IOP Publishing Journal of Physics A: Mathematical and Theoretical

J. Phys. A: Math. Theor. **57** (2024) 435002 (39pp) https://doi.org/10.1088/1751-8121/ad7b8e

Flow of temporal network properties under local aggregation and time shuffling: a tool for characterizing, comparing and classifying temporal networks

Didier Le Bail[*], **Mathieu Génois and Alain Barrat**

Aix Marseille Univ, Université de Toulon, CNRS, CPT, Marseille, France

E-mail: didier.lebail@cpt.univ-mrs.fr

Received 14 October 2023; revised 8 August 2024
Accepted for publication 16 September 2024
Published 10 October 2024

Abstract

Although many tools have been developed and employed to characterize temporal networks (TNs), the issue of how to compare them remains largely open. It depends indeed on what features are considered as relevant, and on the way the differences in these features are quantified. In this paper, we propose to characterize TNs through their behavior under general transformations that are local in time: (i) a local time shuffling, which destroys correlations at time scales smaller than a given scale *b*, while preserving large time scales, and (ii) a local temporal aggregation on time windows of length *n*. By varying *b* and *n*, we obtain a flow of TNs, and flows of observable values, which encode the phenomenology of the TN on multiple time scales. We use a symbolic approach to summarize these flows into labels (strings of characters) describing their trends. These labels can then be used to compare TNs, validate models, or identify groups of networks with similar labels. Our procedure can be applied to any TN and with an arbitrary set of observables, and we illustrate it on an ensemble of data sets describing face-to-face interactions in various contexts, including both empirical and synthetic data.

Keywords: complex systems, temporal networks, social systems, face-to-face interactions, generative models, symbolic analysis

1. Introduction

A large variety of natural and artificial systems can be described as networks, in which nodes represent the elements of the system and links represent their interactions. The corresponding

[*] Author to whom any correspondence should be addressed.

© 2024 IOP Publishing Ltd.
All rights, including for text and data mining, AI training, and similar technologies, are reserved.

data are increasingly available with temporal resolution, which has led to the development of the study of temporal networks (TNs), in which each link can be alternatively active or inactive [1, 2]. The intrinsic dynamic nature of TNs leads however to fundamental differences with respect to static networks, such as changes in the importance of nodes at different times [3], complex temporal correlations [4] that can alter dynamical processes such as information or disease propagation [5–7], and enhanced flexibility that help control the system they describe [8]. To better understand these rich properties, a growing body of literature has recently emerged to deal with the development of analysis tools for these complex objects [4, 9–18] and of models to describe them [19–23]. In this context, here we are interested in the issue of characterizing, comparing and classifying TNs or ensembles of TNs: being able to compare quantitatively networks is indeed needed for instance to detect differences between data sets of an *a priori* similar nature, or to validate models and quantify how well they represent data. This issue is also crucial for static networks, for which a large set of comparison methods has been developed [24–27], but the case of TNs has barely been considered [2, 28, 29]. When are two TNs equivalent or what is their degree of similarity? Depending on the properties that are considered as most relevant, several methods can be devised to answer these questions: if two TNs share such properties, they are then considered as equivalent. For instance one can generalize notions of distance between static networks, using distributions of path lengths [29] or statistics of motifs [28]. One can also consider a TN as a multi-dimensional time series and characterize these series e.g. by their Fourier transform or other tools [30–32].

As in the case of static networks, another approach lies in adopting a statistical point of view, by considering a set of variables (observables) that can be sampled from the network, and their resulting distributions [24]: these distributions are then considered as the set of relevant properties able to characterize a data set or a model, and the similarity of these distributions is seen as corresponding to the similarity of the data or as the ability of the model to represent the data [19–23, 33, 34]. This statistical point of view comes from the underlying hypotheses that the empirical TN under study results from some stochastic process (often implicitly approximated as stationary), and that two sets of similar distributions of observables come from similar stochastic processes. In TNs for instance, observables commonly used to this purpose include the duration of an interaction and the time elapsed between two consecutive interactions [21, 35–38]. This approach has several limitations: (1) the sampling of observables is usually performed at the temporal resolution given by the data at hand, while network dynamics exist typically at many timescales, and moreover different TNs might be collected with different time resolutions [39–42]; the choice of an adequate resolution is non trivial as every scale of observation can potentially give different information about a social system [43, 44]; (2) moreover, sharing the same distributions does not mean sharing the same correlations, so that sampling observables and obtaining their distributions might not be enough to characterize a TN [37]; (3) distributions are in general difficult to compare, especially if they present broad functional shapes [45, 46].

In this article, we present a methodology to go beyond these limitations. To this aim, we propose a labeling procedure of TNs, based on their transformation under reshuffling at a certain scale and temporal aggregation at another scale. By varying the scales of reshuffling and aggregation, we indeed create a two-dimensional flow for the TN and for any attached scalar observable of interest. This allows to go beyond limitation (1) by including information on all time scales in the resulting label. Moreover, by using a reshuffling procedure at a given scale, we remove short scale temporal correlations while preserving long range ones (intuitively, this amounts to apply a low-pass filter acting on the time correlation function). By varying this scale, we include information about not only distributions of observables but about correlations in the TN, going beyond the limitation (2). As comparing flows of observables is not an easy task, we simply summarize each flow of a scalar observable under a one-parameter transformation by the sign sequence of its derivative, which encodes the shape of the corresponding curve. As we have a two-dimensional flow, we have thus two varying parameters (the scales of time shuffling and of time aggregation) and hence two sets of sign sequences that we combine to form two sentences. Each TN is thus finally labeled by strings (two for each observable of interest), which can then be compared. We can now define an equivalence between TNs as the fact that they share the same label, and we can define a distance between two TNs through a distance between labels (e.g. the edit distance). Note that this procedure is limited to scalar observables (having a single realization per TN), while the focus is often on distributions, e.g. of interaction durations. As in [24], we will thus reduce each distribution of interest to its moments.

In the following, we describe step by step the characterization procedure and how to go from a TN to a label. We then introduce tools to analyze those labels, use them to compare TNs or groups of TNs. Finally, we illustrate our procedure and tools of analysis with the study of 27 TNs through the flows of 43 different observables [17, 19, 22, 34, 37, 40, 41, 47–55]. We note that our procedure can be applied to any TNs as long as they are defined in discrete time. Moreover, it can be tailored to any specific field through an adequate choice of observables relevant to that field. Here, we will focus for illustration purposes on TNs describing face-to-face interactions between individuals in various social contexts. On the one hand indeed, a number of data sets describing such contacts in various contexts are publicly available [39, 56]. On the other hand, a number of TN models have been proposed to mimic the mechanisms at play in social networks and the resulting phenomenology, and we will also consider several of those [19, 22, 34, 55].

2. From a TN to a label

The first step in labeling a TN is to create the transformation flow under removal of short-range time correlations and time aggregation. Each of these transformations is characterized by a single integer parameter, that we will denote by b for the reshuffling procedure and n for the time aggregation in the remainder of the paper.

Let us consider a TN $\mathcal{G}$ in discrete time $t = 0, \ldots, T-1$. We denote by $G(t)$ the snapshot at time t, which is formed by all active edges at t. We first proceed with the time shuffling (filtering) transformation using a sliding time window, as illustrated in figure 1, i.e. we apply a sliding time shuffling (STS). Specifically, to apply a STS of range b to a TN, we browse its timeline one time step after the other. At each time step k, we reshuffle at random the b following temporal snapshots $G(k), \ldots, G(k+b-1)$ [57]. The procedure is then iterated at time $k+1$.[1] Successively, we aggregate the TN obtained after reshuffling at level n using sliding windows as well: each snapshot $G(t)$ is replaced by $G^{(n)}(t)$ resulting from the aggregation of snapshots $G(t), G(t+1), \ldots, G(t+n-1)$.

We denote by TN(n,b) the TN obtained after a STS of range b followed by a time aggregation at level n. TN$(n=1, b=1)$ corresponds to the original data set. Then, for each observable $\mathcal{O}$, we can compute its realization $\mathcal{O}(n,b)$ in the transformed network TN(n,b). Note that, to reduce the noise originating from the random shufflings, we average observable values over ten independent STS realizations.

[1] Note that instead of using a sliding procedure, we could divide the TN timeline into disjoint blocks of length b and shuffle each block separately. As discussed in appendix A.1, this turns out to introduce spurious deterministic noise and oscillations in the values of the observables.

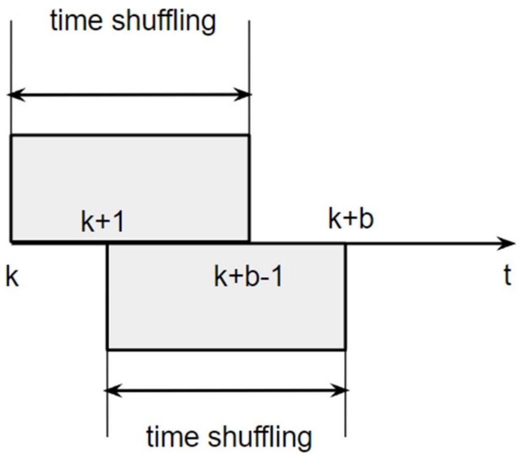
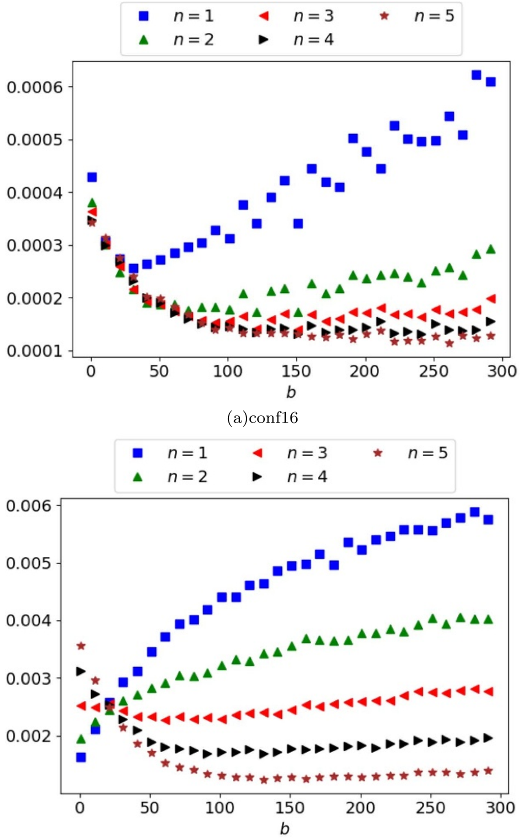


Figure 1. Illustration of sliding time shuffling. b is the integer parameter of the sliding time shuffling, called its range. k denotes a timestamp in the TN timeline. Time shuffling is done iteratively following this timeline, so the transformation is not strictly local. However, the distance travelled by a timestamp follows an exponential law, with average $2b$. This removes any correlation on time scales shorter than $2b$ but preserves correlations on larger time scales.

Collecting values $\mathcal{O}(n,b)$ for various n and b results in a bi-dimensional flow $(n,b) \mapsto \mathcal{O}(n,b)$ for each observable $\mathcal{O}$. This global flow can be viewed as a collection indexed by n (resp. indexed by b) of one dimensional flows versus b (resp. versus n), see figure 2 for examples. For a given value of b, the partial flow versus n is written as $\mathcal{O}(.,b)$. Symmetrically, the partial flow versus b, at fixed n, is written as $\mathcal{O}(n,.)$.

Each partial flow is a one dimensional curve, describing how the associated observable is changing under the associated transformation: $\mathcal{O}(n,.)$ tells us how $\mathcal{O}$ is changing under STS with varying b, while $\mathcal{O}(.,b)$ determines how $\mathcal{O}$ is changing under sliding time aggregation of varying length n. While a standard approach might focus on the values taken by the observable, we make here the crucial hypothesis that the most important information on the TN structure and correlations is contained in the shape of these flows. Therefore, we do not focus on the specific values of the observables but on the trends of their evolution with n and b. We thus need to extract such information from the derivatives of $\mathcal{O}(.,b)$ with respect to n and of $\mathcal{O}(n,.)$ with respect to b. Moreover, in order to focus on trends, we do not collect derivatives in each point of the curves but in the observed succession of trends, such as, e.g. 'the curve first increases then decreases'. We emphasize that two observables taking different values, and varying on different ranges, but having the same succession of increasing and decreasing trends, will thus be summarized in the same way: our procedure completely loses the quantitative information and focuses on qualitative trends, hypothesized to still retain enough relevant information for our purpose of comparing networks.

To automatically extract the information about the trends, we determine the denoised sign sequence of the partial flows derivatives using an artificial neural network that we build for this purpose (see appendix F for the description of the neural network and of the training procedure). The resulting sign sequence describing each partial flow is encoded as a word made up with letters '$+$' (for increasing parts of the curve), '$-$' and '0' (for decreasing and flat parts of the curve). Note that consecutive letters in a word are always different because, as discussed above, we focus on the successive trends (the successive signs taken by the derivative), rather



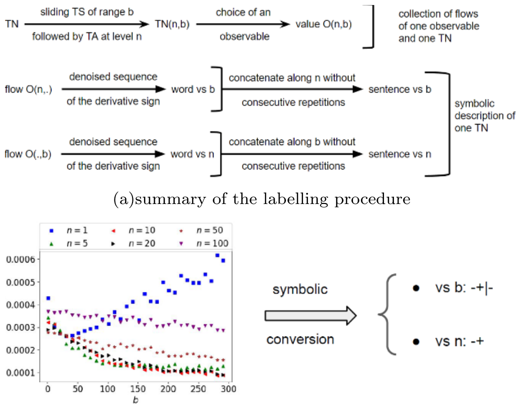
(a)conf16

(b)ADM18conf16

Figure 2. Bi-dimensional flows of a specific observable '3-NCTN_error' in two temporal networks. (See appendix G for the explanation of the nomenclature of observables.) Here, bi-dimensional flows are presented as collections of partial flows versus b, for various values of n (cf text). (a) data set 'conf16' (describing contacts in a scientific conference [41]), (b) data set 'ADM18conf16' (model of temporal network from the ADM class [34]).

than on its sign at every point of the flow. As a result, each partial flow versus n yields a word 'versus n', i.e. describing its behavior versus n, and we thus obtain one such word for each value of b. Similarly, partial flows versus b yield words describing their behavior versus b, one per value of n. For instance, in the two examples shown in figure 2, the obtained word for $n = 1$ are $-+$ (panel (a)) and $+$ (panel (b)), and -0 in both panels for $n = 5$.

