Phasik: a Python package to identify system states in partially temporal networks

Maxime Lucas[1¶], **Alex Townsend-Teague**[2], **Matteo Neri**[1,3], **Simone Poetto**[1,4], **Arthur Morris**[5], **Bianca Habermann**[6], **and Laurent Tichit**[7]

1 CENTAI Institute, Turin, Italy **2** Dahlem Center for Complex Quantum Systems, Freie Universitat Berlin, 14195 Berlin, Germany **3** Institut de Neurosciences de la Timone UMR 7289, Aix Marseille Université, CNRS, Marseille 13005, France **4** Center for Modern Interdisciplinary Technologies, Nicolaus Copernicus University, Toruń, Poland **5** Theory of Condensed Matter Group, Cavendish Laboratory, University of Cambridge, J. J. Thomson Avenue, Cambridge CB3 0HE, United Kingdom **6** Aix Marseille University, CNRS, IBDM UMR 7288, Turing Center for Living Systems, Marseille, France **7** Aix Marseille University, CNRS, I2M UMR 7373, Turing Center for Living Systems, Marseille, France ¶ Corresponding author

DOI: 10.21105/joss.05872

Software

- Review
- Repository
- Archive

Editor: Charlotte Soneson

Reviewers:

- @GiulioRossetti
- @alexbovet

Submitted: 19 August 2023
Published: 21 November 2023

Summary

Phasik is a Python library for analyzing the temporal structure of temporal and partially temporal networks. Temporal networks are used to model complex systems that consist of entities with time-varying interactions. This library provides methods for building temporal networks (including from data), visualizing them, and analyzing their structure. In particular, Phasik focuses on the identification of temporal phases, that is, periods of time during which the system is in a given state. The library supports partially temporal networks for which information about only a subset of the edges' temporal evolution is available. Phasik is implemented in pure Python and integrates with the rest of the Python scientific stack.

Statement of need

Temporal networks allow to model a wide range of complex systems (Holme & Saramäki, 2012), and study both their structure (Longa et al., 2022) and dynamics (Ghosh et al., 2022; Lucas et al., 2018). As such, temporal networks are used by scientists from different fields and backgrounds. To help non-specialists apply temporal network theory, common and ready-to-use computational tools are therefore crucial. For traditional static networks, several Python libraries have had a big impact on the field. Examples include igraph (Csardi & Nepusz, 2006), NetworkX (Hagberg et al., 2008), and graph-tool (Peixoto, 2014). Temporal networks, however, include time as an additional dimension and thus require different approaches. There exist several Python libraries for working with temporal networks, each focusing on different aspects of temporal networks analysis. None of these libraries, however, focuses on identifying the temporal structure of a complex system, which is often intricately linked to the system's function. Systems for which the temporal structure is crucial span many scientific fields, with example including the phases and subphases of the cell cycle (biology) (Koch & Nasmyth, 1994), the five sleep stages the brain goes through (medicine) (Loomis et al., 1937), or the phases of social dynamics a school goes through in a day (Masuda & Holme, 2019). A popular repository for temporal network data is SocioPatterns (Barrat et al., 2013; *SocioPatterns*, 2008). Very often, understanding the temporal structure of such systems is a first step towards understanding the underlying mechanisms at play. A handful of studies have applied temporal network theory to identify these system states from data (Gelardi et al., 2019; Masuda & Holme, 2019; Pedreschi et al., 2020).

