



# Algorithm 1044: PyGenStability, a Multiscale Community Detection with Generalized Markov Stability

ALEXIS ARNAUDON and DOMINIK J. SCHINDLER, Department of Mathematics, Imperial College London, London, United Kingdom

ROBERT L. PEACH, Department of Neurology, University Hospital Würzburg, Würzburg, Germany

ADAM GOSZTOLAI, Signal Processing Laboratory (LTS2), EPFL, Lausanne, Switzerland

MAXWELL HODGES, Spotify, London, United Kingdom

MICHAEL T. SCHAUB, Computational Network Science Group, RWTH Aachen University, Aachen, Germany

MAURICIO BARAHONA, Department of Mathematics, Imperial College London, London, United Kingdom

---

We present PyGenStability, a general-use Python software package that provides a suite of analysis and visualization tools for unsupervised multiscale community detection in graphs. PyGenStability finds optimized partitions of a graph at different levels of resolution by maximizing the generalized Markov Stability quality function with the Louvain or Leiden algorithm. The package includes automatic detection of robust graph partitions and allows the flexibility to choose quality functions for weighted undirected, directed, and signed graphs and to include other user-defined quality functions.

CCS Concepts: • **Mathematics of computing** → **Graph algorithms; Solvers**; • **Networks** → **Topology analysis and generation**; • **Applied computing** → **Mathematics and statistics; Bioinformatics**;

Additional Key Words and Phrases: Multiscale community detection, unsupervised learning, graphs, network science, generalized Markov Stability, modularity, graph clustering, Louvain algorithm, Leiden algorithm, Python

---

Alexis Arnaudon and Dominik J. Schindler contributed equally to this paper.

Alexis Arnaudon, Robert L. Peach, and Mauricio Barahona acknowledge funding through the EPSRC award EP/N014529/1 supporting the EPSRC Centre for Mathematics of Precision Healthcare at Imperial College London. Robert L. Peach acknowledges funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation; Project-ID 424778381-TRR 295). Dominik J. Schindler acknowledges support from the EPSRC (PhD studentship through the Department of Mathematics at Imperial College London). Adam Gosztolai acknowledges support from an HFSP Cross-disciplinary Postdoctoral Fellowship (LT000669/2020-C).

Authors' Contact Information: Alexis Arnaudon (Corresponding author), Department of Mathematics, Imperial College London, London, United Kingdom; e-mail: alexis.arnaudon@imperial.ac.uk; Dominik J. Schindler, Department of Mathematics, Imperial College London, London, United Kingdom; e-mail: dominik.schindler19@imperial.ac.uk; Robert L. Peach, Department of Neurology, University Hospital Würzburg, Würzburg, Germany; e-mail: r.peach13@imperial.ac.uk; Adam Gosztolai, AI Institute, Medical University of Vienna, Vienna; e-mail: a.gosztolai@gmail.com; Maxwell Hodges, Spotify, London, United Kingdom; e-mail: maxwellhodges@gmail.com; Michael T. Schaub, Computational Network Science Group, RWTH Aachen University, Aachen, Germany; e-mail: schaub@cs.rwth-aachen.de; Mauricio Barahona, Department of Mathematics, Imperial College London, London, United Kingdom; e-mail: m.barahona@imperial.ac.uk.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2024 Copyright held by the owner/author(s).

ACM 1557-7295/2024/6-ART15

<https://doi.org/10.1145/3651225>

**ACM Reference format:**

Alexis Arnaudon, Dominik J. Schindler, Robert L. Peach, Adam Gosztolai, Maxwell Hodges, Michael T. Schaub, and Mauricio Barahona. 2024. Algorithm 1044: PyGenStability, a Multiscale Community Detection with Generalized Markov Stability. *ACM Trans. Math. Softw.* 50, 2, Article 15 (June 2024), 8 pages. <https://doi.org/10.1145/3651225>

**1 Introduction**

Unsupervised community detection, or graph clustering, can be traced back to early work in social network analysis in the 1950s [25] and has become a fundamental data analysis tool in the physical and life sciences, as well as in quantitative social science [10]. Various notions of communities (with associated algorithms) have been developed stemming from different mathematical concepts [22], including normalized cut [26], non-negative matrix factorization [9], or modularity maximization [11], among others. Furthermore, in many cases of theoretical and practical interest, graphs have relevant structure at multiple scales (or levels of resolution) [23]; hence, extensions that can deal with multiscale graphs have been proposed based on, e.g., the dynamics of random walks and diffusion processes on graphs [8, 15], graph signal processing [28], or discrete geometry [12]. Indeed, recent work has emphasized that, as for other problems in data clustering, a universally best algorithm for community detection cannot exist [20] and that different partitions may thus be needed to describe various aspects of the structure of a graph [22].

