ZigzagNetVis: Suggesting Temporal Resolutions for Graph Visualization Using Zigzag Persistence

Raphaël Tinarrage, Jean R. Ponciano, Claudio D. G. Linhares, Agma J. M. Traina, and Jorge Poco

Abstract*—Temporal graphs are commonly used to represent complex systems and track the evolution of their constituents over time. Visualizing these graphs is crucial as it allows one to quickly identify anomalies, trends, patterns, and other properties that facilitate better decision-making. In this context, selecting an appropriate *temporal resolution* is essential for constructing and visually analyzing the layout. The choice of resolution is particularly important, especially when dealing with temporally sparse graphs. In such cases, changing the temporal resolution by grouping events (i.e., edges) from consecutive timestamps — a technique known as timeslicing — can aid in the analysis and reveal patterns that might not be discernible otherwise. However, selecting an appropriate temporal resolution is a challenging task. In this paper, we propose ZigzagNetVis, a methodology that suggests temporal resolutions potentially relevant for analyzing a given graph, i.e., resolutions that lead to substantial topological changes in the graph structure. ZigzagNetVis achieves this by leveraging zigzag persistent homology, a well-established technique from Topological Data Analysis (TDA). To improve visual graph analysis, ZigzagNetVis incorporates the colored barcode, a novel timeline-based visualization inspired by persistence barcodes commonly used in TDA. We also contribute with a web-based system prototype that implements suggestion methodology and visualization tools. Finally, we demonstrate the usefulness and effectiveness of ZigzagNetVis through a usage scenario, a user study with 27 participants, and a detailed quantitative evaluation.

***Index Terms*—Graph visualization, persistence barcode, persistent homology, temporal graphs, temporal resolution, timeslicing.**

Received 1 June 2024; revised 17 December 2024; accepted 29 December 2024. This work was supported by the São Paulo Research Foundation (FAPESP) under Grant 2023/18026-8, Grant 2021/07012-0, Grant 2020/10049-0, Grant 2020/07200-9, Grant 2022/13190-1, and Grant 2016/17078-0, in part by the Carlos Chagas Filho Foundation for Research Support of Rio de Janeiro State (FAPERJ) under Grant E-26/204.593/2024, in part by the Fundação Getulio Vargas (FGV), and in part by the Conselho Nacional de Desenvolvimento Cientifico e Tecnologico (CNPq) under Grant 311144/2022-5. The Article Processing Charge (APC) for the publication of this work was funded by Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - CAPES (ROR identifier: 00x0ma614). Recommended for acceptance by Y. Hu. *(Corresponding author: Agma J. M. Traina.)*

Raphaël Tinarrage and Jorge Poco are with the School of Applied Mathematics, Fundação Getulio Vargas, Rio de Janeiro 22250-145, Brazil (e-mail: raphael.tinarrage@fgv.br; jorge.poco@fgv.br).

Jean R. Ponciano and Agma J. M. Traina are with the Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos 13566-590, Brazil (e-mail: jeanponciano@usp.br; agma@icmc.usp.br).

Claudio D. G. Linhares is with the Department of Computer Science and Media Technology, Linnaeus University, 35252 Växjö, Sweden (e-mail: claudio.linhares@lnu.se).

This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2025.3528197, provided by the authors.

Digital Object Identifier 10.1109/TVCG.2025.3528197

I. Introduction

TEMPORAL graphs (or temporal networks) constitute a powerful framework for modeling dynamic and complex systems from a variety of domains, including computer science, social sciences, and biology [26]. The visual representation of temporal graph data provides an intuitive and interactive way to explore complex relationships and dynamic changes over time. By using appropriate visualization techniques, researchers and practitioners are able to gain insights concerning the temporal evolution of the graph structure, to identify trends and anomalies, and detect important events that impact the system being studied.

Many studies have proposed graph drawing methods and visualizations to enhance the analysis of real-world temporal graphs. Examples include animated and timeline-based visualizations [5] (e.g., animated node-link diagrams and *Massive Sequence View* layout [62]), optimization of node positioning [35], [57], edge data sampling [70], summarization of visual representations [56].

Another important type of strategy concerns graph timeslicing, i.e., the choice of a timeslice length that defines the temporal granularity at which the graph will be studied (e.g., daily or weekly). In this context, although non-uniform timeslicing methods have been proposed in recent years [3], [47], [65], the most adopted strategy is uniform timeslicing, where timeslices of equal length represent the graph over time [25], [34], [35], [53], [63], [68], [70].

Once the timeslice length has been chosen, one divides the time interval into windows, and builds in each of them a graph, called a *snapshot*, enabling the use of standard graph analysis techniques. In this paper, in order to present a more general point of view, we will use the term *temporal resolution*, which corresponds to timeslice length, but expressed in terms of the graph's initial resolution rather than an arbitrary unit of time (both quantities are proportional).

Different temporal resolutions reveal different patterns, making the choice of resolution crucial for effective analysis. This is particularly relevant when dealing with temporally sparse graphs; in this case, global pattern identification might not be easy (or even possible) with too-fine resolutions due to the elevated number of timestamps. However, choosing a suitable temporal resolution is not a trivial task. In most cases, it requires exploratory analyses leading to empirical choices or input from a domain expert with prior knowledge of suitable resolutions.

To date, a handful of studies have tackled the problem of *automatic* resolution selection. Some are based on features computed on each snapshot (e.g., mean degree or clustering coefficient) and between consecutive snapshots (e.g., Jaccard similarity between nodes or edges); optimal resolutions are then obtained via maximization or peak detection [28], [55]. However, by considering only the consecutive snapshots, these strategies miss information regarding the graph's global behavior. Incorporating larger-scale dynamics has been explored, notably by finding the largest intervals over which features "persist", through minimization of a trade-off information/variance [17], [21], [44], [55], [58], [59]. However, these methods require additional hyperparameters or only study snapshots through specific features, thereby losing the structural information of the underlying graphs.

In order to study a temporal graph as a whole, and not fragmented into isolated snapshots, we will use tools from Topological Data Analysis (TDA), and more particularly persistent homology (PH) and zigzag persistent homology (zigzag PH). This theory, which aims to capture relevant topological and geometric features from datasets, has already been applied to a wide range of problems concerning the analysis and visualization of graphs [2]. Although its application to dynamic graphs is still in its early stages, a common methodology is emerging: gathering the snapshots into a zigzag module, and analyze its persistence barcode [22], [30], [31], [41], [42]. We emphasize that, in this context, one of the main benefits of employing zigzag PH instead of ordinary PH is that the former allows tracking the appearance, disappearance, merge, and split of connected components, while the latter only allows appearance and merge. To the best of our knowledge, no study has applied PH to the problem of temporal resolution selection.

Our contributions: This paper introduces ZigzagNetVis, a methodology that employs zigzag PH to suggest potentially relevant temporal resolutions for visualizing temporal graphs. These resolutions are identified based on the degree of topological change they induce. As we will discuss throughout the article, leveraging ideas from TDA yields new valuable insights for this problem.

First of all, the structure of zigzag module, by including not only pointwise information (snapshots) but also dynamic information (their relationship), allows one to study a temporal graph as a whole. We propose a topological interpretation of the effect of changing resolution, classified as *timestamps shift* or *structural change*.

Second, compared to certain features used in the literature, PH can be clearly interpreted and visualized through the *persistence barcodes*, a structure that, in the same vein as a tracking graph, captures the dynamics of a temporal graph. An important feature of the barcodes is that we can compare them via the *bottleneck distance*. Based on this idea, we devise an explainability pipeline that spots the most important differences between resolutions.

In addition, to enhance the visual analysis, ZigzagNetVis incorporates a novel timeline-based visualization inspired by the persistence barcodes. It was specifically designed to enhance the analysis of connected components' structure and evolution.

Last, we address an important related issue: the question of selecting an "optimal" resolution is ill-posed. Indeed, different resolutions may be relevant for uncovering different patterns. Furthermore, no reference benchmark is available. We contribute to this problem by bringing together various results scattered in the literature, and by comparing our approach with other traditional methods through an empirical study of two real-world datasets.

In summary, our main contributions are: (i) A layout-agnostic method that leverages zigzag PH to suggest potentially relevant temporal resolutions for graph visualization; (ii) An explainability method for identifying the major topological differences caused by two different resolutions; (iii) A timeline layout inspired by the barcodes from TDA and which depicts the evolutionary behavior of the graph's connected components; (iv) The prototype of a web-based system with interactive linked views to assist in the graph analysis; (v) Evaluation using a usage scenario, a user study (27 participants), and a quantitative comparison with existing features.

II. Background and Related Work

A. Temporal Graphs and Timeslicing

Timeslicing: Let N be an integer representing the maximal time value. A *temporal graph* is a graph G and a collection of pairs (e, t), where e is an edge of G and t is an integer in $[0, N]$. In practice, e represents an interaction between its nodes, occurring at time t. This formalism underpins many models of dynamic phenomena, ranging from communication networks to biological mechanisms [26]. The value $r_0 = 1$ is called the *initial resolution* and the integers $t \in [0, N]$ are referred to as the *initial timestamps*. As in [36], the initial resolution represents the time interval in which the graph data was originally recorded, e.g., timestamps in $r_0 = 1$ span a 1-day interval in the Enron network [29] and 20 seconds in the Primary School network [23].

In the context of temporal graph analysis, one is interested in the graphical representation and analysis of temporal graphs. To this end, the usual approach (used, e.g., in [35], [53], [68]) consists in choosing an integer $r > 1$, regularly cutting the interval $[0, N]$ into $M = \lfloor N/r \rfloor$ sub-intervals $[kr, (k+1)r]$, where $k \in [0, M]$ is an integer, and building $M + 1$ graphs $\{G_k\}_{k=0}^{M}$. Each graph G_k contains the edges (e, t) where $t \in [kr, (k+1)r]$, and and the nodes of these edges. In other words, we build the graphs by collecting the edges active during the corresponding intervals and discarding the isolated nodes. The parameter r is called the *resolution*, and the integers $k \in [0, M]$ are the corresponding *timestamps*. In what follows, we will refer to this process as *partition timeslicing*. The first and second rows of Fig. 1 represent the collection of graphs obtained via this process for resolutions 1 and 2, respectively. One observes that, for the initial resolution, there are two timestamps where the blue nodes are not present. This phenomenon disappears at resolution 2. In general, as the resolution increases, both the number of edges and nodes present at each timestamp may grow.

We will also consider another cutting process, called *sliding-window timeslicing* [42], [63]. As before, let r be a resolution parameter. For each initial timestamp k, we build the graph G_k whose edges are those with the activation time t contained in $[k - r/2, k + r/2]$. Unlike partition timeslicing, which separates edges into disjoint intervals, sliding-window timeslicing allows activation intervals to overlap (see Fig. 1). Note that the graphs obtained for an even resolution $r = 2s$ are identical to

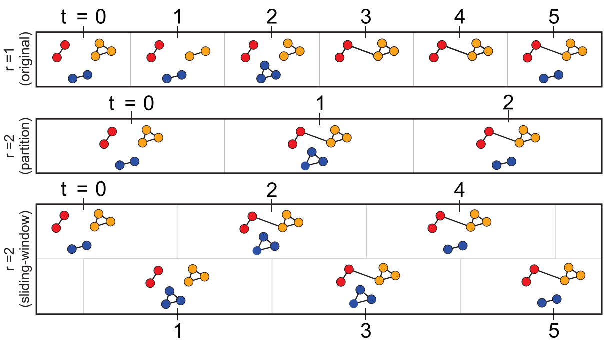


Fig. 1. A temporal graph of initial resolution 1 (first row) and its partition and sliding-window timeslicing at resolution $r = 2$ (second and third rows).

those obtained for the next odd resolution $r = 2s + 1$, since the edges' activation times are integers. Thus, in the rest of this article, we will consider sliding-window timeslicing for even values of resolution only.

A characteristic shared by these two approaches is that all timeslices have the same length, known as *uniform timeslicing*. Although not as popular, the idea of *non-uniform timesciling* has also been considered in recent years. This type of timeslicing allows timeslices with different lengths over time. In graph visualization, we may find timeslices whose lengths depend on how many consecutive timestamps have similar graph structure [3] and how active (in terms of bursts of events) the graph is over time. As an example of this last case, while Ponciano et al. [47] use long timeslices to represent intervals with bursts of events, Wang et al. [65] adopt short timeslices to analyze such intervals. In this paper, we focus on uniform timeslicing, the most commonly adopted approach [25], [28], [34], [63], [68], [70]. In this context, the choice of the resolution r can strongly impact the analysis: an overly coarse cut erases short-duration phenomena, while an overly fine cut disrupts continuous phenomena [47]. This impact has been studied in [32], [53], by comparing features of the resulting temporal graphs (e.g., the average degree of their nodes or the size of their connected components). It is worth noting that the problem persists in the context of non-uniform timeslicing, since such methods often rely on selecting an initial resolution, that has to be chosen wisely [44]. Although it is a crucial parameter, the resolutions are often chosen heuristically, and the question of their selection is barely raised. Keeping in mind the potential applications of temporal graph analysis, our work aims to describe and implement an automatic method of resolution selection.