Related software

There exist a few Python libraries for analyzing temporal networks, each with a different focus. Most of them provide a base `TemporalNetwork` class, methods for manually adding and removing temporal edges, and methods for calculating basic quantities such as degree, snapshots of the temporal network at a given time, or neighbors. Teneto (Thompson et al., 2020) is versatile: it provides different static visualizations, temporal network measures, algorithms for community detection, and has methods to help using neuroimaging data. Tacoma (Maier, 2018) is the most advanced in terms of visualizations, with interactive dynamic plotting functions. It also provides functions to simulate spreading processes. Some of its core routines are written in C++, improving its speed. Reticula (Badie-Modiri & Kivelä, 2023) has a C++ backbone and is a general-purpose package with several generative models and randomization algorithms. Raphtory is also general-purpose and written in Rust for speed, yet accessible as a Python library (Steer et al., 2023). DyNetX (Rossetti et al., 2020) provides a class for directed temporal networks as well as a method to compute time-respecting paths. RandTempNet (Génois, 2019) is more specifically focused on temporal network randomization algorithms. The flow_stability repository (Bovet et al., 2022) provides code for the detection of time-varying communities in temporal networks, similarly to tnetwork (Cazabet, 2019). Pathpy (Hackl et al., 2021)] has a slightly different focus: it provides tools to analyze higher-order path correlations (or memory) in time-stamped data,

A few libraries also exist in other programming languages. In R, ndtv (Bender-deMoll, 2016) focuses on the visualization of temporal networks, both static and animated. In C, P. Holme provides a fast implementation of the susceptible-infected-recovered (SIR) epidemic model on temporal networks (Holme, 2021). TimeNexus (Pierrelée et al., 2021) is a general-purpose Cytoscape app for temporal networks with emphasis on modeling gene expression data.

Similarly to these libraries, Phasik provides general classes, measures, and visualizations functions. However, it distinguishes itself by having a specific focus on the identification of temporal phases, and system states, in complex systems for a range of time-scales from (partially) temporal networks. Concretely, Phasik provides a `PartiallyTemporalNetwork` class and classes dedicated to the clustering of snapshots for the identification of phases. Phasik also naturally handles unevenly spaced time points, which can result from experimental data, especially in biology. Phasik was initially designed with biological networks in mind but the analysis pipeline it provides is general and can be used on any other type of temporal data.

Overview

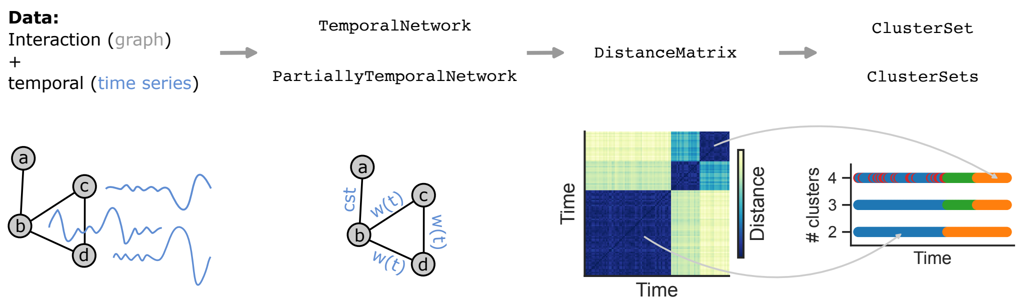

Figure 1: The Phasik pipeline to identify phases. First, a partially temporal network is built from data, such as interaction data (a graph) and (node) time series. Second, the distances between snapshots of the temporal network at different times are encoded in a distance matrix, similar to a recurrence matrix. Finally, clusters of snapshots are computed from the distance matrix. Each snapshot is associated to a time point, so that the resulting clusters of snapshots correspond to periods of time—*phases*. In this example, time series are available for only three out of four nodes. Consequently, a partially temporal network is built, in which the edges without temporal information are set to a constant weight.

The library is composed of two core classes for temporal networks (`TemporalNetwork` and its child class `PartiallyTemporalNetwork`) as well as three additional classes designed for the identification of phases (`DistanceMatrix`, `ClusterSet`, and `ClusterSets`). Phasik's workflow consists of three main steps, as illustrated in Figure 1. First, we build a `TemporalNetwork` by combining time series data to interaction data in the form of a network. Second, we compute a `DistanceMatrix` which encodes the distance between the instantaneous snapshot of the temporal network between any two time points, similarly to a recurrence matrix. Third, we partition the temporal network into phases, through its `DistanceMatrix`, by clustering its snapshots. One can compute a single partition, or several partitions corresponding to different numbers of clusters.