After this procedure, we obtain two indexed sets of words: one set describing behaviors versus n and indexed by b, and one set describing behaviors versus b and indexed by n. We concatenate all words in each set (in order of increasing b or n) into a sentence. As in the case of words, we moreover remove consecutive repetitions of words within the sentence. Overall, the sentence versus n is obtained by (1) ordering words versus n by increasing value of b (2) removing consecutive repetitions (3) adding a separator $|$ between words to obtain a string. For instance, if successive words versus n are $-+$, $-+$, $-+$, -0, $0+$ for the values of b in increasing order, the final sentence versus n is $-+|-0|0+$. The sentence versus b is obtained in the same fashion.



(a)summary of the labelling procedure

(b)application to a particular case

Figure 3. Workflow of the labeling by sentences of one TN viewed through the lens of one observable. (a) Labeling workflow. Flows are averaged over ten realizations of the STS and words are extracted by a neural network. (b) Illustration with the 'conf16' TN and observable '3-NCTN_error' (same as in figure 2(a). To obtain the sentence versus b, we consider the flows versus b at each value of n: for $n = 1$ the flow is first decreasing then increasing, which yields the word $-+$; for larger values of n the flows are always decreasing, yielding words $-$. As we build sentences by removing consecutive repetitions of words, we obtain $-+|-$ as sentence versus b (words are separated by the symbol '|'). The sentence versus n is simply $-+$ because the flows versus n are decreasing then increasing for all values of b.

We finally define the label of the TN under study as the couple of sentences obtained (one versus n and one versus b). Note that this label is relative to the observable whose flows are computed: we obtain one label per observable. A summary and illustration of the labeling procedure described above are given in figure 3.

3. Comparing TNs

The labeling procedure opens the way to various paths of investigation, as one can use them to define a similarity measure between TNs. For instance one can then ask what is the performance of a given generative model by comparing the labels of its instances to the ones of empirical networks. One can also use the labels of several networks to cluster them, or to investigate the heterogeneity of a set of TNs. In the following we propose a concrete way to do this.

3.1. Representation of the space of TNs

We now view each instance of a TN through the lens of a set of scalar observables, each giving rise to a mapping from the TN to a label. For each observable, we can then collect all obtained labels, and build a diagram as follows. Each point in the diagram corresponds to a label, and can also be seen as the set of TNs sharing that label: a diagram is thus dependent on the observable considered. Note that such diagrams are potentially suitable to study both TNs and observables. Here we will focus on their use for TNs.

Since labels are couples of sentences, a diagram can be viewed as a metric space: the difference between two labels can indeed be simply defined as the shortest sequence of operations for rewriting one couple into the other. If we take insertion, deletion and substitution as

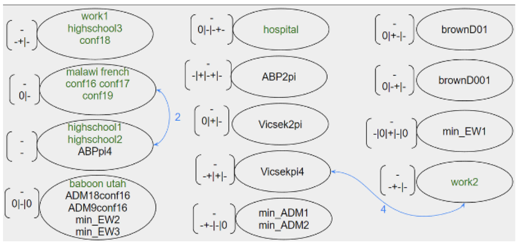


Figure 4. Diagram obtained when labeling temporal networks with respect to a specific observable. (namely '2-ECTN0sim_trunc0vs_n', see appendix G for the explanation of the nomenclature of observables.) Each node of the diagram is both a label and the group of temporal networks sharing this label. The first (resp. second) component of the label is the sentence versus n (resp. b). Note that for this specific observable, the same behavior is observed versus n for all temporal networks. The temporal networks considered are empirical data on face-to-face interactions (data set names written in green) and synthetic data generated from several models (names of models written in black). For readability, only two edges and the corresponding values of $g_{i,j}$ are shown in blue.

elementary operations, the length of this sequence is the edit distance between the two couples: the distance between two couples of sentences (s,t) and (s',t') is the sum of the pairwise edit distances of their members: $d[(s,t),(s',t')] = d(s,s') + d(t,t')$. This metric information can be made explicit by representing a diagram as a fully connected weighted directed graph, whose nodes are labeled sets of TNs and the edge from i to j bears as weight $g_{i,j}$ the length of the shortest sequence of operations that returns j if applied to i (see figure 4 for an example).

3.2. Clustering TNs using diagrams

A diagram tells us how the space of TNs looks like when seen through the lens of a scalar observable. Several clustering are then possible, depending on whether we want to compare TNs or classes of TNs. Let us introduce useful notations to this aim:

- we denote by $D(\mathcal{O})$ the diagram associated to the observable $\mathcal{O}$;
- we call a set or a family of TNs a class C;
- for $i \in D(\mathcal{O})$ a node of the diagram, we denote by $n_C[\mathcal{O}](i)$ the number of TNs from the class C that belong to i, i.e. that are labeled by the label of the node i. $n_C[\mathcal{O}]$ is called the repartition function of the class C in the diagram $D(\mathcal{O})$;
- $I_{\mathcal{O}}$ maps each TN to the node in $D(\mathcal{O})$ that contains it;
- S denotes the space of observables considered; it is a finite subset of the space of all possible observables.

Each diagram provides a clustering of TNs (simply by grouping together the networks with the same label). However, such clusterings are observable-dependent, while we would like to combine information from all the considered observables in a systematic way. A first, naive solution would consist in considering two TNs to be equivalent or part of the same cluster if and only if their labels match for every observable. Such a condition is however too restrictive. Another possibility is to consider the space of observables as a sampling space, and a measure of similarity or distance between two networks as a random sampling experiment: specifically, to compare two networks, we draw at random an observable and we compare the labels of the two networks for the chosen observable. A difficulty of this approach is to define and justify a probability distribution over observables. Here, for simplicity we consider a uniform distribution over all observables considered (see section 4.1 for a presentation of the observables we have used).

3.2.1. Comparing two TNs. Let us consider two TNs G and G'. If we draw an observable at random, the probability that we obtain the same label for G and G' can be written as:

$$\frac{1}{|S|}\sum_{\mathcal{O}\in S}\delta_{I_{\mathcal{O}}(t),I_{\mathcal{O}}(t')}\ .$$

It can be interpreted as a degree of similarity between the two TNs: if it equals 1, G and G' always share the same labels so they behave in similar ways with respect to the reshuffling and aggregation procedures, for all considered observables. If instead this probability is equal to 0, G and G' behave differently for all observables. We call this similarity measure the *raw similarity*, because it does not consider any metric structure associated with the labels.

By taking into account the metric structure, we can define another similarity measure. First, note that we can introduce a distance between TNs relative to any chosen observable:

$$d_{\mathrm{TN}}\left(G,G'|\mathcal{O}\right)=g\left(I_{\mathcal{O}}\left(G\right),I_{\mathcal{O}}\left(G'\right)\right)\ ,$$

where g is the edit distance introduced above. We then rescale this distance between 0 and 1 for each observable through:

$$\hat{d}_{\mathrm{TN}}\left(G,G'|\mathcal{O}\right)=\frac{1}{M}d_{\mathrm{TN}}\left(G,G'|\mathcal{O}\right)$$

where:

$$M=\max_{G,G'}\left(d_{\mathrm{TN}}\left(G,G'|\mathcal{O}\right)\right)$$

is the maximal distance encountered between two TNs among the set we analyze. Assuming we restrict the analysis to a finite set of networks, M is well-defined. The rescaled distance can then be transformed into a similarity depending on the chosen observable, and we obtain a global similarity measure by taking the arithmetic average of the similarities obtained for all considered observables (Note also that one could potentially define a non uniform average by putting more weight on specific observables of interest):

$$\mathrm{Sim}_{ar}\left(t,t'\right)=\frac{1}{|S|}\sum_{\mathcal{O}\in S}\left(1-\hat{d}_{\mathrm{TN}}\left(t,t'|\mathcal{O}\right)\right)\ .$$

When using an arithmetic average however, a strong difference between two networks on one observable can be ‘hidden’ by the averaging procedure. A more stringent way of averaging is the geometric average

$$\mathrm{Sim}\left(t,t'\right)=\left(\prod_{\mathcal{O}\in S}\left(1-\hat{d}_{\mathrm{TN}}\left(t,t'|\mathcal{O}\right)\right)\right)^{\frac{1}{|S|}}\ .$$

Indeed, it is then enough for the two networks to be dissimilar on one specific observable for the similarity to be small. We thus obtain a better contrast when comparing a set of networks with each other by using this geometric average. We call this similarity measure the *metric similarity* in the following.

Both the raw and metric similarities can be used to evaluate how far a generative model is from the empirical network it should represent. They can also be used as objective functions to be maximized by e.g. a genetic algorithm encoding a model to tune its parameter and obtain synthetic data as close as possible to given empirical data (as done e.g. in [34] with a different objective function).

Moreover, each similarity measure between TNs gives rise to a weighted undirected graph, where nodes are TNs and edge weights are similarities between nodes. Such a graph representation can then be leveraged to detect clusters of TNs by running any community detection algorithm on it (see section 4.3 for examples using empirical and synthetic data).

3.2.2. Characterizing a class of TNs. Given a class C of TNs, such as empirical data collected in similar environments (e.g. schools), or as a set of instances of a model with different parameter values, another relevant question concerns its characterization as a class. A first characterization can be given by the set of pairwise similarities between the TNs of the class. It is however possible to go further by using the repartition functions of C for different observables. Namely, given an observable $\mathcal{O}$ and $n_C[\mathcal{O}]$ the associated repartition function, we define the probability for C to occupy the node i as

$$P_C[\mathcal{O}](i) = \frac{1}{|C|} n_C[\mathcal{O}](i)$$

and call this probability the occupancy probability of the class C. For each observable $\mathcal{O}$, we can then extract several properties of C (we omit the label $\mathcal{O}$ to make definitions more readable):

- the *area* of C is the number of nodes with a non-zero probability to be occupied by C. It is an integer bounded between 1 and $|C|$. If it equals 1 (resp. $|C|$), $\mathcal{O}$ is said to be universal (resp. specific) with respect to C.
- the *diameter* of C is the average distance between labels encountered in C: $\sum_{i,j} d(i,j) P_C(i) P_C(j)$. It indicates how different on average are two TNs extracted at random in C.
- the *heterogeneity* is given by the entropy of P_C, which indicates how uniformly the TNs of C are distributed among the nodes occupied by C.

Linking P_C to our previous pairwise distances framework allows us to introduce a further property, which quantifies the relationship between a change in the occupancy probability and a change in labels:

$$\forall i,j \in D(\mathcal{O}),\ K^C_{i,j} = \frac{|P_C(i) - P_C(j)|}{|g_{i,j}|}$$

The definition of K^C is reminiscent of the definition of a derivative. Thus, small values in K^C indicate that the occupancy probability varies smoothly with respect to the labels: close labels contain close amounts of TNs belonging to C. From a statistical point of view, it would mean that variations in P_C are bounded by variations in labels, indicating a correlation between the two.

3.2.3. Comparing two classes of TNs. Let us consider a model of TNs with its parameters, and a class of empirical TNs, such as e.g. TNs describing social interactions in various conferences. An important question about the relevance of the model concerns the probability that the model produces TN instances similar to the empirical ones. We can quantify this by asking, when we vary the model parameters, how many of the resulting artificial TNs share a label with an empirical one? The corresponding overlap between the synthetic class C and the empirical class C' can be defined as the co-occupancy probability of the two classes:

$$\mathrm{Overlap}\left(C, C'|\mathcal{O}\right) = P\left(C \cap C'|\mathcal{O}\right) = \sum_{i \in D(\mathcal{O})} P_C\left(i\right) P_{C'}\left(i\right).$$

The arithmetic average of this overlap over all considered observables yields a similarity measure between classes that we call *overlap similarity* (as in previous cases, one could also define a weighted average giving more importance to specific observables):

$$\mathrm{Overlap}\left(C, C'\right) = \frac{1}{|S|} \sum_{\mathcal{O} \in S} \mathrm{Overlap}\left(C, C'|\mathcal{O}\right).$$

We also note that this similarity measure can be used to map classes of TNs into a weighted undirected network, where nodes are classes and edge weights are global overlaps between classes.

4. Results

We now illustrate the procedure and concepts developed above in concrete cases, using both empirical and synthetic data sets.

4.1. TN data sets

We consider 27 data sets describing TNs, corresponding to (i) 14 publicly available empirical data sets on social interactions with high temporal resolution [33, 39, 56] and (ii) 13 to models of TNs. We consider 6 models representing the dynamics of pedestrians and their physical proximity, 4 TN models from the activity driven with memory (ADM) class [19, 22, 34], and 3 ad hoc models of temporal edge dynamics.

In this sub-section, we give a brief description of the data sets: for the empirical data, we indicate when and where the data were collected and for the models we give the principle behind them. For more details about the sizes of the different data sets, see appendix D. For more details about the models, see appendices B and C.

Note that, while here for illustration purposes we focus on TNs of face-to-face interaction, our framework is applicable to any type of TNs, including networks of higher-order interactions [58].

4.1.1. Empirical data sets. The empirical TNs we use represent face-to-face interaction data collected in various contexts using wearable sensors that exchange low-power radio signals [39, 56]. This allows to detect face-to-face close proximity with here a temporal resolution of about 20 s [38]. All the data we used have been made publicly available by the research collaborations who collected the data [39, 56]. They correspond to data collected among human individuals in conferences, schools, a hospital, workplaces, and also within a group of baboons.

In all cases, individuals are represented as nodes, and an edge is drawn between two nodes each time the associated individuals are interacting with each other.