Automatic resolution detection: Among the works that tackle this problem, two strategies are found. The first is to define the parameter via the maximization of features' values or the minimization of trade-offs between features [28], [58], [59]. For example, MoNetExplorer [28] is a visual analytic system that ranks candidate timeslice lengths (i.e., window sizes) based on three motif-based features: motif stability, motif fidelity, and motif clusterness, which are computed for each candidate. Not every possible length becomes a candidate; only those following a predefined base time unit. For example, if the base is a month, there are 12 candidates (lengths ranging from one to twelve months). Other examples include [58], where the resolution is found by minimizing a trade-off between the compression ratio and the variance of a sequence of features, as well as [59], which seeks to minimize the variance of several features (positional dynamicity, degree, closeness, and betweenness centrality).

The second strategy involves abrupt change detection in time series, for instance, from the Jaccard distance between snapshots [44], from compression ratios [21] or from coefficients of convergence [55]. In the same vein, [16] detects peaks from the autocorrelation of the time series of features. Our method employs this strategy since it can be naturally combined with features from Topological Data Analysis. To highlight the value of our method, we provide a precise theoretical justification and an extensive empirical analysis.

As pointed out by the studies above [32], several distinct resolutions may be relevant for analyzing a temporal graph. Therefore, the question of a "correct" interval length is ill-posed. In this work, we circumvent this issue by suggesting various values — without relying on predefined user-selected candidates or other parameters — and explaining their relevance.

B. Persistent Homology Applied to Graphs

The mathematical tools used in this article are drawn from Topological Data Analysis (TDA), a field at the intersection of computational geometry, algebraic topology, and data analysis [14]. Persistent homology (PH), one of its most popular techniques, allows us to infer homology groups of a dataset [43]. It has been applied to a wide range of problems, including medicine, physics, computer vision, and machine learning, among others. However, its application to the study of temporal graphs is relatively new.

Analysis of graphs: PH is mainly used when the dataset is a point cloud, an image, a scalar function, or a graph. We refer the reader to the survey [2] for an extensive presentation of how TDA has been applied to graph analysis. As an intermediary construction between the input data and PH, the user must choose a *filtration*, i.e., a non-decreasing family of subspaces that covers the data. To this end, several popular filtrations exist, such as the Rips filtration.

However, in our context, the input data is not a single graph but a sequence of graphs, and PH cannot be used directly. This is due to the fact that the sequence may not be non-decreasing: as time progresses, nodes or edges may disappear. As a consequence, the temporal graph may not form a filtration. To get around this problem, one strategy involves applying PH to each graph in the sequence and analyzing the results, as [25] does in the context of temporal graph exploration. Although it allows exhibiting global properties of the data, this method does not use the full potential of PH, since persistence is computed only at the level of each graph, and not throughout the sequence. In particular, no temporal information is contained in the persistence diagram. Moreover, this method lacks the theoretical guarantees of TDA, such as stability.

As an alternative, one can use *zigzag persistent homology* (zigzag PH), which we will describe in Section II-C. This variation of PH has already been used in the context of topological bootstrapping, thresholding, and parameter selection. Unlike ordinary PH, it is based on the notion of *zigzag filtrations*, which do not have to be non-decreasing. In particular, it can be applied to a temporal graph, allowing one to compute the persistence of the sequence of graphs all at once [22], [30], [31], [41], [42]. By computing the persistence barcode, the main object of TDA and described in the next section, one can detect the global behavior of the graph (e.g., the evolution of its connected components, periodic or chaotic patterns). Our work brings these ideas to the problem of resolution selection by investigating the link between the stability of zigzag persistence modules and the choice of a resolution. In addition, we also devised a new visualization layout based on PH.

We point out that, in this article, the topology of the graphs will be studied through the lenses of the homology H_0, that is, the connected components. Ordinary PH enables us to track these components over time, limited to the case of appearance and merge. In addition, employing zigzag PH allows one to study the disappearance and splitting of connected components, phenomena that occur in temporal graphs. As exemplified by numerous articles in the TDA literature, H_0 contains sufficient information to solve certain problems [7], [13], [45], [46]. Furthermore, in the particular context of temporal graph visualization, it has been reported that the analysis of connected components allows for a rich exploration of the data [37], [38], [39], [66]. Since the purpose of this article is to visualize the formation of groups within networks, i.e., of connected components, we will focus on H_0. The higher homology groups H_i, $i > 0$, although they could capture additional information (e.g., tunnels, voids), are beyond the scope of the paper.

Visualization: TDA has also seen applications in the context of (non-temporal) graph visualization. By quantifying the strength of connections between the nodes of the graph, TDA can improve force-directed layouts and facilitate interaction with them [20], [57]. One may also consider the connectivity between communities, resulting in new representations, such as those in [37], [51].

In contrast, applications of TDA to the visualization of *temporal* graphs are few. The first work is found in [38], where the persistence diagram is used as a means to visualize the connected components generated by a scalar field. However, in this case, PH is computed at the level of each snapshot, and therefore does not capture information about the dynamics of the data.To our knowledge, only [25], [42] propose visual layouts incorporating temporal information. The former consists of a curve, exhibiting patterns and changes in behavior over time. However, it does not provide information concerning the topology of the graphs at each snapshot. The latter layout uses the persistence barcodes given by the zigzag PH. It displays the topological properties of the graphs at each timestamp and shows how they evolve over time. Nevertheless, in some contexts, focusing only on graphs' topological properties, such as their number of connected components, can be too coarse and make analysis and visualization difficult for the user. An important contribution of our work is to enhance this representation by incorporating information about the size and composition of the connected components. These enhanced barcodes, that we call "colored barcodes", show promising results for graph visualization.

We draw the reader's attention to the fact that a close connection can be established between the persistence barcodes offered by TDA and certain popular visualization techniques. In particular, the persistence barcodes of (ordinary) PH can be deduced from the merge tree of the data, and that of zigzag PH from its tracking graph [37], [66]. In particular, the barcode graph [18] or formigram [30], [31], a handy tool of TDA, can be understood as a tracking graph. This connection is studied in further detail in Section V-B. Similarly, the visualization proposed in this paper (Fig. 9(A)), which incorporates additional information into the persistence barcodes, is related to the idea of nested tracking visualization [38], [39]. Both approaches draw flows between adjacent timestamps to represent events like merges and splits (in our case, triggered by user interaction). This last connection, however, only concerns visual representation, since these tools are designed to handle different information (nested components vs. disjoint components). Section V-B discusses our design decisions and explains in more detail why nested tracking visualizations are not applicable in our case.

C. Zigzag Persistent Homology

We now succinctly introduce the topological tools used in this paper, and refer the reader to [14] for a thorough presentation.

Persistence modules: Zigzag persistent homology, introduced in [11], is based on the notion of *simplicial homology*. Given an integer $i \geq 0$, the i^{th} homology functor H_i is an operator that takes as input a graph G, and returns a vector space, denoted $H_i(G)$, which contains topological information about G. As already discussed, we will only consider $H_0(G)$, the group of connected components, since it already enables a rich analysis of the graph's structure. It is a vector space whose dimension is equal to the number of connected components of G.

To define a zigzag PH, one has to first build a *zigzag filtration*, that is, a sequence of graphs, such that for each pair of consecutive graphs, one of them is included in the other. In order to build such a filtration, consider the sequence of graphs $\{G_k\}_{k=0}^{M}$ defined in the previous section, using the partition or sliding-window timeslicing. By considering the union graph $G_k \cup G_{k+1}$ for all the pairs of consecutive graphs, one obtains a zigzag filtration

$$G_0 \hookrightarrow G_0 \cup G_1 \hookleftarrow G_1 \hookrightarrow G_1 \cup G_2 \hookleftarrow G_2 \hookrightarrow \dots$$

In this filtration, one is able to *track the evolution* of the connected components: how they merge, split, appear or disappear.

By applying the H_0-homology to this filtration, the graphs are transformed into vector spaces, and the inclusions into linear maps:

$$\begin{aligned} H_0(G_0) \rightarrow H_0(G_0 \cup G_1) \leftarrow H_0(G_1) \\ \rightarrow H_0(G_1 \cup G_2) \leftarrow H_0(G_2) \rightarrow \dots \end{aligned}$$

This sequence forms a *zigzag persistence module*, an algebraic structure that condenses all the information concerning the evolution of the connected components. For instance, one reads directly from these maps whether a connected component splits or is preserved; similarly, one reads whether two connected components merge.

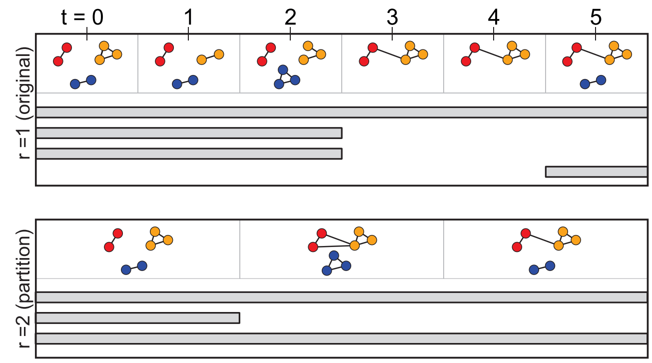


Fig. 2. Barcodes associated with a temporal graph at resolution 1 and 2. Each horizontal bar refers to a connected component throughout time.

Barcodes: To each persistence module is attached a *persistence barcode*, denoted $\mathscr{B}$. It is a collection of intervals $[b, d]$, called *bars*. They are interpreted as follows. For each timestamp k, the number of bars present at this time is equal to the number of connected components in the graph G_k. Moreover, we can see how these connected components evolve: To a bar $[b, d]$ corresponds a connected component of the graph born at time b (either because new points appeared in the graph, or because an existing component split in two) and died at time d (either because the points that compose it disappeared, or because it merged with another component). The barcode is the main object of TDA, and can be understood as a visual representation of persistence modules.

As an example, we give in Fig. 2 the persistence barcodes associated with a temporal graph at resolutions 1 and 2, as in Fig. 1. Let us analyze the first barcode. It contains a long bar [0,5], indicating that there is a connected component that persists all along the filtration. We may think of it as representing the nodes colored in orange, or red. Moreover, there are three smaller bars, depicting connected components that survive for a shorter time: one bar [0,2] represents a component that merges with another (the red nodes with the orange nodes), another bar [0,2] represents a component that disappears (the blue nodes) and reappears at $t = 5$. Besides, the second barcode of Fig. 2 contains only three bars. Indeed, in the corresponding filtration, the blue nodes are always present, resulting in a long bar [0,5] in the barcode.

It should be pointed out that the barcode does not allow one to identify directly which connected components the bars represent. In certain cases, in the presence of many bars, for instance, this task can be difficult to perform visually. One part of our work consisted in defining an improved version of the barcode, called the colored barcode, which allows us to fix this problem (see Section V-A).

Bottleneck distance: Another fundamental feature of TDA is the possibility of comparing two persistence barcodes, through the notion of *bottleneck distance*. In a few words, this distance seeks a pairing between the bars of the barcodes and computes the largest distance between a bar in the first barcode and one in the second. The exact definition is given in our supp. material (Section A). With respect to the bottleneck distance, two barcodes are close if the large bars of one can be matched with the large bars of the other, the short bars being forgotten. Fig. 3 shows a pairing between the barcodes in Fig. 2. The most distant bars in this pairing are the two bottom ones, [0,2] and [0,5]. The distance between these bars is 3, which is also the bottleneck distance, denoted $d_{\mathrm{B}}(\mathscr{B}, \mathscr{B}')$.

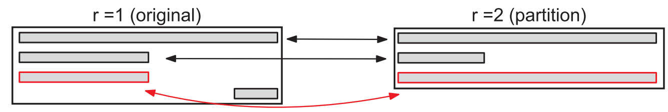


Fig. 3. A pairing between the barcodes of Fig. 2. We outline in red the most distant paired bars (distance 3), causing the bottleneck distance.

The bottleneck distance lies at the core of our method and will be used as a means to select resolutions in Section IV. Namely, we will compare the temporal graphs coming from two different resolutions via the bottleneck distance between the persistence barcodes coming from their zigzag filtrations. This distance computes the global topological agreement between these temporal graphs, allowing us to determine whether they are similar or not, just as in the context of abrupt change detection. In addition, the bottleneck distance offers two advantages. First, it allows for a theoretical treatment: we will study in Section IV-B what values of distance are to be expected, and when they indicate a relevant change. Secondly, and as a consequence of its definition, the bottleneck distance is always caused by a pair of bars or a bar alone. From a practical point of view, one can identify which nodes of the graph are responsible for the topological difference between two barcodes. Based on this observation, we will describe an explainability pipeline in Section IX.

III. Design Tasks and Workflow

Design tasks: Besides suggesting temporal resolutions, we seek to effectively explore the graph, and identify global and local behaviors and patterns, under a given temporal resolution. In that sense, we designed our visual components and interaction to meet high-level tasks derived from low-level tasks and dimensions proposed in Bach et al.'s taxonomy for temporal graph exploration [4].

Specifically, we combine the three task dimensions described in this taxonomy: *temporal/when* (easy identification and reaching of specific time steps); *topological/where* (easy identification, situation, and tracking of elements with properties of interest); and *behavioral/what* (easy understanding of the behaviors and changes that affect elements of interest). These dimensions help generate the following tasks, which should be satisfied during the graph analysis under any temporal resolution.

