MultilayerGraphs.jl: Multilayer Network Science in Julia

Claudio Moroni 1,2* and Pietro Monticone 1,2*

1 University of Turin, Italy 2 Interdisciplinary Physics Team, Italy * These authors contributed equally.

Summary

MultilayerGraphs.jl is a Julia package for the creation, manipulation, and analysis of the structure, dynamics, and functions of multilayer graphs. A multilayer graph consists of multiple subgraphs called layers, which can be interconnected through bipartite graphs called interlayers, composed of the vertex sets of two different layers and the edges between them. The vertices in each layer represent a single set of nodes, although not all nodes have to be represented in every layer. Formally, a multilayer graph can be defined as a triple \( G = (V, E, L) \), where:

- \( V \) is the set of vertices;
- \( E \) is the set of edges, pairs of nodes \((u, v)\) representing a connection, relationship or interaction between the nodes \( u \) and \( v \); and
- \( L \) is a set of layers, which are subsets of \( V \) and \( E \) encoding the nodes and edges within each layer.

Each layer \( \ell \) in \( L \) is a tuple \( (V_\ell, E_\ell) \), where \( V_\ell \) is a subset of \( V \) that represents the vertices within that layer, and \( E_\ell \) is a subset of \( E \) that represents the edges within that layer.

MultilayerGraphs.jl is an integral part of the JuliaGraphs ecosystem extending Graphs.jl (Fairbanks et al., 2021) so all the methods and metrics exported by Graphs.jl work for multilayer graphs, but due to the special nature of multilayer graphs, the package features a peculiar implementation that maps a standard integer-labelled vertex representation to a more user-friendly framework exporting all the objects an experienced practitioner would expect such as nodes, vertices, layers, interlayers, etc.

MultilayerGraphs.jl features specific methods and metrics including the global clustering coefficient, the overlay clustering coefficient, the multilayer eigenvector centrality, the multilayer modularity, and the Von Neumann entropy.

Finally, MultilayerGraphs.jl has been integrated within the JuliaDynamics ecosystem so that any Multilayer(Di)Graph can be utilised as an argument to the GraphSpace constructor in Agents.jl (Datseris et al., 2022).

Figure 1: Logo of MultilayerGraphs.jl.
Statement of Need

Several theoretical frameworks have been proposed to formally subsume all instances of multilayer graphs (Aleta & Moreno, 2019; Artime et al., 2022; Bianconi, 2018; Boccaletti et al., 2014; Cozzo et al., 2018; M. D. Domenico et al., 2013; M. D. Domenico, 2022; Kivela et al., 2014; Lee et al., 2015).

Multilayer graphs have been adopted to model the structure and dynamics of a wide spectrum of high-dimensional, non-linear, multi-scale, time-dependent complex systems including physical, chemical, biological, neuronal, socio-technical, epidemiological, ecological, and economic networks (Aleta et al., 2020, 2022; Amato et al., 2017; Arruda et al., 2017; Azimi-Tafreshi, 2016; Baggio et al., 2016; Buldú & Porter, 2018; Cozzo et al., 2013; M. D. Domenico et al., 2016; M. D. Domenico, 2017; Estrada & Gómez-Gardeñes, 2014; Gosak et al., 2018; Granell et al., 2013; Lim et al., 2019; Mangioni et al., 2020; Massaro & Bagnoli, 2014; Pilosof et al., 2017; Soriano-Paños et al., 2018; Timóteo et al., 2018).

To the best of our knowledge there currently are no software packages dedicated to the creation, manipulation, and analysis of multilayer graphs implemented in the Julia language (Bezanson et al., 2017), apart from MultilayerGraphs.jl itself (Moroni & Monticone, 2022).

![Figure 2: Diagram illustrating a multi-scale multilayer graph applied to model social systems from the molecular to the population level (De Domenico, 2022).](image-url)
Main Features

The two main data structures are collections of layers connected through interlayers called MultilayerGraph and MultilayerDiGraph.

The vertices of a multilayer graph are representations of one set of distinct objects called Nodes. Each layer may represent all the node set or just a subset of it. The vertices of Multilayer(Di)Graph are implemented via the MultilayerVertex custom type. Each MultilayerVertex encodes information about the node it represents, the layer it belongs to and its metadata.