The data sets we consider are:

- ‘conf16’, ‘conf17’, ‘conf18’, ‘conf19’: these data sets were collected in scientific conferences, respectively the 3rd GESIS Computational Social Science Winter Symposium (30 November and 1 December 2016), the International Conference on Computational Social Science (10–13 July 2017), the Eurosymposium on Computational Social Science (December 5 to 7, 2018), and the 41st European Conference on Information Retrieval (14–18 April 2019) [41];
- the ‘utah’ data set describes the proximity interactions which occurred on 28 and 29 November 2012 in an urban public middle school in Utah (USA) [39];
- the ‘french’ data set contains the TN of contacts between the children and teachers that occurred in a french primary school on Thursday, 1 October and Friday, 2 October 2009. It is described in [47, 48];
- the ‘highschool1’, ‘highschool2’ and ‘highschool3’ data set describe the interactions between students in a high school in Marseille, France [40, 49]. They were respectively collected for three classes during four days in December 2011, five classes during seven days in November 2012 and nine classes during 5 days in December 2013;
- the ‘hospital’ data set contains the TN of contacts between patients and health-care workers (HCWs) and among HCWs in a hospital ward in Lyon, France, from 6–10 December 2010. The study included 46 HCWs and 29 patients [50];
- the ‘malawi’ data set contains the list of contacts measured between members of 5 households of rural Kenya between 24 April and 12 May 2012 [51];
- the ‘baboon’ data set contains observational and wearable sensors data collected in a group of 20 Guinea baboons living in an enclosure of a Primate Center in France, between 13 June and 10 July 2019 [52];
- the ‘work1’ and ‘work2” data sets contain the TN of contacts between individuals measured in an office building in France, respectively from 24 June to 3 July 2013 [53] and during two weeks in 2015 [54].

Note that, in all empirical data sets, the night and week-end periods have been removed (long periods without any activity) for simplicity and for a fairer comparison with the models in which no such inactive periods are present. However, large fluctuations of activity are still observed, as documented in previous studies of these data.

4.1.2. Pedestrian models [59]. Pedestrian models consist in stochastic agent-based models implemented in discrete time. These agents move through a two-dimensional space and are point oriented particles. In the simulations considered here, the 2D space is a square with reflecting boundary conditions. A TN is built from the agents’ trajectories according to a rule similar to the one used in empirical face-to-face interactions: an interaction between two agents i and j is recorded at time t if i and j are close enough and oriented towards each other at t.

In all pedestrian models considered in this paper, agents are point particles. Three types of models are considered:

- Brownian particles without any interaction: ‘brownD01’ and ‘brownD001’. These models differ only by the value of the diffusion coefficient of the agents.

- Active Brownian particles [60]: 'ABP2pi' and 'ABPpi4'. The orientation vector of an active Brownian particle follows a Brownian motion, whereas its position vector enjoys an overdamped Langevin equation with a self-propelling force. This force has constant magnitude and is parallel to the orientation vector. In our case, however, the noise contributing to the velocity vector is zero, meaning the velocity is equal to the self-propelling force. Besides, the noise giving the angular velocity is a uniform random variable in $[-\theta,\theta]$. In 'ABP2pi', $\theta=\pi$ and in 'ABPpi4', $\theta=\frac{\pi}{8}$.
- The Vicsek model [55]: 'Vicsek2pi' and 'Vicsekpi4'. In this model, the velocity of a particle at the next time step $t+1$ points in the same direction as the average velocity of its neighbors at time t, and an angular noise is added. The velocity modulus is constant and identical for every particle. In 'Vicsek2pi', the velocity direction is drawn uniformly in an interval of size 2π around the average velocity of the neighbors, i.e. is completely random. In 'Vicsekpi4', the velocity direction is drawn uniformly in an interval of size $\frac{\pi}{4}$ around the average velocity of the neighbors.

We report in appendix D the sizes and durations used in each model.

4.1.3. ADM models. The class of ADM models is an extended framework [34] of the original activity driven (AD) model [19]. In practice, an ADM model is a stochastic agent-based model in discrete time that produces a synthetic TN of interactions between agents.

We refer to [34] and appendix B for a detailed description of the models and their phenomenology, and provide here a brief reminder of their definition. In a nutshell, we consider N agents, each endowed with an intrinsic activity parameter, and who interact with each other at each discrete time step in a way depending on their activity and on the memory of past interactions between agents. This memory is encoded in another TN between the same agents, called the social bond graph: in this weighted and directed temporal graph, the weight of an edge represents the social affinity of an agent towards an other agent. At each time step, agents thus choose partners to interact with depending on their social affinity towards other agents. The affinity is then updated by the chosen interactions through a reinforcement process: social bonds between interacting agents strengthen while the social affinity weakens if two agents do not interact.

The ADM models we considered in this paper are 'ADM9conf16', 'ADM18conf16', 'min_ADM1' and 'min_ADM2'. Here the numbers '9' or '18' stand for the specific dynamical rule of the model (in [34] a large number of possible variations of ADM rules has been explored) and the suffix 'conf16' means that the parameters of the model have been tuned in order to resemble the empirical data set 'conf16'. For more detail about those models and how their parameters have been tuned, we refer to appendix B. We also report in appendix D, table D3 the sizes and durations used for each model.

4.1.4. Edge-weight (EW) models. An EW model is also a stochastic agent-based model in discrete time producing a TN. However, contrarily to the ADM or pedestrian models, here agents are not nodes but edges of this TN.

In these models, edges are independent of each other. Their probability of activation is given by the fraction of time they have been active since their last 'birth', which is defined either as the starting time of the TN, i.e. the time step 0, or as the last time the edge's history has been reset. We refer to appendix C for more details on the three variants we consider here, denoted 'min_EW1', 'min_EW2' and 'min_EW3'. See appendix D for the sizes used in the simulations of the models.

4.2. Observables

For each data set considered, we have computed the flow of 43 observables under time aggregation and shuffling and obtained the corresponding labels. Observables of interest in a TN split in *scalar* observables, which yield a single realization per data set (e.g. the average degree, or the degree assortativity of the aggregated network), and *distribution* observables, which can be sampled into a one dimensional probability distribution per data set (e.g. the contact durations).

4.2.1. Distribution observables. We consider as objects both nodes or edges and, for each, their properties listed below, which have largely been considered and analyzed in previous works:

- *duration*: number of consecutive snapshots the object is active, i.e. present in the TN;
- *interduration*: number of consecutive snapshots the object is inactive, i.e. absent from the TN;
- *event_duration* [37]: equivalent to the duration but applied to trains of the object, it is the number of consecutive packets separated by less than two timestamps (a packet is a maximal time interval over which the object is active).
- *time_weight*: total number of snapshots the object has been active over the TN duration.

For each property x, we measure its distribution over the TN considered. Since, as previously mentioned, our methodology can only handle scalar observables, we then reduce each distribution to its first modified moments $\langle x\rangle$ and $\frac{\langle x^2\rangle}{\langle x\rangle}$ [24] (higher moments could obviously be added to the list at will).

4.2.2. Motif-based observables. We also consider a set of observables based on spatio-temporal motifs called egocentric temporal neighborhood motifs (ETN, see [17]). Indeed, they have recently been shown to be useful tools to characterize TNs, and also to form building blocks able to decompose and reconstruct instances of TNs [23]. Motifs consist in sub-temporal graphs extracted on a few consecutive timestamps. This number of timestamps is called their depth, which is either 2 or 3 in our case. We consider on the one hand the ETNs defined in [17], that we call NCTN, for node-centered temporal neighborhood. In a NCTN, only the interactions between a given node and its neighbors are taken into account. Interactions between two distinct neighbors are not considered, so that the NCTNs do not contain any information on triangles. We moreover use an extension of ETN, the edge-centered temporal neighborhood (ECTN). In an ECTN, the interactions between the nodes of a given edge and the neighboring nodes are reported: thus, ECTNs can include triangles to which that edge belongs [61]. For both NCTNs and ECTNs, we compute:

- the total number of motifs in the data set *nb_tot*, including repetitions of the same motif;
- the number *nb_diff* of distinct motifs, i.e. repetitions are removed;
- the difference *motif_error* between the frequency of an observed motif and its probability predicted under the assumption of statistical independence between its parts (see appendix E), as this is a measure of correlations in the network.

Moreover, if we group motifs in isomorphism classes, we obtain a vector with *nb_diff* components, where each component is given by the number of occurrences of the associated motif.

We call this vector a motif vector (NCTN or ECTN vector). We then consider the following observables:

- *sim 'vs n'* (resp. *'vs b'*): cosine similarity between the motif vectors of $\mathrm{TN}(n, b)$ and $\mathrm{TN}(1, b)$ (resp. $\mathrm{TN}(n, 1)$);
- *sim_trunc*: same as 'sim' but we truncate each motif vector to its 20 largest components.

4.2.3. Other observables. In addition to the distribution and motif-based observables described above, we computed the flows of the average instantaneous clustering coefficient (ICC), i.e. the average of the clustering coefficient of all snapshot networks. We note that many other observables could be added to the list we considered here, such as, e.g. the average instantaneous node degree or the size of connected components. As our aim here is to provide a proof of concept of the whole procedure, we do not extend further our list of observables at this stage.

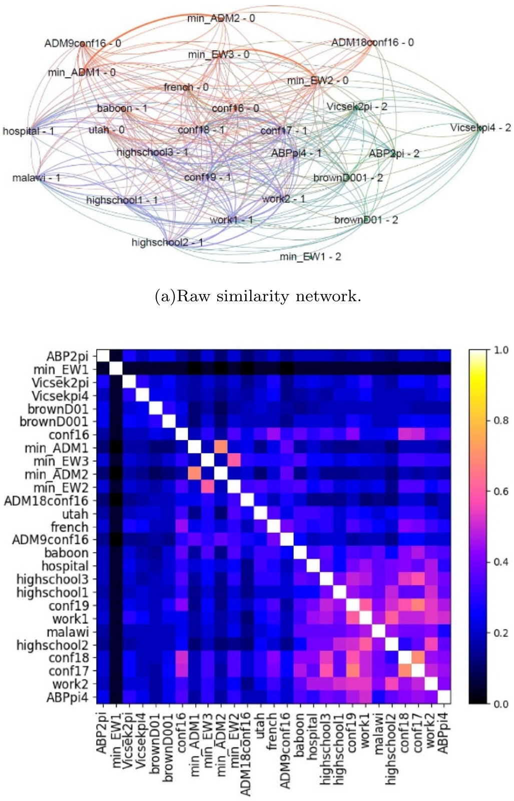
4.3. Diagrams and clusters of TNs

The goal of this subsection is to show that our labeling framework can be combined with usual tools of complex systems (clustering, similarity matrices, statistical analysis, etc) to identify clusters of TNs, describe these clusters by intrinsic properties and evaluate their proximity with each other (i.e. detect clusters of clusters). For this, we apply our framework to the 27 data sets and 43 observables described in the previous subsections, and focus on TN analysis rather than observable analysis. We note here that the removal of long inactive periods in the empirical data affects the distributions of interduration (since they produce large values of the interduration times), but only marginally the other observables, and does not affect our results. In fact, as models do not contain long inactivity periods, keeping them in the data would tend to make models and data less similar, so that removing them makes actually the task of classifying models vs. data harder.

4.3.1. Pairwise analysis. We follow here the steps described in section 3.2.1: we first evaluate the proximity of each pair of TNs using both the raw similarity and the metric similarity. For each similarity, we build a similarity matrix—or equivalently a proximity network—between TNs, on which community detection algorithms can be run.

The resulting proximity networks and similarity matrices obtained are shown in figure 5 for the raw similarity case and in figure 6 for the metric similarity. To extract groups of TNs, we apply the Louvain algorithm to the proximity networks. Note that there is a resolution parameter in the Louvain algorithm, and changing it results in different communities. We thus used the following criterion: we measure the rate of change of the partition into communities as we change the resolution parameter by computing the adjusted Rand score [63] of similarity between partitions obtained at two consecutive values of increasing resolution. Then we selected the resolution maximizing the adjusted Rand score. If a plateau was observed, we chose the resolution maximizing the modularity.

From figure 5, we see that considering raw diagrams allows to separate between ADM and pedestrian models, while most empirical networks form a third class. The 'conf16' empirical data is classified with the ADM models, which is probably due to the fact that the ADM models have been tuned in order to resemble the 'conf16' data set. A different structure is obtained by taking the metric similarity, as shown in figure 6: we then obtain two clusters corresponding



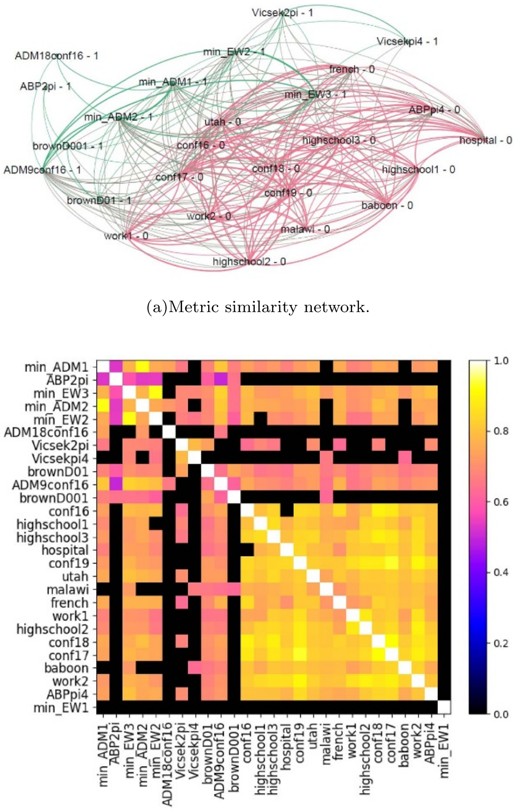
(a)Raw similarity network.

(b)Raw similarity matrix.

Figure 5. Pairwise probability to share the same label, sampled over all observables. (a) We used Gephi [62] to visualize the graph and ran a community detection algorithm based on the Louvain method. Edge thickness is proportional to the edge weight, which is the probability of sharing the exact same label. For a better readibility, only the strongest edges are visible (75.50 % of all edges). Three communities are returned by the Louvain algorithm: the community 0 in orange, the community 1 in purple and the community 2 in green. The community to which nodes belong is written at the right of their labels. Intra-community links are of the same color as the community while inter-community links are plotted in gray. The orange community (0) contains the ADM models, a few empirical data sets and the models min_EW2 and min_EW3. The purple community (1) contains exclusively empirical data sets and the model ABPpi4. The green community (2) contains pedestrian models and the model min_EW1. (b) Similarity matrix, from which some large similarity cases can be readily seen, such as the group of conference data, the two min_ADM models or min_EW2 and min_EW3.

respectively to the synthetic and empirical TNs, with one exception as the model ABPpi4 belongs to the same cluster as empirical data sets.