T1: Analyze particular groups of elements (entire network, connected components, or nodes) in terms of identification, situation, and inspection at a given time of interest.

T2: Analyze the temporal evolution of particular groups of elements, identifying, e.g., the addition or deletion of elements and abrupt increases or decreases of an element property (referred to as *peak* or *valley* events in Ahn et al.'s taxonomy [1]).

T3: Identify and compare structural changes that occur at particular times of interest. In addition to the *when*, *where*, and

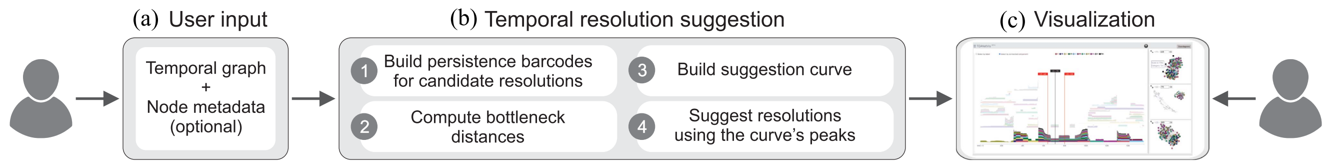


Fig. 4. ZigzagNetVis workflow. (a) Users input a temporal graph and node metadata (optional). (b) We suggest resolutions using a four-step procedure. (c) Users visualize the graph using any resolution through the colored barcode and node-link diagrams, visualizations that compose our prototype.

what dimensions from Bach et al.'s taxonomy, we further consider *why* and *how* task descriptions from Brehmer & Munzner's multi-level typology of visualization tasks [10]. From the *why* point of view, our tasks enable *discoveries*, which include the generation and verification of hypotheses. To achieve that, users first *locate* groups of elements of interest (tasks T1, T2) or at particular times (task T3). Alternatively, they can freely *explore* the visualization to find elements/times of interest (e.g., based on global patterns or anomalies). Once these are found, users may *identify*, *compare*, and *summarize* elements or patterns (T1-T3). From the *how* perspective, our views will meet the tasks by *encoding* the network data and by providing *manipulation* methods such as *selection*, *navigation*, and *filtering*. They will also *introduce* new elements to the visualization by *importing* network data on demand.

Workflow: As illustrated in Fig. 4(a), users first input a temporal graph and its node categorical metadata (optional). The resolution suggestion then proceeds as follows (Fig. 4(b), details in Section IV): we build persistence barcodes for every candidate resolution (predefined range of values, e.g., [1,100]); we compute the bottleneck distance between pairs of barcodes, and build a suggestion curve using the distances. Resolutions are then suggested based on the curve's peaks. Finally, users visualize the graph under any resolution by using our proposed layout — the *colored barcode* (Section V-A) — and associated node-link diagrams, visualizations that compose our system prototype (Fig. 4(c), details in Section V-C).

IV. Temporal Resolution Suggestion

A. Description of the Method

As discussed above, the choice of a resolution significantly impacts the analysis of a temporal graph. In practice, one wishes to select an "optimal" resolution. However, the problem is ill-posed: various resolutions may be relevant, leading to different analyses. To circumvent this issue, our strategy selects a collection of resolutions, each of which reveals different behaviors of the temporal graph.

Let us consider an initial set of resolutions $\{r_0, \dots, r_n\}$, to be tested, and a parameter m, the number of requested resolutions. Our method consists in partitioning this set into m subsets of consecutive resolutions,

$$\{r_{i_0} = r_0, \dots, r_{i_1}\}, \ \{r_{i_1}, \dots, r_{i_2}\}, \ \dots, \\ \{r_{i_{m-1}}, \dots, r_{i_m} = r_n\}, \tag{1}$$

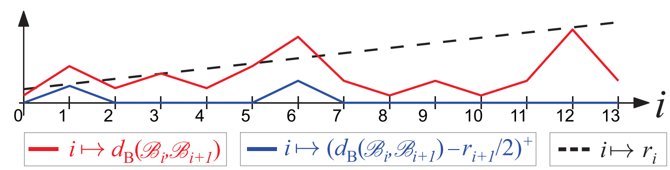


Fig. 5. A suggestion curve in red, its corresponding normalized suggestion curve in blue (for partition timeslicing), and the curve $i \mapsto r_i$ in black. .

where each subset consists of resolutions for which the temporal graphs exhibit similar behavior. We will quantify this similarity using zigzag PH, as explained in the next paragraph. As a last step, we will choose a resolution in each of these subsets — for instance, the first ones, $r_{i_0}, \dots, r_{i_{m-1}}$ — therefore yielding an exhaustive sample of all possible behaviors exhibited by the temporal graph.

Our method for obtaining a partition as in (1) consists in comparing each pair of consecutive resolutions r_i and r_{i+1}, and in detecting abrupt changes in the corresponding temporal graphs. This detection is performed using zigzag PH, as follows. First, we perform timeslicing on the temporal graph G for both resolutions, using partition or sliding-window, as described in Section II-A. Second, we compute the corresponding persistence barcodes $\mathscr{B}_i$ and $\mathscr{B}_{i+1}$, as well as their bottleneck distance $d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1})$, as described in Section II-C. Gathering all the bottleneck distances yields a sequence

$$d_{\mathrm{B}}(\mathscr{B}_0, \mathscr{B}_1), \ d_{\mathrm{B}}(\mathscr{B}_1, \mathscr{B}_2), \ \dots, \ d_{\mathrm{B}}(\mathscr{B}_{n-1}, \mathscr{B}_n),$$

which we represent as a curve, drawn in red in Fig. 5. We refer to it as the *suggestion curve*.

On the suggestion curve, peaks correspond to consecutive resolutions for which the associated barcodes are significantly different, which we interpret as structural topological changes in the temporal graphs. Finally, we identify the peaks of this curve and use them as separators to obtain the partition of (1). We give further explanations in the next section.

In a nutshell, our methodology employs the bottleneck distance between consecutive resolutions as a feature to perform abrupt change detection. While change detection based on features is common in temporal graph analysis (see, e.g., [3]), incorporating PH offers several advantages. First, thanks to the high interpretability of PH, we can give a heuristic analysis in Section IV-B, already yielding important insights. Moreover, as studied further in the supplementary material, the bottleneck distance appears to be a stable and relevant quantity, gathering information from various other features of the literature.

B. Timestamps Shifts and Structural Changes

In the previous section, we have built the suggestion curve $i \mapsto d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1})$. In order to identify relevant peaks of this curve, we need to give some comments regarding the values it can take.

Partition timeslicing: Let us first consider that we have chosen the partition timeslicing. By going from resolution r_i to r_{i+1}, one alters the timestamps: a timestamp for the first resolution will be at a distance at most $r_{i+1}/2$ of a timestamp for the second one. Consequently, we expect that the bars of the persistence barcode will be displaced by a distance of at most $r_{i+1}/2$. This interpretation leads us to distinguish two values of the bottleneck distance.

- If $d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1}) \leq r_{i+1}/2$, the distance is merely caused by artificial changes coming from the modification of the timestamps' values. We call it *timestamps shift*.
- If $d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1}) > r_{i+1}/2$, the distance is no longer just caused by the displacement of the timestamps: we consider that the temporal graph has undergone a *structural change*.

In order to estimate the structural changes only, we must detect the values of the suggestion curve that exceed $r_{i+1}/2$. In other words, we seek the positive values of the *normalized suggestion curve:*

$$i \longmapsto (d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1}) - r_{i+1}/2)^+$$

where $(\cdot)^+$ denotes the positive part of a real number. This curve is represented in Fig. 5. In this example, we would detect the resolutions r_2 and r_7 as values that cause structural changes since they are the first resolutions after the peaks occurring at r_1 and r_6.

Fig. 2 provides another example. Going from resolution $r_i = 1$ to $r_{i+1} = 2$, we have seen previously that the bottleneck distance is equal to 3, greater than $r_{i+1}/2 = 1$, hence we observe a structural change. It is caused by the two blue bars merging together. If we had considered only the red and yellow bars, we would have observed a bottleneck distance of 2, i.e., a timestamps shift.

Sliding-window timescling: We now turn to the case of sliding-window timeslicing. By going from resolution r_i to r_{i+1}, the activation windows of the edges are only altered by a value $(r_{i+1} - r_i)/2$. Consequently, we expect that the bars of the barcode will be displaced by a distance of at most $(r_{i+1} - r_i)/2$. This leads us to define a *timestamps shift* if $d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1}) \leq (r_{i+1} - r_i)/2$, and *structural change* if $d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1}) > (r_{i+1} - r_i)/2$. Accordingly, we define the *normalized suggestion curve* as

$$i \longmapsto (d_{\mathrm{B}}(\mathscr{B}_i, \mathscr{B}_{i+1}) - (r_{i+1} - r_i)/2)^+.$$

As before, we identify structural changes through its positive values.

In practice, users can select the preferred timeslicing method prior to applying the resolution suggestion technique. However, the results obtained for partition or sliding-window may be different. In the case of partition, a particularly inconvenient phenomenon occurs. Two bars of the barcode might merge between r_i and r_{i+1}, provoking a structural change, and then split between r_{i+1} and r_{i+2}, again provoking a structural change. We call this phenomenon *instability*, and we explain the situation in more detail in our supp. material (Section B.1). Consequently, we recommend that users use sliding-window timeslicing, and we make this choice in the rest of this article, except when stated otherwise.

Peak detection: In real-life examples, the normalized suggestion curve may contain many positive values. However, returning all the corresponding resolutions to the user would not be relevant. Instead, we choose to return only the most prominent peaks of the curve. In practice, prominence is computed using the package `signal` of `scipy`. We return only $m = 5$ maxima, five being an arbitrary value that we found suitable. We will give concrete outputs of our algorithm on eight temporal graphs in Section IX.

Other distances: In TDA, one chooses a distance according to the context: while the bottleneck distance calculates the maximal discrepancy between two barcodes, the Wasserstein distance incorporates all perturbations. The latter option is of interest, for instance, when low-persisting features matter [14]. In contrast, our work aims to detect structural changes, which are evidenced by the perturbation of a single bar of the barcode. Therefore, the bottleneck distance appears as a natural choice (see, for instance, Fig. 2). This observation is supported by Section C.2.4 of our supp. material, where it is shown that the Wasserstein distance leads to less interpretable results. In the same vein, one could use, instead of the bottleneck distance, any feature that quantifies the proximity between two temporal graphs. To this end, many quantities exist, such as those presented in Section C.2.3 of our supplementary material (e.g., mean degree, density, or burstiness). However, they all appear to either lack stability or provide limited information. Our experimental study shows that the bottleneck distance acts as a relevant trade-off between stability and information, "incorporating" several popular features.

V. Visualization

A. Colored Barcode Layout

In practice, the barcodes of TDA may not contain enough information: one is not able to identify which nodes are part of which bar. Indeed, the barcode is built from the homology groups $H_0(G_k)$ of the graphs, where the information about the nodes has been lost. A contribution of our work is to adapt and implement an algorithm that identifies the nodes that compose each bar.

Nodes identification: Consider a temporal graph, the sequence of graphs $\{G_k\}_{k=0}^{M}$ obtained by timeslicing, and the H_0-barcode $\mathscr{B}$ of its zigzag filtration. We wish to find, for each bar $I \in \mathscr{B}$ and each timestamp $k \in I$, a connected component C_k^I such that

- for each timestamp k, if $\mathscr{B}^k$ denotes the set of bars living at time k, then the set $\{C_k^I \mid I \in \mathscr{B}^k\}$ is a partition of the set of nodes of G_k,
- for each bar $I \in \mathscr{B}$ and each $k \in I$ such that $k+1 \in I$, we have $C_k^I \cap C_{k+1}^I \neq \emptyset$.

The first point guarantees that we do not attribute the same node to two bars at the same timestamp, and the second point

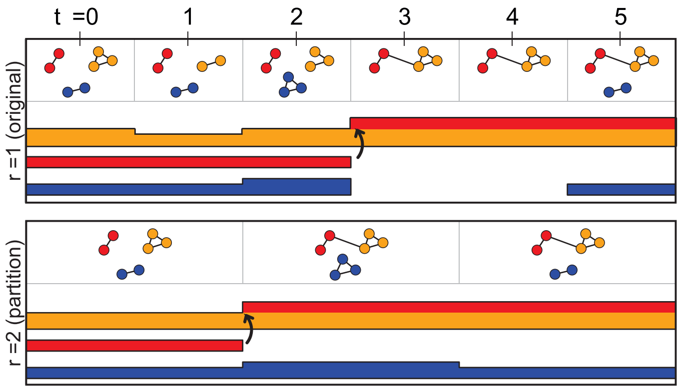


Fig. 6. Colored barcodes corresponding to the barcodes of Fig. 2. Vertical arrows depict component merging.

