A thesis submitted in partial fulfilment of
Master 2 Computer science
Ecole Normale Superieure de Lyon

Defining an evaluation setting for community detection in dynamic graphs

Abstract

Temporal networks offer valuable insights into dynamic complex systems, capturing the evolving nature of social, biological, and technological interactions. Community structure is a critical feature of real networks, revealing the internal organization of nodes. Dynamic community detection algorithms uncover strongly connected node groups, unveiling hidden temporal patterns and community dynamics in temporal networks. Despite existing algorithms, evaluating their performance remains an open issue. A well-established method is to use tests that rely on synthetic graphs. However, such a benchmark does not exist for dynamic graphs with instantaneous edges and continuous time domains. To address this issue, we will introduce a novel benchmark with predefined communities, representing temporal networks with these properties. Additionally, we propose the Mosaic scoring function to evaluate partitions quantitatively to overcome the lack of quality functions. We explore the scoring function's behavior on the benchmark graphs to assess its capabilities and limitations. This work contributes to the improved understanding and evaluation of community detection algorithms in dynamic graphs.

Author:
Yasaman(Yas) Asgari

Research Supervisors:
Pierre Borgnat
Remy Cazabet

Date: 06/28/2023

Acknowledgments

I would like to express my heartfelt appreciation to Rémy Cazabet and Pierre Borgnat for their guidance and supervision throughout my six-month master's thesis. The introduction to the captivating realm of link streams was made possible by Rémy's excellent teaching of the network science course that my classmates and I attended during the M2 program. I am particularly grateful to Pierre, as his support enabled me to pursue my master's thesis in the field that truly interests me. I extend my sincere thanks to both of my advisors not only for their valuable advice but also for their humble and sincere approach to research. They consistently listened to my ideas and were cautious in rejecting them, fostering productive discussions and a strong sense of belonging.

I would like to extend special thanks for the incredible opportunity presented to me through the Ampère Excellence scholarship. This prestigious scholarship has allowed me to become a member of ENSL and granted me the privilege of residing in the enchanting city of Lyon for the last year.

Many special mentions are now deserved by all the friends with whom I shared the period of my master's (Fatemeh, Mahshid, Pouriya) and all of my classmates in Master 2 Lyon who helped me to enjoy the new life in Lyon. I would like to kindly thank also my friend, Khorshid, that was always sending me hope and intimacy. I would also like to mention friends and colleagues (Victor, Fabrice, Saeed, Andrea, Sina,and Avin) that constructive discussions helped me to go further with my research ideas.

Finally, my greatest thank you go to my family, my mother, father, and my sister because my academic career would have never even started without their constant efforts to promote my curiosity, perseverance, and willingness to put myself out there. They have always been and still are my first inspiration, and my accomplishments will never be great enough to pay them back. It was and is still hard to stay away from them, but I know that with my joy and passion for life, and also my success, at some point, I can really compensate for their hard times and give them happiness.

Contents

1 Context

Graph theory is a branch of mathematics that focuses on examining a collection of vertices interconnected by edges. Graphs represent a highly abstract and universal concept. When we employ a graph to depict real-world scenarios, it is commonly referred to as a network, comprising nodes and links [1, 2].

Network science tackles inquiries about international trade, traffic congestion, scientific collaborations, online or offline social networks, and synaptic connections within the brain [2]. By employing the vocabulary of networks, these concepts can be translated into graph theory to gain insights and make meaningful analyses.

There is a fascinating question of whether we can find communities in a network with the property that nodes are more densely connected within those clusters than the rest of the vertices. Community detection or graph clustering aims to answer this question and analyze the organization and structure of networks.

Introducing Community detection techniques can yield significant advantages in real-world settings. For instance, they play a crucial role in improving the performance of services on the World Wide Web. Clustering web clients with similar interests and close geographical proximity allows for implementing dedicated mirror servers, enhancing service delivery [3]. Additionally, identifying clusters of customers with similar interests within the network of purchase relationships enables personalized recommendations and optimized marketing strategies for online retailers like Amazon [4].

However, when in a network, both nodes and edges have the ability to emerge, vanish, or modify their characteristics over time; defining and identifying communities can be considerably more complex due to the increased level of detail and time-dependent nature. This additional temporal dimension can pose significant challenges in defining a temporal community, its recognition, and finally, the evaluation process.

In this master's thesis, our objective is to discover a quality function that establishes a concept of a good temporal community for fine-grained temporal networks named link streams. We aim to develop a measure that evaluates the quality of partitioning, considering both topological and temporal aspects. As evaluating dynamic community detection methods without a reliable ground truth is challenging, we will introduce a benchmark for link streams. This benchmark will serve as a reference for assessing algorithms in subsequent stages of research in this area.

The report is organized as follows. It commences by introducing link streams and exploring different representations in section 2. Section 3 provides a brief overview of existing community detection literature. The significance of benchmarks in evaluating these algorithms is discussed in section 4, where a few are also reviewed. In Section 5, we present a comprehensive framework named Mosaic for defining communities in link streams. In section 6, we expand a well-known benchmark from static networks to generate fine-grained temporal networks efficiently.

Moreover, we propose novel quality functions that build upon the widely-used Modularity method in section 7. We establish their equivalence to the existing literature in some cases. To demonstrate the behavior of our quality functions, we analyze them using a set of samples from the benchmark and a real-world data set in section 8.

2 Link stream

2.1 Mathematical Framework

Static networks are fixed network structures that represent relationships between entities. These connections denote various associations, including social interactions, communication channels, information flow, or any other relevant form of interdependence.

Definition 1. *Static network: A simple graph $G = (V, E)$ consists of a non-empty finite set V of elements called nodes, and a finite set E of unordered pairs of distinct elements of V called edges. We call V the node set and E the edge set of G. An edge u, v is said to join the vertices u and v, and is usually abbreviated to uv. [1].*

Temporal networks offer an expanded perspective where nodes and edges are subject to changes, allowing for additions or removals in time [5]. By incorporating the element of time, these networks capture the dynamic evolution of connections among nodes. In contrast to static networks, which have stable relationships, in temporal networks, edges seize the temporal dependencies and ordering of interactions between these entities.

Definition 2. *Temporal Network: A dynamic graph $\mathcal{T} = (V, E, T)$ consists of a non-empty finite set V, and a finite set E be temporal edges, and T be the time domain. Each temporal edge (u, v, t) signifies that nodes u and v interacted at time $t \in T$. The time domain T defines the set of possible time stamps or intervals when the interactions can occur. It can be discrete or continuous, depending on the nature of the temporal network [5].*

We will use a specific subset of temporal networks known as "Link streams" [6] to narrow our focus. In the context of temporal networks, link streams are characterized by instantaneous edges and a continuous time domain. Conceptually, a link stream can be interpreted as a static network where links interconnect nodes, yet each connection is associated with a set of timestamps indicating its occurrences. Link streams have found applications in various domains, including communication channels such as email and text messages, as well as social networks that involve physical contacts [5, 7].

Before we proceed, let's redefine the mathematical formulation of a link stream.

Definition 3. *Link Stream: A link Stream L is a triple (V, E, T) with $T = [T_s, T_e] \subset \mathbb{R}$ and $E \subseteq V \times V \times T$ models interactions over time: $l = (u, v, t) \in E$ means that an instantaneous interaction occurred between $u \in V$ and $v \in V$ at time $t \in T$ [6].*

The illustration in Figure 1 presents a link stream featuring a set of vertices $V = \{A, B, C, D, E, F\}$ where multiple temporal edges are observed. As an example, nodes A and B establish a connection twice within the given period, specifically at time stamps $\{2, 3\}$.

2.2 Link stream representations

Link streams cannot be considered merely as a straightforward extension of static networks; instead, they constitute a separate realm of study. One particular challenge lies in effectively representing and visualizing them. To tackle this challenge, researchers have introduced several widely-used representations [5, 6, 8, 9].

Given that the link stream is fine-grained data, a common approach is partitioning it into a sequence of static networks known as snapshots. To accomplish this, fixed or variable time windows denoted as Δ are employed to segment the temporal domain of the link stream into distinct intervals. Subsequently, the links occurring within each interval are filtered, forming a static network where nodes represent entities and edges illustrate their interactions.

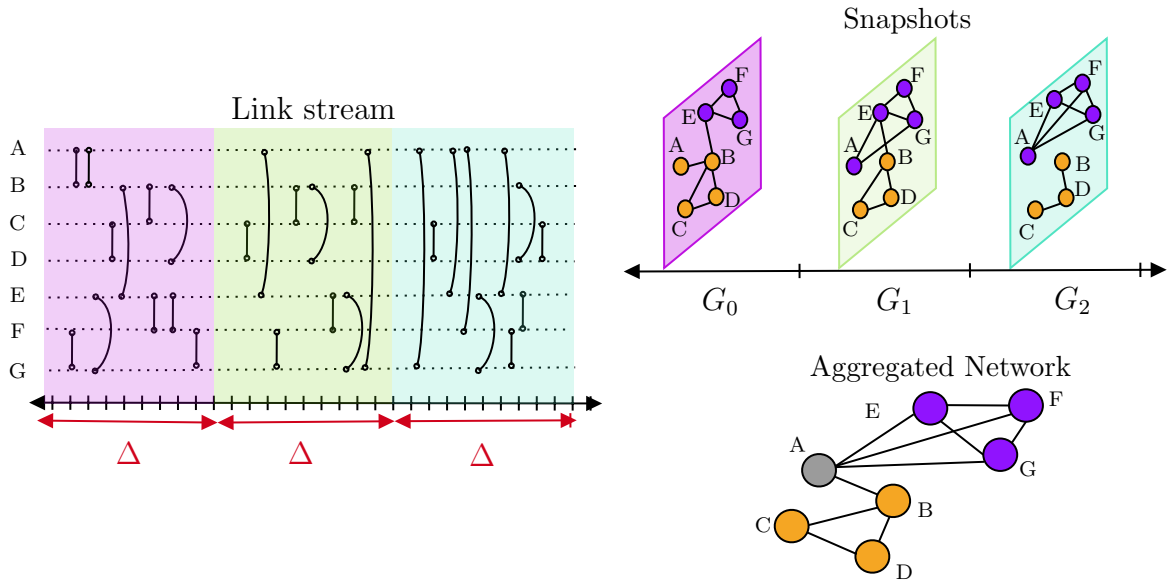

Figure 1: **Link Stream Representations**: On the left, a link stream is depicted, consisting of a set of vertices $V = \{A, B, C, D, E, F\}$ along with multiple temporal edges. On the right, two link stream representations are illustrated. At the top, we observe a sequence of 3 ordered snapshots, showcasing the evolving interactions over time. At the bottom, the link stream is transformed into a static aggregated network, where edges are displayed without a specific order or temporal information.

Definition 4. *Snapshot: A snapshot of a link stream $L = (V, E, T)$ is a static network $G = (V, F)$, where every edge in E adds a unit of weight on the corresponding edge in F:*

$$F = \{(u, v) | u, v \in V; \exists t \in [\tau, \tau + \Delta), (t, u, v) \in E\}$$

A link stream can be divided into N ordered snapshots $G = (G_1, \cdots, G_N)$ with a fixed or time-varying Δ [5, 8, 10]. If we choose the time window Δ to be $T_{end} - T_{start}$, we will get a static aggregated network.