We also note that the model 'ADM9conf16', put forward in [34], reproduces well the empirical distributions of many observables (distributions of contact and intercontact duration, ETN motifs, etc) of the 'conf16' data set. As seen in figure 6(b), our approach indeed finds it closer to the empirical data sets than the model 'ADM18conf16', which had a lower performance (see [34]) at reproducing the empirical properties. However, 'ADM9conf16' still does not belong to the same cluster as the empirical data sets: this illustrates how the mere reproduction of



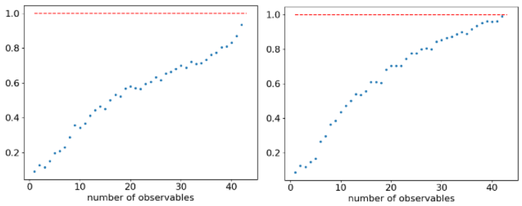
(a)Metric similarity network.

(b)Metric similarity matrix.

Figure 6. Geometric average over observables of the edit similarity between labels. (a) As in figure 5, we used Gephi to visualize the graph and ran the same community detection algorithm. The min_EW1 model is missing, because it has zero similarity with every other TN. Two communities of comparable size are detected, respectively populated with empirical data sets (0, orange) and models (1, green). (b) The two detected communities are visible on the similarity matrix. Note that these communities do overlap, which is expected since most of the models considered here aim at reproducing some properties of the empirical data sets.

statistical distributions of observables of one empirical TN does not ensure that the label of the TN will also be shared. Although the labeling procedure entails only limited information (since we use only information about qualitative shapes of the flows), the fact that it contains information about all scales at once seems to make it more informative about a TN than a collection of distributions sampled at a single resolution level. Said otherwise, there seems to be more information in the way an observable changes under the aggregation and shuffling transformations, than in the precise realizations of this observable.

4.3.2. Sensitivity analysis. The clusters of TNs we obtained above depend *a priori* on the set of observables we choose to compute the similarity between networks. As the framework is flexible and can be extended to an arbitrary large number of observables, one can hope that the partition into clusters becomes stable as the number and diversity of observables considered



(a)raw similarity case (b)metric similarity case

Figure 7. Stability of communities with respect to the number of considered observables. The adjusted Rand score (displayed on the y-axis) increases with the number of considered observables (x-axis), meaning that the detected communities become increasingly stable as more observables are used. Values close to 1 are obtained for both the raw (a) and metric (b) cases.

becomes large enough. To check this, we computed the evolution of the structure of communities yielded by the raw and metric similarities as the number of observables considered increased from 1 to 43. More precisely, for each $1 \leqslant s \leqslant 43$, we draw s observables at random among the 43 available, and compute the similarity network and the resulting community structure. We then compute the adjusted Rand score R_s between the partition in communities obtained with s observables and the one obtained with $s+1$ observables. This measure indeed quantifies the similarity between two partitions of the same set of elements (here the TNs), and we average its value over 100 realizations of the random choice of observables.

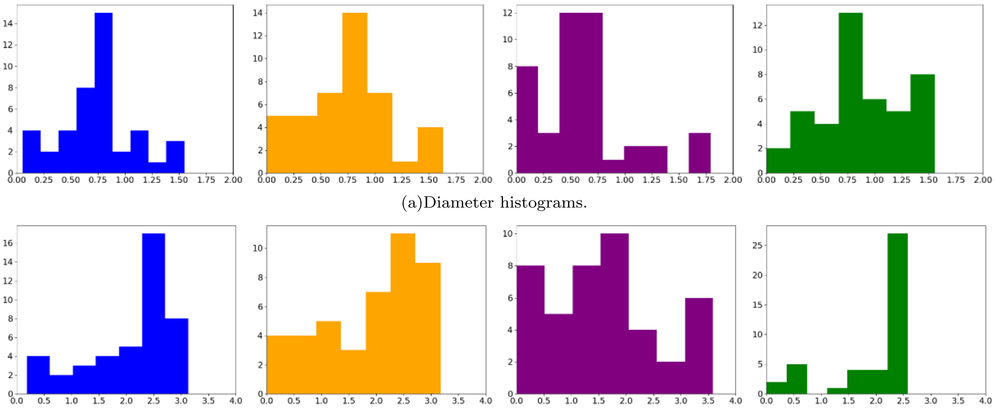
Figure 7 shows that R_s increases with s, i.e. the structure in communities becomes more stable as more observables are added. A larger stability is obtained with the metric similarity than with the raw one, and very large values of the Rand index are obtained when more than 30 observables are considered. To reach a perfect stability, some additional observables might need to be considered, but the results of figure 7 indicates that the clusters identified in figures 5 and 6 are already very stable with respect to the choice of observables.

4.3.3. Analysis of classes. We consider three classes of TNs, corresponding to the three detected communities on figure 5:

- class 0 (orange, 9 elements): ‘min_EW2’, ‘min_EW3’, ‘ADM9conf16’, ‘ADM18conf16’, ‘min_ADM1’, ‘min_ADM2’, ‘conf16’, ‘french’, ‘utah’;
- class 1 (purple, 12 elements): ‘ABPpi4’, ‘highschool1’, ‘highschool2’, ‘highschool3’, ‘conf17’, ‘conf18’, ‘conf19’, ‘work1’, ‘work2’, ‘malawi’, ‘baboon’, ‘hospital’;
- class 2 (green, 6 elements): ‘min_EW1’, ‘Vicsekpi4’, Vicsek2pi, brownD01, brownD001, ABP2pi.

We compute the diameter and heterogeneity of each class (see section 3.2.2 for definitions) for each observable, and display in figure 8 the histograms of these quantities sampled over observables. On the same figure, we also display the histograms obtained for a group of 6 TNs taken at random among the 27 available, averaging over 200 independent realizations of this random class.

Figure 8 shows that each class is characterized by both different diameter and heterogeneity distributions. The class 1, mostly composed of empirical data sets, tends to have lower diameters, indicating that labels of empirical data sets are close to each other for most observables.



(a)Diameter histograms.

(b)Heterogeneity histograms.

Figure 8. Histograms of heterogeneity and diameter. For each class, the heterogeneity (a) and diameter (b) are computed from the repartition function and definitions given in section 3.2.2. The *y*-axis indicates the number of occurrences of the corresponding value reported on the *x*-axis. The colors indicate the considered class following the coloring of figure 5(a): orange for the class 0 (mostly ADM models), purple for the class 1 (mostly empirical data) and green for the class 2 (mostly pedestrian models). Histograms in blue are obtained from the average over 200 independent choices of a group of 6 temporal networks at random among the 27 available.

The heterogeneity distribution indicates how probable it is to pick a specific or a universal observable with respect to the class considered when choosing an observable at random. If the class is built randomly (blue histograms), observables tend to be specific (higher values of heterogeneity more represented than small values), i.e. have different labels for different TNs. Class 1 (purple) shows a clearly different behavior, with more observables having a low heterogeneity (being more universal within the class). This reflects the fact that empirical data sets tend to share many statistical properties.

On the other hand, the class of pedestrian models (class 2, green) has almost only specific observables among the ones we considered. This is in accordance with the similarity matrix of figure 5(b), which shows that models from this class do not share the same labels. This might be due to two reasons: (1) The observables' flows obtained for observables of the TNs created by pedestrian models are mainly flat (to the human eye) but actually noisy; the automatic attribution of labels by the neural network we built might then fail to identify the flat behavior and instead assign complex labels to these flows. For example, it may assign something like $-|+-+|0$ to a flow instead of 0. Thus, random complex labels may be assigned to the pedestrian models, resulting in different labels for almost every TN. We have checked by hand that this was not the case. (2) The pedestrian models do not form a well-defined class because their fundamental mechanisms differ from one model to the other. Moreover, changing parameter values can strongly affect the model's behavior. For instance, Vicseckpi4 and Vicseck2pi are in two distinct parts in the phase space of the Vicseck model [55]. Interestingly, our labeling procedure seems thus to be able to distinguish between realizations of a given model corresponding to different phases and thus to have the potential to detect phase transitions undergone by a model. Note also that the situation is different for the other models we consider, such as e.g. min_EW2 and min_EW3, or min_ADM2 and min_ADM3: here changing the values of the models' parameters does not affect strongly the model behavior, and we indeed obtain very close labels.

4.3.4. Co-occupancy of classes. The three classes of TNs have different diameter and heterogeneity distributions. Do these differences reflect in non-overlapping labels? To answer this question, we compute the overlap between our three classes, as defined in section 3.2.3: recall that the overlap similarity between two classes measures the probability that two members drawn at random from these classes share the same label. We obtain the following values:

- $P(C_0 \cap C_1) \simeq 0.24$
- $P(C_0 \cap C_2) \simeq 0.15$
- $P(C_1 \cap C_2) \simeq 0.18$

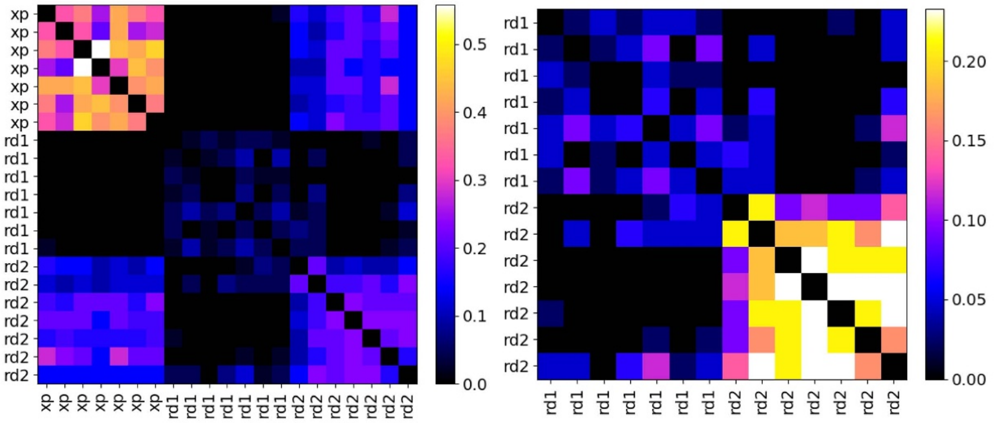
In particular, we see that the ADM class (class 0) has a stronger overlap with the class of empirical data (class 1) than the class of pedestrian models (class 2), and the pedestrian and ADM models almost do not overlap with each other, indicating that their statistical properties fundamentally differ from each other. This qualitative difference may be due to the role of space: in pedestrian models, agents are constrained by a 2D motion while in ADM models, the social network alone is considered.

5. Conclusion

In this article, we have proposed a systematic procedure to associate discrete labels to TNs in order to provide a way to compare instances or whole classes of TNs. This can help not only to validate models but also to assess the heterogeneity of TNs within a class of models or within an ensemble of empirical data sets.

To this aim, we have considered how observables evolve when the TN data is transformed under a reshuffling at a certain scale, leading to a partial removal of short-time correlations in the TN, followed by a temporal aggregation at another scale. For simplicity, we have defined labels as describing the successive trends of the flows of observables under these transformations, without encoding their specific values: networks with very different values of an observable and possibly different range of variations of this observable, but with the same increasing or decreasing successive trends, are then encoded with the same label. Although these labels thus encode only qualitative information, they make it possible to define a metric to compare TNs, and thus to evaluate a model performance, identify clusters of TNs and characterize these clusters by new properties (diameter, heterogeneity, etc). Our hypothesis that important information is contained in the succession of increasing and decreasing trends of the flows is *a posteriori* justified by our results. We have for instance shown how the procedure is able to separate empirical data sets from synthetic data created by models, even if some of these models have been tuned to reproduce several properties of the data. This also highlights how current models need to be improved to take into account non-trivial temporal correlations. As a further illustration of the method, we show in figure 9 that the method can separate empirical data sets from their reshuffled versions, and even separate among the reshuffling methods. To quantitatively confirm this and explore further the ability of the method to distinguish between types of reshuffling, we would need (1) to test our method with a larger set of reshuffling methods [57] and (2) to improve the performance of our neural network. Indeed, the current version of the neural network currently still struggles to assign flat labels ‘0’ to noisy flows that look flat to the human eye. One consequence of this limitation is that the stronger the randomization, the weaker the similarity, as seen in figure 9.

The framework we have put forward is flexible and could be extended at will. For instance, one could consider other reshuffling procedures [57] to create the flows of TNs and observables. It would also be interesting to enrich the qualitative information kept on the trends with



(a)similarity between empirical and randomized data sets

(b)similarity between randomized data sets

Figure 9. Raw similarity matrices between empirical data sets and their randomized versions. For a better readability, the matrix diagonal has been filled with zeros and we considered only one empirical data set of each type: conference, village, groups of baboons, highschool, primary school, hospital and workplace. Moreover, data set have been renamed to three categories: (1) 'xp' for any empirical data set (2) 'rd1' (resp. 'rd2') for any data set randomized according to the method 1 (resp. 2), which we describe now. The randomization 1 preserves only the fully aggregated network: it consists, for each edge, in re-drawing at random its times of activation within the network duration. The randomization 2 preserves both the fully aggregated network and the number of interactions at each time step. (a) We see that empirical data sets and their randomized versions are well separated. (b) The randomized versions of the empirical data sets considered are also separated according to their type, or degree (recall 'rd1' is stronger than 'rd2'). Overall, we note that the stronger the randomization, the weaker the similarity between data sets. This is because the neural network assigning labels to flows is not perfect: it tends to denote flat but noisy flows by random lengthy labels (like e.g. '$- + -$' instead of '0'). As we observe that flows are flattened by a strong randomization, it follows that their labels are random-like, making them diverse and thus different from each other.

partial quantitative information on the values of the observables, or on the amount of their relative variations on each increasing/decreasing part, and investigate whether this enhances the performance of our procedures. We have also here considered a certain list of observables, chosen in part arbitrarily. Depending on the context of the TNs considered, a different set of observables could be used. The set of observables could also be extended, for instance including higher moments of the distributions [24], temporal clustering coefficient [11], or other properties of network snapshots such as average degree, assortativity or sizes of connected components. The similarity metrics could also be defined using a non-uniform average over observables if some are deemed more important than others in specific contexts.

Our work suggest some avenues for future work. First, the performance of the neural network we used to assign a label to a curve could probably be improved. Second, the study could be extended to a larger and more diversified set of empirical and TNs, and potentially therefore also to more observables. As briefly mentioned, the procedure can also give rise to a study focused on the observables themselves: What are the possible or most probable labels for a given observable? Which observables yield the same or similar diagrams? Finally, labels in themselves could be studied in more detail, revealing distinct and well-defined properties for TNs and observables (universality, specificity, etc). This would also raise many theoretical questions about the mechanisms behind the rich phenomenology of the flows: indeed, even seemingly simple models like the min_ADM models of this paper can exhibit large labels if an appropriate choice of their parameters is made.