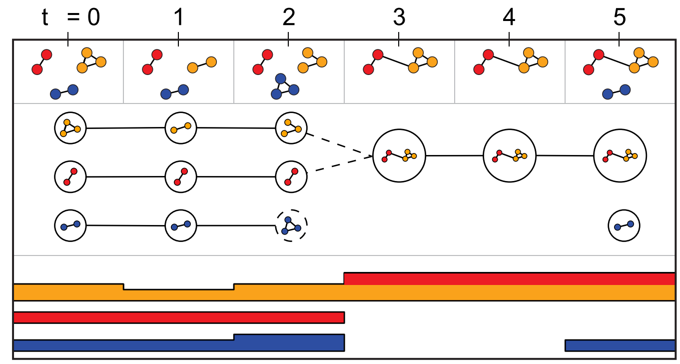


Fig. 7. On top of a zigzag filtration (top) is built the barcode graph (middle). By keeping the nodes' information, an adaptation of [18, Algorithm 1] enables us to compute the colored barcode (bottom).

that, within a bar, we choose a sequence of connected components that are connected one to another. Such a choice is possible as a consequence of previous work, which is detailed below.

Once the node identification has been done, this information can be incorporated into the persistence barcode. By attributing to each node or cluster of nodes a color (representing, e.g., node metadata information), we paint the bars in accordance with the nodes it contains. We also vary the height of the bars to indicate the number of nodes. We call this representation the *colored barcode*. In case it is not possible to assign different colors to nodes (e.g., when there are no node metadata), we use a single color and only consider the variation of the heights of the bars.

We give in Fig. 6 two examples of colored barcodes, where the nodes are divided into three clusters: red, orange, and blue. They correspond to the (non-colored) barcodes of Fig. 2. On the first colored barcode (Fig. 6(top)), one reads that a connected component persists throughout the filtration, initially composed of orange nodes and later receiving the participation of red nodes. One can also visualize the connected component formed by the blue nodes, which disappears and reappears at $t = 5$.

The choice of nodes composing each bar is not unique. For instance, on the first barcode of Fig. 6, the long bar starts with only orange nodes, until $t = 3$, where red nodes connect. In this example, one could have chosen to start this long bar with red nodes instead. The analysis of the colored barcode, however, is independent of this choice. The user must keep in mind that, when two connected components merge, only one of the two has been arbitrarily chosen to appear at the beginning of the corresponding bar.

Algorithm: We now turn to the implementation of nodes identification, based on the work of Dey and Hou [18]. As described in the article, there exists an intermediate construction between the zigzag filtration and the persistence barcode, called the *barcode graph*. It is built recursively by studying how the temporal graph $\{G_k\}_{k=0}^{M}$ evolves: creating, removing, merging, or splitting connecting components. Formally, each node of the barcode graph is associated with a connected component of G_k at a certain time k. Moreover, an edge is added between two components at times k and $k+1$ if they share a node (see Fig. 7). We draw the reader's attention to the fact that the barcode graph is a tracking graph (see Section II-B).

Algorithm 1 of the aforementioned article allows one to deduce, from the barcode graph, the persistence barcode of the zigzag filtration. To do so, the authors recursively build the *barcode forest*, a complementary construction. For the most part of the algorithm, when iterating through the filtration, five events may happen: ENTRANCE, DEPARTURE, NO-EVENT, MERGE and SPLIT. They respectively represent that a node entered the filtration, that a node left the filtration, that an edge entered or left without changing the topology of the graph, that an edge entered the filtration provoking two connected components to merge, or that an edge left the filtration provoking a connected component to split in two. Only DEPARTURE and MERGE provoke the appearance of a new bar.

In our context, since we wish to identify the nodes that compose the bars, we incorporate a further step in this procedure. During the event DEPARTURE (as the dashed node of Fig. 7), we simply collect the connected components written on the path, and add this information to the bar of the barcode. For MERGE (as the dashed edges of Fig. 7), there are two potential paths; we choose one arbitrarily, associate the new bar with the connected components written on it, and remove these components from the graph.

Note that the latter choice is not unique as the "elder rule" holds for ordinary PH, but not in the zigzag context. For instance, if the filtration consists of one connected component, that splits in two (at time t_1) and then merges (at time t_2), then the barcode will consist of two bars, one bar being $[t_1, t_2]$. However, two choices of identification are possible for this bar, and none is canonical. In practice, when choosing a path to remove, we remove the one that starts with the least number of nodes. This allows for maintaining homogeneity within the bars; that is, bars containing many nodes will continue to have many nodes.

One final detail should be noted. The original algorithm takes as input a zigzag filtration such that two consecutive graphs are obtained one from the other by adding or removing a single node or edge. However, partition or sliding-window timeslicings may yield filtrations where several nodes or edges are added or removed at the same time. Consequently, we must apply a pre-processing step to assign each event a unique time. Once the algorithm has been performed, we go back to the initial timestamps.

B. Design Decisions

In practice, our colored barcode is a timeline visualization that can be thought of as a series of stacked area charts, each referring to a connected component and its node members throughout time (Fig. 9(A)). As mentioned above, the color and height indicate the label (node metadata) and number of nodes, respectively.

Alternative designs: We have considered *alternative design choices* before proposing this layout. We decided to use a timeline visualization instead of animations to better meet analyses that rely on multiple and often distant timestamps (tasks T2, T3), complying with [5]. We then studied the suitability of existing timeline visualizations for our context. An option would be a visualization based on tracking graphs [34], [37], [66]. In particular, LargeNetVis's Global View [34] is a grid-based layout where rows and columns represent respectively network communities and timeslices. In this view, communities are encoded as circles with varying sizes, and their temporal evolution is depicted through links connecting communities from consecutive timeslices. Although we could adapt it to encode connected components, it would still not provide immediate information about components' node members. The identification and tracking of the members would also not be immediate with other visualizations, for instance, MSV [62], PAOH [60], and TAM [34].

We have also considered nested tracking graph visualizations [38], [39], but we opted for a different approach due to their inherent limitations in our specific context, particularly those concerning *visual scalability*, in terms of the number of timestamps, and the intrinsic *components' hierarchy* they consider. First, nested tracking graphs depict the graph evolution by representing events (i.e., merges and splits) through horizontal flows drawn between uniformly spaced timestamps. While our visualization also enables the analysis of these events upon interaction, our primary focus lies on depicting the graph structure at each timestamp and the most salient connected components, namely the longest bars throughout time. We, therefore, provide a more scalable solution for our case by employing vertical flows alongside timestamps that are positioned one right after the other. As an example, while Figs. 9 and 10 illustrate our layout for a graph with 1,555 timestamps, all visual analyses from [38], [39] consider a maximum of 100 timestamps.

Secondly, nested tracking graphs leverage hierarchical relationships coming from superlevel sets to derive component visibility, which can potentially lead to occlusions and impair the identification of patterns that would be crucial to our context (e.g., the number of nodes in a given component or its life-cycle). Conversely, we consider graphs with components without a hierarchical relationship (disjoint). Stacking them instead of nesting allows for immediate and effortless recognition of individual components and their attributes. Overall, we consider our colored barcode layout to be a better solution for our tasks and goals. It is suitable for graphs with many timestamps, emphasizes structural clarity, and enables efficient identification and exploration of key connected components and timestamps.

Bars' positioning: We use two representations for the bars' positioning in our colored barcode. The first consists of fixing the bars' bottom ordinate and distributing them upwards only (see Fig. 9) — we call this approach *"bottom-based ordering"*. Inspired by well-established cluster positioning on Sankey- and nested graph visualizations (e.g., [38], [39]), our second representation considers centered bars whose height varies uniformly up and down (see Fig. 10) — we call it *"center-based ordering"*. In both cases, the bars are arranged in such a way as to reduce the space they occupy in the interface and the lengths of the vertical flows.

C. System Prototype

We now describe the interface and interactions that compose the prototype of our ZigzagNetVis system (see a screenshot in Fig. 9), a web-based visual analytics tool that incorporates all steps of our workflow and was used by our user study participants (Section VIII).

When first loaded, the system automatically opens a menu through which it is possible to input a network and node categorical metadata (optional). The system then suggests temporal resolutions for the inputted network following the procedure described in Section IV. To help users choose among the suggested resolutions, they can ask for quantitative network measures (or features). For each resolution, the system will display values for burstiness [53], average lifetime of edges [53], normalized stability [15] and the inverse of the normalized fidelity — the original fidelity [15] gives us a distance measurement and we use the similarity counterpart. In our case, a higher value indicates greater faithfulness of the network under the selected resolution to the original network ($r = 1$). After choosing a resolution, users can filter out bars (i.e., connected components) with less than x node or with duration less than y timestamps, x and y being user-defined. We provide a visual comparison of different filtering parameters in our supp. material (Section C.1).

Once the network, temporal resolution, and the other parameter values are chosen, the system exhibits its first and main view (Fig. 9(a)), which contains the colored barcode and appears with maximized height and width, i.e., also occupying the screen space on Fig. 9(c-e). This view adopts as default the bottom-based component ordering, but the user is free to change it at any time (Fig. 9(h)). Besides zoom in/out and pan, users can select specific connected components or bars representing nodes that share the same label (Fig. 9(g)). In this way, it is possible to analyze their behavior at particular timestamps (tasks T1, T3) and evolution throughout time (task T2). Nodes sharing the same label can be selected in the layout by hovering over the label of interest in the color legend or the bar with the color associated with that label. Likewise, a connected component can be selected by hovering over any of its bars (Fig. 9(A)). It is also possible to persist the current selection (left-click) and select multiple labels (CTRL + left-click).

Two behaviors are expected when marking the checkbox *"See flows under interaction"* (Fig. 9(g)) and hovering over any bar

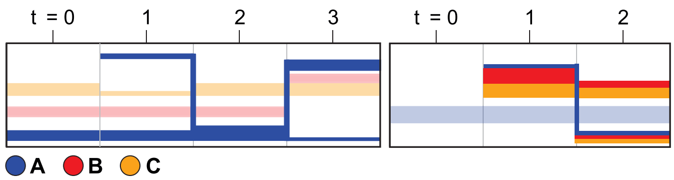


Fig. 8. Toy example showing the behaviors when marking the checkbox *"See flows under interaction"* and hovering over a bar: (left) if selecting by label, the system tracks same label nodes throughout time; (right) if selecting by conn. component, the system shows merges/splits in the selected component.

of a component: (i) if selecting by label (see Fig. 9(g) again), the system enables tracking the nodes with that (or those) label(s) throughout the connected components over time, as illustrated in Fig. 8(left); (ii) if selecting by connected component, the system shows the events that connect that component to others over time through vertical flows that indicate merges and splits (Fig. 8(right)).

After finding a potentially relevant timestamp or interval for analysis, the user double-clicks near it and the system opens three node-link diagrams as presented in Fig. 9(c-e), one showing the network structure at the timestamp of interest (referred to as $t(2)$, see Fig. 9(d)) and two others, by default, for $t(2) \mp 10$ timestamps (referred to as $t(1)$ and $t(3)$, see Fig. 9(c,e)). *Timestamp markers* are inserted in the colored barcode to highlight the three timestamps whose node-link diagrams are opened (Fig. 9(b)).

Users can freely change the three timestamps being analyzed — note, e.g., the values for $t(1)$, $t(2)$, and $t(3)$ in Fig. 9. This way, they can analyze the structure of groups of elements at different granularity levels (from the entire network to individual nodes) for any timestamp (task T1), as well as identify and compare structures and temporal behaviors by analyzing multiple node-link diagrams (tasks T2, T3). There are two ways for the user to reach a new timestamp of interest. If the user knows *a priori* which timestamp is relevant for analysis, they can simply type the new timestamp value in the node-link diagram area to update it; the system then repositions the corresponding timestamp marker accordingly. However, if the user is interested in analyzing a timestamp or interval that caught their attention because of an unexpected behavior found on the colored barcode, they can drag and drop one or more timestamp markers to that timestamp or interval; the system then updates the node-link diagram(s) accordingly.

Node-link diagram: Given a selected timestamp of interest t_k, our node-link diagram shows all nodes and edges active at t_k using a spring-force node positioning. Nodes are colored using the same color scale as in the colored barcode. In addition, the system also shows a tooltip with node id and label whenever a node is hovered over, as illustrated in Fig. 9(f). The user can expand one or more node-link diagrams (button **insert graphic here**) and drag/drop their maximized versions, e.g., to put them side-by-side and optimize comparisons. Depending on the type of selection (recall Fig. 9(g)), a click on a node x in a diagram (expanded or not) selects all nodes that contain the same label as x or all nodes that belong to the same connected component as x (T1). To help users compare structures and temporal behaviors (T2, T3), all node-link diagrams (expanded and non-expanded) are coordinated with each other and with the colored barcode: groups of nodes selected in one of them are automatically selected in the others (see, e.g., the non-selected connected component in Fig. 9(a,d), $t(2) = 710$).

Design decisions: Besides the decisions made on the colored barcode (recall Section V-B), we also studied *alternative approaches* before choosing static node-link diagrams to explore the network structure at particular times. First, we considered using animations to show the evolution of the network during the time interval selected through the timestamp markers. We gave up this idea because animations have limitations on tasks involving multiple and distant timestamps [5]. After opting for "static" visualizations, we considered node-link diagrams and adjacency matrix-based visualizations [6]. We chose the former as it would be easier to identify connected components using the diagram, especially when adopting spring-force node positioning. Finally, we decided to enable the analysis of three timestamps (three node-link diagrams at once) based on the intuitive notion of past, present, and future.

As mentioned, our system prototype associates different colors to nodes (or bars) with different labels when this metadata information is available. Color-blind users can use a color scheme that is safe from color blindness. Our prototype also provides a series of features that help colorblind users in their analysis, e.g., by allowing selections and by showing informative tooltips. In the user study, we validated visualizations and color scheme with two self-declared colorblind participants (see Section VIII-C).

