

# MatrixBandwidth.jl: Fast algorithms for matrix bandwidth minimization and recognition

Luis M. B. Varona<sup>1,2,3</sup>

**1** Department of Politics and International Relations, Mount Allison University **2** Department of Mathematics and Computer Science, Mount Allison University **3** Department of Economics, Mount Allison University

DOI:

Software

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

Licence

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

## Summary

**MatrixBandwidth.jl** offers fast algorithms for matrix bandwidth minimization and recognition. The *bandwidth* of an  $n \times n$  matrix  $A$  is the minimum non-negative integer  $k \in \{0, 1, \dots, n-1\}$  such that  $A_{i,j} = 0$  whenever  $|i - j| > k$ . Reordering the rows and columns of a matrix to reduce its bandwidth has many practical applications in engineering and scientific computing: it can improve performance when solving linear systems, approximating partial differential equations, optimizing circuit layout, and more (Maftiu-Scai 2014). There are two variants of this problem: *minimization*, which involves finding a permutation matrix  $P$  such that the bandwidth of  $PAP^T$  is minimized, and *recognition*, which entails determining whether there exists a permutation matrix  $P$  such that the bandwidth of  $PAP^T$  is less than or equal to some fixed non-negative integer (an optimal permutation that fully minimizes the bandwidth of  $A$  is not required).

## Algorithms

As of version 0.2.1, the following matrix bandwidth reduction algorithms are available:

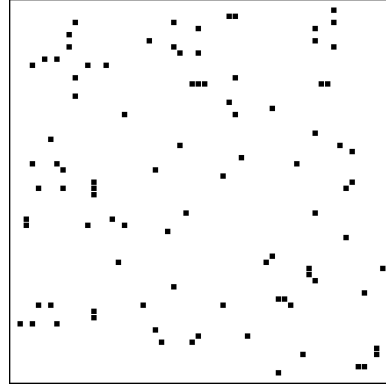
- Minimization
  - Exact
    - \* Caprara–Salazar–González (Caprara and Salazar-González 2005)
    - \* Del Corso–Manzini (Del Corso and Manzini 1999)
    - \* Del Corso–Manzini with perimeter search (Del Corso and Manzini 1999)
    - \* Saxe–Gurari–Sudborough (Saxe 1980; Gurari and Sudborough 1984)
    - \* Brute-force search
  - Heuristic
    - \* Gibbs–Poole–Stockmeyer (Gibbs et al. 1976)
    - \* Cuthill–McKee (Cuthill and McKee 1969)
    - \* Reverse Cuthill–McKee (Cuthill and McKee 1969; George 1971)
- Recognition
  - Caprara–Salazar–González (Caprara and Salazar-González 2005)
  - Del Corso–Manzini (Del Corso and Manzini 1999)
  - Del Corso–Manzini with perimeter search (Del Corso and Manzini 1999)
  - Saxe–Gurari–Sudborough (Saxe 1980; Gurari and Sudborough 1984)
  - Brute-force search

Recognition algorithms determine whether any row-and-column permutation of a matrix induces bandwidth less than or equal to some fixed integer. Exact minimization algorithms always guarantee optimal orderings to minimize bandwidth, while heuristic minimization algorithms produce near-optimal solutions more quickly. Metaheuristic minimization algorithms employ iterative search frameworks to find better solutions than heuristic

methods (albeit more slowly); no such algorithms are already implemented, but several (e.g., simulated annealing) are currently under development.

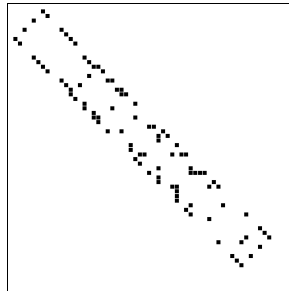
## Example

Consider the following  $60 \times 60$  sparse matrix with initial bandwidth 51:

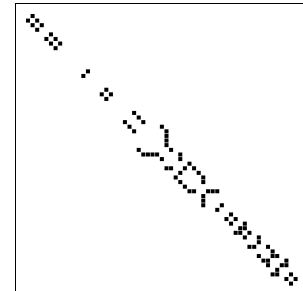


**Figure 1:** Original  $60 \times 60$  matrix with bandwidth 51

`MatrixBandwidth.jl` can both recognize whether the minimum bandwidth of  $A$  is less than or equal to some fixed integer (Figure 2) and actually minimize the bandwidth of  $A$  (Figure 3):



**Figure 2:** The matrix with bandwidth recognized as  $\leq 6$  via the Del Corso–Manzini algorithm



**Figure 3:** The matrix with bandwidth minimized to 5 via the Gibbs–Poole–Stockmeyer algorithm

(Note that since Gibbs–Poole–Stockmeyer is a heuristic algorithm, 5 may not be the *true* minimum bandwidth of  $A$ , but it is likely close.)