In this spirit, we introduce PyGenStability, a publicly available software package for multiscale community detection based on the optimization of the generalized **Markov Stability (MS)** multiscale quality function. The MS framework, which was developed in a series of papers [7, 8, 15, 16, 21, 23], exploits graph diffusion processes to uncover graph partitions at different levels of resolution and has the flexibility to accommodate different notions of graph communities through the modification of a quality function. However, MS has been missing efficient software to boost its adoption by practitioners in data science and in different academic domains. PyGenStability fills this gap and provides a versatile Python package that encompasses several useful variants of the generalized MS quality function to allow for the analysis of undirected, directed, and signed graphs, as well as including fast approximations for large graphs.

The multiscale community detection problem is defined as the following optimization problem. Given a graph  $\mathcal{G}$  with  $N$  vertices, PyGenStability finds a series of optimized graph partitions at different values of a scale parameter  $t$  by maximizing the *generalized MS* function [21]:

$$H^*(t) = \operatorname{argmax}_H Q_{gen}(t, H) := \operatorname{argmax}_H \operatorname{Tr} \left[ H^T \left( F(t) - \sum_{k=1}^m v_{2k-1} v_{2k}^T \right) H \right], \quad (1)$$

where the output is a series of  $N \times c$  indicator matrices  $H^*(t)$  describing the optimized (hard) partitions of the  $N$  nodes into  $c$  communities for different values of the scale parameter  $t$ . Here  $F(t)$  is an  $N \times N$  *node similarity matrix* that measures the similarity between the nodes of the graph as a function of  $t$ , and  $\{v_k\}_{k=1}^{2m}$  is a set of  $N$  dimensional node vector pairs that encode a *null model* of rank  $m$ . The null model provides the reference against which the quality of the partition is compared. The scale parameter  $t$ , sometimes referred to as the *Markov time* or *Markov scale*, regulates the coarseness of the partition  $H^*(t)$ , and the optimization is solved *across all scales*, i.e., for a range of values  $t > 0$  that spans from the finest to the coarsest resolution.

Our package implements constructors to design various quality matrices  $F(t)$  and null models  $\{v_k\}$ , each of which yields different notions of quality and balance for the graph partitions and ensuing communities. For example, we can use the graph heat kernel  $F(t) = \Pi \exp(-Lt)$  with

a null model of rank  $m = 1$  defined by  $v_1 = v_2 = \pi$ , where  $L$  is a graph Laplacian, the vector  $\pi$  is the stationary distribution of the associated Markov process, and  $\Pi = \text{diag}(\pi)$  [7, 16]. In this case,  $F(t) = \Pi \exp(-Lt)$  corresponds to the transition probabilities of a Markov process over time  $t > 0$ , and Equation (1) can be viewed as optimizing the partition of the graph into subgraphs where the Markov process is more likely to remain contained over time  $t$ , as compared to the expected behavior at stationarity. Therefore, this dynamic viewpoint allows for scanning across different levels of coarseness through  $t$ . In Section 2.2, we describe the implementation of various constructors for different graph types such as weighted, directed, and even signed graphs.

To carry out the combinatorial optimization of the generalized MS quality function of Equation (1), PyGenStability provides a Python wrapper around the C++ implementation of two efficient greedy algorithms: the Louvain [4] and Leiden [27] optimizers. Further, it is easy to implement other graph clustering algorithms that can be written as a maximization of a generalized function  $Q_{gen}(t, H)$  [21], making the package easily extendable. PyGenStability also includes a suite of analysis and visualization tools to process and analyze multiscale graph partitions and to facilitate the automatic detection of robust partitions at different scales [24]. As its output, PyGenStability provides a description of the graph in terms of a sequence of robust partitions  $H^*(t_i)$  at scales  $t_i$  of increasing coarseness, yet not necessarily hierarchical.