This methodology captures the evolving dynamics of interactions, thereby unveiling mesoscale temporal patterns within the network structure. For example, when examining a sequence of text messages exchanged between individuals, people typically connect to one or no other people at any given moment. However, by employing the snapshot method, messaging patterns are unveiled on a broader scale, allowing for the tracking of changes over time [6].

To highlight this perspective, the left panel of Figure 1 showcases a link stream, while the right panels depict two different representations of the same stream. At the top, we observe a series of three ordered snapshots that showcase the evolving interactions over time. At the bottom, the link stream is transformed into a static aggregated network, where edges are displayed without a specific order or temporal information, providing an overview of the overall connectivity between nodes.

3 Community detection

Real-world networks deviate from regular structures and exhibit notable heterogeneity, revealing a high degree of organization and order in local and global arrangements. This phenomenon of local heterogeneity is commonly referred to as "community structure" [11, 12].

Qualitatively, a community corresponds to a structure of a network, composed of nodes densely connected and more sparsely connected to other nodes [13, 14], see Fig.1, panel aggregated network.

We can see the concept of a community in various contexts, such as a group of friends in online or offline social networks, proteins interacting for a biological function, or researchers collaborating within their respective fields.

Historically, network science emerged when real-world data for temporal networks, which capture the evolution of network structures over time, were scarce. Therefore, early research in network science focused primarily on static networks [11].

As the availability of low-resolution temporal data, such as snapshots, improved over time, researchers started paying attention to the dynamic nature of networks. This shift in focus led to the development of methods and techniques for dynamic community detection, which aim to identify and track communities or clusters of nodes that evolve in temporal networks [15].

Nowadays, we have access to fine-grained temporal networks, such as link streams. In these networks, defining and identifying communities can be considerably more complex due to the data's increased level of detail and time-dependent nature.

Therefore, we shall commence with a comprehensive analysis of well-established approaches employed within static or snapshot networks. Subsequently, we redirect our attention to why addressing this inquiry within higher-resolution networks is crucial, which refers to link streams.

3.1 Community detection in static networks

Community detection in static networks has received significant attention and investigation [11, 16]. Researchers from various disciplines have proposed numerous definitions for communities and approaches for automatically extracting these structures.

Our particular focus lies in community detection algorithms that effectively partition the "nodes" of a network into distinct, non-overlapping communities. We will obtain partitioning by applying such an algorithm to a given network.

Definition 5. *A partitioning $\mathcal{C}$ of a network $G = (V, E)$ involves the division of the network into k nonempty and mutually exclusive communities, denoted as c_i for each $i \in \{1, \ldots, k\}$. Each community c_i represents a subset of nodes $V_{c_i} \subseteq V$ where $\bigcup_{i=1}^{k} V_{c_i} = V$ and $(V_{c_i} \cap V_{c_j} = \emptyset, \forall i \neq j)$.*

Although numerous partitionings are available for a single network, not all effectively capture its communities' topological structure. While we primarily concentrate on a particular subset of algorithms that seek a globally-optimal community partitioning, we will also provide comprehensive explanations of other notable techniques and community definitions in A.

3.1.1 Globally-optimal community detection

In Globally-optimal community detection, the procedure begins by choosing a scoring function that quantifies communities as groups of nodes with dense connections. Then, one can use specific optimization techniques to find a partitioning that maximizes the scoring function. Afterward, they usually evaluate the outcome of their algorithm based on a reliable ground truth [17].

Globally-optimal algorithms are challenging due to multiple factors: There is no widely recognized definition for communities, and they are usually determined algorithmically, meaning they are the algorithm's outcome [18, 19]. Even if we agree upon a single definition, the formalizations of community detection give rise to NP-hard problems [20]. Moreover, the lack of ground truth makes evaluation tricky.

We can broadly categorize the scoring functions into two main groups: scoring based on connectivity(internal, external, mixed) or a network model [17, 18, 21], refer to appendix E and Figure 9. Within the latter category, one widely-used measure is Modularity. We will explain it in detail in the following subsections.

3.1.2 What is Modularity?

To address the query "What is the optimal partitioning?" scholars have introduced global metrics, such as M. J. Newman's Modularity [14]. Modularity is proportional to the number of edges falling within a community minus the expected number in an equivalent network with links placed randomly, preserving the degree sequence of nodes. Thus, Modularity is defined as below:

$$Q(W, \mathcal{C}) = \frac{1}{2m} \sum_{(i,j) \in V^2} \left[W_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{1}$$

where W_{ij} represents the number of edges between nodes i and j is an entry of adjacency matrix W. k_i represent the degree of node i, m represents the total number of edges in the network, c_i is the group membership of node i, and $\delta(c_i, c_j)$ is the Kronecker delta function equal to 1 if nodes i and j belong to the same community, and 0 otherwise.

Modularity is a scale value between -0.5 (non-modular) and 1 (fully modular). If the number of links within a group is not different from what we would expect by chance, the Modularity value is zero [22].

3.1.3 Why is Modularity a good scoring function?

After defining the Modularity, we need to apply an optimization method to find the optimal partitioning, denoted as $\hat{\mathcal{C}}$, that maximizes the value of $Q(W, \mathcal{C})$. Mathematically, this can be expressed as:

$$\hat{\mathcal{C}} = \arg\max_{\mathcal{C}} Q(W, \mathcal{C})$$

Although considerable effort has been dedicated to solving this optimization problem, it has been proven to be NP-complete [23]. Nevertheless, numerous greedy heuristics have been introduced, including a bottom-up approach [13, 24] or multi-level greedy search [25].

However, it has been discovered that the optimization of Modularity fails to identify modules smaller than a specific scale known as resolution limit [26]. Furthermore, it also finds communities in random networks [27].

Despite these limitations, Modularity remains a valuable framework, widely used and successful in practice. Modularity's simplicity and additive nature enable fast optimization heuristics, making it highly advantageous. Consequently, introducing Modularity to link streams would undoubtedly make a significant and worthwhile contribution to the field.

3.2 Community detection in snapshots

Since snapshots are sets of ordered static networks, static community detection algorithms can be applied separately to each temporal partition. This approach eliminates the need for analytical tools specifically designed to handle temporal data.

Various researchers have carried out several surveys [15,28–30] to present the distinctive features and challenges of dynamic community discovery and propose a classification of published approaches. We will refer to the comprehensive survey by Roessetti and Cazabet (2018) [15] that classifies methods into Instant-optimal, Temporal Trade-off, and Cross-Time. We will briefly explain the two categories that relate more to our proposed framework.

One category of techniques, known as instant-optimal methods, involves applying a static algorithm independently to each snapshot and determining the optimal partitioning for each one. Subsequently, these methods attempt to match communities between snapshot G_t and snapshot G_{t+1} that share similar nodes.

A straightforward approach [31] is to establish a match between communities c_i and c_j in consecutive times when their Jaccard index $\mathcal{J}(c_i, c_j)$ surpasses a predefined threshold Θ.

Jaccard index between two communities c_i and c_j can be defined as:

$$\mathcal{J}(c_i, c_j) = \frac{|V_{c_i} \cap V_{c_j}|}{|V_{c_i} \cup V_{c_j}|}$$

This category of methods does not include temporal smoothing; neither does it explicitly consider temporal dynamics. Additionally, static community detection methods often yield highly variable results, leading to instability in temporal partitioning.

Second, in the Temporal Trade-off category, communities defined at an instant t do not only depend on the topology of the network at that time but also on the past partitions found.

These methods offer incrementally smoothed partitioning based on objective optimization [32,33], balancing instantaneous quality and temporal smoothness. This criterion can be defined as:

$$Q^* = \zeta Q_{\text{snapshot}} + (1 - \zeta) Q_{\text{smoothness}}$$

In which $Q_{\text{stability}}$ is a temporal smoothing function and $\zeta \in [0, 1]$.

3.2.1 What are the limitations of community detection using snapshots

A critical issue related to the low-resolution temporal networks is identifying the optimal window size Δ to use when generating the snapshots, a choice that can profoundly affect the outcome of the subsequent analysis. The problem of choosing an appropriate window size has been extensively investigated in the field of temporal networks [34].

When the window size is chosen to be significantly larger than the timescale of dynamics in the network, the outcomes of community detection will become over-smoothed. Conversely, opting for a relatively small window size can lead to fluctuations within the community assignments [5, 15, 35].

However, considering the overall context of link streams, finding the optimal window size is irrelevant due to the fine time resolution concerning the dynamics. Instead, a distinct realm of study is required to explore community detection methods in this scenario.

3.3 Community detection in link streams

The literature on community detection in link streams is small since it is a more recent concept, and data with fine-grained temporal information was usually rare. There are, however, a few attempts to identify other notions of communities.

One such attempt is to utilize cliques. Cliques, in general, are sets of vertices that interact with all other vertices. Extending this definition to link streams, maximal cliques are cliques that maximize the number of nodes and duration, which can be detected by the algorithm proposed by Viard et al. [36]. Subsequent studies [37–41] have further improved this algorithm.

Although cliques offer a quantitative approach to identifying communities, they may not be suitable for real-world scenarios due to their strictness. Factors such as measurement errors or application-specific considerations can result in the absence of edges within cliques. To address this issue, a more relaxed definition called k-plexes has been proposed [42]. =

Definition 6. *A k-plex is a maximal subgraph with the following property: each vertex of the induced subgraph is connected to at least n-k other vertices, where n is the number of vertices in the induced subgraph. A clique corresponds to a 1-plex.*

This relaxation allows for the presence of communities with missing edges while still maintaining cohesion within the k-plex [42]. However, this notion is still restrictive compared to the original definition.

From another perspective, communities can be conceptualized as sets of edges rather than individual nodes, referred to as link communities [43]. There is a growing interest in detecting

link communities within a link stream. Density measures how consistently pairs of nodes are connected over time [6]. When a link community has a higher density than neighboring, it becomes significant in terms of both structure and temporal connectivity [44].

Moreover, a scoring function inspired by Modularity called Expected Nodes has been developed to detect link communities. It assumes that a link community usually involves fewer individuals than expected, while the surrounding links involve more individuals than expected [45].

While the concept of link communities provides valuable insights into the link stream, our focus lies on a definition that assigns a node to a community at a specific timestamp rather than considering the links themselves.

Despite the advancements in finding communities in link streams, this remains an ongoing research area with unanswered questions. Each solution developed so far has its strengths and limitations, and researchers continue to strive for more effective and comprehensive approaches to address the challenges posed by dynamic network data.

4 Network Benchmarks

Assessing and comparing the community detection algorithm presents a significant challenge. Although real-world datasets could offer valuable insights, it has been shown that node metadata are not the same as ground truth and that treating them as such induces severe theoretical and practical problems [46]. To overcome this limitation, researchers have developed benchmarks to generate synthetic networks for examining algorithm behavior on networks with diverse predefined properties [47].

Network Benchmarks enable checking an algorithm against [48]:

- 'Definition' of communities: Since there is no universal definition of community, a benchmark with its ground truth defines what we want to find and check if the method indeed recognizes it.
- Stability: The effectiveness of a Community detection method can be evaluated by testing it on numerous network instances that share similar characteristics. This test estimates the algorithm's stability, indicating how well it performs consistently across different network scenarios.
- Scalability: By gradually increasing the network size, it becomes possible to determine how well the algorithm handles larger and more complex networks.