Implementation details: We use a client-server architecture. The server side was implemented in Python and uses popular libraries and frameworks (e.g., `NetworkX`, `Flask`, and `Dionysus2`). We used the `D3` library in our views. A demo version of the system, used by our user study participants and already including suggestions, is available at https://github.com/raphaeltinarrage/ZigzagNetVis.

Computational complexity: The overall ZigzagNetVis process can be divided into three steps: open the dataset (1), compute the suggestion curve (2), and compute the colored barcode for one resolution (3). Let m be the number of pairs (edge, time) in the temporal graph, and let n be the number of resolutions tested. Step 1 consists in reorganizing these pairs in a dictionary, and creating a list of unique edges, resulting in a computational complexity of $O(m)$. In Step 2, we create n zigzag filtrations, compute their H_0-homology barcode, and then compute the consecutive bottleneck distances. The respective complexities are $O(nm)$, $O(nm\alpha(m))$, and $O(nm^{1.5})$, where α is the inverse Ackermann's function (approximately constant in practice) [18]. Last, Step 3 consists of one computation of zigzag persistence, which therefore has a computational complexity of $O(n\alpha(n))$. In general, the complexity of the process is $O(nm^{1.5})$. We should mention, however, that our personal implementation of the persistence algorithm does not reach the complexity mentioned above and can potentially yield longer execution times. In our supp. material (Section C.3), we give the running times observed in practice for eight temporal graphs.

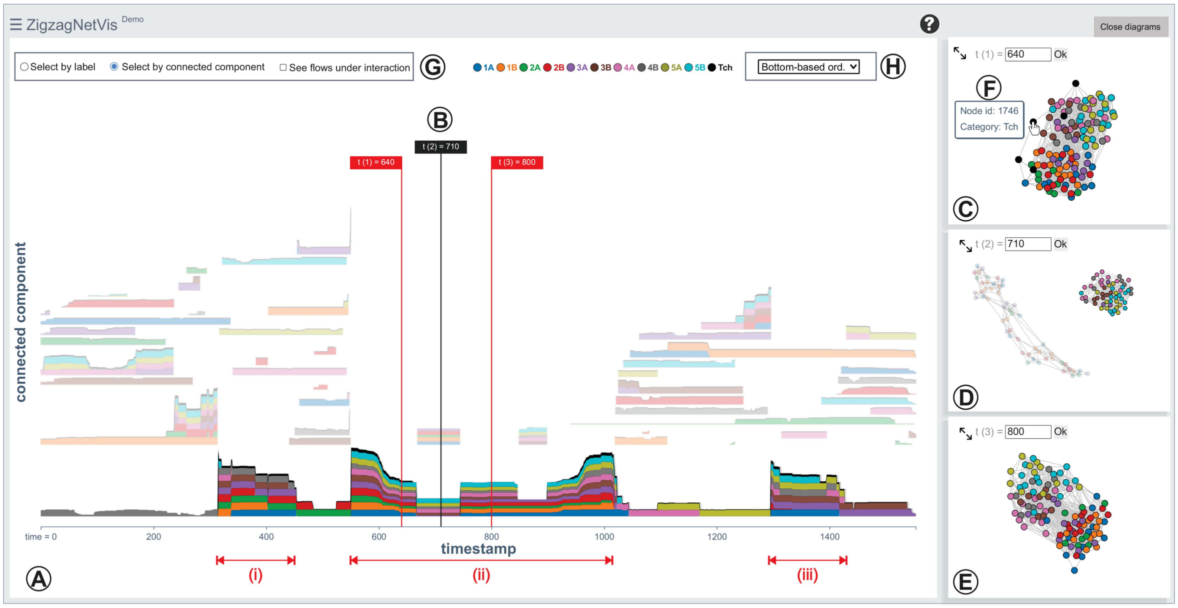


Fig. 9. ZigzagNetVis system prototype, an interactive and web-based system with linked views designed to assist the analysis of temporal graphs by highlighting connected components' structure and evolution. (a) Colored barcode with bottom-based ordering that highlights the longest connected component in the graph — note that (i), (ii), and (iii) represent time intervals with few connected components compared to others. (b) Timestamp markers indicating the three timestamps being depicted by (c-e) the three node-link diagrams. (f) Tooltip showing extra information. (g) Users can select groups of nodes by label or by connected component. (h) Users can choose between two available component positioning strategies.

VI. Datasets

Our usage scenario and user study explore the first day of data from two real-world and face-to-face temporal graphs collected in educational environments, the Primary School [23] and the High School [40] networks. We have chosen these graphs as they have been extensively analyzed in the context of temporal graph visualization [25], [34], [47], [63], [67], [68] and because they contain relevant node metadata information.

The first day of the Primary School network [23] contains 236 nodes (students and teachers from the first to the fifth grade, each having classes A and B) and 60,623 edges, which represent face-to-face interactions. There are 1,555 timestamps in the original resolution ($r = 1$), each comprising a 20-sec interval. Data were collected from 8:45 am to 5:20 pm. There is a lunch break from 12pm to 2pm and two smaller breaks (20-25 min), one in the morning (around 10:30am) and one in the afternoon (around 3:30pm). Each of the 10 school classes has an assigned teacher. For convenience, we will refer to each class using simple terms, for example, 1B to refer to "first grade, class B".

The High School network [40] contains face-to-face interactions between students from nine classes related to different subjects: chemistry and physics (classes PC and PC1), mathematics and physics (classes MP, MP1, and MP2), engineering (class PSI), and biology (classes 2BIO1, 2BIO2, and 2BIO3). The first day contains 312 nodes and 28,780 edges distributed in 899 timestamps (a 20-sec interval each) when adopting the original resolution.

VII. Usage Scenario

We demonstrate the suitability of a suggested resolution for analyzing the Primary School and the usefulness of our colored barcode and system to assist in this analysis. We also show in Section C.4 of our supplementary material that the patterns found using ZigzagNetVis are comparable to those identified using LargeNetVis [34], a state-of-the-art approach.

ZigzagNetVis suggested the resolutions $r = 8$, 18, 76, 154, and 282 for the Primary School. Fig. 9(A) shows our colored barcode for the median resolution $r = 76$, empirically chosen among the suggested ones due to its interesting patterns. This visualization was produced after (i) filtering out components with less than 10 nodes and 10 timestamps and (ii) selecting the component with the longest duration. Disregarding the component selection (we will discuss it later on), we can already enumerate some patterns and interesting behaviors in the graph data. First, we see that most of the non-selected components (i.e., components with low opacity in Fig. 9(a)) are composed of students from a single class, along with their teacher (tasks T1, T2). This is expected since these students were having classes in their respective classrooms. This pattern can also be seen in Fig. 10, which shows the same network and resolution using a different ordering.

There are also time intervals with few connected components compared to others, possibly indicating school breaks (one in the morning, lunch break, and another in the afternoon — see Fig. 9(i,ii,iii), respectively) (tasks T2, T3). The first time we have a single component in the graph delineates the beginning of lunch break (see the selected component near timestamp $t = 580$ in Fig. 9(a)) (task T1). As the students go home for

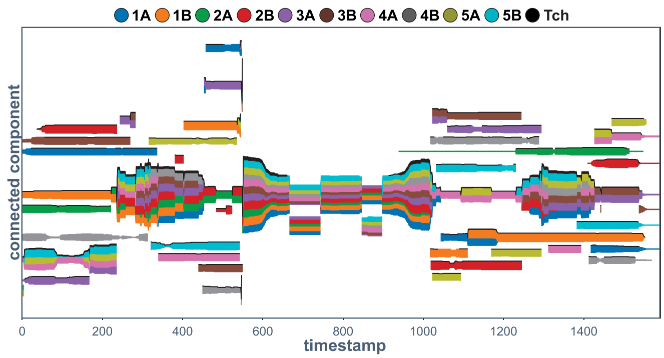


Fig. 10. Colored barcode with center-based ordering for the Primary School and the suggested resolution 76.

lunch [23], we observe a decrease in the number of nodes in the graph (see just after timestamp $t = 600$) (tasks T2, T3). During lunch, this component is eventually decomposed into two parts, as illustrated in Fig. 9(A,D)($t = 710$), one containing students from classes 1A, 1B, 2A, 2B, 3A and the other containing a few other students from 3A and 3B, 4A, 4B, 5A, 5B (task T1). This division is explained by the location of the students that stay at the school: some children stay in the cafeteria while others stay at the courtyard [23]; these groups encounter each other when they switch places, leading to a single component again (see Fig. 9(a,e)($t = 800$)). Note also the absence of teachers during the lunch break: they are present at first (see Fig. 9(a,c,f)($t = 640$)), but they leave (there are no teachers in $t = 710$ and $t = 800$, for example) and come back near the end of the lunch break (task T3), when we start seeing many connected components in the graph again (task T2).

VIII. User Study

A. Participants and Experiment Setup

The experiment recruited 27 participants, including undergraduates (9), Master's students (6), Ph.D.s candidates (7), postdocs (2) and professors (1). According to their self-reports, 4 participants had advanced knowledge in graphs, 4 in visualization, 2 in TDA, and 3 in Informatics in Education. We conducted the experiment using the think-aloud protocol, a common technique to obtain a more accurate perception of the participants' thoughts [12]. To avoid participants being influenced by our presence and not mentioning negative aspects, we explicitly asked them to highlight our approach's limitations. Before the experiment, we conducted a pilot study with two participants not included in the final analysis.

B. Questionnaire

First, the participants were presented with a 7-minute video tutorial that introduced the concepts of graph, temporal resolution, and connected components, and explained the proposed layout and system functionalities. The questionnaire was divided into four main sections: (i) background and experience; (ii) a hands-on experience with defined tasks; (iii) nine questions that address the Primary and High School networks; and (iv) Likert-scale questions to collect the participants' feedback.

The questions were designed to evaluate layout perception, test functionalities, find patterns, and freely explore the given networks. First, we assessed comprehension of the basic functionalities through hands-on experience, where we asked the participants to open the Primary School network using the default configuration. Then, we asked them to verbally describe the definitions of some concepts necessary to understand the experiment (e.g., connected components and temporal resolution) and to follow a set of 12 simple tasks (ST1-ST12) to check if they were familiarized with the system's functionalities (e.g., shortcuts and interaction features). They were also asked to validate our visualization by exploring the Primary School network with resolutions $r = 76$ (SQ1-SQ3) and $r = 154$ (SQ4-SQ6), and the High School under $r = 46$ (SQ7-SQ9). Due to time limitations, we focused on analyzing only these three resolutions, all suggested by ZigzagNetVis using sliding-window timeslicing. In addition, our focus on school networks aimed to provide participants with familiar contexts for understanding nodes and edges, which have the same meaning on both networks.

The SQ1-SQ9 questions were open questions in which we guided the participants to identify specific patterns (SQ1-SQ3 and SQ7-SQ8), asked them to compare the results of two resolutions (SQ4, SQ5), and encouraged them to explore the system freely (SQ6, SQ9). Finally, we evaluated the participants' preferences for ZigzagNetVis using a series of Likert-Scale questions (LQ1-LQ10) and asked them to describe the positive and negative aspects of the system. The complete description of the questions and expected answers are available in the supp. material (Section D.1).

After preliminary tests, we fixed both filters for bars in 10 (recall Section V-C) to avoid receiving too many different results, which would hinder the analysis of the collected data.

C. Results

Hypotheses on data analysis: All participants answered at least one of the points that we expected for each open question (SQ1-SQ5, SQ7, SQ8). Also, during the experiment, we encouraged participants to raise hypotheses that could justify specific patterns considering a school environment. For instance, in question SQ1 (Primary School), we asked them to evaluate the relationship between students and teachers from classes 4 A, 5 A, and 5B (which form a connected component at some point). All participants mentioned that class 4 A was far from the others. Furthermore, 62% of the participants identified that the two subgroups (4 A, 5A-5B) were linked by an edge that involved a teacher; 37% noticed that this edge actually involves two teachers. The hypotheses put forward to explain the strong interaction between classes 5 A and 5B mentioned that, since both classes belong to the fifth grade (30% remembered this information), it could be due to interdisciplinary events such as laboratory activity (22%) or group studies (14.81%).

Exploratory analyses: We proposed questions where the participants could freely explore the system and identify patterns not described by other questions (SQ6, SQ9). More than 85% of the participants mentioned new patterns or anomalies in their exploratory analyses of the Primary School (SQ6), and 74% found new ones in the High School network (SQ9). Among the

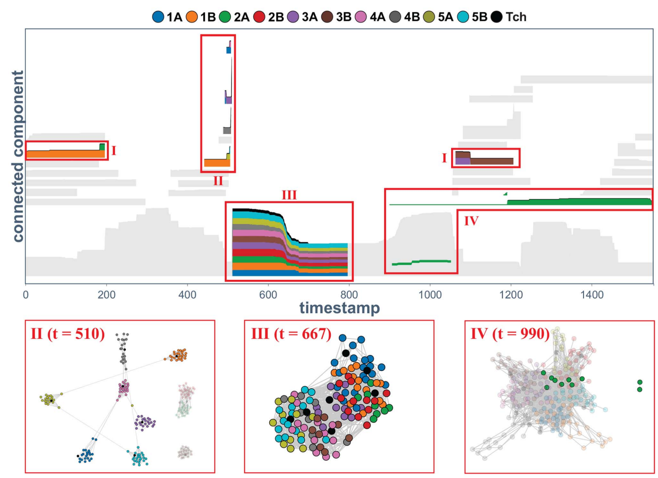


Fig. 11. Four patterns (I-IV) mentioned via SQ6 (Primary School, $r = 154$). The colored barcode adopts bottom-based ordering.

