pymnet: A Python Library for Multilayer Networks

Tarmo Nurmi [1], **Arash Badie-Modiri** [2], **Corinna Coupette** [1,3,4], and **Mikko Kivelä** [1]

1 Aalto University, Finland **2** Central European University, Austria **3** KTH Royal Institute of Technology, Sweden **4** Max Planck Institute for Informatics, Germany

Summary

Many complex systems can be readily modeled as networks and represented as graphs. Such systems include social interactions, transport infrastructures, biological pathways, brains, ecosystems, and many more. A major advantage of representing complex systems as graphs is that the same graph tools and methods can be applied in a wide variety of domains. However, the graph representation has its limitations: many systems contain nodes with multidimensional features, interactions of various types, different levels of hierarchy, or multiple modalities, which deserve to be modeled but cannot be described by simple graphs. Multilayer networks (Kivelä et al., 2014) generalize graphs to capture the rich network data often associated with complex systems, allowing us to study a broad range of phenomena using the same representations, tools, and methods. With pymnet, we introduce a Python package that provides the essential data structures and computational tools for multilayer-network analysis. As highlights, the library offers efficient and scalable implementations for sparse multilayer networks and multiplex networks, integration with bliss to analyze multilayer-network isomorphisms and automorphisms, and versatile methods for multilayer-network visualization.

Statement of Need

pymnet is a Python package for creating, analyzing, and visualizing multilayer networks. It is designed for network scientists with an easy-to-use yet flexible interface, featuring, inter alia, representations of a very general class of multilayer networks, structural metrics of multilayer networks (e.g., clustering coefficients (Cozzo et al., 2015) and graphlet analysis (Sallmen et al., 2022)), multilayer-network transforms, multilayer-network isomorphisms and automorphisms (Kivelä & Porter, 2017) (with bliss (Junttila & Kaski, 2007, 2011)), and random multilayer-network models.

Different kinds of multilayer network data are becoming increasingly available, but our computational tools for handling such data lag behind. Python is a popular programming language for network scientists and data scientists, and pymnet addresses the need for a feature-rich multilayer-networks package in the Python language that is actively maintained.

pymnet implements the general multilayer-network framework described by Kivelä et al. (2014). A *multilayer network* M is defined by $M = (V_M, E_M, V, \mathbf{L})$, where the sequence $\mathbf{L} = (L_a)_{a=1}^d$ defines sets L_a of *elementary layers*, the set V defines the *nodes* of the network, the *node-layers* are $V_M \subseteq V \times L_1 \times ... \times L_d$, and the *edges* $E_M \subseteq V_M \times V_M$ are defined between node-layers. Put simply, a node-layer is an association of a node $v \in V$ with a layer $\in L_1 \times ... \times L_d$ of dimensionality d, nodes can exist on an arbitrary number of layers, and edges can connect node-layers within layers and across arbitrary pairs of layers, which can differ in an arbitrary number of dimensions. The dimensions $1, 2, ..., d$ are called the *aspects* of the network.

Beyond the general multilayer-network framework described by Kivelä et al. ([2014](#)), pymnet also includes a specialized implementation of *multiplex* networks, a common subtype of multilayer networks. In multiplex networks, edges across layers (interlayer edges) only occur between a node and its counterpart(s) on the other layers. The advantages of this specialization include, for example, automatic lazy evaluation of interlayer-coupling edges.

Main Features

pymnet's main data structure is `MultilayerNetwork`, which is implemented as a dictionary of dictionaries with a tensor-like interface, where each key represents a node, and each value is another dictionary containing information about the neighbors of each node, with the neighbors as keys and the weights of their incident edges as values. This structure ensures that adding nodes, removing nodes, querying for existence of edges, or querying for edge weights all have constant average time complexity, and iterating over the neighbors of a node is linear in the number of nodes. Furthermore, the memory requirements are in $O(|V| + |L| + |E|)$ and typically dominated by the number of edges in the network.

To represent multiplex networks, pymnet offers `MultiplexNetwork`, which exploits the special structure of interlayer edges for efficiency, storing intralayer edges separately for each layer and only generating interlayer edges according to the applicable interlayer-coupling rules when they are explicitly needed. This ensures that we can always iterate over intralayer edges in linear time, and that interlayer edges only require constant memory (i.e., the memory to store the rule to generate them).