With a similar structure to the last section, we will begin by thoroughly examining Benchmarks designed explicitly for static scenarios. Subsequently, we will focus on exploring temporal network benchmarks.

4.1 Static Network Benchmarks

According to [11], several Benchmarks for static networks, each capturing different properties of communities, exist. A valuable benchmark called Stochastic block models (SBM, also random planted partition graphs) [12,49–51] was one of the starting points for generating synthetic static networks with communities. It gets a partitioning $\mathcal{C} = \{c_1, c_2, \cdots, c_k\}$ including k communities with a desirable number of nodes, $p_i n$. Moreover it gets two values between $[0, 1]$ as probability p_{in} and probability p_{ext}. Vertices of the same community are connected with a probability p_{in}, whereas vertices of different groups are linked with a probability p_{ext}.

Another widely-used Benchmark named LFR [47] is developed to generate static networks with real-world properties. They assume that the distributions of overall degrees of nodes and community sizes are power laws, with exponents τ_1 and τ_2, respectively. Each vertex shares a

fraction $1 - \mu$ of its edges with the other vertices of its community and a fraction μ with the vertices of the other communities; $0 \leq \mu \leq 1$ is the mixing parameter.

Although many other notable Benchmarks for static networks show diverse perspectives of real networks, we only mentioned methods that are most relevant to our work.

4.2 Temporal Network Benchmarks

However, a major assumption has been made so far by many proposed generators: the networks modeled are static, and the communities stay the same as time goes by.

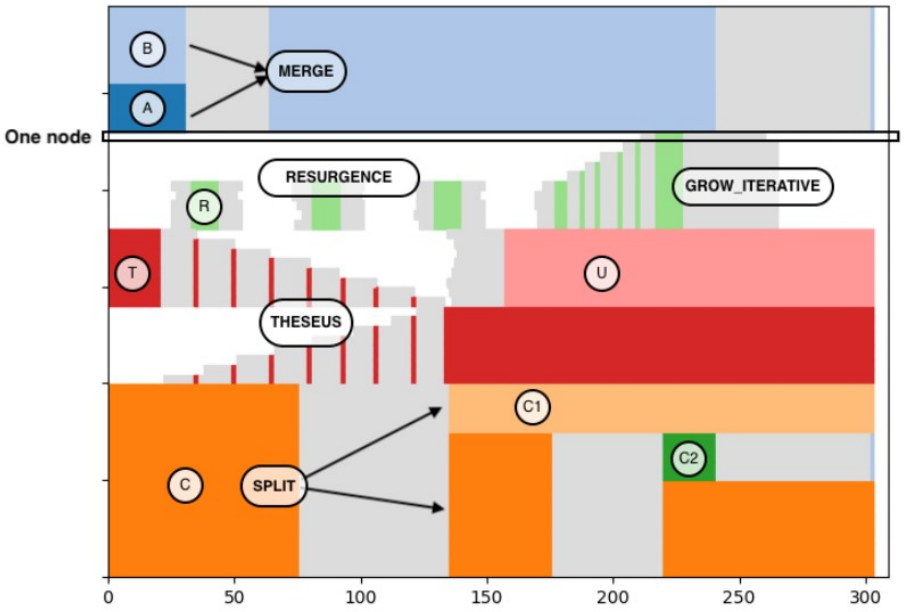

Figure 2: Progressively evolving communities: Every node is depicted as a horizontal line. Different colors show the communities. The flow of events is captured through arrows and labels. [15]

A few methods have already been introduced in the literature to generate benchmark graphs for evolving communities [48, 52–55]. By evolving communities; we mean that the evolution of communities can be characterized by the fundamental events of birth, death, merge, split, expansion, and contraction, iterative continuation, and ship of Theseus, see Figure 2 for an illustration and a complete formalism at [15].

We will explain one of the latest works tailored for progressively evolving networks [56]. This Benchmark is built upon the fact that empirical observations from real-world datasets indicate a trend in which the density of a community tends to decrease as its size increases while the average internal degree of the community increases. Based on a two-step process, first, the experimenter describes the scenario based on events and creates a set of communities denoted as $\mathcal{C}$; next, edges are generated by a random process using two parameters, a density coefficient $\alpha \in (0, 1]$ that defines the internal probability as $p_{in} = (|V_c| - 1)^{\alpha - 1}$ and a parameter of community identifiability that controls the external density.

However, to the best of our knowledge, no benchmarks have yet to be specifically tailored for link streams to account for their continuous time domain and the asynchronous nature of edge occurrences.

5 Mosaic Framework

To answer the question of what temporal community is in link streams, we need to introduce a new definition that expands the non-overlapping community to include the time-related aspect of relationships. We want to match our definition with the communities we observe in snapshots and static situations. To achieve this, we can think of each community in a link stream as a group of nodes that interact during a specific period.

In simple terms, we can define a Mosaic as follows:

Definition 7. *A Mosaic, denoted as c, is defined as a pair of (nodes, period): $c = (V_c, T_c)$. V_c is set of n nodes denoted as $\{v_1, v_2, \cdots, v_n\}$. T_c is a tuple, $T_c = (T_{cs}, T_{ce})$ where, T_{cs} and T_{ce} represent the start and end times of a Mosaic c, respectively. It represents the interval in which nodes V are considered part of the community c.*

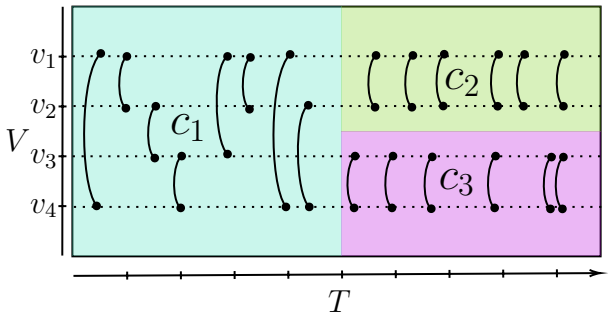

Figure 3: **Mosaic partioning**: A mosaic partitioning $\mathcal{C} = \{c_1, c_2, c_3\}$ is shown. It covers $\{v_1, v_2, v_3, v_4\} \times [0, 10)$ without any overlap.

After defining a Mosaic community, Mosaic partitioning can be defined as follows:

Definition 8. *Mosaic partitioning: Given a link stream $L = (V, E, T)$, $\mathcal{C}$ is a partitioning containing k mosaics $\{c_1, c_2, \cdots, c_k, c_*\}$ that cover the link stream fully without any overlap, refer to Figure 3. The empty community c_* is where nodes inside do not interact with other nodes in the link stream. This requirement can be written as follows:*

$$\bigcup_{c \in \mathcal{C}} V_c \times T_c = V_L \times T_L$$

6 Mosaic Link Stream Benchmark

In this section, we aim to introduce an innovative Benchmark that leverages the unique features of link streams, namely the continuous nature of the time domain and instantaneous edges using the Mosaic communities. This Benchmark addresses the absence of reliable ground truth for evaluating the effectiveness of partitioning in a link stream and testing the stability and scalability.

Although a few algorithms are tailored for link streams, to the best of our knowledge, we will be the first to introduce a benchmark for these dynamic graphs. It is worth mentioning that not only a benchmark offers a platform to explore the strengths and limitations of any algorithm, but it also functions as a playground for creating new methodologies.

We introduce the Random Mosaic Link Stream Benchmark, which aims to incorporate a temporal dimension into the random planted partition graph (refer to section 4.1), following a five-step process; see Figure **??**:

Step A: Mosaic partitioning generation: Given time domain $T = [T_s, T_e]$ and a set of nodes V, a partitioning $\mathcal{C}$ is obtained using Ad-hoc language or random Mosaic Partitioning.

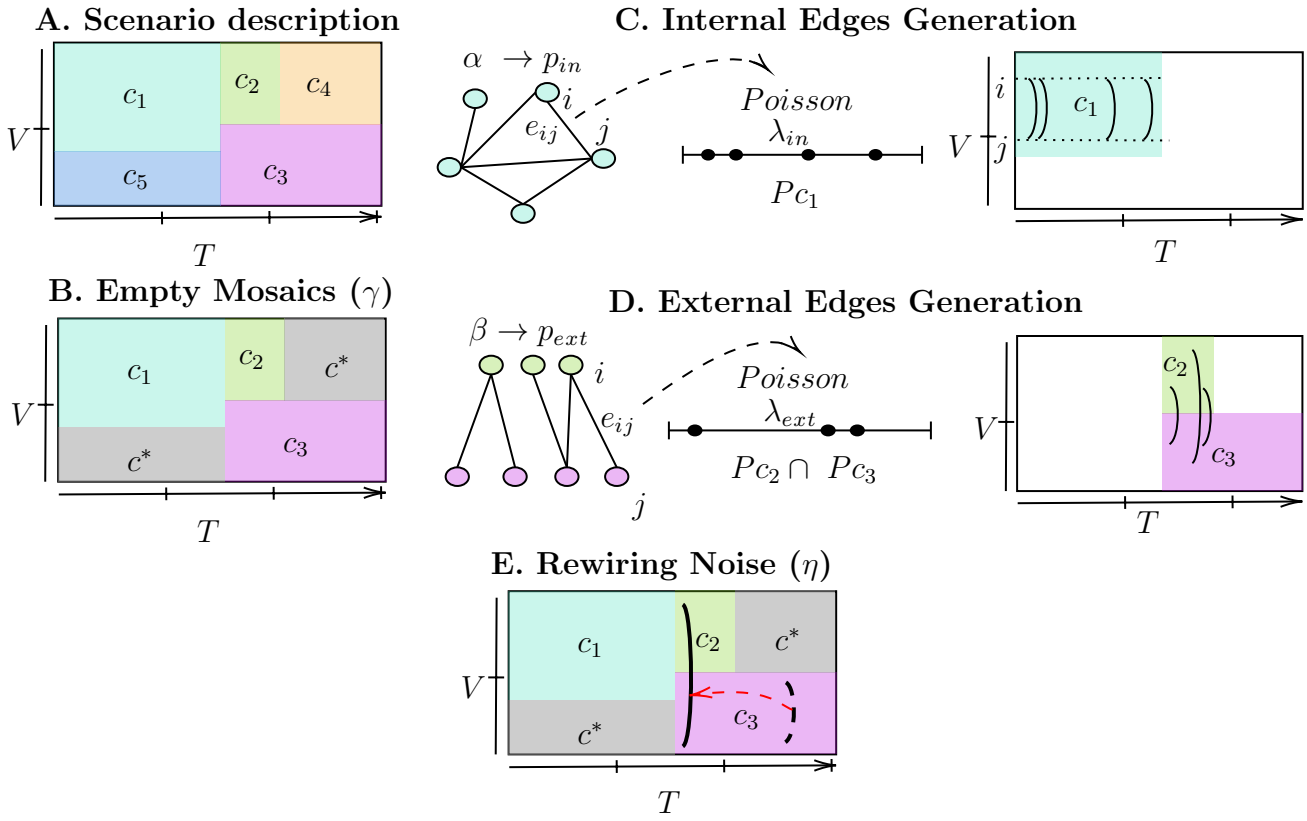

Figure 4: **Random Mosaic Link Stream Benchmark:** This figure illustrates a five-step process. Step A involves creating a scenario. Step B focuses on removing a fraction (γ) of mosaics to create an empty community named c^*, if necessary. Steps C and D add internal and external edges, respectively. Finally, if needed, in the last step, a fraction (η) of edges in the link stream can be rewired to different time intervals, node sources, and targets.

