




# MultilayerGraphs.jl: A Julia package for the creation, manipulation and analysis of the structure, dynamics and functions of multilayer graphs

Claudio Moroni <sup>1,2\*</sup> and Pietro Monticone <sup>1,2\*</sup>

<sup>1</sup> University of Turin, Italy <sup>2</sup> Interdisciplinary Physics Team, Italy \* These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))

## Summary

**MultilayerGraphs.jl** is a Julia package for the creation, manipulation and analysis of the structure, dynamics and functions of multilayer graphs extending **Graphs.jl** ([Fairbanks et al., 2021](#)) and fully integrating with the [JuliaGraphs](#) ecosystem.

A multilayer graph is a graph consisting of multiple standard 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  $v$ ;
- $L$  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.

[A FEW WORDS ABOUT THE MAIN FEATURES, POSSIBLY EXTRACTED FROM TUTORIAL / README]

## Statement of Need

Multiple theoretical frameworks have been proposed to formally incorporate 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](#)).

Multilayer graphs have been adopted to model the structure and dynamics of a wide spectrum of high-dimensional and heterogeneous complex systems, including physical, chemical, biological, neuronal, socio-technical, ecological and economic networks ([Baggio et al., 2016](#); [Buldú & Porter, 2018](#); [Dickison et al., 2016](#); [Lazega & Snijders, 2016](#); [Lim et al., 2019](#); [Timóteo et al., 2018](#)).

We have chosen the [Julia language](#) for this software package because it is a modern, open-source, high-level, high-performance dynamic language for technical computing ([Bezanson et al., 2017](#)). At the best of our knowledge there are currently no software packages dedicated to

the creation, manipulation and analysis of multilayer graphs implemented in the Julia language apart from MultilayerGraphs.jl itself (Moroni & Monticone, 2022).

## Main Features

- Main structs
- Different formalisms
- Main methods and metrics
- Extension of Graphs.jl (Fairbanks et al., 2021), fully integrated within the JuliaGraphs ecosystem
- Integration with Agents.jl (Datseris et al., 2022), fully integrated within the JuliaDynamics ecosystem

Although being part of the Graphs.jl's ecosystem, due to the special nature of multilayer graphs this package features a peculiar implementation that maps a standard integer-labelled vertex representation to a more user friendly framework that exports all the objects a practitioner would expect (Nodes, MultilayerVertexs, Layers, Interlayers, etc). The details are briefly described hereafter. The package revolves around two data structures, MultilayerGraph and MultilayerDiGraph. As said above, they are collection of layers whose couplings form the edge sets of the so-called interlayers. The vertices of a multilayer graph are representations of one set of distinct objects named Nodes. Each layer may represent all or just part of such set. The vertices of Multilayer(Di)Graph are implemented via the MultilayerVertex custom type. Each MultilayerVertex carries information about the node it represents, the layer it belongs to and its metadata. Edges, both intra- and inter-layer, are embodied in the MultilayerEdge struct, whose fields are the two MultilayerVertexs involved, the edge weight and its metadata. Note that Multilayer(Di)Graphs are weighted and able to carry metadata by default (i.e. they are given the IsWeighted and IsMeta traits from SimpleTraits.jl). Layers are implemented via the Layer struct, which is constituted by an underlying graph from the Graphs.jl ecosystem and a mapping from its integer-labelled vertices to the collection of MultilayerVertexs the layer represents. Interlayers are similarly implemented via the Interlayer mutable struct, and they are generally constructed by providing the two Layers involved, the (multilayer) edge list between them and an underlying graph. This usage of underlying graphs allows for easy debugging during construction and more intuitive analysis afterwards. It also allows the package to leverage all the features of the ecosystem, and acts as a proving ground of its consistency and coherence. Now we may understand why Multilayer(Di)Graph are weighted and able to carry both vertex and edge-level metadata by default: since they are designed so that at any moment the user may add or remove a Layer or specify an Interlayer, and since it could be that different layers and interlayers are better substantiated by graphs that are weighted or unweighted and with or without metadata, it was necessary to provide a structure capable to adapt to the most general scenario. As specified in the Future Developments section of the package README, future enhancements may provide more stringent multilayer graphs data structures, by restricting to specific traits, types and/or special cases defined in the literature. A Multilayer(Di)Graph is instantiated by providing to the constructor the ordered list of layers and the list of interlayers. The latter are automatically specified, so there is no need to instantiate all of them. Another way of constructing a Multilayer(Di)Graph uses a configuration model-like signature: it allows to select the degree distribution or the degree sequence (indegree and outdegree distributions or sequences may be provided separately for the MultilayerDiGraph) and uses the Havel-Hakimi algorithm from {Hakimi (1962)}(havelhakimi?) (or {Kleitman and Wang (1973)}[Kleitman1973] for the directed MultilayerDiGraph). Please note that, although inspired from BIANCONI??, this is not a complete implementation of a multilayer configuration model: it lacks the capability to specify a different distributions for different groups of layers and/or interlayers (aspects). Once specified, the full API of Graphs.jl works on Multilayer(Di)Graphs as they were ordinary extensions of the ecosystem. Moreover, multilayer-specific methods or implementations thereof have been developed, mainly drawing from {De Domenico et al,