In conclusion, the study of the flows of TNs and attached observables under partial reshuffling and aggregation is a promising tool for the study of TNs, which contains valuable information to compare, cluster and characterize TNs.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: www.sociopatterns.org/datasets/.

Acknowledgments

We thank Razieh Masoumi for providing us with the synthetic data sets of the Pedestrian models. This work was supported by the Agence Nationale de la Recherche (ANR) Project DATAREDUX (ANR-19-CE46-0008).

Appendix A. Disjoint and sliding time shuffling (STS)

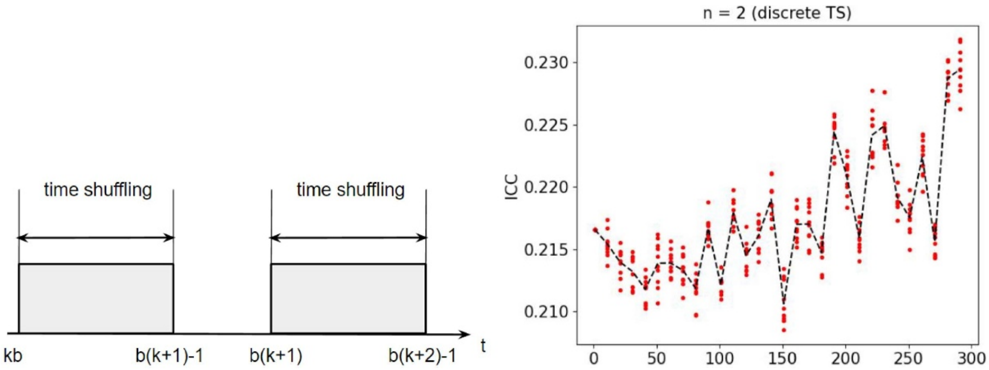
A.1. Noise removal by STS

As explained in the main text, we want to compute the sign sequence of the derivative of a given curve (given in our specific case by the evolution of an observable $\mathcal{O}$ under shuffling or aggregation when the corresponding time scale b or n is varied). As we are dealing with noisy data or random realizations of models, each curve will contain a certain amount of noise. To obtain the sign sequence automatically, we need thus first to reduce the noise level as much as possible. Therefore, we average each value $\mathcal{O}(n,b)$ over ten realizations of the local time shuffling with range b. If we use a time shuffling on successive, disjoint time windows (see figure A1(a) for illustration), it can however happen that the noise of the curve is of greater amplitude than the noise originating from the time shuffling. We show an example of such a noisy flow in figure A1(b).

These oscillations are *a priori* surprising because the only source of randomness is precisely the local time shuffling. It means that this additional noise is not random but might contain information. However, this information is too specific to the TN. Indeed, this pseudo-noise originates from the combination of (1) the split of the TN timeline in disjoint blocks and (2) the temporal fluctuations of the edge activity of a TN, i.e. the number of active ties at a given timestamp. To see this, consider the toy situation: of a temporal network (TN) with timeline for the edge activity $E(t)$ (which denotes the number of active ties at time $t \geqslant 0$):

$$E(t) = E_0 + \Delta\theta(t - t_0)$$

(θ denotes the Heaviside function). Then if b divides t_0, snapshots with different levels edge activities are not mixed by the reshuffling at scale b. However, if b is not such a divider, snapshots from different states do mix, which results in a different realization for most observables. This in turn causes spurious oscillations in the observables' flows. To avoid these oscillations, we thus consider a sliding window for the shuffling procedure, as described in the main text, instead of disjoint windows. More precisely, the transformation we propose, that we call a STS (figure 1), consists in shuffling the snapshots in the block $[\![t, t+b-1]\!]$ when t goes iteratively



(a)disjoint time shuffling (DTS) (b)example of a noisy flow under DTS

Figure A1. Illustration of disjoint time shuffling and a noisy flow. (a) To perform a disjoint time shuffling, the timeline of the temporal network is split in successive disjoint blocs of length b, where $b \in \mathbf{N}$ is the parameter of the transformation. Then a time shuffling is performed within each bloc. This removes any correlation on time scales shorter than b but preserves correlations on time scales larger than b. (b) Flow of the instantaneous clustering coefficient ('ICC') under DTS for the aggregation level $n = 2$ and the TN 'conf16'. The red dots stand for different realizations of each time shuffling while the black dashed line is the average of these realizations. The dispersion of the red dots appears to be smaller than the dashed line discontinuities, indicating a source of randomness that is present in the TN itself.

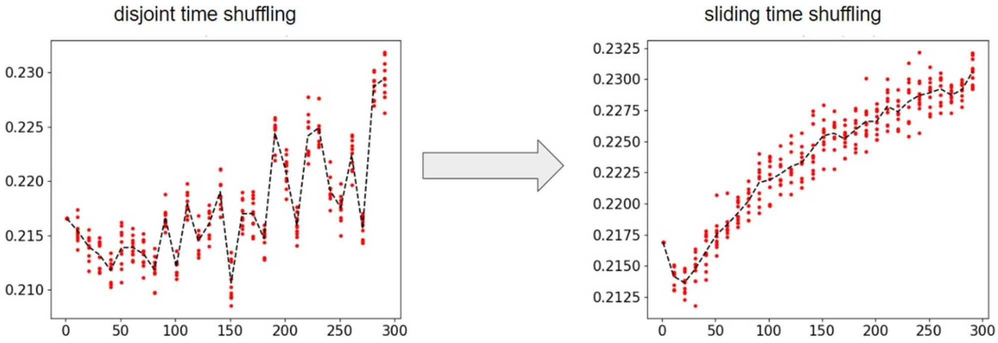


Figure A2. Noise reduction by sliding time shuffling. We consider the observable ICC at aggregation level 2, like in figure A1(b). We see that using STS instead of DTS removes most of the spurious oscillations in the ICC flow versus b. Indeed, the fluctuations of the black line are now fully explainable by the dispersion of the red dots.

from 0 to $T - b$, T being the total number of snapshots. Figure A2 shows that the STS indeed results in a smoothing of the observables' flows.

A.2. Range of STS

Note that the STS is not strictly local anymore, since there is a non-zero probability that snapshots distant by more than b get exchanged. Thus, we need to check how often it occurs, otherwise what is the range of the STS compared to the disjoint time shuffling? To answer this, let us compute the average distance travelled by a snapshot as we apply successive shufflings.

To avoid boundary effects, let us consider a snapshot initially at a position $t_0 \geqslant b$. Let us denote by t_k the position of this snapshot after k successive shufflings have been performed. As initial shufflings do not matter, we initialize the shuffling window at the time step $g_0 = t_0 - b + 1$. We denote the boundaries of the shuffling window by g_k, d_k. We have:

$$\begin{cases} g_k = t_0 - b + 1 + k \\ d_k = t_0 + k \end{cases}$$

so that either $t_{k+1} = t_k$, or t_{k+1} is drawn uniformly in $[\![g_k, d_k]\!]$. The sequence $(t_k)_{k \geqslant 1}$ is a stochastic process, which almost surely will become stationary, i.e. above a given rank k, we have $t_{k+1} = t_k$. Let us define

$$K = \min\left\{k | n \geqslant k \implies t_{n+1} = t_n\right\}.$$

Since t_k is drawn in $[\![g_{k-1}, d_{k-1}]\!]$, the only possibility which ensures $(t_k)_{k \geqslant K}$ to be stationary is that $t_K = g_{K-1}$. Here it writes

$$t_K - t_0 = K - b.$$

To determine how much the snapshot has been displaced by the STS, we want to determine the law of $X = |t_K - t_0|$. This law can be derived from the law of K, i.e. the number of shufflings to perform before the snapshot finds its final position. We have the following:

$$\begin{aligned} P(K = 1) &= \frac{1}{b} \\ P(K = 2) &= \frac{1}{b}\left(1 - \frac{1}{b}\right) \\ \forall k \geqslant 1, P(K = k) &= \frac{1}{b}\left(1 - \frac{1}{b}\right)^{k-1}. \end{aligned}$$

We deduce the law of X:

$$\begin{cases} \forall 1 \leqslant x \leqslant b - 1, P(X = x) = b^{-1} q^{b-1}\left(q^x + q^{-x}\right) \\ \forall x \geqslant b \vee x = 0, P(X = x) = b^{-1} q^{b-1} q^x \end{cases}$$

where we introduced $q = 1 - b^{-1}$ for conciseness.

Note that the map $x \mapsto q^x + q^{-x}$ is strictly increasing for $0 < q < 1$ so the most probable value for X is $b - 1$. Let us compute $\langle X \rangle$. To do this, we can first compute the generating function of X:

$$f(t) = \left\langle t^X \right\rangle = \sum_{x=0}^{\infty} t^x P(X = x) = b^{-1} q^{b-1}\left[1 + \sum_{x=1}^{b-1} t^x\left(q^x + q^{-x}\right) + \sum_{x \geqslant b} (tq)^x\right]$$

$$\begin{aligned} f(t)\, b q^{-b+1} &= 1 + \frac{tq}{1 - tq}\left(1 - (tq)^{b-1}\right) + q^{-b+1}\frac{t}{q - t}\left(q^{b-1} - t^{b-1}\right) + \frac{(tq)^b}{1 - tq} \\ &= \frac{1}{1 - tq} + q^{-b+1}\frac{t}{q - t}\left(q^{b-1} - t^{b-1}\right). \end{aligned}$$

As we have $\langle X\rangle = \frac{\mathrm{d}f}{\mathrm{d}t}(t=1)$, we deduce (recall that $q = 1-b^{-1}$):

$$\begin{aligned}\langle X\rangle &= b^{-1}q^{b-1}q(1-q)^{-2} + b^{-1}\left[\left(q^{b-1}-1\right)q(1-q)^{-2} + (b-1)(1-q)^{-1}\right]\\ &= 2q^b b - qb + b - 1 = 2q^b b\\ \rightarrow \langle X\rangle &= 2b\left(1-b^{-1}\right)^b \leqslant 2b\exp(-1)\end{aligned}$$

Thus, a STS with a window of size b will displace a snapshot from its initial position by an average distance less than (and approximately equal to when b becomes large) $2b\exp(-1) \simeq 0.7b$. For comparison, for a disjoint time shuffling, the average travelling distance is equal to:

$$\begin{aligned}\langle X\rangle_{\mathrm{DTS}} &= \frac{1}{b^2}\sum_{t=0}^{b-1}\sum_{t_0=0}^{b-1}|t-t_0| = \frac{2}{b^2}\sum_{t_0=0}^{b-2}\sum_{t=t_0+1}^{b-1}(t-t_0) = \frac{2}{b^2}\sum_{t_0=0}^{b-2}\binom{b-t_0}{2}\\ &= \frac{2}{b^2}\sum_{t_0=0}^{b-2}\left[\binom{b-t_0+1}{3} - \binom{b-t_0}{3}\right] = \frac{2}{b^2}\binom{b+1}{3}\\ \rightarrow \langle X\rangle_{\mathrm{DTS}} &= \frac{1}{3}\left(b-b^{-1}\right) \simeq 0.3b\end{aligned}$$

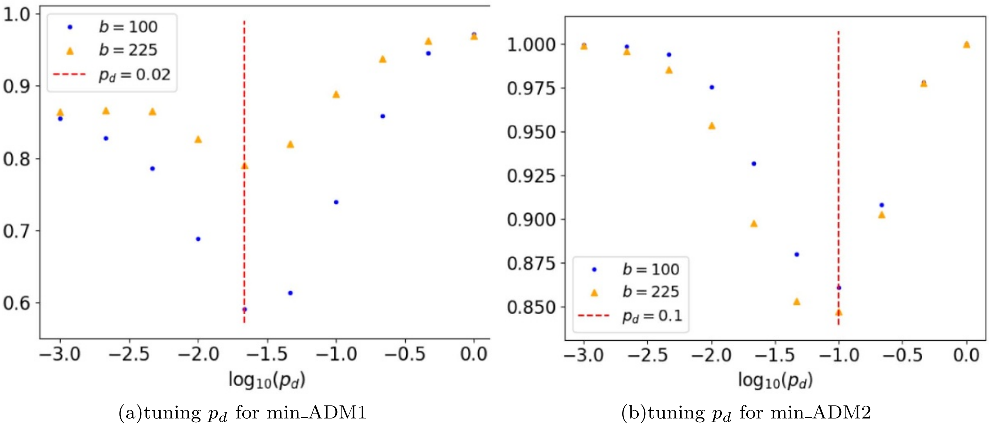
Thus the displacement induced by a STS is approximately twice the one induced by a disjoint time shuffling, when the same parameter b is considered for both shufflings.

Appendix B. ADM models

The ADM models are agent-based models in which a fixed population of N agents interact following specific rules at each time step. Specifying an instance of the ADM class comes in two steps: (1) we choose the dynamics for the agents, i.e. the rules for their behavior; (2) we choose values for the parameters associated to these rules. For example, if we choose that agents can lose memory of all their past interactions at each time step (rule), we have to precise what is the probability of such an event (parameter).

Once dynamical rules for the agents have been set, two options are available to specify values for the parameters. Instead of entering them arbitrarily, it is indeed possible to use a genetic algorithm to tune them automatically, so that our model produces a TN as similar as possible to a specified target, with respect to a set of observables. This is the procedure put forward in [34].

The ADM models we considered in this paper are called ‘ADM9conf16’, ‘ADM18conf16’, ‘min_ADM1’ and ‘min_ADM2’. The numbers ‘9’ and ‘18’ stand for two different dynamical rules (a broad variety of rules has been studied in [34]) and the suffix ‘conf16’ means that the parameters of have been obtained by genetic tuning in order to resemble the empirical data set ‘conf16’. On the other hand, most parameters of ‘min_ADM1’ and ‘min_ADM2’ have been chosen by hand (given in table B1). Only one parameter has been tuned, but without using any genetic algorithm. This parameter is denoted by p_d and corresponds to the probability for an agent to clear its memory of past interactions at each time step. For both models ‘min_ADM1’



(a)tuning p_d for min_ADM1 (b)tuning p_d for min_ADM2

Figure B3. Choice of the probability p_d of deleting memory for the models min_ADM1 and min_ADM2. We plot the ETN similarity between TN(n, b) and TN$(n, 1)$ for large values of b as a function of p_d. The aggregation level n differs in the two models. In each case, a minimum similarity is achieved for an intermediate value of p_d, corresponding to a maximum sensitivity under STS. (a) min_ADM1: $n = 10$ and we obtain $p_d = 0.02$; (b) min_ADM2: $n = 1$ and we obtain $p_d = 0.1$.

and ‘min_ADM2’, we choose the value of p_d to maximize the sensitivity to STS at a certain, arbitrarily chosen level of aggregation. Said otherwise, we wanted models that are not invariant under STS. Indeed, being invariant under STS means having no time correlations, which is not an interesting situation for a TN. We thus quantified, for a given aggregation level n, the degree of invariance of a TN under STS as the cosine similarity between $\mathcal{O}(n,b)$ and $\mathcal{O}(n,1)$ in the limit $b \gg 1$, using as observable $\mathcal{O}$ is a vector indexed by spatio-temporal motifs (its component $\mathcal{O}_i$ is the number of occurrences of the spatio-temporal motif i), namely the Egocentric Temporal Neighborhoods of depth 3 (ETN, see [17]), written as ‘3-NCTN’ in our nomenclature for observables, see figure B3.