pymnet contains submodules for advanced analysis of multilayer networks. One example is graphlet-degree analysis, a powerful tool for investigating the structure of graphs that has been generalized to multilayer networks ([Sallmen et al., 2022](#)) and is implemented in pymnet for single-aspect multiplex networks. A graphlet is an isomorphism class of (connected) induced subgraphs that are typically small. pymnet can generate all graphlets of a specified size, i.e., all isomorphic multiplex networks with a user-specified number of nodes and layers (coming from a user-defined set of layers), user-defined interlayer couplings, and a user-defined type of multilayer isomorphism. From the graphlets, pymnet can compute the automorphism orbits of nodes or node-layers in the graphlets, with a user-specified type of isomorphism. For example, we can use pymnet to enumerate and visualize all automorphism orbits of nodes in single-aspect multiplex graphlets with two or three nodes and two layers under node-layer isomorphism. The results are depicted in [Figure 1](#).
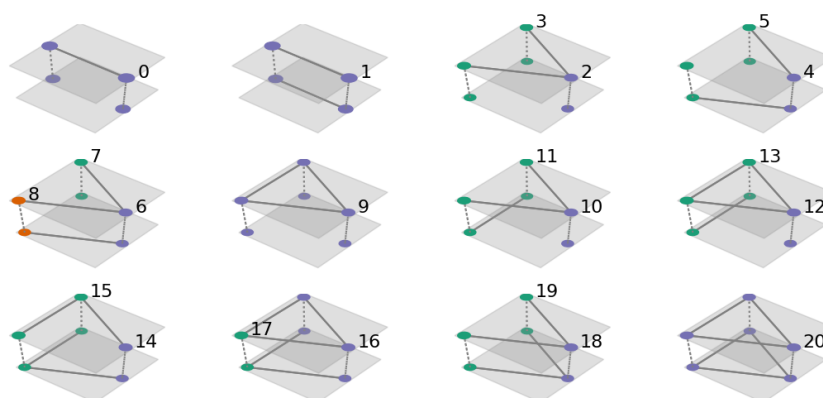

Figure 1: Using pymnet to enumerate and visualize automorphism orbits of nodes in single-aspect multiplex graphlets under node-layer isomorphism. [Visualization script](#) adapted from Sallmen et al. ([2022](#)).

Other amenities shipped with pymnet include graph generators for generalizations of popular random-graph models to multilayer networks (e.g., Erdős-Rényi models and configuration models) as well as utilities for multilayer-network visualization.

Real-World Example

As an example of how pymnet can be used to analyze real-world data, we explore data documenting the *legal* international trade in endangered species, which is crucial for monitoring progress toward the United Nations' Sustainable Development Goal 15. The Convention on International Trade in Endangered Species of Wild Fauna and Flora (CITES) regulates this trade, and the CITES database provides highly granular trade data, accompanied by great documentation.

To keep matters simple, we analyze an excerpt from the CITES database, investigating the trade in endangered species in two different *years* (2010 and 2020), for two different *trade terms* (live specimens and meat), and restricting ourselves to species taken from the *wild* and traded for *commercial* purposes. In the multilayer-network model representing this data, our nodes are countries or territories importing and exporting endangered species (represented by their ISO-2 codes), our aspects are years (two elementary layers: 2010 and 2020) and trade terms (two elementary layers: "live" and "meat"), and our edges represent bilateral trades, measured in the number of specimens ("live") or the number of kilograms ("meat") traded. More details on the data provenance and our preprocessing steps are available in the pymnet repository.

In Figure 2, we visualize the data with pymnet, only drawing undirected edges representing aggregate trade volumes of at least one million specimens ("live") or one thousand kilograms ("meat"), scaling nodes by degree and highlighting OECD countries in red, as well as mapping trade volume to edge widths and edge colors. Between 2010 and 2020, we observe roughly equal levels of trade in live specimens, but a pronounced increase in both trade volume and diversity of trade partners for trade in meat.
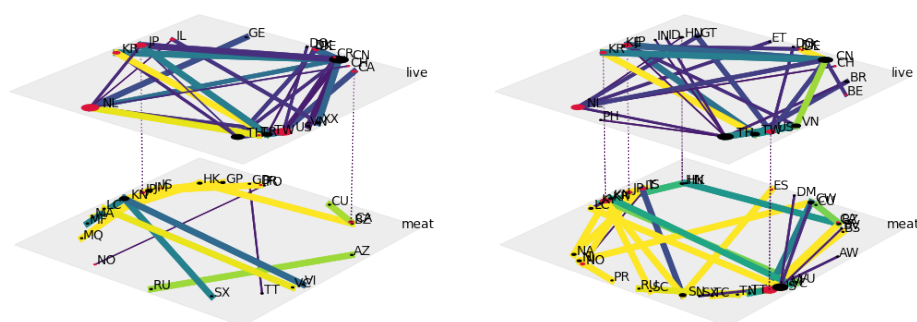

Figure 2: Trade in live specimens (top) and meat (bottom) of endangered species in the years 2010 (left) and 2020 (right).

