




matlab-igraph: bringing igraph to MATLAB

David R. Connell ¹

¹ Independent Researcher, United States

DOI: [10.21105/joss.08622](https://doi.org/10.21105/joss.08622)

Software

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

Editor: [Daniel S. Katz](#)  

Reviewers:

- [@gartavanis](#)
- [@platipodium](#)

Submitted: 25 June 2025

Published: 05 September 2025

License

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

Summary

High-throughput lab techniques have enabled systems biology to view medicine through networks rather than through individual genes or gene pairs ([Goh et al., 2007](#)). Using graph approaches, we can learn generic patterns that apply to many sets of genes, proteins, diseases, etc. simultaneously, vastly improving the rate we can find disease genes and enabling the transfer of knowledge between disorders. I created `matlab-igraph`, a toolbox that integrates the efficient graph algorithms and tools of the `igraph` C library ([Csardi & Nepusz, 2006](#)) into MATLAB's environment to aid in graph based research. By representing graphs using MATLAB's builtin types, I maintain a simple syntax while leveraging compiled C code to simplify graph analysis without sacrificing efficiency, enabling users to focus on their experiments.

Statement of need

MATLAB is a powerful tool for numerical analysis that is widely used in academia alongside Python and R, but unlike Python and R, it lacked support for `igraph`, a gap filled by `matlab-igraph`. Without `matlab-igraph`, researchers would need to rewrite algorithms specifically for MATLAB, which not only takes time but also risks errors. Development of `matlab-igraph` begun with work on the `SpeakEasy2` community detection algorithm ([Gaiteri et al., 2023](#)), which needed to score partitions, using methods like modularity and normalized mutual information, and compare results to current community detection algorithms such as Leiden and InfoMAP. By making graph algorithms available to MATLAB, the `matlab-igraph` toolbox allows easy comparisons against state-of-the-art methods. This package has two goals: to bring `igraph` functions into MATLAB for end users, and to ease development of new graph methods in C using the `mxIgraph` interface library and the `igutils` namespace.

Use in MATLAB

To take advantage of MATLAB's strong matrix support, I designed the toolbox to support representing graphs as adjacency matrices, allowing cooperation between MATLAB builtins and `igraph` functions. For graphs that require more metadata, MATLAB's own graph and digraph classes can be used as well, which explicitly track directedness and can store `igraph`'s node and edge attributes. All functions that accept a graph can read graph classes or matrices and functions that return a graph accept the `repr` keyword to choose between a full matrix, sparse matrix, or graph class. This design choice streamlines workflows, allowing researchers to combine MATLAB's native functions with `igraph`'s algorithms without the need for a new `igraph` specific data type, converting between representations, or redundant coding.

Comparison to builtin graph algorithms

While MATLAB provides graph datatypes and a few algorithms for those graphs, the set of available graph algorithms is limited and does not provide recently published methods. The `igraph` library provides many missing algorithms made accessible to MATLAB through

this toolbox, including those pertaining to generating graphs (such as through `generate` or `randgame` for stochastic graphs), reading and writing common graph file types, community detection, comparing community structure, and rewiring graphs. All methods supplied by `matlab-igraph` work on the graph datatypes but they can also be applied directly to matrices, in contrast to the builtin graph algorithms which can only be used with graphs types, allowing users to continue working with the matrix syntax experienced MATLAB programmers have become familiar with. For this reason, `matlab-igraph` provides replacements for several builtin functions like `numnodes` and `numedges`, `isisomorphic`, `degree`, etc. that will work equivalently on both adjacency matrices and graphs, making it simple to create higher level graph functions that can be used with either datatype.

mxlgraph

In addition to the `igraph` functions, I provide a standard for creating new graph functions using MATLAB's mex engine and the `igraph` C library to compile code for use in MATLAB. To aid with this, the toolbox includes the bridge C library `mxIgraph` for communicating between MATLAB's C API and `igraph` and the `igutils` namespace for consistent argument handling throughout the toolbox. The `mxIgraph` library exposes functions for converting between MATLAB and `igraph` data types, predicates for working with graphs, and a set of methods for parsing argument structures created by MATLAB's argument blocks in C. When large graphs are involved, C can reduce the memory demand over writing in MATLAB. Specifically, memory usage can be an issue in MATLAB when using "pairwise-reduce" patterns or parallel computation. For computation on vectors, a common pattern is performing a pairwise function (like an outer-product) on two vectors and then reducing over an axis. For an $m \times 1$ and an $n \times 1$ vector, this leads to an $m \times n$ intermediary matrix that is then reduced back down to a vector with a sum along the rows or similar operation. By using for-loops in C instead of high-level linear algebra packages, the final vector can be created directly. This reduces the $\mathcal{O}(mn)$ memory requirements to $\mathcal{O}(m)$. If the code can be written to run in parallel, C supports multithreading, in contrast to MATLAB's parallel toolbox which is based on multiprocessing, a technique that requires cloning data across each worker, potentially exhausting memory when used with large graphs, while high overhead can cause performance to be worse than serial processing for smaller graphs. Multithreading allows memory to be shared across threads, enabling efficient parallel processing on graphs of all sizes and does not require the MATLAB parallel toolbox.

The `igutils` namespace within the toolbox complements the `mxIgraph` by standardizing parsing of common arguments, ensuring consistency across functions. For example, `igutils` provides argument parser classes for functions that accept graphs and a second for those that return a graph, these classes organize common arguments and define acceptable values in a single location. If `mxIgraph` is extended to handle other types in the future, all MATLAB functions using the `igutils` classes and predicates for argument parsing will automatically be updated as well. Together, these components lower the barrier to implement efficient graph code for MATLAB.

References

- Csardi, G., & Nepusz, T. (2006). The `igraph` software. *Complex Syst*, 1695, 1–9.
- Gaiteri, C., Connell, D. R., Sultan, F. A., Iatrou, A., Ng, B., Szymanski, B. K., Zhang, A., & Tasaki, S. (2023). Robust, scalable, and informative clustering for diverse biological networks. *Genome Biology*, 24(1), 228. <https://doi.org/10.1186/s13059-023-03062-0>
- Goh, K.-I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., & Barabási, A.-L. (2007). The human disease network. *Proceedings of the National Academy of Sciences*, 104(21), 8685–8690. <https://doi.org/10.1073/pnas.0701361104>