Let us now describe the rules for the agents’ dynamics in the four ADM models considered here. We also refer to [34] for more details and results on these models. Each agent i is endowed with an activity parameter a_i that will determine its level of activity, taken from a certain distribution, or set to the same value for all agents. We denote by G the TN of their interaction, called the *interaction graph*. In the four models, the N agents keep a long-term memory of their past interactions, encoded in another TN, called the *social bond graph* and denoted by B. $B_{i,j}(t)$ is the directed strength of the social affinity of the agent i towards the agent j. At the initial time, the N agents are initialized with an empty social bond graph.

Before going into more details, let us precise that the models ‘ADM18conf16’, ‘min_ADM1’ and ‘min_ADM2’ share similar dynamics, while the model ‘ADM9conf16’ is more complex. Thus we will describe first the simpler models and then the model ‘ADM9conf16’.

B.1. Models ‘ADM18conf16’, ‘min_ADM1’ and ‘min_ADM2’

At time t, the snapshot $G(t)$ is built as follows:

1. We determine which agents will be ‘active’, i.e. able to propose an interaction to other agents. Every agent has the same probability to be active, denoted by a.

2. Active agents try to interact with a given number of other agents. This number is denoted by m and is the same for every agent.
3. For each proposal, the active agent will either pick a partner among its memorized contacts (probability $1-p_g$) or will grow its egonet, i.e. will pick an unknown partner (probability p_g).
4. Two options:
   (a) To grow its egonet, two options are available to the agent: with probability p_u it will pick a partner at random among the agents it does not know. Otherwise (so with probability $1-p_u$), it will grow his egonet by triadic closure.
   (b) If the agent does not grow its egonet, it picks a partner among the members of its egonet. The probability that the agent i picks the agent j is proportional to the strength of the social affinity of i towards j:

$$P(i \to j, t) = \frac{B_{i,j}(t)}{\sum_{k=1}^{N} B_{i,k}(t)}$$

   $G(t)$ is given by the interactions determined by the previous steps, and the social bond interaction is then updated:
5. $B(t)$ is updated through a linear Hebbian process, meaning that for each edge (i,j) in $G(t)$:

$$B_{i,j}(t+1) = B_{i,j}(t) + 1.$$

6. $B(t+1)$ is pruned, taking possible loss of agents' memory into account. In the four models considered, this pruning consists in removing all directed bonds starting from i with probability p_d for each agent i. Once the pruning has been done, the previous steps are repeated for step $t+1$ instead of t.

The only difference in rules between these three models is that triadic closure is allowed in 'ADM18conf16', while in the two others, agents always grow their egonets by picking an unknown agent at random, i.e. $p_u = 1$. Parameters of each model are given in table B1.

B.2. Model 'ADM9conf16'

This model differs from the ADM previously described on the following points:

- the construction of $G(t)$
- the Hebbian process updating B
- the pruning process updating B
- the agent activity a_i and the parameter m_i (number of interactions emitted per agent) are agent-dependent. a_i is drawn from a power-law of exponent -1 bounded between $a^{\min}$ and $a^{\max}$. m_i is drawn from a uniform law on integers between 1 and $m^{\max}$. These three parameters $a^{\min}$, $a^{\max}$ and $m^{\max}$ are subjected to the genetic tuning evoked in the previous sub-section.

Table B1. Parameter values of the four ADM models considered in this paper. The number of agents and the TN duration have been taken equal to the case of the empirical data set 'conf16'. For the models 'ADM9conf16' and 'ADM18conf16', the other parameters have been obtained by a genetic algorithm to match as closely as possible the TN 'conf16'. For the two other models, all parameters except the rate of memory loss p_d have been set arbitrarily by hand. p_d has been chosen according to the procedure described in figure B3. Contrary to the three other models, the a and m parameters are not the same for every agent in the model 'ADM9conf16'. The agent activity a_i is drawn for each agent i from a power-law of exponent -1 and bounded in the interval given in the table. The parameter m_i is drawn for each agent i from a uniform law on the set $\{1,2\}$.

Parameter notation	Function	Value in 'ADM18conf16'	in 'min_ADM1'	in 'min_ADM2'	in 'ADM9conf16'
a	probability of interaction proposal	0.2	0.3	0.3	$\in [0.03, 0.91]$
m	number of emitted interactions	1	1	1	$\in [\![1,2]\!]$
p_g	rate of egonet growth	0.004	0.085	0.085	0.102
$1-p_u$	probability of triadic closure	0.3	0	0	0.46
p_d	rate of agents' memory loss	0.02	0.02	0.1	—
α	evolution rate of social ties	—	—	—	0.115
p_c	dynamic triadic closure	—	—	—	0.009
λ	rate of edge pruning	—	—	—	0.225

B.2.1. Construction of G(t). Given a time step t, the network of interactions at that time $G(t)$ is built with two differences with respect to the ADM models from the previous sub-section:

- taking into account a *social context*: $G(t)$ is built both on the basis on $B(t)$ and $G(t-1)$. More precisely, the social affinity of i towards j is modulated by the number $c_{i,j}$ of their common partners at previous time:

$$B_{i,j}(t) \to (1 + c_{i,j}(t-1))\, B_{i,j}(t)$$

- taking into account *contextual interactions*: we distinguish between *intentional* and *contextual* interactions. Intentional interactions are 'chosen' by the agents; for the agent i, they consist in the m_i interactions it emits. Contextual interactions are consequences of the fact that social interactions tend to be transitive. Hence, if i is talking to j who is talking to k, it is likely that i and k will also talk to each other. Thus contextual interactions can be viewed as a dynamic triadic closure mechanism. In our model, it is implemented as follows. Once the intentional interactions of every agent have been added in $G(t)$, we close every open triangle with some probability proportional to a new parameter p_c, which is subjected to genetic tuning [34].

B.2.2. Update of the social bond graph B. Three points must be considered.

First, we distinguish between intentional and contextual interactions. In the 'ADM9conf16' model, we consider the latter as *neutral*. This means that contextual interactions do not lead to any change in the social ties: they do not take part in the update of the social bond graph.

Second, social ties are reinforced through an exponential Hebbian process, instead of the linear process described before. One important conceptual difference between the two processes is their sensitivity to the time ordering of past interactions: only the number of past interactions matters in the linear Hebbian process, but the temporal order in which these interactions occur matters in the exponential Hebbian process. This process is described in detail in [64].

Third, the pruning of B consists in removing edges instead of nodes. This pruning is not uniform: the stronger a social tie is, the less likely it is to be removed. Moreover, ties that have been reinforced during the Hebbian process at time t cannot be removed during the pruning at time t. The probability of removing the directed edge (i,j) from $B(t)$ is given by (see [34] for more details)

$$P_d(ij) = \exp\left(-\lambda d_i^{\text{out}} P(i \to j)\right) ,$$

where d_i^{out} is the number of out-neighbors of i in $B(t)$ and $P(i \to j)$ is the probability that i chooses to interact with j at time t (see previous sub-section). λ is a model parameter submitted to genetic tuning.

In table B1, we give the values of each parameter of the four ADM models.

Appendix C. Edge-weight (EW) models

We introduced the EW models to have a baseline. Contrary to the ADM class, this class of models does not aim at reproducing empirical data. They aim instead at answering the question: How close from the empirical case are the flows of the almost simplest models we can build?

In the EW models, the number of nodes is fixed in time and denoted by N like in the ADM models. There is also a social bond graph B, indicating the strength of the social affinity between nodes. Let us consider a time step t and describe how the interaction graph $G(t)$ is built.

1. Each edge of non-zero weight in the social bond graph is activated at step t with some probability. To be activated at step t means for an edge to belong to $G(t)$. The activation probability is proportional to the edge weight, but the precise expression depends on the chosen rules for the TN dynamics:
   (a) rule 'no shift': the probability that the edge (i,j) is activated writes $\frac{B_{i,j}(t)}{t+1}$;
   (b) rule 'with shift': the probability of activation writes $\frac{B_{i,j}(t)}{t+1-t_b(i,j)}$ where $t_b(i,j)$ is the time of last birth of the edge (i,j). The birth of an edge is defined by its entering into the social bond graph, i.e. the time at which $B_{i,j}$ becomes non-zero.

Table C2. Parameter values of the three EW models considered in this paper. Just like in the ADM models, the number of nodes and the TN duration were taken as equal to the case of the empirical data set 'conf16'. The value of p_d has been chosen by hand, so that the number of temporal edges in the models match approximately the number of temporal edges in 'conf16'.

Rule name	Associated parameter	Value in min_EW1	in min_EW2	in min_EW3
shift	t_b	0	last birth time	last birth time
birth rate	n_{new}	2	2	2
pruning	p_d	no pruning	node pruning, $p_d = 0.01$	edge pruning, $p_d = 0.02$

2. A constant number of edges selected at random among the edges of weight zero are activated.
3. Weights of active edges are updated according to

$$B_{i,j}(t+1) = B_{i,j}(t) + 1 \ .$$

4. Before moving to step $t+1$, some edges are removed from the social bond graph, meaning their weights are set to zero. In the EW class, two pruning processes are available:
   (a) a node pruning, meaning that each node i has the probability p_d to be removed from $B(t+1)$; in practice, it means that all its edges are removed and it is replaced by an identical but isolated node;
   (b) an edge pruning, meaning that each edge (i,j) has the probability p_d to be removed from $B(t+1)$.

Parameter values as well as the TN dynamics are described in table C2 for the three models considered in this paper: min_EW1, min_EW2 and min_EW3.

Appendix D. Sizes of the considered data sets

Table D3. Sizes of the 27 data sets considered in this paper.

Data set name	Number of nodes	Duration	Number of temporal edges
'conf16'	138	3 635	153 371
'conf17'	274	7 250	229 536
'conf18'	164	3 519	96 362
'conf19'	172	7 313	132 949
'utah'	630	1 250	353 708
'french'	242	3 100	125 773
'highschool1'	126	5 609	28 560
'highschool2'	180	11 273	45 047
'highschool3'	327	7 375	188 508
'hospital'	75	9 453	32 424
'malawi'	86	43 438	102 293
'baboon'	13	40 846	63 095
'work1'	92	7 104	9 827
'work2'	217	18 488	78 249
'brownD01'	1 000	4 000	635 266
'brownD001'	988	4 000	661 878
'ABP2pi'	201	4 000	17 698
'ABPpi4'	1 000	4 000	156 173
'Vicsek2pi'	500	4 000	38 137
'Vicsekpi4'	500	4 000	41 841
'ADM9conf16'	138	3 635	187 774
'ADM18conf16'	138	3 635	86 793
'min_ADM1'	138	3 635	154 628
'min_ADM2'	138	3 635	151 078
'min_EW1'	138	3 635	57 376
'min_EW2'	138	3 635	193 203
'min_EW3'	138	3 635	185 901

Appendix E. The motif_error observable

Given a TN, the motif_error consists in comparing the observed and predicted probability of occurrence of motifs. The observed probability of a motif is the ratio between its number of occurrences and the total number of motifs (nb_tot) appearing in the considered TN. Its predicted probability is instead computed under the hypothesis of statistical independence between its parts. To simplify the discussion, we make the following hypotheses: (1) the motif we consider is an NCTN, (2) we denote its depth by d and (3) we sample the motif on the untouched TN, i.e. $n = b = 1$.

An NCTN is composed of one central node and n_{sat} satellites, with $n_{sat} \geqslant 1$. Each satellite is characterized by an activity profile, which is a binary string of length equal d, with '1' indicating the satellite is interacting with the central node and '0' indicating it is not. An NCTN is then given by the tuple $(s_a)_{a\in A}$, where A is the set of possible activity profiles and n_a is the number of satellites with a as activity profile.

The statistical independence hypothesis states that no correlation between satellites exist, so the occurrence probability of a motif is then the product of probabilities of the activity profiles of its satellites, as well as the probability of having the correct number of satellites. This occurrence probability can thus be written as:

$$P^{\mathrm{th}}\left((s_a)_{a\in A}\right) = P\left(n_{\mathrm{sat}} = s\right) s! \prod_{a\in A} \frac{P(a)}{s_a!}$$

where $s = \sum_{a\in A} s_a$ denotes the total number of satellites n_{sat}, and $P(a)$ denotes the probability to have a as a activity profile. Note that n_{sat} can be obtained from the instantaneous degree k_d at aggregation level d, taking care that it cannot be zero:

$$P\left(n_{\mathrm{sat}} = s\right) = P\left(k_d = s | k_d \geqslant 1\right) = \frac{P\left(k_d = s\right)}{1 - P\left(k_d = 0\right)}$$

This makes possible the computation of $P(n_{\mathrm{sat}} = s)$ without sampling the NCTN occurring in our network. The activity profiles can also be sampled without explicitly computing the NCTN in the following way. Activity profiles are collected from every edge and every time of the TN. Only the zero activity profile $\underbrace{0\ldots0}_{d}$ must be excluded, because empty motifs are not collected. More precisely, for each time t and each edge (i,j) such that (i,j) is active at least once between times t and $t+d-1$, we collect the activity profile of (i,j) at t of length d. This profile is a binary string of '0' and '1'. If positions along this string are indexed from 0 to $d-1$, a '1' in position k means that the edge (i,j) is active at time $t+k$. Then, the probability $P(a)$ is given by the fraction of occurrences (i,j,t) that an edge (i,j) has a as activity profile from time t to $t+d-1$.