## 2 Implementation

### 2.1 Overall Organization

The Python package PyGenStability consists of four parts:

- (1) *Quality function and null model constructors*: This module inputs the node similarity matrix function  $F(t)$  and a null model described by vectors  $\{v_k\}$ . To maintain the flexibility of the package, we provide an object-oriented module to write user-defined constructors for these objects. To facilitate usage, we also provide several constructors already implemented that can be chosen by the user (see Section 2.2).
- (2) *Generalized MS maximizers*: The combinatorial optimization of the generalized MS function (Equation (1)) is carried out by interfacing with two fast algorithms, Louvain [4] or Leiden [27], both implemented in C++. The choice of the optimizer is left to the user: Louvain is widely and successfully used in many fields; Leiden is a recent refinement of Louvain, which introduces several improvements, e.g., ensuring connected communities.
- (3) *Post-processing tools*: We provide several steps to facilitate the detection of robust optimized partitions and to ease the analysis of multiscale clusterings (see Section 2.3).
- (4) *Plotting*: We provide a module to plot the multiscale clustering results, as illustrated in Figure 2.

These four components are tied together via a single, configurable entry point or can be used independently, depending on user needs.

### 2.2 Quality Function Constructors

To aid users, we have already implemented several constructors for different versions of the generalized MS quality function based on graph Laplacians. For weighted, undirected graphs, we have included [7, 8, 16] (1) MS based on the continuous-time random-walk (normalized) graph Laplacian; (2) MS based on the continuous-time combinatorial graph Laplacian; (3) linearized MS based on the normalized graph Laplacian (also referred to as ‘modularity with resolution parameter’) for a more computationally efficient analysis of larger graphs (see Figure 1). For weighted, directed graphs, we have included [16, 21] (4) MS based on the continuous-time random-walk Laplacian with teleportation; (5) linearized MS for the random-walk Laplacian with teleportation (more

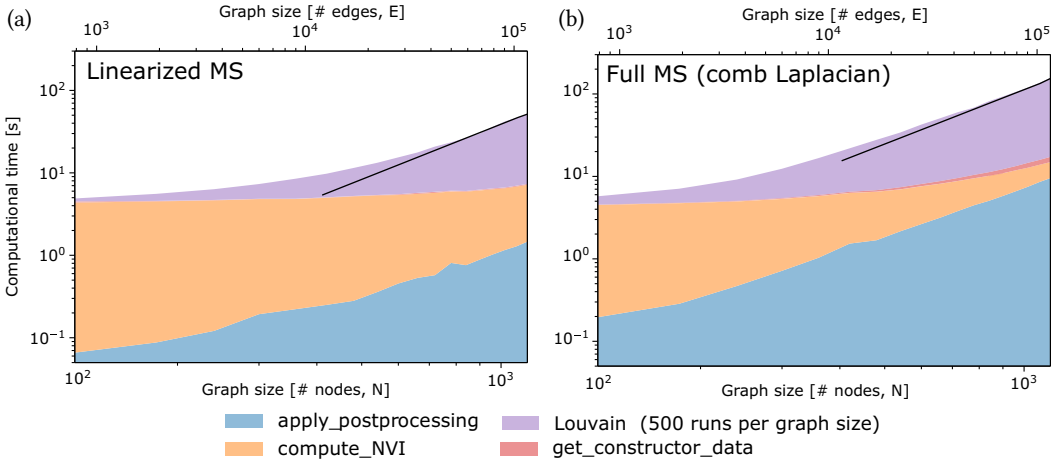


Fig. 1. Code benchmarking. To assess the computational efficiency and scalability of each component of the code, we analyzed **stochastic block model (SBM)** graphs of increasing size. An example of these graphs with  $N = 270$  nodes is shown in Figure 2. We show benchmarking results for (a) ‘modularity with resolution parameter,’ equivalent to linearized Markov Stability (MS) and (b) MS with combinatorial graph Laplacian. The computations were performed on a single CPU and involved 500 Louvain optimizations for each graph size (i.e., 50 Louvain runs computed at 10 scales  $t$ ). The cost of the ‘Louvain optimization’ and ‘post-processing’ steps increase with graph size more sharply for full MS in (b) as compared to linearized MS in (a). This is due to the decreased sparsity of the quality matrix, and the computational cost scales approximately as  $O(E)$  (black line), where  $E$  is the number of edges of the graph.

efficient for larger graphs). For weighted, signed graphs, we have included (6) MS based on the signed Laplacian as given in [21]; and (7) a version of signed modularity with resolution [13] (more efficient for larger graphs). More detailed information about these constructors can be found in our code documentation hosted on GitHub. Our object-oriented module facilitates the simple implementation of further custom constructors.