Both the intra-layer and inter-layer edges are embedded in the MultilayerEdge struct, whose arguments are the two connected multilayer vertices, the edge weight and its metadata. It’s important to highlight that Multilayer(Di)Graphs are weighted and able to store metadata by default (i.e., they have been assigned the IsWeighted and IsMeta traits from SimpleTraits.jl).

The layers are implemented via the Layer struct composed of an underlying graph and a mapping from its integer-labelled vertices to the collection of MultilayerVertices the layer represents. Interlayers are similarly implemented via the Interlayer mutable struct, and they are generally constructed by providing the two connected layers, the (multilayer) edge list between them, and a graph. This usage of underlying graphs allows for an easier debugging procedure during construction and a more intuitive analysis afterwards, allowing the package to leverage all the features of the JuliaGraphs ecosystem so that it can be effectively considered as a real proving ground of its internal consistency.

The Multilayer(Di)Graph structs are weighted and endowed with the functionality to store both vertex-level and edge-level metadata by default, so that at any moment the user may add or remove a Layer or specify an InterLayer and since different layers and interlayers could be better represented by graphs that are weighted or unweighted and with or without metadata, it was crucial for us to provide the most general and adaptable structure. A Multilayer(Di)Graph is instantiated by providing the ordered list of layers and the list of interlayers to the constructor. The latter are automatically specified, so there is no need to instantiate all of them.

Alternatively, it is possible to construct a Multilayer(Di)Graph making use of a graph generator-like signature allowing the user to set the degree distribution or the degree sequence and employs graph realisation methods such as the Havel-Hakimi algorithm for undirected graphs (Hakimi, 1962) and the Kleitman-Wang algorithm for directed ones (Kleitman & Wang, 1973).

Multilayer(Di)Graphs structure may be represented via dedicated WeightTensor, MetadataTensor, and SupraWeightMatrix structs, all of which support indexing with MultilayerVertices. Once a Multilayer(Di)Graph has been instantiated, its layers and interlayers can be accessed as their properties.

For a more comprehensive exploration of the package features and functionalities we strongly recommend consulting the package README and documentation.

Installation and Usage

We invite the user to read the documented example and run the associated script we have designed to synthetically illustrate:

- how to install the package;
- how to define layers and interlayers with a variety of constructors and underlying graphs;
- how to construct a directed multilayer graph with those layers and interlayers;
- how to add nodes, vertices and edges to the multilayer graph; and
- how to compute some multilayer metrics as defined in M. D. Domenico et al. (2013).
Related Packages

R

Here is a list of related software packages implemented in the R language:

- **muxViz** implements functions to perform multilayer correlation analysis, multilayer centrality analysis, multilayer community structure detection, multilayer structural reducibility, multilayer motifs analysis, and utilities to statically and dynamically visualise multilayer graphs (D. Domenico et al., 2014);
- **multinet** implements functions to import, export, create, and manipulate multilayer graphs, several state-of-the-art multiplex graph analysis algorithms for centrality measures, layer comparison, community detection, and visualization (Magnani et al., 2021);
- **mully** implements functions to import, export, create, manipulate, and merge multilayer graphs and utilities to visualise multilayer graphs in 2D and 3D (Hammoud & Kramer, 2018); and
- **multinets** implements functions to import, export, create, and manipulate multilayer graphs and utilities to visualise multilayer graphs (Lazega et al., 2008).

Python

Here is a list of related software packages implemented in the Python language:

- **MultiNetX** implements methods to create undirected networks with weighted or unweighted links, to analyse the spectral properties of adjacency or Laplacian matrices and to visualise multilayer graphs and dynamical processes by coloring the nodes and links accordingly;
- **PyMNet** implements data structures for multilayer graphs and multiplex graphs, methods to import, export, create, and manipulate multilayer graphs, and for the rule-based generation and lazy-evaluation of coupling edges and utilities to visualise multilayer graphs (Kivela et al., 2014).

Acknowledgements

This open-source research software project received no financial support.

References