Step B: Empty Mosaics: In real-world scenarios, it is common to see a subset of nodes be inactive over a while. For example, when we examine the physical contact among students in a class, we observe that these students interact with each other solely during school hours. However, once the night falls, their contact with one another ceases. Therefore, we consider a fraction γ of Mosaics assigned to an Empty Mosaic c^*. We mean that within this Empty Mosaic, no edges can be active that originate from either inside or outside, affecting the nodes contained within it.

Step C: Internal edge generation: To generate edges within a community $c \in \mathcal{C}$, excluding the empty community c^*, we employ a semi-random procedure. Initially, we create a backbone of connections using a density coefficient α_c that determines the probability of two nodes being connected within the community period P_c. Next, we use a Poisson point process for each edge within the backbone network with a rate proportional to λ_{in}^c to determine those edges' active times. The Poisson point process introduces randomness into the generation of edges, allowing connections to be randomly distributed within the period of community. Refer to part 6.2 for a formal description.

Step D: External edge generation: To establish connections between communities c and $c\prime$, we employ a similar procedure described in step C. First, a backbone of connections is created using a community identifiability parameterβ, which determines the $p_{ext}^{cc\prime}$, the likelihood of a connection between nodes in the overlapping period $P_c \cap P_{c\prime}$.

Subsequently, a Poisson point process is utilized for each edge within the backbone connectivity network, with the rate of occurrence being proportional to $\lambda_{ext}^{cc'}$ to find external edges' activation time.

Step E: Rewiring noise: In our Benchmark, we introduce a parameter $\eta = [0, 1]$ to incorporate an element of noise. A portion η of the edges undergo a rewiring process, aiming to highlight the imperfections in community structures. In this step, for edges u, v, t selected in the sample $\eta * |E|$ edges, select two other nodes such that $u' \in c_u$ and $v' \in c_v$ and a timestamp randomly selected in the period of $T_{c_u} \cap T_{c_v}$.

Steps B and E are straightforward. In what follows, we will explain the process of the Mosaic Partitioning Generation (step A in part 6.1). Subsequently, we will delve into the underlying procedure of generating internal and external edges(Steps C and D in part 6.2).

6.1 Mosaic partitioning generation

Provided a user-defined node set V and time domain T, we proceed to generate diverse scenarios for benchmarking link streams to evaluate the dynamic community detection algorithms from various perspectives. We have devised four types of Mosaic partitioning: Experimental, Snapshots, Hierarchical, and Random. All these partitionings follow the primary assumption that Mosaics are communities containing a set of nodes and a period, and they fully cover the link stream. Detailed explanations of these algorithms and their properties will be investigated in further research steps; Figure 5 provides a brief overview.

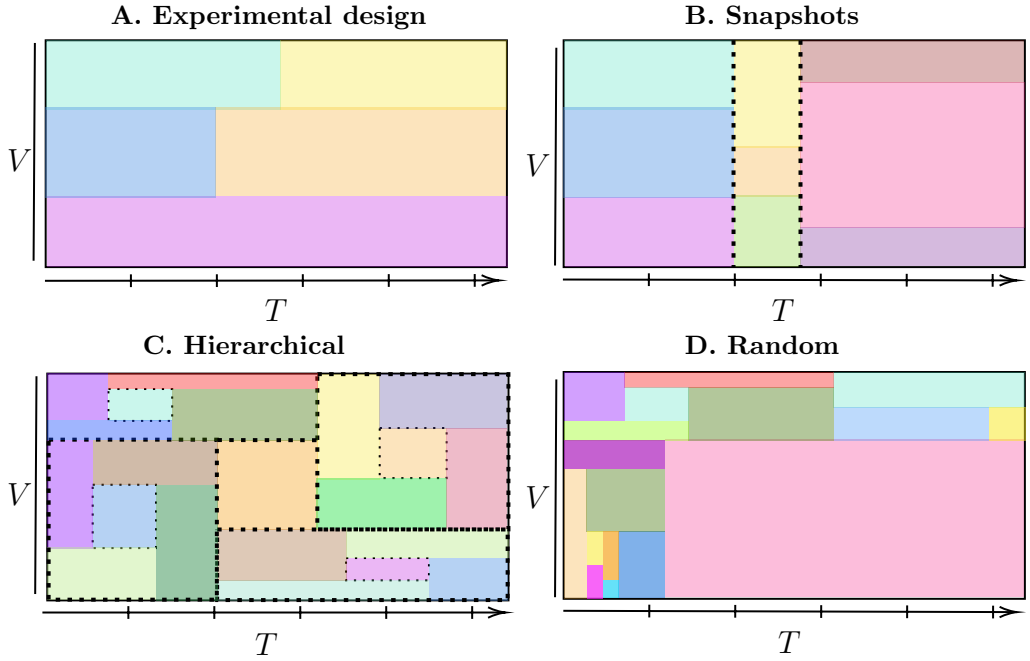

Figure 5: **Mosaic Partitioning Generation** This figure illustrates four different approaches for generating Mosaic Partitioning in the Benchmark. **Type A**: User-defined nodes and periods create the desired scenario. **Type B**: The time domain is divided into multiple frames or snapshots, and node sets are randomly allocated within each frame. **Type C**: Hierarchical Mosaics are generated based on the requested depth. **Type D**: Period and Node set sizes are distributed inhomogeneously, covering the entire link stream.

6.2 Generating edges

In this section, our primary focus is generating edges between nodes within and across different communities. We will follow two steps: Creating a Backbone connectivity network (details in part 6.2.1) and using the Poisson point process (details in part 6.2.2); refer to Algorithm 1.

Algorithm 1 Edges Generation

procedure EDGESGENERATION($\mathcal{C}, \alpha, \lambda, \beta$)
  Create an empty list E
  for c in $\mathcal{C}\backslash c^*$ **do** ▷ Generate internal edges
    $p_{in}^{c} = (|V_c| - 1)^{\alpha-1}$
    List S =**BackboneConnectivity**(c,c,p_{in}^{c})
    for e in S **do**
      Add **PoissonProcessEdge**(e,P_c,λ^{cc}) to E
  for (c, c') in $\binom{\mathcal{C}\backslash c^*}{2}$ **do** ▷ Generate external edges
    $p_{ext}^{cc'} = \beta((|V_c| + |V_c'|) - 1)^{\alpha-1}$
    List S =**BackboneConnectivity**(c,c',$p_{ext}^{cc'}$)
    for e in S **do**
      Add **PoissonProcessEdge**(e,$P_c \cap P_{c'}$,$\lambda^{cc'}$) to E
  return E

6.2.1 Backbone connectivity network

This Benchmark assumes that the connectivity between nodes, whether through internal or external edges, remains stable throughout the specified period. This is why we refer to it as the backbone connectivity network. A backbone connectivity network with a parameter p is a random graph in which each edge is present with probability p, independent of others.

Algorithm 2 Create BackBone connectivity

procedure BACKBONECONNECTIVITY(p, V_1, V_2)
  Create an empty list S
  for (v_1, v_2) in $\binom{V_1 \times V_2}{2}$ **do**
    $r \leftarrow$ random uniform between 0 and 1
    if $r \leq p$ and $v_1 \neq v_2$ **then**
      add (v_1,v_2) to s
  return S

We would like to emphasize that for establishing a well-defined internal structure of a community, it is necessary to utilize an appropriate range of values for p. This range's selection should depend on the number of vertices within the community. To achieve this, we will adopt the model described in [56], which provides the formula for p_{in}^{c} as follows:

$$p_{in}^{c} = (|V_c| - 1)^{\alpha-1}$$

Here, $\alpha \in (0, 1]$ is a hyperparameter named community density coefficient shared between communities. When the value of α is increased, the probability of p_{in}^{c} also increases, leading to denser clusters. If α is set to 1, each community in Mosaic becomes a clique.

The external probability between two communities c and c' denoted as $p_{ext}^{cc'}$ is defined as:

$$p_{ext}^{cc'} = \beta((|V_c| + |V_c'|) - 1)^{\alpha-1}$$

This hyperparameter $\beta \in [0, 1]$, called "community identifiability," is shared among all communities. Increasing the value of β results in more external edges between communities, making it more challenging for algorithms to identify each community as a separate cluster. In other words, β controls the external density of backbone connectivity by treating two communities as a single entity.

6.2.2 Poisson Point Process

To simplify the analysis, we assumed that the edges in a given backbone connectivity network follow a memory-less Stochastic Process for their activation times. First, we will review the definition of a Poisson process [57].

Definition 9. *Poisson Process: The counting process $N(t), t \leq 0$ is said to be a Poisson process having rate $\lambda, \lambda > 0$, if:*

(i) N(0) = 0.

(ii) The process has independent increments.

(iii) The number of events in any interval of length t is Poisson distributed with mean λt. That is, for all $s, t \leq 0$:

$$Pr(N(t+s) - N(s) = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!}, n = 0, 1, \cdots$$

Algorithm 3 Poisson Process of an edge

```
procedure PoissonProcessEdge(e, λ, P)
    Create an empty list S
    n ~ Poisson(λ|P|)
    for i in n do
        sample t from Uniform(P_s, P_e)
        add (e, t) to S
    return S
```

For each edge $e = (i, j)$ in the backbone connectivity network, we generate an i.i.d random Poisson point process with a rate parameter $|P|\lambda$. This rate parameter determines the average number of this edge active times within the time frame P. Then, we use the uniform distribution to distribute this number of occurrences in the selected period. This means the edge time arrivals are uniformly spread over the interval P [57].

To establish internal edges within each community c, we require a parameter λ_{in}^{c}. Furthermore, to generate external edges between communities c and c', we utilize a coefficient $\lambda_{ext}^{cc'}$. Combining these, we need a symmetric matrix λ of size $k \times k$, where k represents the number of communities. The main diagonal of this matrix will be utilized for generating internal edges, and non-diagonal elements can be employed for external edges if there is a non-empty time overlap ($P_c \cap P'_c \neq \emptyset$) between the communities c and c'.