### 2.3 Post-Processing Tools

*Quantifying the robustness of partitions through the Normalized Variation of Information (NVI).* Louvain and Leiden are greedy algorithms that provide local maxima to the combinatorial optimization problem (1) without guarantees of global optimality. The optimization can thus produce different maxima depending on the starting point of the iterations. Louvain/Leiden is run a large number of times for each  $t$  starting from different random initializations to obtain an ensemble of optimized solutions.

To evaluate the consistency of this ensemble of solutions, we use the NVI [14], which measures the distance between partitions. We thus compute the average  $NVI(t)$  between all pairs of partitions (or a random subset thereof to reduce computational cost) obtained at scale  $t$ . A low value of the average  $NVI(t)$  indicates a reproducible (robust) solution for the optimization (1), suggesting a well-defined maximum and hence increased confidence in the optimal partition found.

This quantitative notion of robustness is also applied to compare the partitions obtained across scales by computing  $NVI(t, t')$ , i.e., the distance between the optimal partitions at scales  $t$  and  $t'$ . In this case, persistently low  $NVI(t, t')$  across a long stretch of  $t$  indicates that a partition (or a set of similar partitions) is found robustly across graph scales.

*Post-processing of optimal partitions.* Given the greedy nature of the Louvain/Leiden optimizers, it is possible that the optimal partition found at scale  $t'$  could in fact be a better partition for scale  $t$  than the partition found by Louvain/Leiden at  $t$ . We run a post-processing step that checks for and selects any such improved partition for scale  $t$  even if found at any other  $t'$ .

*Automated scale selection.* We aim to find relevant scales at which partitions are robust both with respect to the optimization (low  $NVI(t)$ ) and across scales (extended blocks of low  $NVI(t, t')$ ). The partitions found at such scales give a good description of the graph structure at a level of coarseness (or resolution). The selection of scales can be done by visual inspection of the result summary plot (see Figure 2) or using the automated scale selection criterion introduced by [24], which combines the robustness to the optimization and the persistence across scales.

## 2.4 Main Parameters and Default Values

To make PyGenStability easier to use for non-experts, we have set default values for several parameters (default values in parentheses). The chosen quality function is optimized over  $n\_scale$  (= 20) scales, chosen equidistantly between  $min\_scale$  (= -2.0) and  $max\_scale$  (= 0.5) on a log scale. Hence,  $min\_scale$  and  $max\_scale$  determine the minimal and maximal coarseness of the partitions, respectively, and  $n\_scale$  increases the resolution of the analysis. Operationally, we recommend starting with the default  $n\_scale$  and increasing it for more fine-grained results.

To quantify the robustness of the partitions with respect to the optimization of the quality function, an ensemble of  $n\_tries$  (= 100) solutions is computed using Louvain or Leiden, and the similarity of the solutions is estimated by computing the average pairwise  $NVI(t)$  of a random subset of  $n\_NVI$  (= 20) partitions. Increasing  $n\_tries$  leads to a better estimation of the robustness, at a computational cost since the total number of Louvain/Leiden optimizations performed is  $n\_scale \times n\_tries$ .

The scale selection follows a sequential algorithm developed by Schindler et al. [24]. To detect intervals over which partitions remain similar, we apply average pooling to  $NVI(t, t')$  with kernel size,  $kernel\_size$  (=  $0.1 \times n\_scale$ ), followed by smoothing of its diagonal with a triangular moving mean, where the smoothness is controlled by the window size,  $window\_size$  (=  $0.1 \times n\_scale$ ). This gives the curve *Block  $NVI(t)$* . Increasing  $kernel\_size$  enlarges the interval over which scales need to be persistent and increasing  $window\_size$  further smoothes out random variability across scales. We then define basins with radius  $basin\_radius$  (=  $0.01 \times n\_scale$ ) around all the local minima of *Block  $NVI(t)$* . From each basin, we select a scale, given by the solution with minimal  $NVI(t)$  within the basin. This procedure selects scales that are both persistent across  $t$  and robust to the combinatorial optimization.