Installation and Usage

Detailed installation and usage instructions, including tutorials demonstrating pymnet's main functionality, can be found in the pymnet documentation.

Related Packages

pymnet extends the popular networkx package developed for single-layer network analysis such that `networkx` functions can be applied to the individual layers of a multilayer network. These functions are automatically wrapped for use in `pymnet`, which has the benefit of automatically including new functionality added to `networkx`. To solve multilayer-network isomorphisms, pymnet uses a backend package, which can be either networkx (limited functionality) or bliss (Junttila & Kaski, 2007, 2011) (full functionality).

For multilayer-network visualization, pymnet uses matplotlib as its default backend, enabling users to exert low-level control over figure aesthetics to produce publication-quality plots. Support for interactive figures is provided via JavaScript and D3.js as a backend.

To the best of our knowledge, the only other libraries offering tools to work with multilayer networks in Python are multiNetX and py3plex, both of which appeared after pymnet. Support for working with multilayer networks in Julia is offered by MultilayerGraphs.jl from Moroni & Monticone (2023), who also compile a list of R packages offering similar functionality.

Projects Using pymnet

pymnet has been used in numerous scientific publications across different disciplines, such as Kivelä et al. (2014), Cozzo et al. (2015) Kivelä & Porter (2017), Danchev et al. (2019), Rio-Chanona et al. (2020), Zhou et al. (2020), Baek et al. (2021), Bergermann & Stoll (2021), Sallmen et al. (2022), and Nurmi & Kivelä (2023).

Acknowledgments

References

Baek, E. C., Porter, M. A., & Parkinson, C. (2021). Social network analysis for social neuroscientists. *Social Cognitive and Affective Neuroscience*, *16*(8), 883–901. https://doi.org/10.1093/scan/nsaa069

Bergermann, K., & Stoll, M. (2021). Orientations and matrix function-based centralities in multiplex network analysis of urban public transport. *Applied Network Science*, *6*, 1–33. https://doi.org/10.1007/s41109-021-00429-9

Cozzo, E., Kivelä, M., De Domenico, M., Solé-Ribalta, A., Arenas, A., Gómez, S., Porter, M. A., & Moreno, Y. (2015). Structure of triadic relations in multiplex networks. *New Journal of Physics*, *17*(7), 073029. https://doi.org/10.1088/1367-2630/17/7/073029

Danchev, V., Rzhetsky, A., & Evans, J. A. (2019). Centralized scientific communities are less likely to generate replicable results. *Elife*, *8*, e43094. https://doi.org/10.7554/eLife.43094

Junttila, T., & Kaski, P. (2007). Engineering an efficient canonical labeling tool for large and sparse graphs. In *2007 proceedings of the ninth workshop on algorithm engineering and experiments (ALENEX)* (pp. 135–149). Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611972870.13

Junttila, T., & Kaski, P. (2011). Conflict propagation and component recursion for canonical labeling. In *Lecture notes in computer science* (pp. 151--162). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19754-3_16

Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y., & Porter, M. A. (2014). Multilayer networks. *Journal of Complex Networks*, *2*(3), 203–271. https://doi.org/10.1093/comnet/cnu016

Kivelä, M., & Porter, M. A. (2017). Isomorphisms in multilayer networks. *IEEE Transactions on Network Science and Engineering*, *5*(3), 198–211. https://doi.org/10.1109/TNSE.2017.2753963

Moroni, C., & Monticone, P. (2023). MultilayerGraphs.jl: Multilayer network science in Julia. *Journal of Open Source Software*, *8*(83), 5116. https://doi.org/10.21105/joss.05116

Nurmi, T., & Kivelä, M. (2023). *Subnetwork enumeration algorithms for multilayer networks*. https://doi.org/10.48550/arXiv.2308.00083

Rio-Chanona, R. M. del, Korniyenko, Y., Patnam, M., & Porter, M. A. (2020). The multiplex nature of global financial contagions. *Applied Network Science*, *5*, 1–23. https://doi.org/10.1007/s41109-020-00301-2

Sallmen, S., Nurmi, T., & Kivelä, M. (2022). Graphlets in multilayer networks. *Journal of Complex Networks*, *10*(2), cnac005. https://doi.org/10.1093/comnet/cnac005

Zhou, Y., Li, Z., Liu, Y., & Deng, F. (2020). Network proximity and communities in innovation clusters across knowledge, business, and geography: Evidence from China. *IEEE Transactions on Engineering Management*, *68*(5), 1388–1397. https://doi.org/10.1109/TEM.2020.3032160