6.2.3 Estimated number of edges and Time complexity

The expected number of internal edges in a Mosaic community $c = (V_c, T_c)$ equals to $|T_c|\lambda_{cc} \cdot \frac{|V_c|(|V_c|-1)^\alpha}{2}$. So, the overall number of internal edges is:

$$E_{internal}(\mathcal{C}) = \sum_{c \in \mathcal{C} \setminus c*} |T_c|\lambda_{cc} \cdot \frac{|V_c|(|V_c|-1)^\alpha}{2}$$

The expected number of edges between two communities $c = (V_c, T_c)$ and $c' = (V'_c, P'_c)$, the expected number of edges can be determined as $|T_c \cap T'_c|\lambda_{cc'} \cdot p_{ext}^{cc'} \frac{|V_c||V'_c|}{2}$. Thus, the overall number of external edges is:

$$E_{external}(\mathcal{C}) = \sum_{(c,c') \in \binom{\mathcal{C}\setminus c* \times \mathcal{C}\setminus c*}{2}} |P_c \cap P'_c|\lambda_{cc'} \cdot \beta((|V_c||V'_c|) - 1)^{\alpha-1} \frac{|V_c||V'_c|}{2}$$

Each community is processed independently, and the generation process can be efficiently parallelized. Finding the upper bound for memory and time complexity will be very complex due to dependence both on the time and structure, but we will assume an extreme case:

- Every node is connected to the other nodes in the community and between communities, $\alpha = 1, \beta = 1$.
- The node set, V, is divided into k equal parts, $V_{c_i} = \frac{|V|}{k}$.
- Each part lasts from the beginning to the end, maximizing $|P_c|$ and also $|P_c \cap P'_c|$.
- $\lambda_{cc'} = r\lambda_{cc}$ where $r \geq 1$.

With this simplification and with knowing that creating backbone connectivity takes less time and memory compared to the point process, we can write for both memory and time complexity:

$$\begin{aligned}\text{Algorithm complexity} &= |T|\lambda_{cc}\Big(\sum_{c \in \mathcal{C}\setminus c*} \frac{|V_c|(|V_c|-1)}{2} + r \sum_{(c,c') \in \binom{\mathcal{C}\setminus c* \times \mathcal{C}\setminus c*}{2}} \frac{|V_c||V'_c|}{2}\Big) \\ &= |T|\lambda_{cc} \frac{k + r\binom{k}{2}}{k^2} \mathcal{O}(|V|^2) = \mathcal{O}(r\lambda_{cc}|T||V|^2)\end{aligned}$$

This enables handling large networks in a reasonable time. Moreover, deterministic hash functions can be utilized for their generation to reduce Memory complexity.

7 Mosaic evaluation score

Based on our understanding of the state-of-the-art, we will propose a novel scoring function for evaluating partitioning quality in a link stream. This definition of the Mosaic evaluation score is a powerful tool that helps define a measure to evaluate the goodness of temporal community, where to the best of our knowledge, it did not exist before. In the next steps, by optimizing such a function for the link stream, we can find the optimal partitioning capturing both topological and temporal aspects of a temporal network.

The Mosaic scoring function simultaneously preserves two essential temporal community behavior properties. A partitioning is considered good if nodes belonging to the same Mosaic exhibit stronger connections than nodes in Mosaics. Moreover, nodes of the link stream should undergo minimal changes in their assigned Mosaics over time.

It is worth noting that the second property contradicts the first one, as a dense small Mosaic often scores higher with the topological structure. However, introducing a smoothness factor will balance this artifact and force communities to continue in time to increase stability.

The proposed scoring function combines these two factors to balance the strength of communities, Mosaic Modularity Q_{MM}, and the smoothness of transitions, Mosaic Smoothness Q_{MS}, using a weighing value $\zeta \in [0, 1]$.

We can define Q^* as follows:

$$Q^* = (1 - \zeta)Q_{\text{MM}} + \zeta Q_{\text{MS}} \tag{2}$$

If ζ is close to 0, it indicates a higher emphasis on the strength of communities, meaning that the topological structures inside the Mosaics play a more significant role in determining the score. On the other hand, if ζ gets larger, it indicates a higher emphasis on the smoothness of transitions, highlighting the importance of temporal dependencies and stability among the communities.

7.1 Mosaic Modularity

Modularity can be defined as a metric that quantifies the difference between the observed and an expected network structure that would arise in a "null" network. A null network breaks the connections between nodes and establishes them randomly to avoid any specific patterns of significance, preserving desired properties.

Given a link stream $L = (V, E, T)$, we can generalize the Modularity 1 as below:

$$Q_{\text{MM}} = \frac{1}{2|E|} \sum_{c \in \mathcal{C}} \sum_{(i,j) \in V_c^2} W_{ij}|_{T_c} - B_{ij}|_{T_c} \tag{3}$$

where $W_{ij}|T_c$ represents the number of edges between nodes i and j in the period of community c, $|E|$ represents the total number of edges in the link stream, and the $B_{ij}|_{T_c}$ represents the expected number of edges between nodes i and j during the period T_c within the link stream L in the null model.

In what follows, we will introduce different null models designed explicitly for link streams. We will categorize these proposed equations into global, local, and mixed groups of Mosaic Modularity.

7.1.1 Global Mosaic Modularity

As mentioned earlier, the degree-preserved null model is commonly employed for static networks, as represented by Equation 1. To extend this concept, we generalize it to encompass the overall node degree during the time domain T of the link stream:

$$B_{ij}|_{T_c} = \frac{|T_c|}{|T_L|} \frac{k_i k_j}{2|E|}|_{T_L}$$

Here, k_i represents the degree of the node i when aggregating the entire network into a static form.

7.1.2 Local Mosaic Modularity

In link streams, an alternative approach is available for determining the expected number of edges between nodes i and j. This approach, known as local Mosaic Modularity, emphasizes preserving the node degrees within each community.

$$B_{ij}|_{T_c} = \frac{k_i k_j}{2m}|_{T_c}$$

Here, m represents the total number of edges within the period T_c of the Mosaic c in the link stream L. The term $k_i|_{T_c}$ denotes the degree of the node i in the period T_c, which is calculated as the number of edges connected to node i during that period.

7.1.3 Mixed Mosaic Modularity

In the category of Mixed Mosaic Modularity, we will focus on a null model that integrates the local and global structure of link streams. It is calculated based on the average degrees of the nodes involved and the overall connectivity of the network.

The formula for expected number of edges between i and j during the period of T_c is as follows:

$$B_{ij}|_{T_c} = \frac{\hat{k}_i \hat{k}_j}{2\hat{m}}$$

We can break down the formula as follows:

- $\hat{k}_i$ and $\hat{k}_j$ represent the average degrees of nodes i and j in the link stream L respectively. They are calculated as follows:

$$\hat{k}_i = \frac{|T_c|}{|T_L|} k_i|_{T_L}$$

- $\hat{m}$ represents the expected number of edges in the entire link stream L during the period T_c. It indicates the average connectivity of the network during that period. To understand $\hat{m}$, we need to consider the vertices that are connected to at least one vertex in V_c within the period T_c. Let's denote this set of vertices as U. We can express $\hat{m}$ as follows:

$$2\hat{m} = \sum_{i=1}^{|U|} \hat{k}_i = \begin{cases} \hat{k}_i & i \in V_c \\ \hat{k}_i \dfrac{|T_x \cap T_c|}{|T_x \cup T_c|} & i \notin V_c \rightarrow i \in V_x \end{cases}$$

7.2 Mosaic smoothness

The Mosaic smoothness measures another aspect of the goodness of Mosaic partitioning $\mathcal{C}$; it wants nodes of the link stream to undergo minimal changes in their assigned Mosaics over time. We are interested in the average number of changes per node, so we will use a mediator network $G = (\mathcal{C}, \mathcal{E})$ that captures the temporal connectivity of communities. To construct this network, we add an edge (c_i, c_j) in $\mathcal{E}$ for any pair of consecutive non-empty communities (c_i and c_j) that share a common set of nodes, refer to Figure 6.

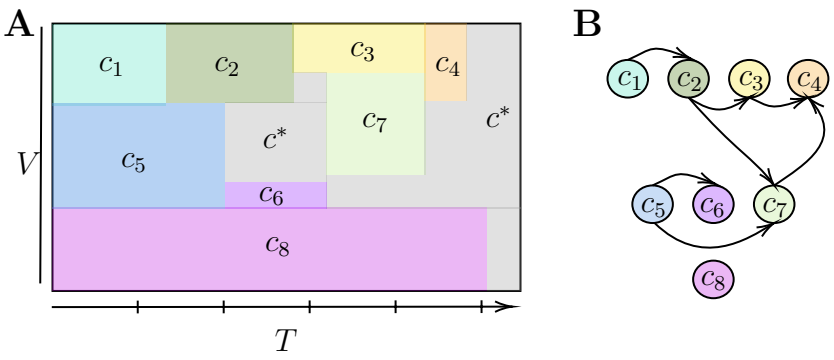

Figure 6: **Mosaic Smoothness**: The left panel illustrates a Mosaic partitioning. On the right panel, we observe the mediator temporal connectivity network for that partitioning. It is worth noting that during the construction process, any connections from or to the empty community (c^*) are disregarded, highlighting the emphasis on preserving the smoothness of the Mosaic partitioning.

We can express the average number of changes per node denoted as $\mathcal{X}$ as follows:

$$\mathcal{X} = \frac{\sum_{(c_i,c_j)\in\mathcal{E}} |V_{c_i} \cap V_{c_j}|}{|V|}$$

To minimize this value, we use the transformation $\frac{1}{1+\mathcal{X}}$. Adding 1 in the denominator considers the scenario where $\mathcal{E}$ is empty, indicating complete smoothness. Overall, we can write our Mosaic smoothness as:

$$Q_{MS} = \frac{1}{1 + \frac{\sum_{(c_i,c_j)\in\mathcal{E}} |V_{c_i} \cap V_{c_j}|}{|V|}}$$

8 Results

In this section, we will explore the characteristics of our Mosaic scoring function to see if it aligns with previous studies on static networks and snapshots. First, we expect Mosaic Modularity(first term of Equation 2) to be similar to certain notions of Modularity under specific conditions.

To the best of our knowledge, a single paper [56] has been published so far comparing empirically dynamic community detection algorithms considering the smoothness(second term of Equation 2). However, in this paper, the smoothness parameters were only tested on progressively evolving networks and not on the link streams.

In what follows, we shall first prove that the structural term is equivalent to some notions of Modularity. Next, we will use our techniques on examples from a benchmark and a real-world data set to demonstrate how the proposed scoring function can account for the presence of communities in temporal networks.

8.1 Mosaic Modularity and Static Modularity Equivalence

If we assume a partitioning where nodes within communities retain their community assignments over time, $Q_{MS} = 1$, the Mosaic Modularity term will align with the static Modularity.

Theorem 1. *For a link stream $L = (V, E, T)$ with $Q_{MS} = 1$, the local/global/mixed Mosaic Modularity can be simplified to static Modularity.*

In appendix C, we prove the equivalence for global mosaic modularity. One can use similar arguments to mixed and local Modularity, resulting in an equivalence between all levels of Mosaic Modularity and static Modularity.

8.2 Local Mosaic Modularity and Average Modularity Equivalence in Snapshots

Given link stream $L = (V, E, T)$, the network can be partitioned into R snapshots with a fixed or varying windowsize Δ, resulting in a lower resolution temporal network represented as $L = (\pi_1, \pi_2, \cdots, \pi_R)$. There exists a scoring function that is designed for snapshots, named average modularity [58]; which can be expressed as:

$$Q_{avg} = \frac{1}{\sum_{r=1}^{R} \omega_r} \sum_{r=1}^{R} \omega_r Q(\pi_r)$$

where Q_{avg} denotes a weighted formula combining static Modularity across different snapshots.