## 3 Benchmarking

To assess computational efficiency as a function of graph size, we timed the core functions called during a computation with Louvain for (1) linearized MS and (2) MS with combinatorial graph Laplacian (Figure 1). We find that the rate-limiting function for larger graphs is the Louvain optimization, followed by the growing computational cost of obtaining the matrix exponential, whereas the other computations have a near-constant computational cost. Hence, for large graphs, we provide the linearized MS quality function to avoid the loss of sparsity induced by the matrix exponential.

## 4 Example and Applications

As a simple illustration of the use of the package, we provide an example of the multiscale analysis of a toy graph: a multiscale SBM with planted partitions at three scales. Figure 2 shows that PyGenStability is able to accurately recover the expected partitions at the three scales.

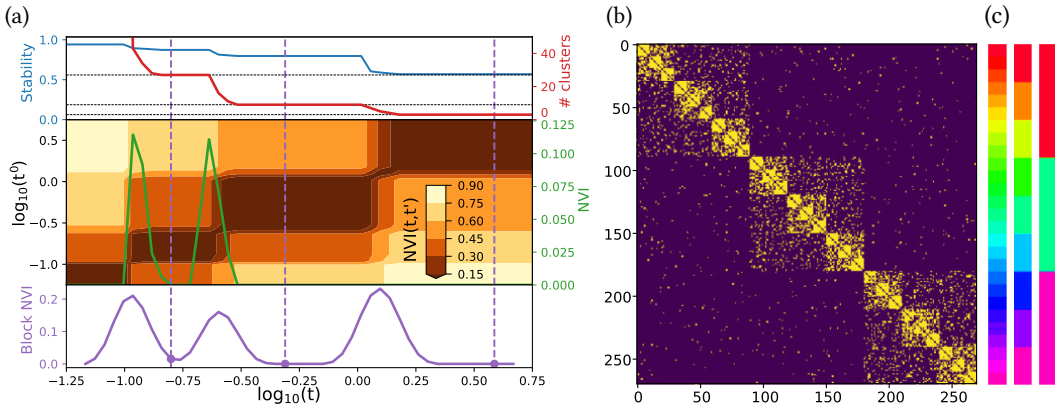


Fig. 2. Example of multiscale community detection. (a) Summary plot for a multiscale SBM graph. The top row shows the value of the optimized generalized MS function  $Q_{gen}(H^*(t))$  together with the number of communities in the optimal partition  $H^*(t)$  as a function of the scale  $t$ . The second row shows the two robustness measures for the obtained partitions:  $NVI(t)$  for each scale and  $NVI(t, t')$  across scales. The bottom row shows the automated scale selection criterion, with basins corresponding to blocks in  $NVI(t, t')$  and robust scales identified as local minima of  $NVI(t)$  within each basin (purple dots). (b) Adjacency matrix of the graph in this toy example: a multiscale SBM graph with  $N = 270$  nodes and ground truth of 3 planted scales with 27, 9, and 3 clusters. (c) The communities determined by the scale selection criterion in (a) are indicated by different colors for the three detected scales and correspond to the ground truth.

The MS framework, which is now made available through `PyGenStability`, has already been used extensively to analyze multiscale community structures in real-world graphs, also called *networks* in the literature, from diverse domains facilitating a range of applications. These include detecting functional and anatomical constituents in the directed neuronal network of *C. elegans* [2], interest communities in the Twitter network of the 2011 UK riots [3], spatial and dynamical subunits in protein structures [6, 18], hospital catchment areas in surgical admission networks [5], learning behaviors among online students [19], multiscale human mobility patterns under lockdown [24] and in hospitals [17] during COVID-19, topic modeling with semantic networks derived from free text [1], and quantifying information flow and bottlenecks using discrete network geometry [12]. Detailed illustrations and examples of applications to several synthetic and real-world networks are provided as examples in the code, including an analysis of a power grid network and protein structural graphs.