## Statement of need

Many matrix bandwidth reduction algorithms exist in the literature, but implementations in the open-source ecosystem are scarce, with those that do exist primarily tackling older, less efficient algorithms. The `Boost` libraries in C++ (Lumsdaine et al. 2001), the `NetworkX` library in Python (NetworkX Developers 2025), and the MATLAB standard library (MATLAB Developers 2025) all only implement the aforementioned reverse Cuthill–McKee algorithm from 1971. In Julia, the only other relevant package identified by the author is `SymRCM.jl` (Krysl 2020), which also only implements reverse Cuthill–McKee.

Furthermore, not enough attention is given to recognition algorithms or exact minimization algorithms. Although more performant modern alternatives are often neglected, at least

reverse Cuthill–McKee is a widely implemented method of approximating a minimal bandwidth ordering. However, no such functionality for recognition or exact minimization is widely available, requiring researchers with such needs to re-implement these algorithms from scratch every time.

These two gaps in the ecosystem not only make it difficult for theoretical researchers to benchmark and compare new proposed algorithms but also preclude the application of the most performant modern algorithms in real-life industry settings. `MatrixBandwidth.jl` aims to bridge this gap by presenting a unified interface for matrix bandwidth reduction algorithms in Julia, designed with extensibility to further methods in mind.

## Research applications

The author either has used or is using this package to do the following:

- Develop a new polynomial-time algorithm for “bandwidth  $\leq k$ ” recognition efficient for both small and large  $k$ , and benchmarking it against other approaches (Saxe 1980; Gurari and Sudborough 1984)
- Speed up  $k$ -coherence checks of quantum states in many cases by confirming that the density matrix’s minimum bandwidth is greater than  $k$  (Johnston et al. 2025)
- Compute the spectral graph property of “ $S$ -bandwidth” (Johnston and Plosker 2025) via the `SDiagonalizability.jl` package (Varona et al. 2025), which depends critically on `MatrixBandwidth.jl` for bandwidth recognition
- Investigate the precise performance benefits of reducing the propagation graph’s bandwidth when training a recurrent neural network, building on Balog et al. (2019)

The first three use cases rely on the recognition and exact minimization functionality unique to `MatrixBandwidth.jl` (indeed, they largely motivated the package’s development). The last (ongoing) research project *could* be facilitated by `SymRCM.jl` instead, but the author intends to use more performant metaheuristic minimization algorithms currently under development when producing the final computational results.

## Limitations

Currently, `MatrixBandwidth.jl`’s core functions generically accept any input of the type `AbstractMatrix{<:Number}`, not behaving any differently when given sparsely stored matrices (e.g., from the `SparseArrays.jl` standard library package). Capabilities for directly handling graph inputs (aiming to reduce the matrix bandwidth of a graph’s adjacency) are also not available. Given that bandwidth reduction is often applied to sparse matrices and graphs, these limitations will be addressed in a future release of the package.

## Conflict of interests

The author declares no conflict of interest.

## Acknowledgements

I owe much to my research supervisors—Nathaniel Johnston, Sarah Plosker, and Craig Brett—for supporting and guiding me in my work. I would also like to thank Liam Keliher, Peter Lelièvre, and Marco Cогnetta for useful discussions. Finally, credit for the package’s telepathic-cat-and-turtle logo goes to Rebekka Jonasson.

