate the quality of the detected communities in terms of their internal connectivity and separation from the rest of the graph, with lower conductance and higher modularity indicating better community structure.

6.1 Real-world graph benchmarks

Our experiments are reported on seven real-world datasets, summarized in Table **??**, selected based on their prominence in several temporal and static graph clustering benchmarks.

In a general description, *arXivAI* and *arXivCS* (Wang et al., 2020) are public citation graphs from the arXiv website, where each node represents a paper and edges represent citation relationships. These datasets are derived from OGB benchmarks for temporal graph clustering. *Brain* (Preti et al., 2017) is a human brain connectivity graph, where nodes represent small brain tissue regions and edges denote inferred connections. *DBLP* (Zuo et al., 2018) is a co-author network from the DBLP website, featuring 10 research areas (clusters). Nodes correspond to researchers, and edges indicate collaborative relationships. *PubMed* (Nikita et al. 2008?, Passos et al., 2024) is a network of papers in the field of medicine, where categories are three areas of research in diabetes and edges are timestamped (yearly) citations. *Patent* (Hall et al., 2001) is a citation network of US patents, with each patent assigned to one of six categories. *School* (Mastrandrea et al., 2015) is a high school contact and friendship network, where nodes are students and edges their interactions. *Travian* (Schleipen et al., 2019) is a temporal network of player interactions in the online game, where nodes represent players and edges represent interactions such as trades.

6.2 Synthetic graph benchmarks

To study model convergence on controlled scenarios, synthetic graphs with $N = 1024$ nodes, $E = 10^4$ edges, and $T = 8$ snapshots were created with a time-varying degree-corrected stochastic block model (DC-SBM) and node attributes sampled from a multivariate Gaussian distribution, as described in generative processes for temporal graphs and GNN benchmarking by . Adjacencies for each snapshot are generated from a degree-corrected stochastic block model and node attributes correspond to d-dimensional vectors of $d = 64$ features sampled from a k-multivariate normal (Gaussian) distribution, where $k = 8$ is the number of communities in the graph. Evolutionary dynamics are introduced by modeling community stability as a Markovian process where nodes transition or remain in their communities based on their current or initial membership affiliation with different likelihoods. Ablation studies were also performed to determine the contributions of each component in the model architecture.

insert table

insert figure

...

6.3 Ablation studies

We compare the performance of LMoN on benchmark graphs with several ablated versions of the model, so to evaluate the contribution of each component in the architecture.

— **LMoN w/o longitudinal** is a version of the model where the longitudinal modularity objective is replaced by the static modularity objective, so that the optimization focuses solely on maximizing modularity at each time step (with temporal decay).

— **LMoN w/o temporal decay** is a version of the model where the temporal decay mechanism is disabled by setting $\alpha = 0$ and $\mu = 1$, so that the adjacency matrix is not weighted by interaction strength and the model learns on the static graph structure.

insert table

insert figure

...

7 Conclusions

...

— Summary of our contributions (spectral longitudinal modularity, Hawkes-Laplacian convolutions, application for neural community detection in attributed temporal graphs)

— Current limitations found (e.g., only transductive setting due to GCN, but model is agnostic to the choice of encoder, so inductive setting is possible with other architectures)

— Future work scope (higher-order interactions, dynamic features, etc.)

☑ **To-Do:**
— **Introduction:** add references, perhaps focus a bit more on the problem we addressing (differentiable dynamic community detection, supposing evolutionary, non-random patterns).
— **Related work:** community detection methods (static/temporal) with special attention to modularity; neural network models (static/temporal) with special attention to DMoN; make it clear where related work fall short: temporal communities + high-dimensional attributes
— **Experiments:** we are now experimenting with encoders. add plots, tables, and more details on datasets, baselines, and evaluation metrics; add ablation study to confirm the independent contributions of the objective and encoder operator; add discussion of results and limitations.

References

N. author. Undefined. *Uncategorized*, 1(1):1–1, 2024.

V. Brabant, Y. Asgari, P. Borgnat, A. Bonifati, and R. Cazabet. Longitudinal modularity, a modularity for link streams. *EPJ Data Science*, 14(1), Feb. 2025. ISSN 2193-1127. doi: 10.1140/epjds/s13688-025-00529-x.

F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *National Academy of Sciences*, 99(25):15879–15882, 12 2002. doi: 10.1073/pnas.252631999.

A. Ghasemian, P. Zhang, A. Clauset, C. Moore, and L. Peel. Detectability thresholds and optimal algorithms for community structure in dynamic networks. *Physical Review X*, 6 (3), July 2016. ISSN 2160-3308. doi: 10.1103/physrevx.6.031005. URL `http://dx.doi.org/10.1103/PhysRevX.6.031005`.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `https://proceedings.mlr.press/v9/glorot10a.html`.

R. Guidotti and M. Coscia. On the equivalence between community discovery and clustering. In *Smart Objects and Technologies for Social Good. Third International Conference, GOODTECHS 2017, Pisa, Italy, November 29-30, 2017, Proceedings*, 3 2018. doi: 10.1007/978-3-319-76111-4_34.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR*, 2017.

M. Liu, Y. Liu, K. Liang, W. Tu, S. Wang, S. Zhou, and X. Liu. Deep temporal graph clustering. In *The 12th International Conference on Learning Representations*, 2024a.

Y. Liu, J. Li, Y. Chen, R. Wu, E. Wang, J. Zhou, S. Tian, S. Shen, X. Fu, C. Meng, W. Wang, and L. Chen. Revisiting modularity maximization for graph clustering: A contrastive learning perspective. In *Proc. of the 30th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, KDD '24, page 1968–1979, New York, NY, USA, 2024b. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671967.

A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 5 2010. doi: 10.1126/science.1184819. URL `https://doi.org/10.1126/science.1184819`.

M. E. J. Newman. Modularity and community structure in networks. *National Academy of Sciences*, 103(23):8577–8582, 2006. doi: 10.1073/pnas.0601602103.

A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. B. Schardl, and C. E. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs, 2019.

L. Peel, D. B. Larremore, and A. Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5), May 2017. ISSN 2375-2548. doi: 10.1126/sciadv.1602548. URL `http://dx.doi.org/10.1126/sciadv.1602548`.

A. T. B. A. R. J. P. B. Perozzi. Synthetic graph generation to benchmark graph learning. In *Workshop on Graph Learning Benchmarks*, 2021. URL `https://graph-learning-benchmarks.github.io/glb2021/`.

Y. Salehi and D. Giannacopoulos. Deep modularity networks with diversity-preserving regularization. *CoRR*, abs/2501.13451, 2025. doi: 10.48550/ARXIV.2501.13451.

A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller. Graph clustering with graph neural networks. *J. Mach. Learn. Res.*, 24(1), mar 2024. ISSN 1532-4435.