## 5 Conclusion and Outlook

The Python package `PyGenStability` is primarily designed for multiscale community detection within the MS framework but can be extended for the optimization of a range of graph clustering quality functions. `PyGenStability` allows researchers to identify robust graph partitions at different resolutions in graphs of different types and has been already applied widely to unsupervised learning tasks for real-world networks from various domains. In future work, we plan to further improve the automatic scale selection functionality, extend the range of constructors for different quality functions, and perform a quantitative comparison of the multiscale optimization using the Louvain and Leiden algorithms.

## Acknowledgment

We thank Vincent Traag for help with the implementation of the Leiden optimizer in `PyGenStability`.

## References

- [1] M. Tarik Altuncu, Erik Mayer, Sophia N. Yaliraki, and Mauricio Barahona. 2019. From free text to clusters of content in health records: An unsupervised graph partitioning approach. *Applied Network Science* 4, 2 (2019), 1–23. Retrieved from <https://appliednetsci.springeropen.com/articles/10.1007/s41109-018-0109-9>
- [2] Karol A. Bacik, Michael T. Schaub, Mariano Beguerisse-Díaz, Yazan N. Billeh, and Mauricio Barahona. 2016. Flow-based network analysis of the *Caenorhabditis elegans* connectome. *PLoS Computational Biology* 12, 8 (2016), e1005055. Retrieved from <https://dx.plos.org/10.1371/journal.pcbi.1005055>
- [3] Mariano Beguerisse-Díaz, Guillermo Garduño-Hernández, Borislav Vangelov, Sophia N. Yaliraki, and Mauricio Barahona. 2014. Interest communities and flow roles in directed networks: The Twitter network of the UK riots. *Journal of The Royal Society Interface* 11, 101 (2014), 20140940. Retrieved from <https://royalsocietypublishing.org/doi/10.1098/rsif.2014.0940>
- [4] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 10 (Oct. 2008). DOI: <https://doi.org/10.1088/1742-5468/2008/10/p10008>
- [5] Jonathan M. Clarke, Mauricio Barahona, and Ara W. Darzi. 2019. Defining hospital catchment areas using multiscale community detection: A case study for planned orthopaedic care in England. Retrieved from <https://www.biorxiv.org/content/10.1101/619692v1>
- [6] Antoine Delmotte, Edward W. Tate, Sophia N. Yaliraki, and Mauricio Barahona. 2011. Protein multi-scale organization through graph partitioning and robustness analysis: Application to the myosin–myosin light chain interaction. *Physical Biology* 8, 5 (2011), 055010.
- [7] Jean-Charles Delvenne, Michael T. Schaub, Sophia N. Yaliraki, and Mauricio Barahona. 2013. The stability of a graph partition: A dynamics-based framework for community detection. In *Dynamics on and of Complex Networks*. A. Mukherjee, M. Choudhury, F. Peruani, N. Ganguly, and B. Mitra (Eds.), Vol. 2. Springer, New York, NY 221–242. DOI: [https://doi.org/10.1007/978-1-4614-6729-8\\_11](https://doi.org/10.1007/978-1-4614-6729-8_11)
- [8] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona. 2010. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences* 107, 29 (Jul. 2010), 12755–12760. DOI: <https://doi.org/10.1073/pnas.0903215107>
- [9] Rundong Du, Da Kuang, Barry Drake, and Haesun Park. 2017. Hierarchical community detection via rank-2 symmetric nonnegative matrix factorization. *Computational Social Networks* 4, 1 (2017), 7.
- [10] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3–5 (Feb. 2010), 75–174. DOI: <https://doi.org/10.1016/j.physrep.2009.11.002>
- [11] M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12 (Jun. 2002), 7821–7826. DOI: <https://doi.org/10.1073/pnas.122653799>
- [12] Adam Gosztolai and Alexis Arnaudon. 2021. Unfolding the multiscale structure of networks with dynamical Ollivier-Ricci curvature. *Nature Communications* 12, 1 (2021), 4561. DOI: <https://doi.org/10.1038/s41467-021-24884-1>
- [13] Sergio Gómez, Pablo Jensen, and Alex Arenas. 2009. Analysis of community structure in networks of correlated data. *Physical Review E* 80, 1 (2009), 016114. Retrieved from <https://link.aps.org/doi/10.1103/PhysRevE.80.016114>
- [14] Alexander Kraskov, Harald Stögbauer, Ralph G. Andrzejak, and Peter Grassberger. 2003. Hierarchical clustering based on mutual information. arXiv:q-bio/0311039. Retrieved from <http://arxiv.org/abs/q-bio/0311039>
- [15] R. Lambiotte, J. C. Delvenne, and M. Barahona. 2008. Laplacian dynamics and multiscale modular structure in networks. arXiv:0812.1770. Retrieved from <https://doi.org/10.48550/arXiv.0812.1770> [physics.soc-ph]
- [16] Renaud Lambiotte, Jean-Charles Delvenne, and Mauricio Barahona. 2014. Random walks, Markov processes and the multiscale modular organization of complex networks. *IEEE Transactions on Network Science and Engineering* 1, 2 (Jul. 2014), 76–90. DOI: <https://doi.org/10.1109/TNSE.2015.2391998>
- [17] Ashleigh C. Myall, Robert L. Peach, Andrea Y. Weiße, Siddharth Mookerjee, Frances Davies, Alison Holmes, and Mauricio Barahona. 2021. Network memory in the movement of hospital patients carrying antimicrobial-resistant bacteria. *Applied Network Science* 6, 1 (2021), 1–23.
- [18] Robert L. Peach, Dominik Saman, Sophia N. Yaliraki, David R. Klug, Liming Ying, Keith R. Willison, and Mauricio Barahona. 2019a. Unsupervised graph-based learning predicts mutations that alter protein dynamics. bioRxiv: 847426. DOI: <https://doi.org/10.1101/847426>
- [19] Robert L. Peach, Sophia N. Yaliraki, David Lefevre, and Mauricio Barahona. 2019b. Data-driven unsupervised clustering of online learner behaviour. *npj Science of Learning* 4, 1 (2019), 1–11.
- [20] Leto Peel, Daniel B. Larremore, and Aaron Clauset. 2017. The ground truth about metadata and community detection in networks. *Science Advances* 3, 5 (May 2017), e1602548. DOI: <https://doi.org/10.1126/sciadv.1602548>
- [21] Michael T. Schaub, Jean-Charles Delvenne, Renaud Lambiotte, and Mauricio Barahona. 2019. Multiscale dynamical embeddings of complex networks. *Physical Review E* 99, 6 (2019), 062308.