Denoting the observed probability of occurrence of a motif by P^{obs} and the set of motifs observed in our data set by M, we can now define the motif_error as:

$$\sqrt{\frac{1}{|M|}\sum_{m\in M}\left(P^{\mathrm{obs}}(m) - P^{\mathrm{th}}(m)\right)^2}\,.$$

Appendix F. Extracting the derivative sign with a neural network

To extract the denoised sequence of the sign derivative of a given curve, we built and used a neural network.

F.1. Architecture

The input layer is the rescaled data curve, where the rescaling of a curve $x_i, i = 1,\ldots,n$ consists in the following:

- if the curve is constant, replace its value by 0.5;
- otherwise, replace x_i by $\frac{x_i - m}{M - m}$, with $m = \min_i\{x_i\}$ and $M = \max_i\{x_i\}$.

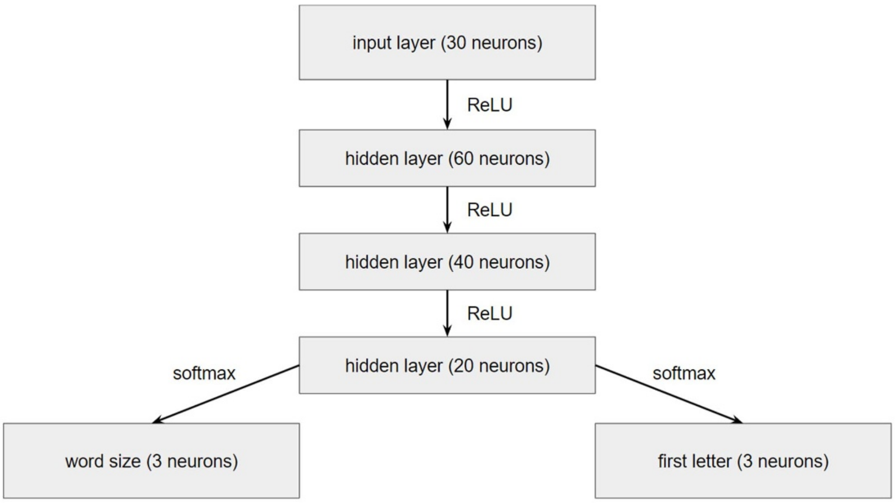


Figure F4. Architecture of the neural network used for analog-symbolic conversion. The input layer receives the rescaled and completed curve (see text). Since the word size is encoded by three neurons and one-hot encoding is used, this neural network can only produce words with a size either of 1, 2 or 3 letters. ReLU was used in every hidden layer as activation function, while softmax was used for the two outputs. The direction of the black arrows gives the information flow inside the network.

Since each curve has 30 points at most in our case, the input layer has 30 neurons. It can deal with curves with less points by repeating their last value until 30 points are filled.

Two output layers are used: one for the word size and one for its first letter. Indeed, a word is fully determined by its first letter and its size. This holds because curves we consider are either flat (so described by the word ‘0’), or they have no flat parts (so described by a word with no ‘0’ in it). The maximal word size we encountered in empirical cases was 3 so we assigned 3 neurons to the output coding for the word size. As there are three possible letters, the output coding for the first letter has 3 neurons. One-hot encoding was used for both outputs.

The overall architecture of the neural network is given in figure F4. Three intermediate layers were used.

F.2. Training

The training is supervised and the cost function is the categorical cross-entropy. The number of epochs used for training has been chosen to maximize the validation accuracy, which was 91% in our case. Training set consists in 20 000 examples of curves with various sizes, with known derivative sign sequence and endowed with white Gaussian noise as well as salt-and-pepper noise. Salt-and-pepper noise consists in randomly choosing one point $1 \leqslant i \leqslant n$ of the curve and replacing its value by either the curve maximum (probability 0.5) or the curve minimum (probability 0.5).

The training set is composed by curves described by various words. Flat curves constitute $\frac{1}{7} \simeq 14\%$ of the training set. The remaining $\frac{6}{7}$ are equally split in three parts, each corresponding to a word size 1, 2 or 3. Thus, the word '0' put apart, each size represents $\frac{2}{7} \simeq 28\%$ of the training set. For each size, examples with either letter '$+$' or '$-$' as first letter are present in equal proportion. Here is the procedure to generate an example:

1. choose a word size and first letter, as well as the curve length (number of points) randomly between 11 and 30 included;
2. generate a curve of the chosen length whose derivative sign sequence matches the chosen word;
3. rescale the curve according to the steps described in previous section F.1;
4. add white Gaussian noise of standard deviation 0.05, then add salt-and-pepper noise with probability 0.05;
5. rescale again the curve and repeat its last value to get 30 points:

$$x_1, \ldots, \underbrace{x_n, \ldots, x_n}_{30-n+1\text{copies}} \ .$$

Let us detail the step 2, or how we generate a curve with a given sign sequence for its derivative. Curves described by the word '0' are the simplest to generate: we take $x_i = 0.5$, $\forall 1 \leqslant i \leqslant n$. Now let us consider a word of length l. To build a curve described by this word, we first place the extrema of the curve and then fit a continuous function to fill the gaps between the extrema. For a word of length l, we have $l+1$ points to place in the plane. We denote those points by $(x_0, y_0), \ldots, (x_l, y_l)$. We put the first one of these points at the origin $(0,0)$ and the second one at the coordinates $(1,1)$ (resp. $(1,-1)$) if the first letter is '$+$' (resp. '$-$'). Said otherwise, introducing ε such that $\varepsilon = 0$ if the first letter is '$-$' and $\varepsilon = 1$ if the first letter is '$+$', we have:

$$\begin{cases} x_0 = 0; \ y_0 = 0 \\ x_1 = 1; \ y_1 = 2\epsilon - 1 \ . \end{cases}$$

The remaining points are placed one after the other, by pursuing to the right in a recursion process.

Note that we cannot place points at random, since we need to control two aspects of the curve, its maximum height ratio r_h and its maximum slope ratio r_s. Height and slope are properties of segments joining two consecutive points (x_k, y_k) and (x_{k+1}, y_{k+1}), where $0 \leqslant k \leqslant l-1$. The height h_k and the slope s_k of such a segment are defined as:

$$\begin{cases} h_k = |y_{k+1} - y_k| \\ s_k = \frac{h_k}{x_{k+1} - x_k} \end{cases}$$

The maximum height and slope ratios are then defined as:

$$\begin{cases} r_h = \frac{M_h}{m_h} \\ r_s = \frac{M_s}{m_s} \\ m_h = \min_k (h_k); \ M_h = \max_k (h_k) \\ m_s = \min_k (s_k); \ M_s = \max_k (s_k) \end{cases}$$

To build our training set, we set $r_h = r_s = 5$. Ratios too high would lead to curves that seem discontinuous, and ratios too low would lead to curves that look flat, making impossible to guess the correct word. A value of 5 seems to be a good compromise according to human eye appreciation.

Let us now describe how we generate our $l+1$ points recursively. Let us assume k points $(x_0, y_0), \ldots, (x_{k-1}, y_{k-1})$ have already been placed, $k \geqslant 2$. Let us denote the extrema height and slope computed at step $k-1$:

$$\begin{cases} m_h^{(k-1)} = \min\limits_{0 \leqslant i \leqslant k-2} (h_i)\,;\; M_h^{(k-1)} = \max\limits_{0 \leqslant i \leqslant k-2} (h_i) \\ m_s^{(k-1)} = \min\limits_{0 \leqslant i \leqslant k-2} (s_i)\,;\; M_s^{(k-1)} = \max\limits_{0 \leqslant i \leqslant k-2} (s_i) \end{cases}.$$

We place the $(k+1)$th point at the right end of a segment line, starting from the last placed point. Its height and slope must be compatible with the maximum ratios:

$$\begin{cases} m_h^{(k-1)} r_h \leqslant h_{k-1} \leqslant \frac{M_h^{(k-1)}}{r_h} \\ m_s^{(k-1)} r_s \leqslant s_{k-1} \leqslant \frac{M_s^{(k-1)}}{r_s} \end{cases}$$

To ensure this, we draw h_{k-1} and s_{k-1} uniformly at random between their bounds. The recursion relation follows:

$$\begin{cases} y_k = y_{k-1} + (-1)^{k+\epsilon} h_{k-1} \\ x_x = x_{k-1} + \frac{h_{k-1}}{s_{k-1}} \\ m_h^{(k)} = \min\left(m_h^{(k-1)}, h_{k-1}\right)\,;\; M_h^{(k)} = \max\left(M_h^{(k-1)}, h_{k-1}\right) \\ m_s^{(k)} = \min\left(m_s^{(k-1)}, s_{k-1}\right)\,;\; M_s^{(k)} = \max\left(M_s^{(k-1)}, s_{k-1}\right) \end{cases}.$$

Once our $l+1$ points $(x_0, y_0), \ldots, (x_l, y_l)$ have been placed, we rescale both coordinates between 0 and 1. For now, we only have the skeleton of our final curve ; it remains to dress it by adding intermediate points between the extrema.

The dressing principle is to fit a polynomial of degree l, so that the sign sequence of its derivative matches exactly the word we chose. However, we squeeze our skeleton before fitting to ensure diversity in the examples we feed to the neural network: squeezing is done by applying a map to the x-coordinates, chosen among 6 different maps. The dressing procedure is fully described in figure F5.

The six maps we use are:

- 'raw': $x \mapsto x$
- 'inv': $x \mapsto \frac{1}{1+\gamma x}$
- 'exp': $x \mapsto \exp(\gamma x)$
- 'expinv': $x \mapsto \exp(-\gamma x)$
- 'log': $x \mapsto \log(1+\gamma x)$
- 'tanh': $x \mapsto \tanh(\gamma x)$

where γ is a float parameter drawn at random between 1 and 10 after a map is chosen.

Once the skeleton has been dressed into a curve of n points, the final curve consists in the rescaled sequence of the y-coordinates $y_0, \ldots, y_{n-1}$.

Examples of final curves are displayed in figure F6 together with their associated words.

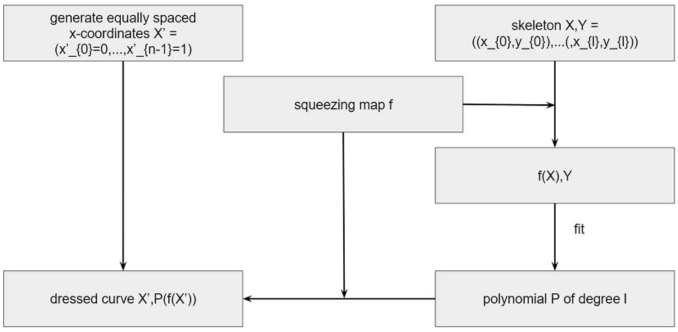

generate equally spaced x-coordinates X' = (x'_{0}=0,...,x'_{n-1}=1)
skeleton X,Y = ((x_{0},y_{0}),...(,x_{l},y_{l}))
squeezing map f
f(X),Y
fit
dressed curve X',P(f(X'))
polynomial P of degree l

Figure F5. Completion of a skeleton into one curve. Once a dressed curve is obtained, we check that its maximum height and slope ratios do not exceed the specified bounds $r_\mathrm{h} = r_\mathrm{s} = 5$. If so, a new dressing is performed with another squeezing map drawn at random among the 6 available, combined with a parameter γ drawn at random too (see text).

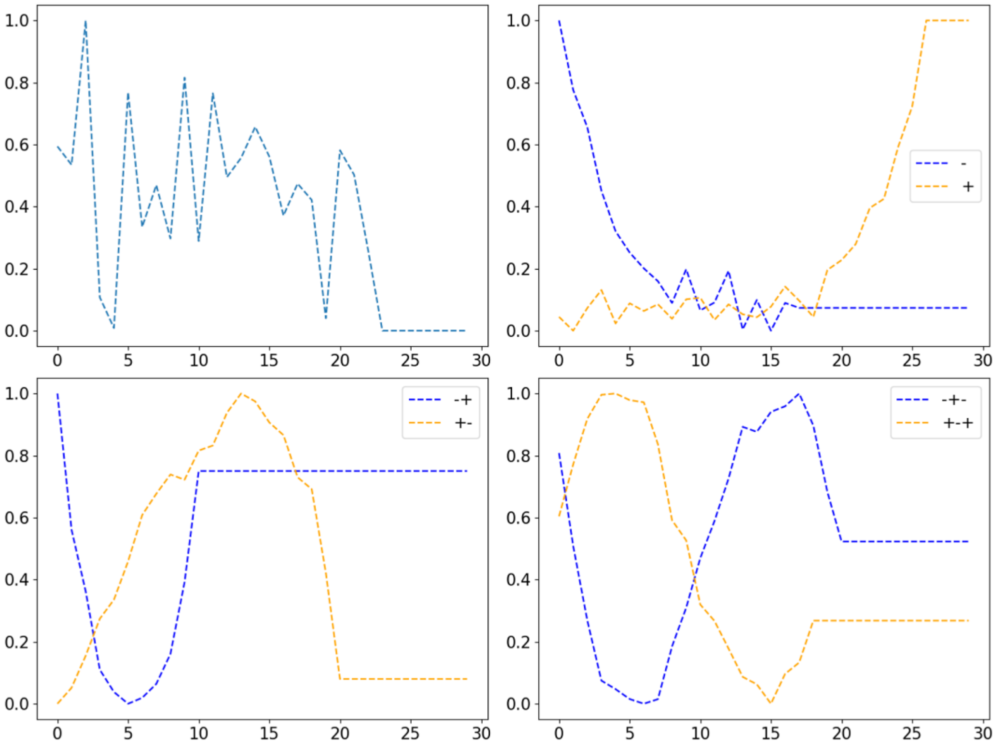


Figure F6. Examples of curves used for training the neural network. The displayed curves have been drawn at random from the training set of 20 000 labeled examples fed to the neural network for its supervised training. We drew one example per word present in the training set. The legend indicates the training labels of the curves. The curve shown at the top left is labeled by the word ‘0’.

