The Journal of Open Source Software

IGraph/M: graph theory and network analysis for Mathematica

Szabolcs Horvát [1,2¶], **Jakub Podkalicki**[3], **Gábor Csárdi** [4], **Tamás Nepusz** [5], **Vincent Traag** [6], **Fabio Zanini** [7], **and Daniel Noom**[8]

1 Max Planck Institute for Cell Biology and Genetics, Dresden, Germany
2 Center for Systems Biology Dresden, Dresden, Germany
3 Independent Developer, Poland
4 RStudio
5 Independent Developer, Hungary
6 Centre for Science and Technology Studies, Leiden University, Leiden, Netherlands
7 Lowy Cancer Research Centre, University of New South Wales, Kensington, NSW, Australia
8 Independent Developer, Netherlands
¶ Corresponding author

DOI: 10.21105/joss.04899

Software

- Review
- Repository
- Archive

Editor: Prashant K Jha
Reviewers:

- @leotrs
- @pitsianis

Submitted: 30 August 2022
Published: 08 January 2023

Summary

IGraph/M[1] is an efficient general purpose graph theory and network analysis package for Mathematica (Wolfram Research, Inc., 2022). IGraph/M serves as the Wolfram Language interfaces to the igraph C library (Csárdi et al., 2022; Csárdi & Nepusz, 2006), and also provides several unique pieces of functionality not yet present in igraph, but made possible by combining its capabilities with Mathematica's. The package is designed to support both graph theoretical research as well as the analysis of large-scale empirical networks.

Statement of need

Mathematica contains extensive built-in functionality for working with graphs. IGraph/M extends this graph framework with many new functions not otherwise available in the Wolfram Language, and also provides alternative and more featureful open-source implementations of many of Mathematica's existing built-ins. This makes it possible for Wolfram Language users to easily double-check results, just as Python and R users can already do thanks to the multiple different graph packages available in those languages. This is particularly useful in graph theory where many results are just as difficult to verify as to compute.

The only other independent graph theory package for Mathematica was Combinatorica (Pemmaraju & Skiena, 2003), which has been mostly unmaintained since the introduction of the `Graph` expression type with the release of Mathematica 8.0 in 2011. Despite this, not all of Combinatorica's functions have built-in equivalents in Mathematica. IGraph/M provides replacements for almost all of these old Combinatorica functions while offering much better performance thanks to being implemented in a mix of C and C++ instead of the Wolfram Language.

Design goals and features

One of the major appeals of Mathematica is its tightly integrated nature: different functionality areas of the system can smoothly and seamlessly interoperate with each other. In order to

[1]Available at szhorvat.net/mathematica/IGraphM and github.com/szhorvat/IGraphM

preserve this benefit, a major design goal for IGraph/M was to integrate well into the rest of the system. This is achieved by working directly with Mathematica's native `Graph` data type, which is transparently converted to igraph's internal format as needed. This makes IGraph/M different from igraph's other high-level interfaces: igraph's internal graph data structure is not exposed to users and the package does not provide trivial operations which are already present in Mathematica, such as adding or removing vertices and edges. Instead, priority is given to functionality that delivers a true benefit over Mathematica's own built-ins. While some of IGraph/M's functions may appear to simply duplicate built-in functions, almost all of them provide additional features not otherwise available in Mathematica. For example, unlike the built-in `BetweenessCentrality[]` function, `IGBetweenness[]` supports weighted graphs; unlike the built-in `PageRankCentrality[]`, `IGPageRank[]` takes into consideration self-loops and parallel edges; in contrast to `IsomorphicGraphQ[]`, `IGIsomorphicQ[]` supports non-simple graphs as well as vertex and edge colours; etc. In order to make it easy to distinguish built-in symbols from those of IGraph/M, all functions in this package have names starting with `IG`. The naming and interface of functions is chosen so as to be familiar both to Mathematica users and users of igraph's Python and R interfaces.

IGraph/M fully leverages the igraph C library's capabilities to integrate into high-level host languages: Almost all computations are interruptible in the usual manner. This feature is particularly important for a graph theory package that provides multiple algorithms with exponential or slower time complexity. Despite being implemented in a compiled language, IGraph/M uses Mathematica's built-in random number generator by default. Therefore, all its stochastic algorithms respect seeds set with `SeedRandom[]` or `BlockRandom[]` the same way as built-in functions would.

IGraph/M aims to exploit the interactive features of Mathematica notebooks to improve user productivity. In this spirit, it includes an interactive graph editor, `IGGraphEditor[]`, and supports dynamically displaying the progress of many functions.

Use cases and unique features

IGraph/M provides multiple unique features that are not present in the core igraph library. Examples include exact graph colouring (Van Gelder, 2008), functions for working with planar graphs and combinatorial embeddings, proximity graph functions (Kirkpatrick & Radke, 1985; Toussaint, 1980) and aids for working with spatial networks, functions for performing tests related to the graph automorphism group, and several others. Some features, such as `IGRealizeDegreeSequence`, are based on original research of the authors (Horvát & Modes, 2021). Additionally, there are many convenience functions that are helpful when working with Mathematica's `Graph`, including a simplified system for working with edge and vertex attributes based on the concept of mapping functions over attribute values (`IGEdgeMap`, `IGVertexMap`).

The following examples show a few of these features while also demonstrating the concise idioms made possible by this package. The output is shown in Figure 1. Many more examples are found in the package's documentation.