2013}(DeDomenico?). They include: - Global Clustering Coefficient - Overlay Clustering Coefficient - (Multilayer) Eigenvector Centrality - (Multilayer) Modularity - Von Neumann Entropy

Multilayer(Di)Graphs structure may be represented via dedicated WeightTensor, MetadataTensor and SupraWeightMatrix structs, all of which support indexing with MultilayerVertexs.

Once a Multilayer(Di)Graph has been instantiated, its layers and interlayers may be accessed as they where its properties. In order to simplify the code and improve performance, Layers and Interlayerss are not fully stored within Multilayer(Di)Graphs, only enough information to reconstruct them when accessed as properties is saved, in the form of LayerDescriptor and InterlayerDescriptors.

## Installation and Usage

To install MultilayerGraphs.jl it's sufficient to activate the pkg mode by pressing ] in the Julia REPL and then run the following command:

```
pkg> add MultilayerGraphs
```

[HERE WE SHOULD INSERT A FEW LINES OF CODE SHOWACASING THE MAIN FEATURES WRITTEN ABOVE]

In the package documentation you can find a comprehensive [tutorial](#) that illustrates all its main features and functionalities.

## Related Packages

### R

Here is a list of software packages for the creation, manipulation, analysis and visualisation of multilayer graphs 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);
- [multinets](#) implements functions to import/export, create, manipulate multilayer graphs and utilities to visualise multilayer graphs (Lazega et al., 2008).

### Python

Here is a list of software packages for the creation, manipulation, analysis and visualisation of multilayer graphs 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;

- 130     ▪ [PyMNet](#) implements data structures for multilayer graphs and multiplex graphs, methods  
131       to import/export, create, manipulate multilayer graphs and for the rule-based generation  
132       and lazy-evaluation of coupling edges and utilities to visualise multilayer graphs ([Kivela](#)  
133       et al., 2014).