## References

- Balog, Matej, Bart van Merriënboer, Subhodeep Moitra, Yujia Li, and Daniel Tarlow. 2019. *Fast Training of Sparse Graph Neural Networks on Dense Hardware*. <https://doi.org/10.48550/arXiv.1906.11786>.
- Caprara, Alberto, and Juan-José Salazar-González. 2005. “Laying Out Sparse Graphs with Provably Minimum Bandwidth.” *INFORMS Journal on Computing* 17 (3): 356–73. <https://doi.org/10.1287/ijoc.1040.0083>.
- Cuthill, E., and J. McKee. 1969. “Reducing the Bandwidth of Sparse Symmetric Matrices.” *Proceedings of the 24th National Conference of the ACM*, 157–72. <https://doi.org/10.1145/800195.805928>.
- Del Corso, G. M., and G. Manzini. 1999. “Finding Exact Solutions to the Bandwidth Minimization Problem.” *Computing* 62: 189–203. <https://doi.org/10.1007/s006070050002>.
- George, J. Alan. 1971. “Computer Implementation of the Finite Element Method.” PhD thesis, Department of Computer Science, Stanford University. <https://apps.dtic.mil/sti/tr/pdf/AD0726171.pdf>.
- Gibbs, Norman E., William G. Poole Jr., and Paul K. Stockmeyer. 1976. “An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix.” *SIAM Journal on Numerical Analysis* 13 (2): 236–50. <https://doi.org/10.1137/0713023>.
- Gurari, Eitan M., and Ival Hal Sudborough. 1984. “Improved Dynamic Programming Algorithms for Bandwidth Minimization and the MinCut Linear Arrangement Problem.” *Journal of Algorithms* 5 (4): 531–46. [https://doi.org/10.1016/0196-6774\(84\)90006-3](https://doi.org/10.1016/0196-6774(84)90006-3).
- Johnston, Nathaniel, Shirin Moein, and Sarah Plosker. 2025. “The Factor Width Rank of a Matrix.” *Linear Algebra and Its Applications* 716: 32–59. <https://doi.org/10.1016/j.laa.2025.03.016>.
- Johnston, Nathaniel, and Sarah Plosker. 2025. “Laplacian  $\{-1,0,1\}$ - and  $\{-1,1\}$ -Diagonalizable Graphs.” *Linear Algebra and Its Applications* 704: 309–39. <https://doi.org/10.1016/j.laa.2024.10.016>.
- Krysl, Petr. 2020. *SymRCM: Reverse Cuthill-McKee Node-Renumbering Algorithm for Sparse Matrices*. GitHub. <https://github.com/PetrKryslUCSD/SymRCM.jl>.
- Lumsdaine, Andrew, Lie-Quan Lee, Jeremy G. Siek, Doug Gregor, and Kevin D. McGrath. 2001. *Example/Cuthill\_mckee\_ordering.cpp*. Boost v1.37.0 documentation. [https://www.boost.org/doc/libs/1\\_37\\_0/libs/graph/example/cuthill\\_mckee\\_ordering.cpp](https://www.boost.org/doc/libs/1_37_0/libs/graph/example/cuthill_mckee_ordering.cpp).
- Maftaiu-Scai, Liviu Octavian. 2014. “The Bandwidths of a Matrix. A Survey of Algorithms.” *Annals of West University of Timisoara - Mathematics and Computer Science* 52 (2): 183–223. <https://doi.org/10.2478/awutm-2014-0019>.
- MATLAB Developers. 2025. *Symrcm - Sparse Reverse Cuthill-McKee Ordering - MATLAB*. MATLAB R2025b documentation. <https://www.mathworks.com/help/matlab/ref/symrcm.html>.
- NetworkX Developers. 2025. *Source Code for Networkx.utils.rcm*. NetworkX v3.5 documentation. [https://networkx.org/documentation/stable/\\_modules/networkx/utils/rcm.html](https://networkx.org/documentation/stable/_modules/networkx/utils/rcm.html).

- Saxe, James B. 1980. “Dynamic-Programming Algorithms for Recognizing Small-Bandwidth Graphs in Polynomial Time.” *SIAM Journal on Algebraic and Discrete Methods* 1 (4): 363–69. <https://doi.org/10.1137/0601042>.
- Varona, Luis M. B., Nathaniel Johnston, and Sarah Plosker. 2025. *S-Diagonalizability: A Dynamic Algorithm to Minimize or Recognize the S-Bandwidth of an Undirected Graph*. GitHub. <https://github.com/GraphQuantum/SDiagonalizability.jl>.