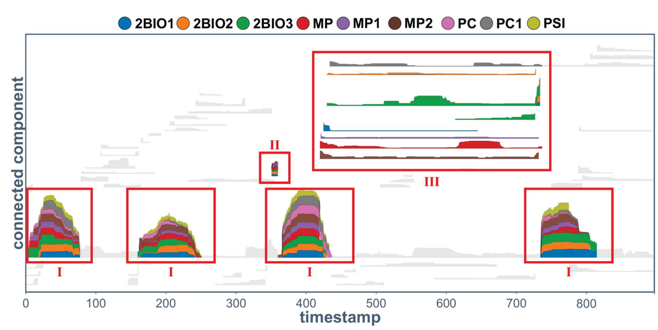


Fig. 12. Three patterns (I-III) mentioned via SQ9 (High School, $r = 46$). The colored barcode adopts bottom-based ordering.

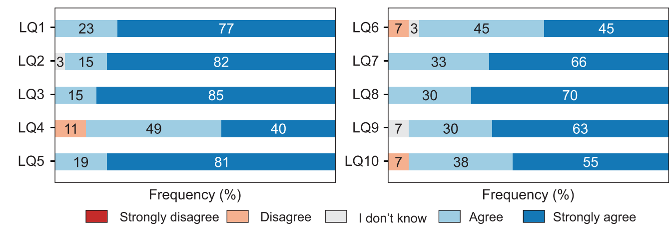


Fig. 13. Participants' feedback using a Likert scale.

patterns and anomalies found, the most cited for the Primary School using $r = 154$ were (Fig. 11): (I) merges and splits between related students; (II) peaks of interaction in a short time period; (III) a single connected component containing all students and teachers (even though the teachers leave the network at some point); and (IV) same-class students divided into two connected components. Although this question was not designed to compare patterns identified in different resolutions, most participants tried to compare patterns visible with $r = 154$ with those from $r = 76$. For instance, Fig. 9(a,d) illustrates that there are two connected components around timestamp 710 when using $r = 76$, which is hidden in the higher resolution (see Fig. 11(III)). About that, a participant mentioned that *"you can clearly see how patterns vary according to the selected resolution when analyzing the primary school"*.

The participants identified three common patterns in the second exploratory question (SQ9, High School). The first refers to peaks of activity in the same connected component over time (see Fig. 12(I)). In the High School, there are also intervals where all students merge into a single and highly connected component. The participants could see that these intervals correspond to break periods, lunch break, or group activities. The second pattern is related to a small connected component just before a large peak (Fig. 12(II)). Based on the node-link diagram, there were just a few connections between the students, which represented the beginning of a group activity or a break. Finally, the third pattern refers to connected components with varying lengths over time but composed of single classes (Fig. 12(III)). According to the participants, they allow one to see the class hours, but, contrary to the primary school, where the number of students per component is quite stable over time during classes (see Fig. 9), this network presents classes with non-uniform activity over time.

Interactive features: We also validated the functionalities mainly used to answer representative questions. In summary, the participants preferred to move timestamp markers rather than type new timestamp values for the exploratory tasks. On average, the feature used mainly in the node-link diagrams was zoom (41.48%), which is justified by the small size of nodes and edges initially applied. Not least, the similar rate of usage involving selection by label (44.44%) and by component (41.48%) indicates that both were appreciated. Please refer to the supp. material (Section D.2) for details.

Likert-scale questions and participants' feedback: Fig. 13 shows the participants' assessments of the colored barcode's (LQ1) and node-link diagrams' (LQ2) quality and usefulness, their coordination and interaction (LQ3), and the system's intuitiveness and ease of use (LQ4), usefulness (LQ5), and response time (LQ6). There were also questions related to specific tasks, such as understanding the temporal evolution (LQ7), comparing structure at different times (LQ8) or at node level (LQ9), and analyzing the network under different resolutions (LQ10).

First, considering the negative evaluations, three participants mentioned that the system was not intuitive (LQ4) because it lacked a "help" button summarizing the main functionalities. Regarding response time (LQ6), two users complained about loading time, although the system's interactions worked satisfactorily. One of the experts added that *"I can't say about speed, for the tested datasets I agree but generally I don't know, it depends on the network size"*. At last, about the analysis under different resolutions (LQ10), two participants considered that the comparison was difficult since it depended on the memory load of the user.

Besides the negative evaluations, ZigzagNetVis achieved a 95% of acceptance rate for the raised criteria (LQ1-LQ10), considering the average agreement (29%) and strong agreement (66%) rates. Several participants raised positive points about the system and colored barcode, claiming that *"The proposed system is simpler and more efficient in analyzing temporal networks than the other tools I know"*, and *"the colored barcode is great*

TABLE I
SUGGESTED RESOLUTIONS FOR EIGHT DISTINCT NETWORKS

Network	Suggested resolutions	Used in the literature
Primary School [23]	8, 18, 76, 154, 282	10 [67], 25 [47]
High School [40]	8, 12, 46, 92, 104	18 [63], 180 [25], 45 [68]
Hospital [64]	14, 26, 32, 74, 352	{9, 45, 60, 90} [33], [35], 69 [50]
InVS [24]	66, 148, 158, 164, 202	—
Museum [27]	6, 12, 36, 52, 320	1 [49]
Enron [29]	6, 12, 24, 36, 68	1 [21], [36], [62], 2 [47], 5 [50], {1, 7, 15, 30, 90, 180} [44], {1, 5, 12} [58]
Conference [27]	12, 22, 30, 42, 224	30 [21]
Sexual [52]	6, 160, 186, 226, 240	1 [49]

(pretty and very interesting), both for the color distinction and the subtlety of increases and decreases in a bar over time". For another participant, *"It is the union of both views (colored barcode and node-links) that is most useful. Each alone would not allow us to understand well what is happening"*. At last, one expert complemented that *"the barcodes are very good for quickly visualizing long interactions, while the node-link diagrams allow you to understand to what degree these interactions are happening"*.

It should be noted that we tested the system with two color-blind participants, who validated that there were enough features (such as tooltips, different color scales, and interactive color legend) to perform all tasks without hindering the analyses. Finally, some participants suggested improvements already incorporated, such as the selection of multiple labels, improvements in readability (e.g., better contrast in menus) and the mentioned "help" button.

IX. Analysis of the Suggested Resolutions

This section is devoted to the analysis of the resolutions suggested by ZigzagNetVis' methodology. We aim to demonstrating empirically that these suggestions are relevant. As mentioned in Section II, few studies have tackled the problem of choosing a resolution. In particular, no reference data sets are available. As a means of comparison, we will use, for each of the temporal graphs considered in this paper, the resolutions used in the literature — that we stress are mainly chosen "by hand". However, a direct comparison cannot be made. Indeed, as pointed out in Section IV-A, one cannot define "optimal" resolutions, but rather *meaningful ranges* of resolutions. Consequently, to assess the quality of ZigzagNetVis' suggested resolutions, one must analyze them qualitatively by understanding the behaviors of the corresponding dynamic graphs and comparing them with the literature. This study will be conducted here, in particular using the visual tool provided by the bottleneck distance. In our supplementary material (Section C.2.3), we extend this study by comparing our suggestion curves with commonly used feature of temporal graphs, revealing when they coincide and when they complement each other.

Visualization of suggested values: Table I presents the resolutions suggested by our approach, using sliding-window timeslicing and considering eight different graphs of varying characteristics and sizes. The corresponding normalized suggestion curves are shown in the supp. material (Section C.2.1). To construct the curves for these eight temporal graphs, we used, respectively, a maximal time of 2000, 2000, 2000, 20000, 1300, 1400, 3000, and 1000, and resolutions up to a quarter of these values. Note that the maximal time for Primary and High Schools covers the first day, as in Section VI.

Fig. 14 shows the colored barcodes for four networks in Table I to emphasize the usefulness of both the resolution suggestion method and this visualization in assisting the analysis of real-world networks. The colored barcodes exhibit the entire graphs, except for the InVS network (Fig. 14(c)), which shows only the first day of data to better present the visual pattern we want to discuss.

Even though the Enron network (Fig. 14(a)) does not provide node metadata, it is easy to identify global patterns that do not rely on such information, for example, the gradual increase in the number of connections and node activity over time [29]. The increasing size of the main connected component, followed by an abrupt decrease near the end of the network, is related to important events in the context of these network data, including the CEO resignation and bankruptcy. Temporal patterns related to circadian rhythms can also be identified in face-to-face networks, as shown in Fig. 14(b) for the Hospital network [64]. We can easily identify intervals with bursts of events (five days) followed by intervals with few or no interaction (four nights).

Incorporating node metadata greatly improves network analysis by allowing us to observe local patterns in the data. In the InVS network [24], for example, most connected components contain only nodes that share the same label (in this case, employees of the same department), as illustrated in Fig. 14(c). That makes sense in the context of this network, as most of the employees are of type "residents", i.e., they interact mainly with others in their own department. This is a pattern we do not observe in the Sexual network [52] (Fig. 14(d)). Since it is a bipartite graph, all connected components will have at least one node from each label, i.e., a buyer and a seller. Note that the Sexual network is much larger than the others we have considered. Its original form (resolution 1) contains 12,157 nodes, 34,060 edges, and 1,000 timestamps, each representing a 1-day interval [52].

Resolutions used in the literature: In general, studies that analyze temporal graphs use resolution directly or indirectly. Some focus on comparing different resolutions [44], [47], [58], while others select arbitrary resolutions according to the analysis needs [50], [62], [68]. For instance, some works prioritize high resolution values for global pattern identification [25], [68], while others focus on small ones and local patterns [21], [49], [62]. Note that ZigzagNetVis suggests resolutions suitable for both types of analysis (Table I).

Table I summarizes our suggested resolutions and others used in literature for eight popular graphs. For the well-known Enron network [29], while some studies use the original resolution $r = 1$ as an arbitrary value to perform local analyses [21], [36], [62], others compare resolutions coming from a small set of arbitrary values [44], [58]. For example, Sulo et al. [58] analyze this network under resolutions $r = 1, r = 5$, and $r = 12$, highlighting the different patterns each resolution allows one to identify. According to the authors, the pattern "CEO resignation" is easily identified when adopting resolutions between 4 and 7 [58]. Note that ZigzagNetVis suggested resolutions $r = 6$

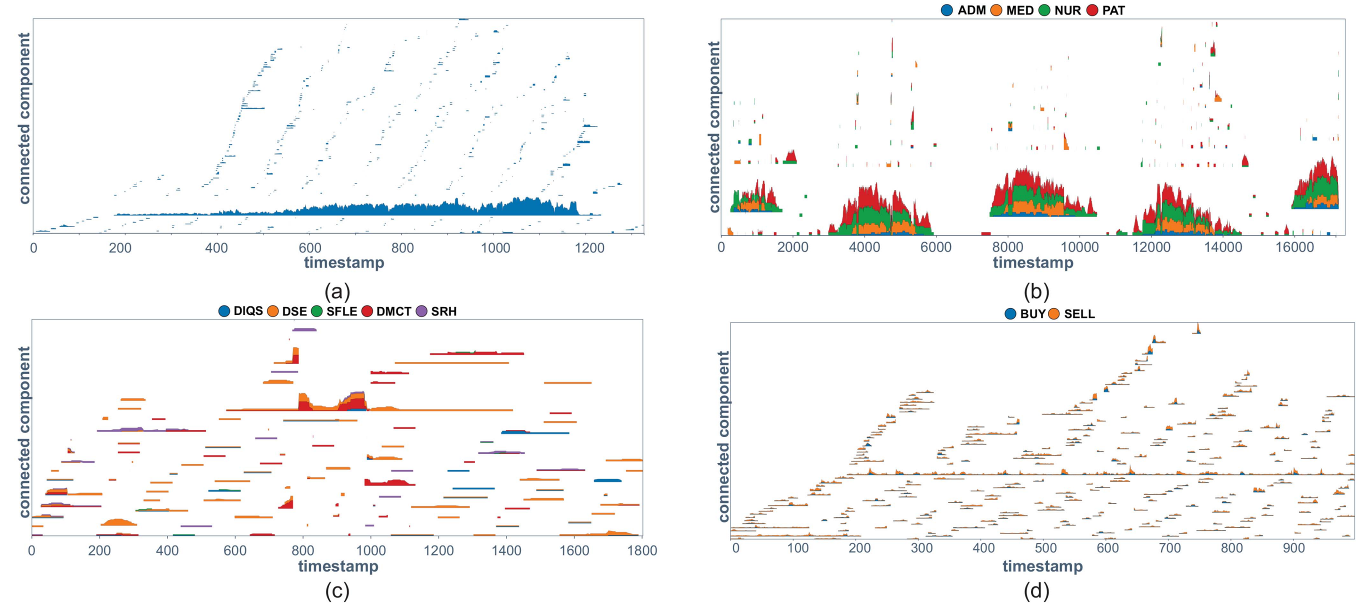


Fig. 14. Colored barcodes with bottom-based ordering for four networks from Table I. (a) Enron with $r = 6$. (b) Hospital with $r = 74$. (c) InVS with $r = 66$. (d) Sexual with $r = 6$. All resolutions adopted were suggested by our method (see Table I). There is no component filtering except for the Sexual network (f), whose colored barcode shows only components with at least 10 node members and a duration of at least 10 timestamps.