This workflow is conveniently implemented through constructor methods: at each step, the new class instance is built from that of the previous step with ".from_*"-named methods. For example, the second step takes the form `DM = pk.DistanceMatrix.from_temporal_network(TN)`, where `TN` is a `TemporalNetwork` (a distance metric also needs to be specified).

The library provides several ways to build a `TemporalNetwork` from data. Temporal information is always required and can be of three formats: (1) temporal edges, i.e. tuples (i, j, t, w) to represent an interaction between nodes i and j and at time t with weight w, (2) time series relative to nodes (see Figure 1), and (3) time series relative to edges. Node time series are automatically combined to form edge time series: by default, two node series are multiplied and normalized to form an edge time series (other ways can be specified by the user). Interaction data in the form of a graph can optionally be used as input. In some experimental scenarios, temporal information is not available for all edges in a graph. This then results in a `PartiallyTemporalNetwork`, where edges without temporal information are assigned a constant default weight of one. Phasik provides basic functions to visualize, animate, and analyze the resulting temporal network.

The `DistanceMatrix` can be computed after specifying a distance metric (by default, Euclidean distance is used). Finally, after specifying a clustering method with related parameters, and a number of clusters to compute, one can compute a `ClusterSet` (by default, hierarchical clustering is used). If a range of numbers of clusters is given, the library computes a `CluserSets` instead. At each step, Phasik provides visualization functions that can be fine-tuned by the user.

Projects using Phasik

Phasik was originally designed to conduct the research presented in (Lucas et al., 2023). Phasik has also been compared to related software in (Badie-Modiri & Kivelä, 2023; Steer et al., 2023).

Acknowledgements

We thanks Alain Barrat for useful discussions during the development of the library.

References

Badie-Modiri, A., & Kivelä, M. (2023). Reticula: A temporal network and hypergraph analysis software package. *SoftwareX*, *21*, 101301. https://doi.org/10.1016/j.softx.2022.101301

Barrat, A., Cattuto, C., Colizza, V., Gesualdo, F., Isella, L., Pandolfi, E., Pinton, J.-F., Ravà, L., Rizzo, C., Romano, M., & others. (2013). Empirical temporal networks of face-to-face human interactions. *The European Physical Journal Special Topics*, *222*, 1295–1309. https://doi.org/10.1140/epjst/e2013-01927-7

Bender-deMoll, S. (2016). *ndtv: Network Dynamic Temporal Visualizations* (Version 0.10). https://github.com/statnet/ndtv

Bovet, A., Delvenne, J.-C., & Lambiotte, R. (2022). Flow stability for dynamic community detection. *Science Advances*, *8*(19), eabj3063. https://doi.org/10.1126/sciadv.abj3063

Cazabet, R. (2019). *tnetwork: A Python software package to manipulate temporal networks* (Version 0.5.0). https://github.com/Yquetzal/tnetwork/

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, *Complex Systems*, 1695. https://igraph.org

Gelardi, V., Fagot, J., Barrat, A., & Claidière, N. (2019). Detecting social (in) stability in primates from their temporal co-presence network. *Animal Behaviour*, *157*, 239–254. https://doi.org/10.1016/j.anbehav.2019.09.011

Génois, M. (2019). *RandTempNet: A collection of Python script for temporal networks*. https://github.com/mgenois/RandTempNet

Ghosh, D., Frasca, M., Rizzo, A., Majhi, S., Rakshit, S., Alfaro-Bittner, K., & Boccaletti, S. (2022). The synchronized dynamics of time-varying networks. *Physics Reports*, *949*, 1–63. https://doi.org/10.1016/j.physrep.2021.10.006

Hackl, J., Scholtes, I., Petrović, L. V., Perri, V., Verginer, L., & Gote, C. (2021). Analysis and visualisation of time series data on networks with pathpy. *Companion Proceedings of the Web Conference 2021*, 530–532. https://doi.org/10.1145/3442442.3452052