- [22] Michael T. Schaub, Jean-Charles Delvenne, Martin Rosvall, and Renaud Lambiotte. 2017. The many facets of community detection in complex networks. *Applied Network Science* 2, 1 (Dec. 2017), 1–13. DOI: <https://doi.org/10.1007/s41109-017-0023-6>
- [23] Michael T. Schaub, Jean-Charles Delvenne, Sophia N. Yaliraki, and Mauricio Barahona. 2012. Markov dynamics as a zooming lens for multiscale community detection: Non clique-like communities and the field-of-view limit. *PLoS One* 7, 2 (Feb. 2012). e32210. DOI: <https://doi.org/10.1371/journal.pone.0032210>
- [24] Dominik J. Schindler, Jonathan Clarke, and Mauricio Barahona. 2023. Multiscale mobility patterns and the restriction of human movement. *Royal Society Open Science* 10, 10 (2023), 230405. Retrieved from <https://royalsocietypublishing.org/doi/10.1098/rsos.230405>
- [25] Dominik J. Schindler and Matthew Fuller. 2023. Community as a vague operator: Epistemological questions for a critical heuristics of community detection algorithms. *Computational Culture* 9 (2023). Retrieved from <http://computationalculture.net/community-as-vague-operator/>
- [26] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.
- [27] V. A. Traag, L. Waltman, and N. J. van Eck. 2019. From Louvain to Leiden: Guaranteeing well-connected communities. *Scientific Reports* 9, 1 (Mar. 2019), 5233. DOI: <https://doi.org/10.1038/s41598-019-41695-z>
- [28] Nicolas Tremblay and Pierre Borgnat. 2014. Graph wavelets for multiscale community mining. *IEEE Transactions on Signal Processing* 62, 20 (2014), 5227–5239. DOI: <https://doi.org/10.1109/TSP.2014.2345355>

Received 5 May 2023; revised 6 November 2023; accepted 23 February 2024