The Journal of Open Source Software

grapherator: A Modular Multi-Step Graph Generator

Jakob Bossek[1]

1 University of Münster

DOI: 10.21105/joss.00528

Software

- Review
- Repository
- Archive

Submitted: 22 December 2017
Published: 19 February 2018

Summary

Benchmarking algorithms for computationally hard (multi-criteria) optimization problems on graphs are usually carried out by running the set of algorithms on a set of test problems. Typically the test set consists of artificially generated benchmark graphs. Artificial problems allow for 1) the generation of arbitrary many instances in short time and 2) the generation of problems with different hardness levels or different characteristics of optimal solutions. E.g., it is well known, that the structure of edge weight combinations decides on the shape and size of the Pareto-front for multi-criteria minimum spanning tree (mcMST) problems which in turn may affect performance of algorithms (Bossek and Grimme 2017; Knowles and Corne 2001).

The `R` (R Core Team 2017) package `grapherator` implements different methods for random graph generation. The focus is on weighted graphs with one or more weights per edge. `Grapherator` thus targets researchers who study single- or multi-criteria optimization problems on graphs. Originally, an early predecessor was part of the `R` package `mcMST` (Bossek 2017). As complexity increased, the methods were outsourced into a dedicated package. In contrast to most graph generation libraries e.g., NetworkX (Hagberg, Schult, and Swart 2008) for Python, `grapherator` implements a flexible three-step workflow.

Grapherator workflow

The technical pipeline (see Figure 1) follows a three-step approach: 1) node generation (e.g., lattice, uniform etc.), 2) edge generation (e.g., Erdos-Renyi (Erdös and Rényi 1959), Waxman-model (Waxman 1988) etc.) and 3) weight generation (e.g., distance-based, random, correlated etc.). Each step may be repeated multiple times with different generator functions yielding high flexibility (see Figure 2 for some examples). The set of predefined generator functions can be easily expanded with custom functions.

Support

Bug reports and feature requests are highly appreciated via the GitHub issue tracker (https://github.com/jakobbossek/grapherator/issues).

References

Bossek, J. 2017. "mcMST: A Toolbox for the Multi-Criteria Minimum Spanning Tree Problem." *Journal of Open Source Software* 2 (17). https://doi.org/10.21105/joss.00374.

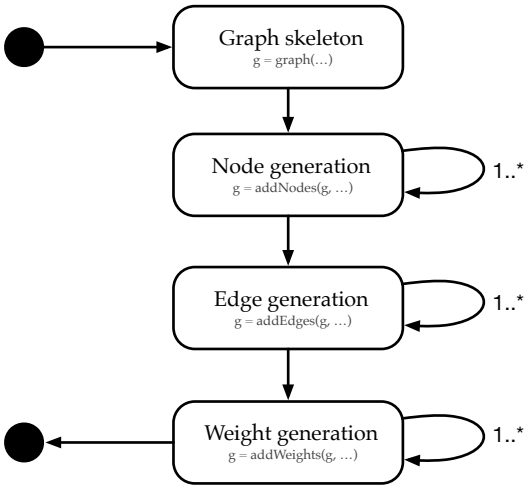

Figure 1: The grapherator workflow

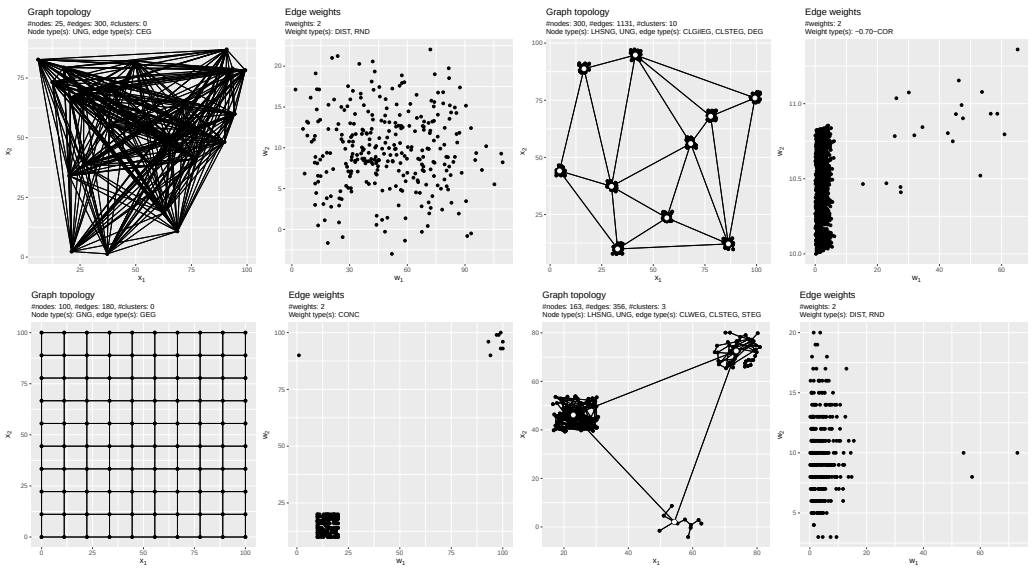

Figure 2: Example graphs with two weights per edge. Both graph topology and a scatterplot of the edge weights is shown.

Bossek, J., and C. Grimme. 2017. "A Pareto-Beneficial Sub-Tree Mutation for the Multi-Criteria Minimum Spanning Tree Problem." In *Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence*, 3280–7. Honolulu, Hawaii, USA.

Erdös, P., and A. Rényi. 1959. "On random graphs, I." *Publicationes Mathematicae (Debrecen)* 6:290–97.

Hagberg, A. A., D. A. Schult, and P. J. Swart. 2008. "Exploring Network Structure, Dynamics, and Function Using NetworkX." In *Proceedings of the 7th Python in Science Conference (Scipy2008)*, 11–15. Pasadena, CA USA.

Knowles, J. D., and D. W. Corne. 2001. "Benchmark Problem Generators and Results for the Multiobjective Degree-Constrained Minimum Spanning Tree Problem." In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, 424–31. GECCO'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Waxman, B. M. 1988. "Routing of Multipoint Connections." *IEEE Journal on Selected Areas in Communications* 6 (9):1617–22. https://doi.org/10.1109/49.12889.