(therefore included in the mentioned "good-quality" range) and $r = 12$ (a resolution also used by the authors). The suggestion of a resolution that matches exactly the one used by previous studies also occurred with the Conference network ($r = 30$, as depicted in Table I).

As another example, some studies mention the same circadian rhythm pattern discussed in Fig. 14(b) for the Hospital network, i.e., days with bursts of activity and idle nights [33], [35], [50]. ZigzagNetVis and these studies allow one to identify this pattern, even though they use different but close resolution values. Our method also suggests a resolution many times greater than those used in the literature for this network ($r = 352$). This is probably the resolution in which the idle intervals are lost. In general, our approach suggests resolutions that are close to those used by the related literature. In addition, it can also suggest other resolution values that potentially lead to unexplored visual patterns.

Resolution comparison and explainability: As discussed in SectionII-C, the bottleneck distance offers a clear interpretability. Namely, the distance $d_{\mathrm{B}}(\mathscr{B}, \mathscr{B}')$ between two barcodes is always caused by a pair of bars or a bar alone, that is, such that the cost of this pair, or of this bar alone, is equal to the distance. Consequently, highlighting these bars allows us to observe precisely *where* the barcodes differ the most. This is particularly useful for understanding the suggested resolutions of ZigzagNetVis.

Taking into account the first day of the Primary School network, the algorithm suggests resolution $r = 8$ (see Table I). The resolution just before this one is $r = 6$, since sliding-window timeslicing only accepts even values of resolutions. In order to visualize what structural change has occurred between resolutions 6 and 8, we show in Fig. 15(a,b) the corresponding colored barcodes, while highlighting the pair of bars that provoked the bottleneck distance. As we can see, when going from $r = 6$ to $r = 8$, a large bar is formed, which lasts throughout the observation period. Please refer to the supp. material (Section C.2.2) for other networks.

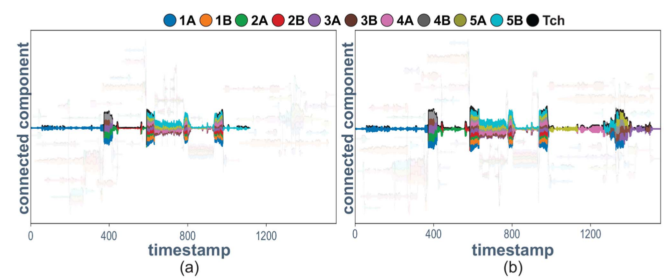


Fig. 15. Visualization of the bottleneck distance for the first day of the Primary School. (a-b) $r = 6$ and $r = 8$ showing only bars with height larger than 50. Highlighted components represent the bars that differ the most between these two resolutions, according to the bottleneck distance.

Classification of structural changes: A manual analysis of the resolutions suggested by ZigzagNetVis compels us to classify the structural changes into three categories. The first category contains the initial resolutions. We have observed, in the suggestion curves, the phenomenon of a chaotic start, followed by a relatively flat phase. These resolutions correspond to critical points indicating the formation of the first persisting connected component. A second type of easily identifiable structural change is that of the connection between days of the temporal graph. At the critical resolution connecting two consecutive days, assuming no activity is recorded during the night, the suggestion curve shows a significant peak. The last group of resolutions generally contains those that cause a persistent connected component to merge with a larger one.

This classification allows, at least heuristically, to divide the range of resolutions into three intervals: a chaotic start, followed by a range where the resolution curve only exhibits relevant peaks, and at last a few values caused by the merging of the days. This observation can be used when the user, through manual inspection, seeks relevant resolutions to study. We stress that the last case is not observed in the figures for the Primary and High School networks, since we selected resolutions smaller than the length of a night.

X. Discussion and Limitations

Timeslicing: ZigzagNetVis is not designed for graphs with continuous real-valued timestamps as timeslicing approaches fail to represent these graphs faithfully [34]. Considering graphs with discrete times, we have described two uniform timeslicing approaches that may be used with ZigzagNetVis: partition and sliding-window-based. Regardless of the chosen approach, the suggested resolution is a global and static value that is used to represent the entire graph. In future work, we intend to investigate whether non-uniform timeslicing would lead to better results.

Visual scalability: Our colored barcode is better suited for small to mid-size graphs, in terms of the number of timestamps or connected components (even though too few and large components also hinder the analysis). Although we provide filters and interactions that help with large networks, we intend to improve our visual scalability to better meet this type of network. Specifically, we plan to extend the representation to deal with more timestamps and components, e.g., by collapsing/expanding based on the graph dynamics. We also intend to incorporate sampling strategies and more sophisticated filters, e.g., based on structural properties such as the strength of the connected components or edge weights (explicit or inferred).

Resolution comparison: Some participants would also like to simultaneously compare suggested resolutions with each other and with non-suggested ones. Although one could open the system many times or perform a side-by-side comparison using multiple instances of the system, we believe that incorporating such a capability into our system would enhance the identification of patterns coming from different resolutions, help users to understand and follow changes that regions of interest suffer when varying resolutions (recall Section IX), and also increase the user's confidence in the suggestions or reveal room for improvement in the suggestion procedure, e.g., by incorporating user feedback.

Zigzag persistent homology: Through the lens of homology, all connected components are treated identically, regardless of the number of nodes they contain. Consequently, in extreme cases, a structural change in the temporal network can be provoked by a single node. Since this situation might not be convenient for the analysis of large networks, where relevant features are commonly understood as those involving many nodes, we intend to design and adopt a variation of the bottleneck distance that would take into account the number of nodes. Besides, our work focused on homology H_0. The inclusion of higher topological features, such as in [42], may contain further relevant information, that we intend to add in future works. Not least, we adopted in this work a simple peak detection via their prominence, which is well established and easy to interpret. In a follow-up study, we intend to test more sophisticated approaches, for example, peak detection via Z-scores, or TDA-inspired techniques based on peaks' persistence.

Running time: Table I in our supp. material shows that, in practice, the most time-consuming step of our algorithm is the computation of m persistence diagrams, m being the number of resolutions tested. To reduce this cost, we could take advantage of the fact that two consecutive resolutions should yield barcodes close to each other; an idea known as *updating barcodes* [19]. Although we have not investigated this aspect further, since the running times obtained empirically were satisfactory, such a technique could open the door to larger-scale graphs.

Visual improvements and new features: Based on feedback from reviewers and participants, we've added new features to the system prototype: a center-based component positioning, merge/split visual representation, and a table with quantitative measurements for suggested resolutions. While participants did not test these features, they do not directly affect the results outlined in this paper.

XI. Conclusion

This paper presented ZigzagNetVis, a methodology that suggests potentially relevant temporal resolutions for graph analysis using zigzag PH, a well-established technique from TDA, and the tool system that implements it. Our methodology can be summarized as follows. First, we build persistence barcodes for candidate resolutions. Then, we compute the bottleneck distance between pairs of barcodes and build a suggestion curve based on the distance values. Finally, we suggest resolutions based on the curve's peaks. ZigzagNetVis also incorporates a timeline-based visualization inspired by the persistence barcodes of TDA. Our visualization assists researchers and practitioners in exploring temporal graphs by highlighting the connected components' structure and evolution. We validated ZigzagNetVis and our web-based and interactive system prototype through a usage scenario and a user study with 27 participants, who assessed its usefulness and effectiveness.

Acknowledgment

For open access purposes, the authors have attributed a Creative Commons CC BY license to any accepted version of the article.

References

[1] J.-W. Ahn, C. Plaisant, and B. Shneiderman, "A task taxonomy for network evolution analysis," *IEEE Trans. Vis. Comp. Graph.*, vol. 20, no. 3, pp. 365–376, Mar. 2014.

[2] M. E. Aktas, E. Akbas, and A. E. Fatmaoui, "Persistence homology of networks: Methods and applications," *Appl. Netw. Sci.*, vol. 4, no. 1, Aug. 2019, Art. no. 61, doi: 10.1007/s41109-019-0179-3.

[3] B. Bach, N. Henry-Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski, "Small multipiles: Piling time to explore temporal patterns in dynamic networks," *Comput. Graph. Forum*, vol. 34, no. 3, pp. 31–40, 2015, doi: 10.1111/cgf.12615.

[4] B. Bach, E. Pietriga, and J.-D. Fekete, "GraphDiaries: Animated transitions and temporal navigation for dynamic networks," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 5, pp. 740–754, May 2014.

[5] F. Beck, M. Burch, S. Diehl, and D. Weiskopf, "A taxonomy and survey of dynamic graph visualization," *Comp. Graph. Forum*, vol. 36, no. 1, pp. 133–159, 2016.

[6] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete, "Matrix reordering methods for table and network visualization," *Comput. Graph. Forum*, vol. 35, no. 3, pp. 693–716, 2016, doi: 10.1111/cgf.12935.

[7] P. Bendich et al., "Topological and statistical behavior classifiers for tracking applications," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 6, pp. 2644–2661, Dec. 2016.

[8] C. Bodnar, C. Cangea, and P. Liò, "Deep graph mapper: Seeing graphs through the neural lens," *Front. Big Data*, vol. 4, 2021, Art. no. 680535, doi: 10.3389/fdata.2021.680535.

[9] U. Brandes, *Force-Directed Graph Drawing*. Boston, MA, USA: Springer, 2008, pp. 1–6.

[10] M. Brehmer and T. Munzner, "A multi-level typology of abstract visualization tasks," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 2376–2385, Dec. 2013, doi: 10.1109/TVCG.2013.124.

[11] G. Carlsson and V. De Silva, "Zigzag persistence," *Found. Comput. Math.*, vol. 10, no. 4, pp. 367–405, 2010.

[12] S. Carpendale, "Evaluating information visualizations," *Inf. Vis.: Human-Centered Issues Perspectives*, pp. 19–45, 2008, doi: 10.1007/978-3-540-70956-5_2.

[13] F. Chazal, L. J. Guibas, S. Y. Oudot, and P. Skraba, "Persistence-based clustering in riemannian manifolds," *J. ACM*, vol. 60, no. 6, pp. 1–38, 2013.

[14] F. Chazal and B. Michel, "An introduction to topological data analysis: Fundamental and practical aspects for data scientists," *Front. Artif. Intell.*, vol. 4, 2021, Art. no. 667963.

[15] A. Chiappori and R. Cazabet, "Quantitative evaluation of snapshot graphs for the analysis of temporal networks," in *Complex Networks & Their Applications X*, R. M. Benito, C. Cherifi, H. Cherifi, E. Moro, L. M. Rocha, and M. Sales-Pardo, eds. Cham, Switzerland: Springer, 2022, pp. 566–577.

[16] A. Clauset and N. Eagle, "Persistence and periodicity in a dynamic proximity network," 2012, *arXiv:1211.7343*.

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

[18] T. K. Dey and T. Hou, "Computing zigzag persistence on graphs in near-linear time," in *Proc. 37th Int. Symp. Comput. Geometry*, Dagstuhl, Germany, 2021, pp. 30:1–30:15, doi: 10.4230/LIPIcs.SoCG.2021.30.

[19] T. K. Dey and T. Hou, "Updating barcodes and representatives for zigzag persistence," 2021, *arXi:v2112.02352*.

[20] B. Doppalapudi, B. Wang, and P. Rosen, "Untangling force-directed layouts using persistent homology," in *Proc. 2022 Topological Data Anal. Vis.*, 2022, pp. 81–91.

[21] B. Fish and R. S. Caceres, "A. supervised approach to time scale detection in dynamic networks," 2017, *arXi:v 1702.07752*.

[22] J. Gamble, H. Chintakunta, and H. Krim, "Applied topology in static and dynamic sensor networks," in *Proc. 2012 Int. Conf. Signal Process. Commun.*, 2012, pp. 1–5.

[23] V. Gemmetto, A. Barrat, and C. Cattuto, "Mitigation of infectious disease at school: Targeted class closure vs school closure," *BMC Infect. Dis.*, vol. 14, no. 1, pp 695, Dec. 2014, doi: 10.1186/PREACCEPT-6851518521414365.

[24] M. Génois, C. L. Vestergaard, J. Fournet, A. Panisson, I. Bonmarin, and A. Barrat, "Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers," *Netw. Sci.*, vol. 3:326-347, 2015, doi: 10.1017/nws.2015.10.

[25] M. Hajij, B. Wang, C. Scheidegger, and P. Rosen, "Visual detection of structural changes in time-varying graphs using persistent homology," in *Proc. IEEE Pacific Visual. Symp.*, 2018, pp. 125–134, doi: 10.1109/PacificVis.2018.00024.

[26] P. Holme and J. Saramäki, "Temporal networks," *Phys. Rep.*, vol. 519, no. 3, pp. 97–125, 2012.

