grapherator: A Modular Multi-Step Graph Generator

Jakob Bossek

1 University of Münster

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., Erdős-Rényi (Erdős 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

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.