We will prove that the local Mosaic Modularity is equivalent to the average Modularity when employing a particular weight function ω_r. The equivalency of two formalisms coming from two different research directions is interesting. The proposed weight function is not mentioned in the paper [58] itself. Thus, it can lead us to find meaningful communities with a new evaluation score.

Theorem 2. *Local Mosaic Modularity is equal to the average Modularity, where the weight assigned to each snapshot is the total number of edges in that particular snapshot.*

The proof of this theorem is written in Appendix D. It is worth noting that this equivalency does not hold in global or mixed Mosaic Modularity.

8.3 Testing Mosaic Modularity on Link stream Benchmark

To complete the discussion on the evaluation setting, we will apply the proposed scoring functions to the benchmark samples and find their capabilities and limitations. In this section, we want to see the effect of the community structure's strength in a link stream varying the benchmark initialization hyperparameters.

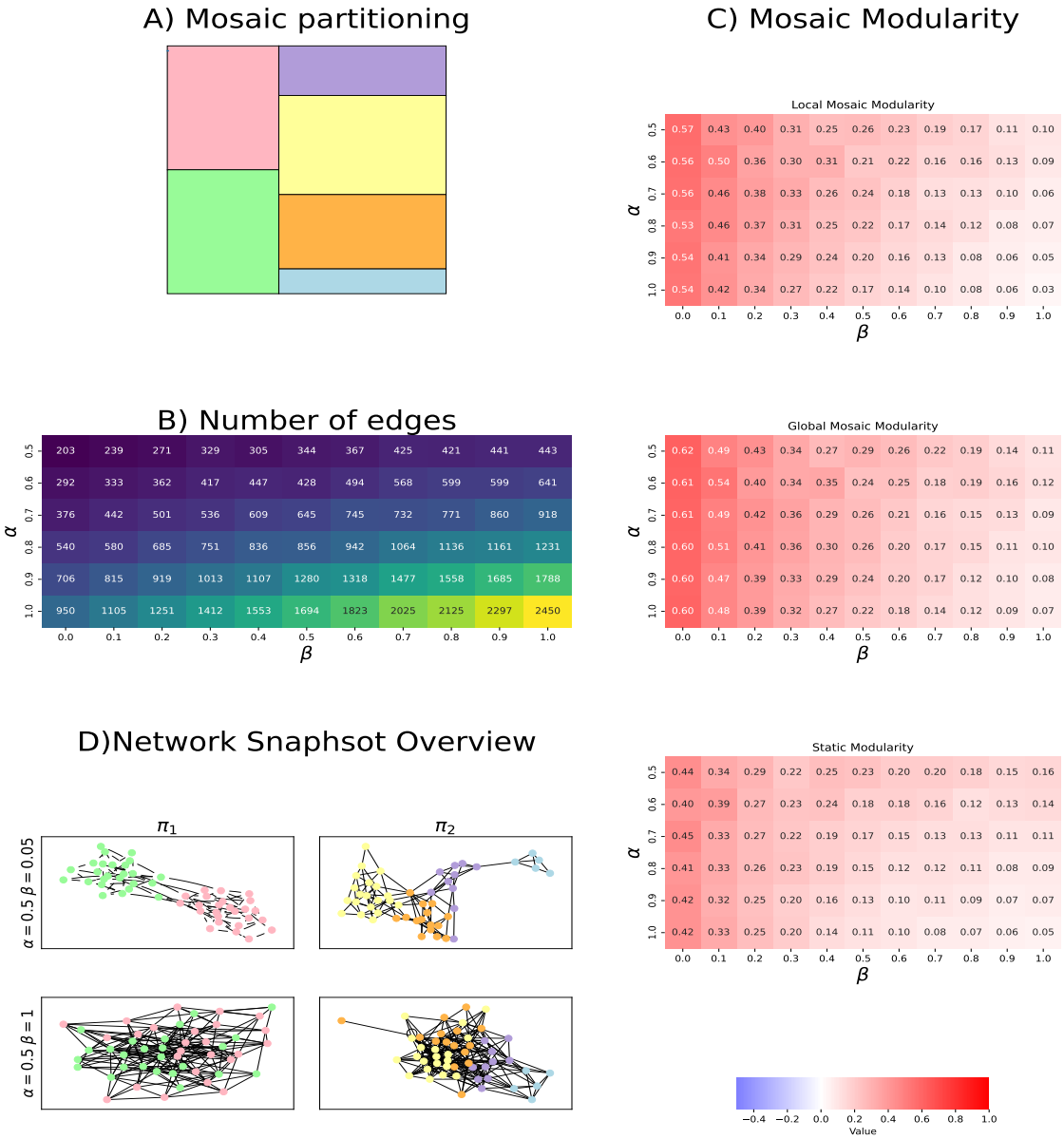

Figure 7: **Testing Mosaic Modularity on the Link Stream Benchmark:** The Mosaic partitioning of the Link Stream Benchmark is depicted in panel A, where 50 nodes interact within the time interval $[0, 50)$. To create edges, we used $\lambda_{cc'} = \lambda_c = 0.01$, and we varied α from 0.5 to 1 and β from 0 to 1, resulting in an increased number of edge as each parameter was increased(Panel B). The values of Mosaic Modularity and the optimal value of static Modularity are displayed in panel C. Panel D shows two samples from this benchmark, illustrating snapshots π_1 and π_2. Increasing β while fixing α makes the communities less distinctive and more interconnected.

As was mentioned in section 6.2, $\beta \in [0, 1]$ is the community identifiability parameter and the $\alpha \in (0, 1]$ is the internal density coefficient. The choices of α and β can affect the

community structures by changing the probability of connections in the backbone connectivity networks, thereby affecting their ability to identify them. If β is zero, we have a few disconnected communities. As we increase β, the Mosaic Modularity should decrease since it is more difficult to distinguish separate communities.

We create a snapshot Mosaic using Mosaic partitioning Type B for the empirical testing. Given the number of nodes and time steps of 50, we start testing different values for hyperparameters $\alpha \in [0.5, 1]$ and $\beta \in [0, 1]$ and then simulate the stream and find the corresponding values for local, global Mosaic modularity. Moreover, we also calculate the static Modularity using greedy optimization for the static aggregated network. In Figure 7 and its caption, we have explained the effect of α and β on the number of edges and network representation briefly.

Interesting results can be obtained by analyzing the behavior of local and global Mosaic Modularity values and comparing them to static Modularity. When we decrease the value of β from 1 to zero while keeping α fixed, we observe an increase in both local and global Mosaic Modularity values. This increase indicates that the scoring function can successfully identify communities when they are present. On the other hand, if we increase both α and β simultaneously, the connectivity between nodes increases, but the evaluation scores decrease rapidly, depicting less clear communities, which is true. Specifically, when $\alpha = \beta = 1$, the value approaches zero. All these observations confirm that the proposed Mosaic Modularity has a well-defined behavior that aligns with the expectations and is compatible with existing methods in the literature.

8.4 Testing the Mosaic scoring function on Real world Dataset

In this section, we aim to elucidate how Mosaic scoring function can be utilized to determine the ideal windowsize Δ when employing well-established methods in snapshots or static networks.

To evaluate our scoring function using real-world datasets, we require an interaction dataset with high temporal resolution. Sociopatterns[1] is a database that offers such data in various contexts. In these scenarios, individuals are equipped with RFID sensors, enabling the measurement of their real-time proximity. For instance, a notable example is a primary school study [59]. Over the course of two consecutive days, 230 pupils and 10 teachers wore sensors, resulting in a total of 125,000 face-to-face interactions recorded for 32 hours at a time resolution of 20 seconds . The left panel of Figure 8 illustrates the interactions between individuals over time, with each interaction color-coded based on their corresponding classes. Additionally, it presents the static network formed by merging all the interactions between individuals.

Initially, we converted our link stream into a snapshot network by employing the described methodology, where the windowsize ranged from 1 minute to 16 hours. Next, within the snapshot network, we utilized the Louvain algorithm [25], implemented in Python's Networkx library, to derive the optimal communities. Following that, we assessed these obtained communities by employing our scoring function on both local and global levels, which incorporates various values of ζ. ζ acts as a weight factor that balances smoothness and Mosaic Modularity. The optimal window size corresponds to the maximum value of $Q*$ with a specific ζ.

By observing the right panel of Figure 8, we can determine that smaller window sizes yield higher values for $Q*$ as ζ approaches 0. Conversely, larger window sizes result in the maximum value of $Q*$ as ζ approaches 1. When considering the specific value of $\zeta = 0.5$, we find that the majority of values hover around 0.45. This indicates that the Mosaic scoring function shares a limitation with Modularity, known as degeneracy. In other words, there are various distinct partitions that produce modularities so similar that heuristics cannot differentiate between them.

[1] http://www.sociopatterns.org/

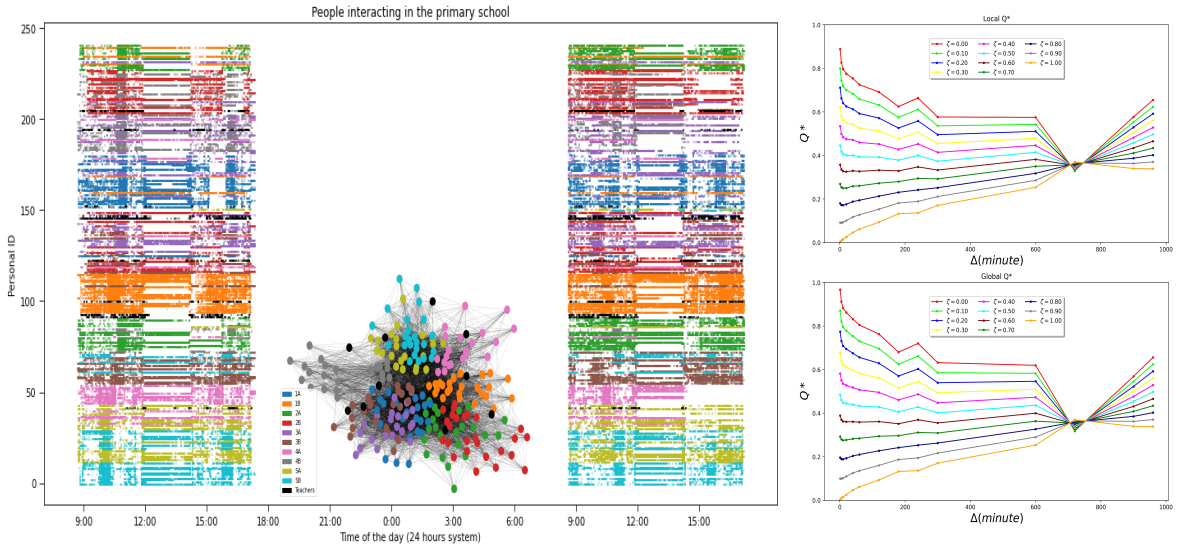

Figure 8: **Sociopatterns Primary School**: In the left panel, the link stream and its aggregated static network is visualized, with colors indicating the pupil's class. The center panel shows the static aggregated network. The right panel presents the results of the scoring function Q^*, computed using the Louvain algorithm on a snapshot of size Δ, highlighting the impact of the parameter ζ on the optimal partitioning.