[27] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck, What's in a crowd? Analysis of face-to-face behavioral networks., *J. Theor. Biol.*, vol. 271, no. 1, pp. 166–180, 2011, doi: 10.1016/j.jtbi.2010.11.033.

[28] S. Jung, D. Shin, H. Jeon, K. Choe, and J. Seo, "Monetexplorer: A visual analytics system for analyzing dynamic networks with temporal network motifs," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 10, pp. 1–15, Oct. 2024, doi: 10.1109/TVCG.2023.3337396.

[29] P. S. Keila and D. B. Skillicorn, "Structure in the enron email dataset," *Comput. Math. Organ. Theory*, vol. 11, pp. 183–199, 2005, doi: 10.1007/s10588-005-5379-y.

[30] W. Kim and F. Mémoli, "Formigrams: Clustering summaries of dynamic data," in *Proc. 30th Can. Conf. Comput. Geometry*, 2018, pp. 180–188.

[31] W. Kim, F. Mémoli, and Z. Smith, "Analysis of dynamic graphs and dynamic metric spaces via zigzag persistence," in *Proc. Topological Data Analysis: Abel Symp.*, 2020, pp. 371–389.

[32] G. Krings, M. Karsai, S. Bernhardsson, V. D. Blondel, and J. Saramäki, "Effects of time window size and placement on the structure of an aggregated communication network," *EPJ Data Sci.*, vol. 1, no. 1, May 2012, Art. no. 4, doi: 10.1140/epjds4.

[33] C. Linhares, J. Ponciano, L. Rocha, J. G. Paiva, and B. Travençolo, "Análise temporal de uma rede de contato hospitalar utilizando técnicas de visualização de informação," in *Proc. XVII Workshop de Informática Médica*, 2017, pp. 1794–1803, doi: 10.5753/sbcas.2017.3696.

[34] C. D. G. Linhares, J. R. Ponciano, D. S. Pedro, L. E. C. Rocha, A. J. M. Traina, and J. Poco, "LargeNetVis: Visual exploration of large temporal networks based on community taxonomies," *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 1, pp. 203–213, Jan. 2023, doi: 10.1109/TVCG.2022.3209477.

[35] C. D. G. Linhares, J. R. Ponciano, F. S. F. Pereira, L. E. C. Rocha, J. G. S. Paiva, and B. A. N. Travençolo, "A scalable node ordering strategy based on community structure for enhanced temporal network visualization," *Comput. Graph.*, vol. 84, pp. 185–198, 2019.

[36] C. D. G. Linhares, B. A. N. Travençolo, J. G. S. Paiva, and L. E. C. Rocha, "Dynetvis: A system for visualization of dynamic networks," in *Proc. Symp. Appl. Comput.*, New York, NY, USA, 2017, pp. 187–194, doi: 10.1145/3019612.3019686.

[37] J. Lukasczyk et al., "Viscous fingering: A topological visual analytic approach," *Appl. Mechan. Mater.*, vol. 869, pp. 9–19, 2017.

[38] J. Lukasczyk, C. Garth, G. H. Weber, T. Biedert, R. Maciejewski, and H. Leitte, "Dynamic nested tracking graphs," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 1, pp. 249–258, Jan. 2020, doi: 10.1109/TVCG.2019.2934368.

[39] J. Lukasczyk, G. Weber, R. Maciejewski, C. Garth, and H. Leitte, "Nested tracking graphs," *Comput. Graph. Forum*, vol. 36, no. 3, pp. 12–22, 2017, doi: 10.1111/cgf.13164.

[40] R. Mastrandrea, J. Fournet, and A. Barrat, "Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys," *PLoS One*, vol. 10, no. 9, pp. 1–26, Sep. 2015, doi: 10.1371/journal.pone.0136497.

[41] A. Myers, C. Joslyn, B. Kay, E. Purvine, G. Roek, and M. Shapiro, "Topological analysis of temporal hypergraphs," in *Proc. 18th Int. Workshop Algorithms Models Web-Graph*, 2023, pp. 127–146.

[42] A. Myers, D. Muñoz, F. A. Khasawneh, and E. Munch, "Temporal network analysis using zigzag persistence," *EPJ Data Sci.*, vol. 12, no. 1, pp. 1–19, 2023.

[43] P. Niyogi, S. Smale, and S. Weinberger, "Finding the homology of submanifolds with high confidence from random samples," *Discrete Comput. Geom.*, vol. 39, no. 1–3, pp. 419–441, 2008.

[44] G. K. Orman, N. Türe, S. Balcisoy, and H. A. Boz, "Finding proper time intervals for dynamic network extraction," *J. Statist. Mech: Theory Exp.*, vol. 2021, no. 3, 2021, Art. no. 033414.

[45] S. Paris and F. Durand, "A. topological approach to hierarchical segmentation using mean shift," in *Proc. 2007 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.

[46] G. Plonka and Y. Zheng, "Relation between total variation and persistence distance and its application in signal processing," *Adv. Comput. Math.*, vol. 42, pp. 651–674, 2016.

[47] J. R. Ponciano, C. D. Linhares, E. R. Faria, and B. A. Travençolo, "An online and nonuniform timeslicing method for network visualisation," *Comput. Graph.*, vol. 97, pp. 170–182, 2021, doi: 10.1016/j.cag.2021.04.006.

[48] J. R. Ponciano, C. D. G. Linhares, S. L. Melo, L. V. Lima, and B. A. N. Travençolo, "Visual analysis of contact patterns in school environments," *Informat. Educ.*, vol. 19, no. 3, pp. 455–472, 2020, doi: 10.15388/infedu.2020.20.

[49] J. R. Ponciano, C. D. G. Linhares, L. E. C. Rocha, E. R. Faria, and B. A. N. Travençolo, "Combining clutter reduction methods for temporal network visualization," in *Proc. 37th ACM/SIGAPP Symp. Appl. Comput.*, New York, NY, USA, 2022, pp. 1748–1755, doi: 10.1145/3477314.3507018.

[50] J. R. Ponciano, C. D. G. Linhares, L. E. C. Rocha, E. R. Faria, and B. A. N. Travençolo, "A. streaming edge sampling method for network visualization," *KAIS*, vol. 63, no. 7, pp. 1717–1743, 2021, doi: 10.1007/s10115-021-01571-7.

[51] B. Rieck, U. Fugacci, J. Lukasczyk, and H. Leitte, "Clique community persistence: A topological visual analysis approach for complex networks," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 01, pp. 822–831, Jan. 2018, doi: 10.1109/TVCG.2017.2744321.

[52] L. E. C. Rocha, F. Liljeros, and P. Holme, “Simulated epidemics in an empirical spatiotemporal network of 50,185 sexual contacts,” *PLoS Comput. Biol.*, vol. 7, no. 3, Mar. 2011, Art. no. e1001109.

[53] L. E. C. Rocha, N. Masuda, and P. Holme, “Sampling of temporal networks: Methods and biases,” *Phys. Rev. E*, vol. 96, Nov. 2017, Art. no. 052302, doi: 10.1103/PhysRevE.96.052302.

[54] A. E. Sizemore and D. S. Bassett, “Dynamic graph metrics: Tutorial, toolbox, and tale,” *NeuroImage*, vol. 180, pp. 417–427, 2018.

[55] S. Soundarajan et al., “Generating graph snapshots from streaming edge data,” in *Proc. World Wide Web Conf.*, 2016, pp. 109–110.

[56] N. Stanley, R. Kwitt, M. Niethammer, and P. J. Mucha, “Compressing networks with super nodes,” *Sci. Rep.*, vol. 8, no. 1, Jul. 2018, Art. no. 10892.

[57] A. Suh, M. Hajij, B. Wang, C. Scheidegger, and P. Rosen, “Persistent homology guided force-directed graph layouts,” *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 1, pp. 697–707, Jan. 2020, doi: 10.1109/TVCG.2019.2934802.

[58] R. Sulo, T. Berger-Wolf, and R. Grossman, “Meaningful selection of temporal resolution for dynamic networks,” in *Proc. 8th Workshop Mining Learn. Graphs*, 2010, pp. 127–136.

[59] S. Uddin, N. Choudhury, S. Farhad, and M. Rahman, “The optimal window size for analysing longitudinal networks,” *Sci. Rep.*, vol. 7, no. 1, 2017, Art. no. 13389.

[60] P. Valdivia, P. Buono, C. Plaisant, N. Dufournaud, and J.-D. Fekete, “Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization,” *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 1, pp. 1–13, Jan. 2021.

[61] S. van den Elzen, D. Holten, J. Blaas, and J. van Wijk, “Dynamic network visualization with extended massive sequence views,” *IEEE Trans. Vis. Comput. Graph.*, vol. 20, pp. 1087–1099, 2014.

[62] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk, “Reordering massive sequence views: Enabling temporal and structural analysis of dynamic networks,” in *Proc. 2013 IEEE Pacific Visual. Symp.*, 2013, pp. 33–40, doi: 10.1109/PacificVis.2013.6596125.

[63] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk, “Reducing snapshots to points: A visual analytics approach to dynamic network exploration,” *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 1, pp. 1–10, Jan. 2016, doi: 10.1109/TVCG.2015.2468078.

[64] P. Vanhems et al., “Estimating potential infection transmission routes in hospital wards using wearable proximity sensors,” *PLoS One*, vol. 8, 2013, Art. no. e73970, doi: 10.1371/journal.pone.0073970.

[65] Y. Wang, D. Archambault, H. Haleem, T. Moeller, Y. Wu, and H. Qu, “Nonuniform timeslicing of dynamic graphs based on visual complexity,” in *Proc. 2019 IEEE Visual. Conf.*, 2019, pp. 1–5.

[66] W. Widanagamaachchi, C. Christensen, V. Pascucci, and P.-T. Bremer, “Interactive exploration of large-scale time-varying data using dynamic tracking graphs,” in *Proc. IEEE Symp. Large Data Anal. Visual.*, 2012, pp. 9–17, doi: 10.1109/LDAV.2012.6378962.

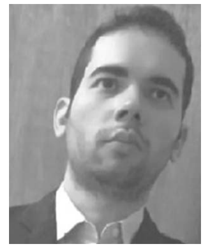
[67] P. Wills and F. G. Meyer, “Metrics for graph comparison: A practitioner’s guide,” *PLoS One*, vol. 15, no. 2, pp. 1–54, 2020, doi: 10.1371/journal.pone.0228728.

[68] L. Xie, J. O’Donnell, B. Bach, and J.-D. Fekete, “Interactive time-series of measures for exploring dynamic networks,” in *Proc. Int. Conf. Adv. Vis. Interfaces*, 2020, pp. 1–9.

[69] V. Yoghourdjian, T. Dwyer, K. Klein, K. Marriott, and M. Wybrow, “Graph thumbnails: Identifying and comparing multiple graphs at a glance,” *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 12, pp. 3081–3095, Dec. 2018, doi: 10.1109/TVCG.2018.2790961.

[70] Y. Zhao et al., “EOD edge sampling for visualizing dynamic network via massive sequence view,” *IEEE Access*, vol. 6, pp. 53006–53018, 2018.

Raphaël Tinarrage received the PhD degree in pure and applied mathematics from the Université Paris-Saclay, in 2020. He passed the agrégation (national teaching qualification) while studying with the École normale supérieure Paris-Saclay. He is currently a research fellow with the Institute of Science and Technology Austria, and previously at the Fundação Getúlio Vargas EMAp. His work focuses on Topological Data Analysis, in its theoretical developments and practical applications.



Jean R. Ponciano is an assistant professor with the Mathematics and Computer Science Institute - University of São Paulo, Brazil. His research interests include information visualization, visual analytics, network science, and data streams. He frequently serves as a program committee member for relevant conferences, including IEEE Vis, EuroVis, and ASONAM, and as an external reviewer for other relevant venues, such as *IEEE Transactions on Visualization and Computer Graphics*, Compter Graphics Forum, Computers & Graphics.

Claudio D. G. Linhares is a senior lecturer with Linnaeus University, Department of Computer Science and Media Technology Faculty of Technology, in Växjö, Sweden. His research interests include information visualization, network visualization, human-computer interaction, visual analytics, and human-in-the-loop AI, with applications especially in forestry and healthcare.

Agma J. M. Traina received the BSc and MSc degrees in computer science and PhD degree in computational applied physics from the University of São Paulo, Brazil, in 1983, 1987 and 1991 respectively. She also spent a sabbatical leave as a visiting researcher with the Computer Science Department, Carnegie Mellon University (1998-2000) working on Multimedia Databases. She is a full professor with the Mathematics and Computer Science Institute - University of São Paulo, since 2008. Her research interests include indexing and retrieval of complex data by content, similarity queries, data visualization, visual data mining, as well as image and video processing. She has supervised over 60 graduate students in those areas. She has been working in the integration of the results of her research lines with applications to medicine, aimed at the development of applied computational systems. She is a member of the Brazilian Computer Society, ACM and IEEE Computer Society.

Jorge Poco received the MSc degree in computer science from the University of São Paulo, Brazil, and the PhD degree in computer science from New York University. He is an associate professor with the School of Applied Mathematics, Fundação Getúlio Vargas (FGV), Brazil. His research interests include focuses on data visualization, visual analytics, machine learning, and data science. He has actively contributed to program committees for IEEE SciVis, IEEE InfoVis, VAST, and EuroVis.