Load the package:

```
In[1]:= << IGraphM`
Out[1]= IGraph/M 0.6.5 (December 21, 2022)
        Evaluate IGDocumentation[] to get started.
```

Create a random maze as a random spanning tree of a regular lattice confined to a hexagon, and colour it according to betweenness centrality:

```
In[2]:=
g = IGMeshGraph@IGLatticeMesh["CairoPentagonal", Polygon@CirclePoints[6, 6]];
t = IGRandomSpanningTree[g,
        VertexCoordinates -> GraphEmbedding[g], GraphStyle -> "ThickEdge"];
IGEdgeMap[ColorData["Rainbow"], EdgeStyle -> IGEdgeBetweenness /* Rescale, t]
```

Compute and visualize a minimum vertex and edge colouring of the Gabriel graph of a set of spatial points:

```
In[3]:=
IGGabrielGraph[RandomPoint[Disk[], 30],
    GraphStyle -> "Monochrome", VertexSize -> 1, EdgeStyle -> Thickness[1/40]
] //
  IGVertexMap[ColorData[106], VertexStyle -> IGMinimumVertexColoring] //
  IGEdgeMap[ColorData[106], EdgeStyle -> IGMinimumEdgeColoring]
```

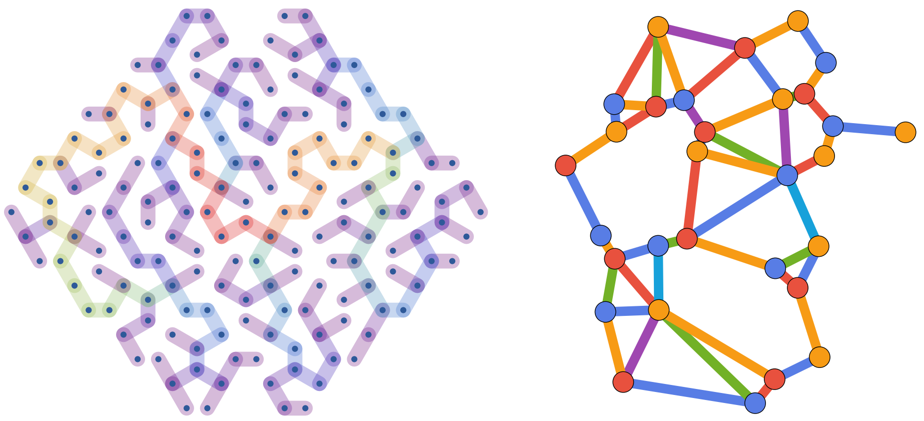

Figure 1: Output of the above example code. `Out[2]` is on the left, `Out[3]` on the right.

Implementation notes

IGraph/M is built using LTemplate (Horvát, 2018), an open-source system that makes it easier to extend Mathematica using C++ code. IGraph/M also serves as the primary driver of LTemplate development. Planar graph-related functionality is implemented using the LEMON graph library (Dezső et al., 2011; Egerváry Research Group on Combinatorial Optimization, 2014). Spatial graph functions make use of the nanoflann nearest neighbour search library (Blanco & Rai, 2022).

Acknowledgements

This project has been made possible in part by grant number 2021-237461 from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation. We thank Juho Lauri for feedback and ideas, as well as for guidance with the implementation of the SAT-based exact graph colouring functionality. Henrik Schumacher provided the basis for the implementation of the mesh/graph conversion functions and some of the proximity graph functions. Patrick Scheibe made available his Wolfram Language plugin for IntelliJ IDEA, without which this project would have been much more difficult to manage. Sz. H. is grateful to Carl Modes for ongoing support and feedback regarding the paper, and to the Wissenchaftskolleg zu Berlin for time to work on spatial networks functionality in this package.

References

Blanco, J. L., & Rai, P. K. (2022). *Nanoflann: A C++ header-only fork of FLANN, a library for nearest neighbor (NN) with KD-trees* (Version 1.4.3) [Computer software]. GitHub.

https://github.com/jlblancoc/nanoflann

Csárdi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal Complex Systems*, *1695*(5), 1–9. http://www.interjournal.org/manuscript_abstract.php?361100992

Csárdi, G., Nepusz, T., Horvát, Sz., Traag, V., Zanini, F., & Noom, D. (2022). *The igraph C library* (Version 0.9.9) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.3630268

Dezső, B., Jüttner, A., & Kovács, P. (2011). LEMON – an open source C++ graph template library. *Electron. Notes Theor. Comput. Sci.*, *264*, 23–45. https://doi.org/10.1016/j.entcs.2011.06.003

Egerváry Research Group on Combinatorial Optimization. (2014). *LEMON – library for efficient modeling and optimization in networks* (Version 1.3.1) [Computer software]. https://lemon.cs.elte.hu/

Horvát, Sz. (2018). *LTemplate* (Version 0.5.4) [Computer software]. GitHub. https://github.com/szhorvat/LTemplate/

Horvát, Sz., & Modes, C. D. (2021). Connectedness matters: Construction and exact random sampling of connected networks. *Journal of Physics: Complexity*, *2*, 015008. https://doi.org/10.1088/2632-072X/abced5

Kirkpatrick, D. G., & Radke, J. D. (1985). A framework for computational morphology. *Machine Intelligence and Pattern Recognition*, *2*(C), 217–248. https://doi.org/10.1016/B978-0-444-87806-9.50013-X

Pemmaraju, S., & Skiena, S. (2003). *Computational discrete mathematics: Combinatorics and graph theory with Mathematica*. Cambridge University Press. https://doi.org/10.1017/CBO9781139164849

Toussaint, G. T. (1980). The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, *12*(4), 261–268. https://doi.org/10.1016/0031-3203(80)90066-7

Van Gelder, A. (2008). Another look at graph coloring via propositional satisfiability. *Discrete Applied Mathematics*, *156*(2), 230–243. https://doi.org/10.1016/j.dam.2006.07.016

Wolfram Research, Inc. (2022). *Mathematica* (Version 13.1) [Computer software]. Wolfram Research, Inc. https://www.wolfram.com/mathematica/