9 Conclusion

When addressing optimization problems rooted in real-world scenarios within computer science, the typical approach commences with formulating the problem within a mathematical framework. Subsequently, algorithms are sought out to solve this problem efficiently. Finally, researchers assess the efficacy of their proposed methodology by comparing it against established ground truths. In our study on link stream community detection, we primarily concentrate on the initial and final stages of the optimization process. Initially, we define the problem mathematically and put forth a quality function to optimize it. Subsequently, we extend our efforts to generate a benchmark that establishes a dependable ground truth, allowing for the accurate evaluation of future algorithms' quality based on it.

Further efforts can be devolved into refining or modifying the Mosaic scoring function in terms of Modularity and smoothness to overcome possible limitations that emerged after the comprehensive analysis. For example, we can have the intuition that if we look at the backbone connectivity of two consecutive communities in time and they share no differences, it is better to merge them to have a higher inertia.

During the remaining month of my internship, I aim also to contribute to the broader research community by making my benchmark accessible to the public. This will address the existing challenge of lacking a reliable ground truth in Modular link streams.

In these five months of exploratory research in this newborn area, I extended Modularity in six forms and tried five versions of a benchmark. In each case, I started with an intuition, then implemented it with Python, looking for limitations and advantages. During this period, I started to believe this quote from Thomas Edison:

> "I have not failed 10,000 times—I've successfully found 10,000 ways that will not work.

I would like to thank my supervisors again for meaningful discussions, who gave me guidance and freedom to work independently and follow my intuition. For these reasons, I consider the internship a truly enriching experience that made me academically and personally grow.

References

[1] M. Newman, *Networks*. Oxford university press, 2018.

[2] A.-L. Barabási, "Network science," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1987, p. 20120375, 2013.

[3] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," in *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 97–110, 2000.

[4] P. Krishna Reddy, M. Kitsuregawa, P. Sreekanth, and S. Srinivasa Rao, "A graph based approach to extract a neighborhood customer community for collaborative filtering," in *Databases in Networked Information Systems: Second International Workshop, DNIS 2002 Aizu, Japan, December 16–18, 2002 Proceedings 2*, pp. 188–200, Springer, 2002.

[5] P. Holme and J. Saramäki, "Temporal networks," *Physics reports*, vol. 519, no. 3, pp. 97–125, 2012.

[6] M. Latapy, T. Viard, and C. Magnien, "Stream graphs and link streams for the modeling of interactions over time," *Social Network Analysis and Mining*, vol. 8, pp. 1–29, 2018.

[7] N. Gaumont, T. Viard, R. Fournier-S'Niehotta, Q. Wang, and M. Latapy, "Analysis of the temporal and structural features of threads in a mailing-list," in *Complex Networks VII: Proceedings of the 7th Workshop on Complex Networks CompleNet 2016*, pp. 107–118, Springer, 2016.

[8] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro, "Time-varying graphs and dynamic networks," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 27, no. 5, pp. 387–408, 2012.

[9] J. Saramäki, M. Kivelä, and M. Karsai, "Weighted temporal event graphs," *Temporal Network Theory*, pp. 107–128, 2019.

[10] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani, "The architecture of complex weighted networks," *Proceedings of the national academy of Sciences*, vol. 101, no. 11, pp. 3747–3752, 2004.

[11] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[12] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[13] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.

[14] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[15] G. Rossetti and R. Cazabet, "Community discovery in dynamic networks: a survey," *ACM computing surveys (CSUR)*, vol. 51, no. 2, pp. 1–37, 2018.

[16] M. Coscia, F. Giannotti, and D. Pedreschi, "A classification for community discovery methods in complex networks," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 4, no. 5, pp. 512–546, 2011.

[17] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, pp. 1–8, 2012.

[18] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the national academy of sciences*, vol. 101, no. 9, pp. 2658–2663, 2004.

[19] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of statistical mechanics: Theory and experiment*, vol. 2005, no. 09, p. P09008, 2005.

[20] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007.

[21] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly, "Metrics for community analysis: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 4, pp. 1–37, 2017.

[22] P. Van Mieghem, *Graph spectra for complex networks*. Cambridge University Press, 2010.

[23] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE transactions on knowledge and data engineering*, vol. 20, no. 2, pp. 172–188, 2007.

[24] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[25] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[26] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the national academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[27] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral, "Modularity from fluctuations in random graphs and complex networks," *Physical Review E*, vol. 70, no. 2, p. 025101, 2004.

[28] T. Aynaud, E. Fleury, J.-L. Guillaume, and Q. Wang, "Communities in evolving networks: definitions, detection, and analysis techniques," *Dynamics on and of complex networks, volume 2: applications to time-varying dynamical systems*, pp. 159–200, 2013.

[29] T. Hartmann, A. Kappes, and D. Wagner, "Clustering evolving networks," *Algorithm engineering: Selected results and surveys*, pp. 280–329, 2016.

[30] N. Dakiche, F. B.-S. Tayeb, Y. Slimani, and K. Benatchba, "Tracking community evolution in social networks: A survey," *Information Processing & Management*, vol. 56, no. 3, pp. 1084–1102, 2019.

[31] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *2010 international conference on advances in social networks analysis and mining*, pp. 176–183, IEEE, 2010.

[32] F. Folino and C. Pizzuti, "Multiobjective evolutionary community detection for dynamic networks," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 535–536, 2010.

[33] C. Shi, P. S. Yu, Z. Yan, Y. Huang, and B. Wang, "Comparison and selection of objective functions in multiobjective community detection," *Computational Intelligence*, vol. 30, no. 3, pp. 562–582, 2014.

[34] R. K. Darst, C. Granell, A. Arenas, S. Gómez, J. Saramäki, and S. Fortunato, "Detection of timescales in evolving complex systems," *Scientific reports*, vol. 6, no. 1, p. 39713, 2016.

[35] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics reports*, vol. 659, pp. 1–44, 2016.

[36] T. Viard, M. Latapy, and C. Magnien, "Computing maximal cliques in link streams," *Theoretical Computer Science*, vol. 609, pp. 245–252, 2016.

[37] A. Baudin, M. Danisch, S. Kirgizov, C. Magnien, and M. Ghanem, "Clique percolation method: memory efficient almost exact communities," in *Advanced Data Mining and Applications: 17th International Conference, ADMA 2021, Sydney, NSW, Australia, February 2–4, 2022, Proceedings, Part II*, pp. 113–127, Springer, 2022.

[38] A. Baudin, C. Magnien, and L. Tabourier, "Faster maximal clique enumeration in large real-world link streams," *arXiv preprint arXiv:2302.00360*, 2023.

[39] T. Viard, C. Magnien, and M. Latapy, "Enumerating maximal cliques in link streams with durations," *Information Processing Letters*, vol. 133, pp. 44–48, 2018.

[40] A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge, "Enumerating maximal cliques in temporal graphs," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 337–344, IEEE, 2016.

[41] A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge, "Adapting the bron–kerbosch algorithm for enumerating maximal cliques in temporal graphs," *Social Network Analysis and Mining*, vol. 7, pp. 1–16, 2017.

[42] M. Bentert, A.-S. Himmel, H. Molter, M. Morik, R. Niedermeier, and R. Saitenmacher, "Listing all maximal k-plexes in temporal graphs," *Journal of Experimental Algorithmics (JEA)*, vol. 24, pp. 1–27, 2019.

[43] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *nature*, vol. 466, no. 7307, pp. 761–764, 2010.

[44] N. Gaumont, C. Magnien, and M. Latapy, "Finding remarkably dense sequences of contacts in link streams," *Social Network Analysis and Mining*, vol. 6, pp. 1–14, 2016.

[45] N. Gaumont, F. Queyroi, C. Magnien, and M. Latapy, "Expected nodes: A quality function for the detection of link communities.," in *CompleNet*, pp. 57–64, 2015.

[46] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Science advances*, vol. 3, no. 5, p. e1602548, 2017.

[47] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[48] G. Rossetti, ": graph benchmark handling community dynamics," *Journal of Complex Networks*, vol. 5, no. 6, pp. 893–912, 2017.

[49] E. Abbe, "Community detection and stochastic block models: recent developments," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6446–6531, 2017.

[50] C. Lee and D. J. Wilkinson, "A review of stochastic block models and extensions for graph clustering," *Applied Network Science*, vol. 4, no. 1, pp. 1–50, 2019.

[51] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," in *Algorithms-ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11*, pp. 568–579, Springer, 2003.

[52] C. Granell, R. K. Darst, A. Arenas, S. Fortunato, and S. Gómez, "Benchmark model to assess community structure in evolving networks," *Physical Review E*, vol. 92, no. 1, p. 012805, 2015.

[53] M. Bazzi, L. G. Jeub, A. Arenas, S. D. Howison, and M. A. Porter, "A framework for the construction of generative models for mesoscale structure in multilayer networks," *Physical Review Research*, vol. 2, no. 2, p. 023100, 2020.

[54] N. Sengupta, M. Hamann, and D. Wagner, "Benchmark generator for dynamic overlapping communities in networks," in *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 415–424, IEEE, 2017.

[55] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng, "Facetnet: a framework for analyzing communities and their evolutions in dynamic networks," in *Proceedings of the 17th international conference on World Wide Web*, pp. 685–694, 2008.

[56] R. Cazabet, S. Boudebza, and G. Rossetti, "Evaluating community detection algorithms for progressively evolving graphs," *Journal of Complex Networks*, vol. 8, no. 6, p. cnaa027, 2020.

[57] S. M. Ross, *Introduction to probability models*. Academic press, 2014.

[58] T. Aynaud and J.-L. Guillaume, "Multi-step community detection and hierarchical time segmentation in evolving networks," in *Proceedings of the 5th SNA-KDD workshop*, vol. 11, 2011.

[59] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina, *et al.*, "High-resolution measurements of face-to-face contact patterns in a primary school," *PloS one*, vol. 6, no. 8, p. e23176, 2011.

[60] L. Tang and H. Liu, "Graph mining applications to social network analysis," *Managing and mining graph data*, pp. 487–513, 2010.

[61] B. Abrahao, S. Soundarajan, J. Hopcroft, and R. Kleinberg, "A separability framework for analyzing community structure," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 1, pp. 1–29, 2014.

[62] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20*, pp. 284–293, Springer, 2005.

[63] M. Rosvall and C. T. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," *PloS one*, vol. 6, no. 4, p. e18209, 2011.

[64] V. Zlatić, A. Gabrielli, and G. Caldarelli, "Topologically biased random walk and community finding in networks," *Physical Review E*, vol. 82, no. 6, p. 066109, 2010.

[65] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.

[66] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Demon: a local-first discovery method for overlapping communities," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 615–623, 2012.

A Community definitions

- **Vertex similarity**: Communities are often presumed to be clusters of vertices that share similarities. One approach to determining such similarities is to calculate the distance between every pair of vertices based on a chosen reference property.

- **Local definition**: The community can be seen as a relatively independent entity distinct from the larger network as a whole to some degree. Several methods are employed to identify communities on a local level, including complete mutuality, reachability, vertex degree, and assessing internal cohesion versus external cohesion [60].