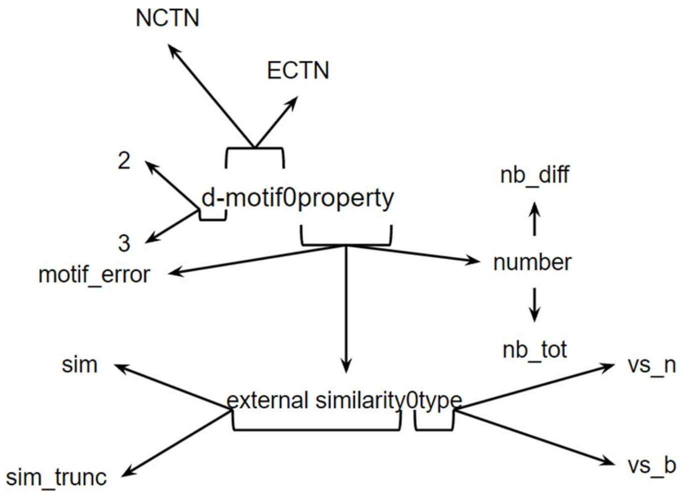


Figure G7. Nomenclature of motif-based observables. All possible names have been encoded in a tree. A name is read by choosing recursively a child for each parent until leaves are reached. Going from a parent to a child is done by following the arrows. Here are examples of names generated according to this principle: '3-ECTN0sim_trunc0vs_b', '2-NCTN0nb_tot'.

Appendix G. Nomenclature of observables

In section 4.2, we described various observables of a TN. Here, we explain how we name an observable in a standardized way, which is suitable in particular for numerical analysis.

Recall that we introduced two main types of observables: distribution and motif-based observables. We name distribution observables by concatenation of a prefix and a suffix separated by a '0'. The prefix indicates the observed object and the suffix gives the object property we care about. In this paper, an object is a node or an edge while a property is e.g. its duration, time_weight, etc (cf section 4.2.1). Each distribution observable gives rise to two scalar ones by following the naming convention:

object0property0scalar_type.

We considered only two scalar types: 'avg' for $\langle x\rangle$ and 'frac' for $\frac{\langle x^2\rangle}{\langle x\rangle}$. For example, 'node0duration0avg' refers to the average number of consecutive time steps a node is active.

The naming convention of motif-based observables is simpler:

object0property.

However, an object is given by a motif type and its depth d together, written as:

d-motif_type.

A property is then any item from the list of section 4.2.2. For example, 2-NCTN0nb_tot is the total number of NCTN of depth 2, and 3-ECTN0sim_trunc0vs_b is the degree of invariance of the ECTN vector of depth 3 under STS. At level n, this invariance is quantified by the cosine similarity between the vectors obtained for $\mathrm{TN}(n,1)$ and $\mathrm{TN}(n,b)$.

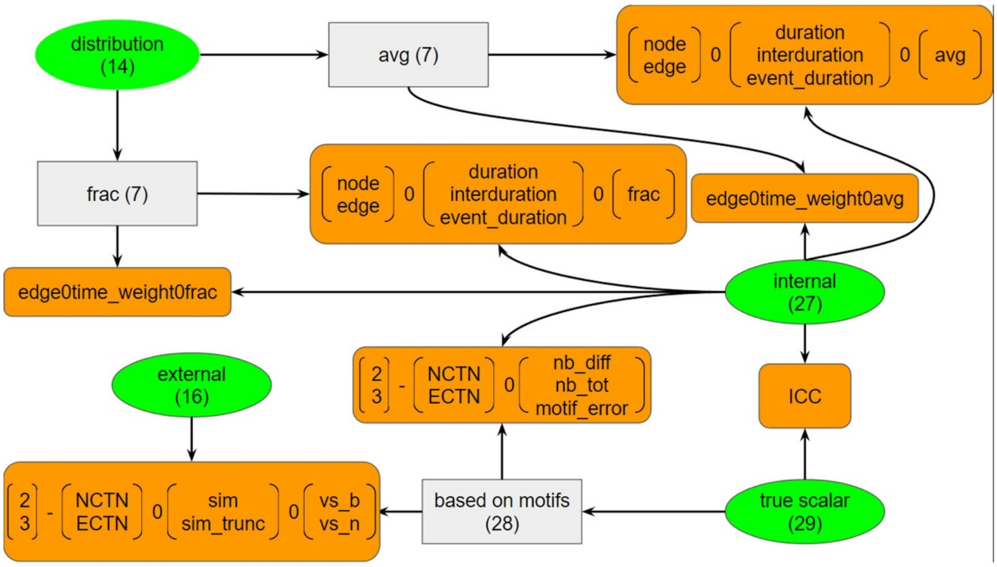

distribution (14)
avg (7)
node edge 0 duration interduration event_duration 0 avg
frac (7)
node edge 0 duration interduration event_duration 0 frac
edge0time_weight0avg
edge0time_weight0frac
internal (27)
external (16)
2 3 - NCTN ECTN 0 nb_diff nb_tot motif_error
ICC
2 3 - NCTN ECTN 0 sim sim_trunc 0 vs_b vs_n
based on motifs (28)
true scalar (29)

Figure G8. Recap of the types of observables considered in this paper. Observables have been displayed by using a directed acyclic graph. Leaves are the rounded boxes colored in orange. They contain observables of the same type. In the non-leaves boxes are indicated a type and the number of observables of this type among the ones considered in this paper. Roots are highlighted as elliptic boxes colored in green. The gray rectangular boxes correspond to the levels intermediate between roots and leaves. An arrow from a box i to a box j means that i contains j. For example, the observable edge0time_weight0avg is both an internal and a scalarized observable of type avg. 'True scalar' observables are just the observables that are not considered as deriving from a distribution. Note that some boxes contain column vectors. It means that these boxes contain every observable that is possible to write by choosing any component for each column vector. For example, the out-neighbor of the box 'frac (7)' contains $2\times3\times1=6$ observables.

Note that, when computing this similarity, we must precise whether we compute it versus b or versus n. Thus, we distinguish between internal and external observables, because we cannot use the same algorithm to compute them: *Internal* observables can be computed for every couple (n,b) using only the transformed data set $\mathrm{TN}(n,b)$. They are the majority of the observables considered here, like e.g. the total number of motifs or the average node duration. *External* observables need some additional data to be computed. For example, the similarity between $\mathrm{TN}(n,b)$ and $\mathrm{TN}(1,b)$ is an external observable because it requires $\mathrm{TN}(1,b)$ besides $\mathrm{TN}(n,b)$ to be computed.

On figure G7, we propose a diagrammatic representation of the nomenclature of all motifs-based observables. On figure G8, we summarize every type of observable encountered in this paper, and how they relate to each other.

ORCID iDs

Didier Le Bail https://orcid.org/0009-0007-8816-2678
Alain Barrat https://orcid.org/0000-0001-8683-269X

References

[1] Holme P 2015 *Eur. Phys. J.* B **88** 234
[2] Holme P and Saramäki J 2012 *Phys. Rep.* **519** 97
[3] Braha D and Bar-Yam Y 2006 *Complexity* **12** 59
[4] Pfitzner R, Scholtes I, Garas A, Tessone C J and Schweitzer F 2013 *Phys. Rev. Lett.* **110** 198701
[5] Scholtes I, Wider N, Pfitzner R, Garas A, Tessone C J and Schweitzer F 2014 *Nat. Commun.* **5** 5024
[6] Valdano E, Ferreri L, Poletto C and Colizza V 2015 *Phys. Rev.* X **5** 021005
[7] Masuda N, Klemm K and Eguíluz V M 2013 *Phys. Rev. Lett.* **111** 188701
[8] Li A, Cornelius S P, Liu Y-Y, Wang L and Barabási A-L 2017 *Science* **358** 1042
[9] Pan R K and Saramäki J 2011 *Phys. Rev.* E **84** 016105
[10] Kovanen L, Karsai M, Kaski K, Kertész J and Saramäki J 2011 *J. Stat. Mech.* **2011** 11005
[11] Chen B, Hou G and Li A 2024 *Phys. Rev.* E **109** 064302
[12] Masuda N and Lambiotte R 2016 *A Guide to Temporal Networks* (World Scientific)
[13] Masuda N and Holme P 2019 *Sci. Rep.* **9** 795
[14] Gelardi V, Fagot J, Barrat A and Claidière N 2019 *Animal Behav.* **157** 239
[15] Galimberti E, Bonchi F, Gullo F and Lanciano T 2020 *ACM Trans. Knowl. Discov. Data* **14** 11
[16] Pedreschi N, Battaglia D and Barrat A 2022 *Nat. Phys.* **18** 931
[17] Longa A, Cencetti G, Lepri B and Passerini A 2022 *Data Min. Knowl. Discovery* **36** 355
[18] Li A, Zhou L, Su Q, Cornelius S P, Liu Y-Y, Wang L and Levin S A 2020 *Nat. Commun.* **11** 2259
[19] Perra N, Gonçalves B, Pastor-Satorras R and Vespignani A 2012 *Sci. Rep.* **2** 469
[20] Zhao K, Stehlé J, Bianconi G and Barrat A 2011 *Phys. Rev.* E **83** 056109
[21] Vestergaard C L, Génois M and Barrat A 2014 *Phys. Rev.* E **90** 042805
[22] Laurent G, Saramäki J and Karsai M 2015 *Eur. Phys. J.* B **88** 301
[23] Longa A, Cencetti G, Lehmann S, Passerini A and Lepri B 2024 *Commun. Phys.* **7** 22
[24] Berlingerio M, Koutra D, Eliassi-Rad T and Faloutsos C 2013 *Proc. 2013 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining* (*ASONAM '13*) (Association for Computing Machinery) p 1439–40
[25] Bagrow J P and Bollt E M 2019 *Appl. Netw. Sci.* **4** 45
[26] Tantardini M, Ieva F, Tajoli L and Piccardi C 2019 *Sci. Rep.* **9** 17557
[27] Harrison H, Brennan K, Stefan M, Alexander D, Guillaume S-O, Charles M and Laurent H-D 2020 *Proc. R. Soc.* A **476** 20190744
[28] Tu K, Li J, Towsley D, Braines D and Turner L D 2018 arXiv:1807.03733
[29] Zhan X-X, Liu C, Wang Z, Wang H, Holme P and Zhang Z-K 2021 (arXiv:2111.01334)
[30] Andres E, Barrat A and Karsai M 2024 Detecting periodic time scales of changes in temporal networks *J. Complex Netw.* **12** 2
[31] Sikdar S, Ganguly N and Mukherjee A 2016 *Eur. Phys. J.* B **89** 11
[32] Nie C-X 2023 *Nonlinear Dyn.* **111** 481
[33] Barrat A and Cattuto C 2013 *Temporal Networks* (Springer) pp 191–216
[34] Le Bail D, Génois M and Barrat A 2023 *Phys. Rev.* E **107** 024301
[35] Tang J, Musolesi M, Mascolo C and Latora V 2009 *Proc. 2nd ACM Workshop on Online Social Networks* pp 31–36
[36] Nicosia V, Tang J, Mascolo C, Musolesi M, Russo G and Latora V 2013 Graph metrics for temporal networks *Temporal Networks* (*Understanding Complex Systems*) (Springer) p 15
[37] Karsai M, Kaski K, Barabási A-L and Kertész J 2012 *Sci. Rep.* **2** 397
[38] Cattuto C, Van den Broeck W, Barrat A, Colizza V, Pinton J-F and Vespignani A 2010 *PLoS One* **5** e11596
[39] Toth D J, Leecaster M, Pettey W B, Gundlapalli A V, Gao H, Rainey J J, Uzicanin A and Samore M H 2015 *J. R. Soc. Interface* **12** 20150279
[40] Mastrandrea R, Fournet J and Barrat A 2015 *PLoS One* **10** 315
[41] Génois M, Zens M, Oliveira M, Lechner C, Schaible J and Strohmaier M 2023 Combining sensors and surveys to study social interactions: a case of four science conferences *Pers. Sci.* **4** 1
[42] Sapiezynski P, Stopczynski A, Lassen D D and Lehmann S 2019 *Sci. Data* **6** 1
[43] Krings G, Karsai M, Bernhardsson S, Blondel V D and Saramäki J 2012 *EPJ Data Sci.* **1** 4
[44] Sulo R, Berger-Wolf T and Grossman R 2010 *Proc. 8th Workshop on Mining and Learning With Graphs* pp 127–36
[45] Clauset A, Shalizi C R and Newman M E J 2009 *SIAM Rev.* **51** 661
[46] Voitalov I, van der Hoorn P, van der Hofstad R and Krioukov D 2019 *Phys. Rev. Res.* **1** 033034

[47] Gemmetto V, Barrat A and Cattuto C 2014 *BMC Infect. Dis.* **14** 695
[48] Stehlé J *et al* 2011 *PLoS One* **6** e23176
[49] Fournet J and Barrat A 2014 *PLoS One* **9** e107878
[50] Vanhems P, Barrat A, Cattuto C, Pinton J-F, Khanafer N, Régis C, Kim B-a, Comte B and Voirin N 2013 *PLoS One* **8** e73970
[51] Kiti M C *et al* 2016 *EPJ Data Sci.* **5** 21
[52] Gelardi V, Godard J, Paleressompoulle D, Claidiere N and Barrat A 2020 *Proc. R. Soc* A **476** 20190737
[53] Génois M, Vestergaard C L, Fournet J, Panisson A, Bonmarin I and Barrat A 2015 *Netw. Sci.* **3** 326
[54] Génois M and Barrat A 2018 *EPJ Data Sci.* **7** 11
[55] Vicsek T, Czirók A, Ben-Jacob E, Cohen I and Shochet O 1995 *Phys. Rev. Lett.* **75** 1226
[56] SocioPatterns collaboration n.d. (available at: www.sociopatterns.org/) (accessed 5 October 2021)
[57] Gauvin L, Génois M, Karsai M, Kivelä M, Takaguchi T, Valdano E and Vestergaard C L 2022 *SIAM Rev.* **64** 763
[58] Battiston F, Cencetti G, Iacopini I, Latora V, Lucas M, Patania A, Young J-G and Petri G 2020 *Phys. Rep.* **874** 1
[59] Masoumi R, Gambaudo J and Génois M 2024 Simple crowd dynamics to generate complex temporal contact networks (arXiv:2405.06508 [Preprint])
[60] Solon A P, Cates M E and Tailleur J 2015 *Eur. Phys. J. Spec. Top.* **224** 1231
[61] Le Bail D 2024 From ego to inter-ego motifs in temporal networks (in preparation)
[62] Bastian M, Heymann S and Jacomy M 2009 Gephi: an open source software for exploring and manipulating networks (www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154)
[63] Pedregosa F *et al* 2011 *J. Mach. Learn. Res.* **12** 2825
[64] Gelardi V, Le Bail D, Barrat A and Claidiere N 2021 *Proc. R. Soc.* B **288** 20211164