Beating the Worst Case
Winterterm 2025/2026
Jean-Pierre von der Heydt & Marcus Wilhelm

Exercise Sheet 3: More Graphs

Published on 26.11.2025. Duration: 2 Weeks

In the previous exercise sheets, we have learned about network properties that help predict the performance of our algorithms. In this exercise sheet, the goal is to collect more graphs that cover a broader range of properties. To this end, we will both generate graphs and analyze real-world networks.

Exercise 1: Generate Graphs

Real-world instances in exactly the right shapes and sizes are often difficult to find. Therefore, it is often useful to run an algorithm on randomly generated instances. In particular, this allows us to study its scalability. Using randomly generated instances also has the advantage that the parameters of the model (e.g., number of nodes and average degree) can be chosen freely. In this exercise, we want to expand our collection of test instances to enable more robust conclusions.

There is a large variety of models for generating graphs, each with its own advantages and disadvantages depending on the application. Research the most common models and use them to generate test instances. A good starting point for this are the graph generators of various frameworks, such as NetworKit[1], NetworkX[2], or igraph[3].

Choose at least 5 network models and generate random graphs with each of them. Analyze these networks according to the methods used in the previous exercise sheets.

Exercise 2: GIRG Generator

The framework already integrates a generator[4] for so-called *GIRGs* (geometric inhomogeneous random graphs). The generator comes as a C++ library with a command-line interface for GIRGs and for hyperbolic random graphs. The GIRG model has several parameters. What influence do the parameters `ple` and `alpha` have on the performance of the algorithms? Can you generate networks with high heterogeneity and low locality? Are there parameter values for which GIRGs behave similarly to the generators from Exercise 1? Can you perhaps even determine how the graphs in the first exercise sheet were generated?

Hint 1: `ple` is restricted to the interval $(2, 3]$ in the command-line interface. However, values greater than 3 can also be meaningful. To use larger values, you may need to adjust the command-line

[1]`https://networkit.github.io/`
[2]`https://networkx.org/`
[3]`https://igraph.org/`
[4]`https://github.com/chistopher/girgs/`

interface or use the C++ library directly.

Hint 2: The parameter `alpha` is restricted to $(1, \infty)$. When experimenting with different values for `alpha`, think rather in terms of `1/alpha`. It is helpful to choose values such that the corresponding `1/alpha` values are roughly evenly distributed in $[0, 1)$.

Exercise 3: Real-World Networks

In practice, of course, one rarely works with generated graphs, and the real world is seldom as ideal as our models. Several good collections of networks can be found, for example, at Network Repository[5], Konect[6], and on the chair's website[7].

Analyze the heterogeneity and locality of these graphs. Can we transfer our insights about graph parameters and random graphs to real-world graphs? Repeat the previous experiments on a **large** set of real-world networks.

[5] `https://networkrepository.com/`

[6] `http://konect.cc/networks/`

[7] `https://external-validity.iti.kit.edu/`