## Acknowledgements

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

## References

- 137 Aleta, A., & Moreno, Y. (2019). Multilayer networks in a nutshell. *Annual Re-*  
138 *view of Condensed Matter Physics*, 10(1), 45–62. [https://doi.org/10.1146/](https://doi.org/10.1146/annurev-conmatphys-031218-013259)  
139 [annurev-conmatphys-031218-013259](https://doi.org/10.1146/annurev-conmatphys-031218-013259)
- 140 Artime, O., Benigni, B., Bertagnolli, G., dAndrea, V., Gallotti, R., Ghavasieh, A., Raimondo,  
141 S., & Domenico, M. D. (2022). *Multilayer network science*. Cambridge University Press.  
142 <https://doi.org/10.1017/9781009085809>
- 143 Baggio, J. A., BurnSilver, S. B., Arenas, A., Magdanz, J. S., Kofinas, G. P., & Domenico, M.  
144 D. (2016). Multiplex social ecological network analysis reveals how social changes affect  
145 community robustness more than resource depletion. *Proceedings of the National Academy*  
146 *of Sciences*, 113(48), 13708–13713. <https://doi.org/10.1073/pnas.1604401113>
- 147 Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to  
148 numerical computing. *SIAM Review*, 59(1), 65–98. <https://doi.org/10.1137/141000671>
- 149 Bianconi, G. (2018). *Multilayer networks*. Oxford University Press. [https://doi.org/10.1093/](https://doi.org/10.1093/oso/9780198753919.001.0001)  
150 [oso/9780198753919.001.0001](https://doi.org/10.1093/oso/9780198753919.001.0001)
- 151 Boccaletti, S., Bianconi, G., Criado, R., Genio, C. I. del, Gómez-Gardeñes, J., Romance, M.,  
152 Sendiña-Nadal, I., Wang, Z., & Zanin, M. (2014). The structure and dynamics of multilayer  
153 networks. *Physics Reports*, 544(1), 1–122. <https://doi.org/10.1016/j.physrep.2014.07.001>
- 154 Buldú, J. M., & Porter, M. A. (2018). Frequency-based brain networks: From a multiplex  
155 framework to a full multilayer description. *Network Neuroscience*, 2(4), 418–441. [https://doi.org/10.1162/netn\\_a\\_00033](https://doi.org/10.1162/netn_a_00033)
- 157 Cozzo, E., Arruda, G. F. de, Rodrigues, F. A., & Moreno, Y. (2018). *Multiplex networks*.  
158 Springer International Publishing. <https://doi.org/10.1007/978-3-319-92255-3>
- 159 Datseris, G., Vahdati, A. R., & DuBois, T. C. (2022). Agents.jl: A performant and  
160 feature-full agent-based modeling software of minimal code complexity. *SIMULATION*,  
161 003754972110688. <https://doi.org/10.1177/00375497211068820>
- 162 Dickison, M. E., Magnani, M., & Rossi, L. (2016). *Multilayer social networks*. Cambridge  
163 University Press. <https://doi.org/10.1017/cbo9781139941907>
- 164 Domenico, D., Porter, & Arenas. (2014). MuxViz: A tool for multilayer analysis and  
165 visualization of networks. *Journal of Complex Networks*, 3(2), 159–176. <https://doi.org/10.1093/comnet/cnu038>
- 167 Domenico, M. D. (2022). *Multilayer networks: Analysis and visualization*. Springer International  
168 Publishing. <https://doi.org/10.1007/978-3-030-75718-2>
- 169 Domenico, M. D., Solé-Ribalta, A., Cozzo, E., Kivela, M., Moreno, Y., Porter, M. A., Gómez,  
170 S., & Arenas, A. (2013). Mathematical formulation of multilayer networks. *Physical Review*  
171 *X*, 3(4). <https://doi.org/10.1103/physrevx.3.041022>

- 172 Fairbanks, J., Besançon, M., Simon, S., Hoffiman, J., Eubank, N., & Karpinski, S. (2021).  
173 *JuliaGraphs/graphs.jl: An optimized graphs package for the julia programming language*.  
174 <https://github.com/JuliaGraphs/Graphs.jl/>
- 175 Hammoud, Z., & Kramer, F. (2018). Mully: An r package to create, modify and visualize  
176 multilayered graphs. *Genes*, 9(11), 519. <https://doi.org/10.3390/genes9110519>
- 177 Kivela, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y., & Porter, M. A. (2014).  
178 Multilayer networks. *Journal of Complex Networks*, 2(3), 203–271. <https://doi.org/10.1093/comnet/cnu016>  
179
- 180 Lazega, E., Jourda, M.-T., Mounier, L., & Stofer, R. (2008). Catching up with big fish in  
181 the big pond? Multi-level network analysis through linked design. *Social Networks*, 30(2),  
182 159–176. <https://doi.org/10.1016/j.socnet.2008.02.001>
- 183 Lazega, E., & Snijders, T. A. B. (Eds.). (2016). *Multilevel network analysis for the social*  
184 *sciences*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-24520-1>
- 185 Lim, S., Radicchi, F., Heuvel, M. P. van den, & Sporns, O. (2019). Discordant attributes of  
186 structural and functional brain connectivity in a two-layer multiplex network. *Scientific*  
187 *Reports*, 9(1). <https://doi.org/10.1038/s41598-019-39243-w>
- 188 Magnani, M., Rossi, L., & Vega, D. (2021). Analysis of multiplex social networks with r.  
189 *Journal of Statistical Software*, 98(8). <https://doi.org/10.18637/jss.v098.i08>
- 190 Moroni, C., & Monticone, P. (2022). *MultilayerGraphs.jl: A julia package for the creation,*  
191 *manipulation and analysis of the structure, dynamics and functions of multilayer graphs*.  
192 University of Turin (UniTO); Interdisciplinary Physics Team (InPhyT). <https://doi.org/10.5281/zenodo.7009172>  
193
- 194 Timóteo, S., Correia, M., Rodríguez-Echeverría, S., Freitas, H., & Heleno, R. (2018). Multilayer  
195 networks reveal the spatial structure of seed-dispersal interactions across the great rift  
196 landscapes. *Nature Communications*, 9(1). <https://doi.org/10.1038/s41467-017-02658-y>