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th Python in Science Conference* (pp. 11–15). https://conference.scipy.org/proceedings/SciPy2008/paper_2/

Holme, P. (2021). Fast and principled simulations of the SIR model on temporal networks. *PLOS ONE*, *16*(2), e0246961. https://doi.org/10.1371/journal.pone.0246961

Holme, P., & Saramäki, J. (2012). Temporal networks. *Physics Reports*, *519*(3), 97–125. https://doi.org/10.1016/j.physrep.2012.03.001

Koch, C., & Nasmyth, K. (1994). Cell cycle regulted transcription in yeast. *Current Opinion in Cell Biology*, *6*(3), 451–459. https://doi.org/10.1016/0955-0674(94)90039-6

Longa, A., Cencetti, G., Lehmann, S., Passerini, A., & Lepri, B. (2022). Neighbourhood matching creates realistic surrogate temporal networks. *arXiv:2205.08820*. https://doi.org/10.48550/arXiv.2205.08820

Loomis, A. L., Harvey, E. N., & Hobart, G. A. (1937). Cerebral states during sleep, as studied by human brain potentials. *Journal of Experimental Psychology*, *21*(2), 127. https://doi.org/10.1037/h0057431

Lucas, M., Fanelli, D., Carletti, T., & Petit, J. (2018). Desynchronization induced by time-varying network. *Europhysics Letters*, *121*(5), 50008. https://doi.org/10.1209/0295-5075/121/50008

Lucas, M., Morris, A., Townsend-Teague, A., Tichit, L., Habermann, B. H., & Barrat, A. (2023). Inferring cell cycle phases from a partially temporal network of protein interactions. *Cell Reports Methods*, 100397. https://doi.org/10.1016/j.crmeth.2023.100397

Maier, B. F. (2018). *tacoma: A Python library for TemporAl COntact Modeling and Analysis*. https://github.com/benmaier/tacoma

Masuda, N., & Holme, P. (2019). Detecting sequences of system states in temporal networks. *Scientific Reports*, *9*(1), 1–11. https://doi.org/10.1038/s41598-018-37534-2

Pedreschi, N., Bernard, C., Clawson, W., Quilichini, P., Barrat, A., & Battaglia, D. (2020). Dynamic core-periphery structure of information sharing networks in entorhinal cortex and hippocampus. *Network Neuroscience*, *4*(3), 946–975. https://doi.org/10.1162/netn_a_00142

Peixoto, T. P. (2014). The graph-tool Python library. *Figshare*. https://doi.org/10.6084/m9.figshare.1164194

Pierrelée, M., Reynders, A., Lopez, F., Moqrich, A., Tichit, L., & Habermann, B. H. (2021). Introducing the novel cytoscape app TimeNexus to analyze time-series data using temporal MultiLayer networks (tMLNs). *Scientific Reports*, *11*(1), 1–17. https://doi.org/10.1038/s41598-021-93128-5

Rossetti, G., bot, pyup.io, Utku Norman, dormanh, & Dorner, M. (2020). *DyNetx: Dynamic network analysis library* (Version v0.2.1). Zenodo. https://doi.org/10.5281/zenodo.3953119

SocioPatterns: A collection of contacts datasets. (2008). http://www.sociopatterns.org/datasets/

Steer, B., Arnold, N., Ba, C. T., Lambiotte, R., Yousaf, H., Jeub, L., Murariu, F., Kapoor, S., Rico, P., Chan, R., & others. (2023). Raphtory: The temporal graph engine for rust and python. *arXiv:2306.16309*. https://doi.org/10.48550/arXiv.2306.16309

Thompson, W. H., granitz, Harlalka, V., & Icandeago. (2020). *Teneto: A Python library for Temporal Network Tools* (Version 0.5.0). Zenodo. https://doi.org/10.5281/zenodo.3626827