- **Global definition**: A community corresponds to a structure of a network composed of nodes densely connected together and more sparsely connected to the rest of the network. Based on this well-known definition of a non-overlapping community, we can write: Consider the network graph G(V,E). Assume the vertices' sets $C_1, \cdots, C_k$ meet the following conditions:

$$C_1 \cup C_2 \cup \cdots \cup C_k = G$$

$$C_i \cap C_j = \emptyset, \forall i \neq j$$

$$Pr(V_{C_i}, V_{C_i}) > Pr(V_{C_i}, V_{C_j}), \forall i \neq j$$

  Then, $C = C_1, \cdots C_k$ is a partition of network G. Each set C_i represents a community in the network. $Pr(V_{C_i}, V_{C_i})$ indicates the probability that two vertices inside C_i have edges. $Pr(V_{C_i}, V_{C_j})$ indicates the probability that there exist edges between C_i and C_j.

B Other notable community detection methods

Random-walk-based algorithms have demonstrated their effectiveness in identifying communities that closely resemble the ground truth communities [61]. Pons et al. [62] introduced a hierarchical agglomerative algorithm called **WalkTrap**, which utilizes random walks to measure vertex similarity and partitions the network into non-overlapping communities. Additionally, **Infomap**, proposed by Rosvall & Bergstrom [63], allows for the discovery of hierarchical structures in networks by compressing a description of a random walker to represent actual flow in the networks. A variant of this technique, known as biased random walk [64], has also been employed for community detection.

Another notable research approach involves the **label propagation** method [65], which simulates the spread of labels among network vertices by establishing rules. For instance, the **DEMON** algorithm [66] allows each vertex to democratically vote for the communities it observes in its local view of the global system using a label propagation algorithm. Subsequently, these local communities are merged into a global collection.

C Proof of Mosaic Modularity and Static Modularity Equivalence

Theorem 3. *For a link stream $L = (V, E, T)$ with atemporal communities, the Mosaic Modularity, at any level, can be simplified to static modularity.*

The proof of this theorem focuses on global mosaic modularity. However, similar arguments apply to other forms of mosaic modularity, resulting in an equivalence be- tween all levels of mosaic modularity and static modularity.

Proof. The equation for global Mosaic Modularity is expressed as:

$$Q_{\text{MM}} = \frac{1}{2|E|} \sum_{c \in \mathcal{C}} \sum_{(i,j) \in V_c^2} W_{ij}|_{P_c} - \frac{|P_c|}{|T_L|} \frac{k_i k_j}{2|E|}|_{T_L}$$

In the absence of temporal dynamics in the link stream, each Mosaic community c exists throughout the entire time domain T of L. This allows us to simplify the formula by setting $|P_c| = |T_L|$:

$$Q_{\text{MM}} = \frac{1}{2|E|} \sum_{c \in \mathcal{C}} \sum_{(i,j) \in V_c^2} W_{ij}|_{T_L} - \frac{k_i k_j}{2|E|}|_{T_L}$$

To further simplify the equation and transform it into a node-based formulation, we replace the first two summations with a delta function since each node is assigned to only one community within its time cycle:

$$Q_{\text{MM}} = \frac{1}{2|E|} \sum_{(i,j) \in V^2} [W_{ij} - \frac{k_i k_j}{2|E|}] \delta(c_i, c_j)$$

where c_i and c_j represent the group memberships of nodes i and j respectively. □

D Proof of Local Mosaic Modularity and Average Modularity Equivalence

Theorem 4. *The Local Mosaic Modularity is equal to the average modularity, where the weight assigned to each snapshot is the total number of edges in that particular snapshot.*

Proof. Starting with the definition of local Mosaic Modularity, we can express it as follows:

$$Q_{\text{MM}} = \frac{1}{2|E|} \sum_{c \in \mathcal{C}} \sum_{(i,j) \in V_c^2} W_{ij}|_{P_c} - \frac{k_i k_j}{2m}|_{P_c}$$

If we categorize the link stream into a set of mutually exclusive sets $\{\pi_1, \pi_2, \cdots, \pi_R\}$, where communities within each set share a period P_c, we can rewrite the formula as follows:

$$Q_{\text{MM}} = \frac{1}{2|E|} \sum_{r=1}^{R} \sum_{c \in \pi_r} \sum_{(i,j) \in V_c^2} W_{ij}|_{P_c} - \frac{k_i k_j}{2m}|_{P_c}$$

Then, we name P_{π_r} as shared P_c for communities like c inside π_r. Next, we combine the sums based on the fact that within each period P_{π_r}, nodes do not change their communities:

$$Q_{\text{MM}} = \frac{1}{2|E|} \sum_{r=1}^{R} \sum_{(i,j) \in V_{\pi_r}^2} (W_{ij}|_{P_{\pi_r}} - \frac{k_i k_j}{2m}|_{P_{\pi_r}}) \delta_{\pi_r}(c_i, c_j)$$

We bring coefficient two inside the sum and define w_r as the number of edges within π_r. We can then multiply it as follows:

$$Q_{\text{MM}} = \frac{1}{|E|} \sum_{r=1}^{R} \frac{\omega_r}{2m} \sum_{(i,j) \in V_{\pi_r}^2} (W_{ij}|_{P_{\pi_r}} - \frac{k_i k_j}{2m}|_{P_{\pi_r}}) \delta_{\pi_r}(c_i, c_j)$$

Simplifying further, we obtain the following:

$$Q_{\text{MM}} = \frac{1}{|E|} \sum_{r=1}^{R} \omega_r Q_{\pi_r}$$

Since $|E| = \sum_{r=1}^{R} w_r$, we can rewrite the equation as:

$$Q_{\text{MM}} = \frac{1}{\sum_{r=1}^{R} w_r} \sum_{r=1}^{R} \omega_r Q_{\pi_r} \equiv Q_{avg}$$

□

E Scoring functions

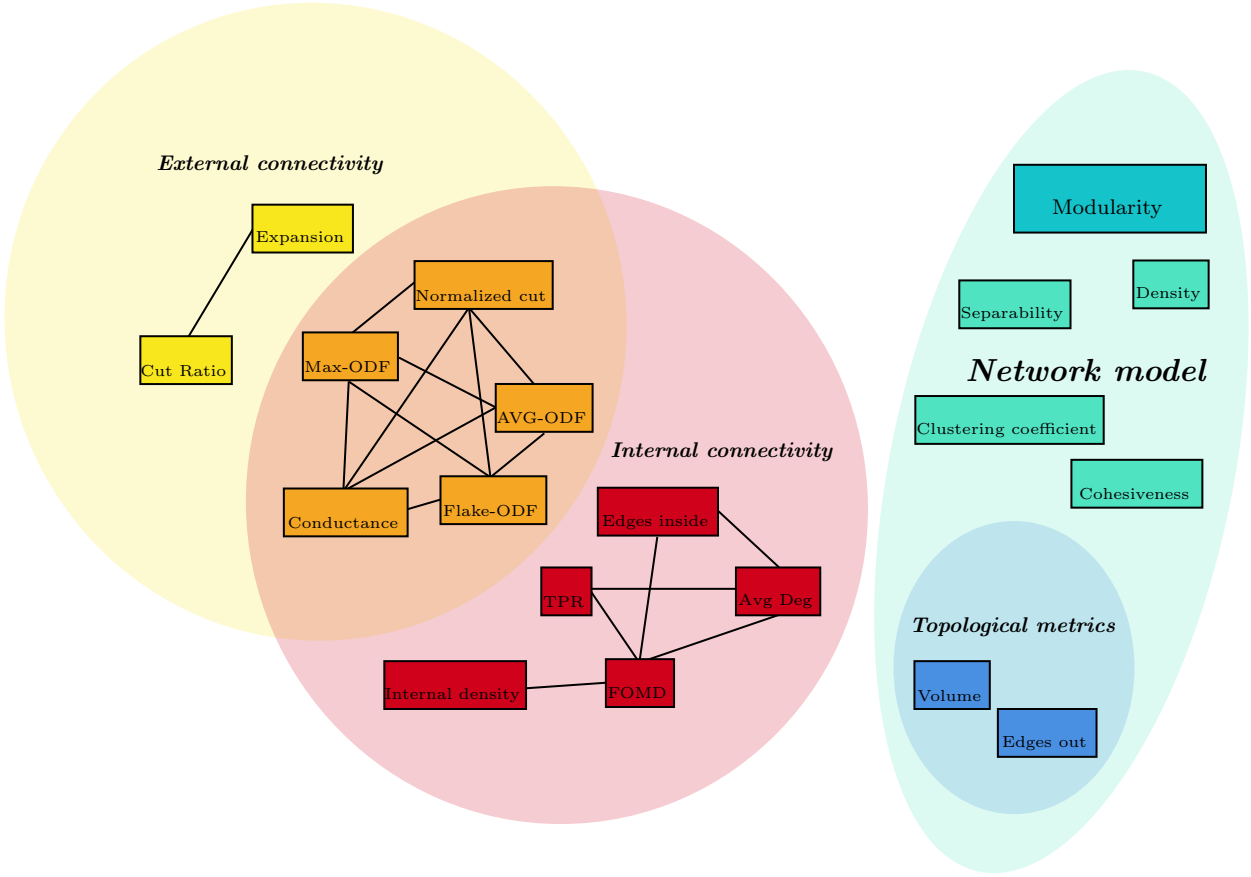

Figure 9: Scoring functions: scoring based on connectivity(internal, external, mixed) or a network model [17, 21]

A summary of these quality functions based on connectivity is represented in table 1.

	Algorithm	Formula
Internal Connectivity	Internal density	$\frac{\lvert E_{C_i}^{inter}\rvert}{\binom{\lvert V_{C_i}\rvert}{2}}$
	Edge inside	$\lvert E_{c_i}^{inter}\rvert$
	Average internal degree	$2\frac{\lvert E_{c_i}^{inter}\rvert}{\lvert V_{C_i}\rvert}$
	Triangle Participation ratio (TPR)	Fraction of nodes in C_i belong to a triad
External Connectivity	Algorithm	Formula
	Expansion	$\frac{\lvert E_{C_i}^{intra}\rvert}{\lvert V_{C_i}\rvert}$
	Cut ratio	$\frac{\lvert E_{C_i}^{intra}\rvert}{\binom{\lvert V_{C_i}\rvert}{2}}$
Mixed Connectivity	Algorithm	Formula
	Conductance	$\frac{\lvert E_{C_i}^{inter}\rvert}{2\lvert E_{C_i}^{inter}\rvert + \lvert E_{C_i}^{intra}\rvert}$
	Normalized Cut	$\frac{\lvert E_{C_i}^{intra}\rvert}{\binom{\lvert V_{C_i}\rvert}{2}}$
	Maximum ODF	$max_{u \in V_{C_i}} \frac{\lvert (u,v) \in E_{C_i} : v \notin V_{C_i}\rvert}{d(u)}$
	Average ODF	$\frac{1}{\lvert V_{C_i}\rvert} \sum_{u \in V_{C_i}} \frac{\lvert (u,v) \in E_{C_i} : v \notin V_{C_i}\rvert}{d(u)}$
	Flake ODF	$\frac{\lvert u : u \in V_{C_i}, \lvert (u,v) \in E_{C_i} : v \in V_{C_i}\rvert < d(u)/2\rvert}{\lvert V_{C_i}\rvert}$

Table 1: Connectivity quality functions for non-overlapping community detection algorithms in static graphs