NETWORKIT: AN INTERACTIVE TOOL FOR HIGH-PERFORMANCE NETWORK ANALYSIS

Christian L. Staudt, Aleksejs Sazonovs, Henning Meyerhenke

Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT)

Summary

We introduce NetworKit, an open-source package for high-performance analysis of large complex networks. Complex networks are equally attractive and challenging targets for data mining, and novel algorithmic solutions as well as parallelism are required to handle data sets containing billions of connections [4, 5]. Our goal is to package results of our algorithm engineering efforts and put them into the hands of domain experts.

The package is a hybrid combining the performance of kernels – written in C++ and parallelized with OpenMP – with a convenient interactive interface written in Python. The package supports general multicore platforms and scales from notebooks to workstations to compute servers. In comparison with related software for network analysis, we propose NetworKit as the package which satisfies all of three important criteria: High performance enabled by parallelism, interactive workflows and integration into an ecosystem of tested tools for data analysis and scientific computation. The feature set includes standard network analytics kernels such as connected components, clustering coefficients, community detection, core decomposition, assortativity and centrality. Applying these to massive networks is enabled by efficient algorithm design, parallelism and approximation. Furthermore, the package comes with a collection of graph generators and has basic support for visualization and dynamic networks. With the current release, we aim to present and open up the project to a community of both algorithm engineers and domain experts.

Features

NetworKit has a growing feature set and is built for extensibility. Current features include:

Community detection is the task of identifying groups of nodes which are significantly more densely connected among each other than to the rest of the network. NetworKit includes state-of-the-art heuristics with efficient parallel implementations for partitioning the network into natural modules [7].

Clustering coefficients quantify the tendency of relations in a network to become transitive by looking at the frequency of closed triangles. NetworKit supports both exact calculation and approximation.

Degree distribution and assortativity play an important role in characterizing a network: Complex networks tend to show a heavy tailed degree distribution which follow a power-law with a characteristic exponent [1]. Degree assortativity is the correlation of degrees for connected nodes. NetworKit makes it easy to estimate both.

Components and cores are related concepts for subdividing a network: All nodes in a connected component are reachable from each other. k-cores/k-shells result from successively peeling away nodes of degree k.

Centrality refers to the relative importance of a node within a network. Different ideas of importance are expressed by betweenness, PageRank and eigenvector centrality. Betweenness is approximated with a bounded error to be applicable to large networks.

Standard graph algorithms such as finding independent sets, computing approximate maximum weight matchings, breadth-first and depth-first search or finding shortest paths.

Generative models aim to explain how networks form and evolve specific structural features. NetworKit has efficient generators for basic Erdős-Rényi random graphs, the Barabasi-Albert and Dorogovtsev-Mendes models (which produce power law degree distributions), the Chung-Lu and Havel-Hakimi model (which replicate given degree distributions, the former in expectation, the latter only realizable ones).

Visualization functionality which enables the user to draw smaller networks to the IPython Notebook or files.

Design Goals

NetworKit is designed to stand out in three areas:

Performance Algorithms and data structures are selected and implemented with high performance and parallelism in mind. Some implementations are among the fastest in published research. For example, community detection in a 3 billion edge web graph can be performed on a machine with 16 physical cores and 256 GB of RAM in a matter of minutes.

Interface Networks are as diverse as the series of questions we might ask of them - for example, what is the largest connected component, what are the most central nodes in it and how do they connect to each other? A practical tool for network analysis should therefore avoid restricting the user to fixed and predefined tasks, as most static command line interfaces do. Rather, the aim must be to create convenient and freely combinable functions. In this respect we take inspiration from software like R, MATLAB and Mathematica, as well as a variety of Python packages. An interactive shell, which the Python language provides, meets these requirements. While NetworKit works with the standard Python 3 interpreter, combining it with the IPython Notebook allows us to integrate it into a fully fledged computing environment for scientific workflows [6]. It is also straightforward to set up and control a remote server for heavy computations.

Integration As a Python module, NetworKit enables seamless integration with Python libraries for scientific computing and data analysis, e. g. pandas for data frame processing and analytics, matplotlib for plotting, networkx for additional network analysis tasks, or numpy and scipy for advanced numeric and scientific computation. Furthermore, NetworKit aims to support a variety of input/output formats, for example export to the graphical network analysis software Gephi [2].

Implementation

Core data structures and algorithms of NetworKit are implemented in C++ using the C++11 standard, which allows the use of object-oriented and functional programming concepts without sacrificing performance. The graph data structure provides parallel iterators over node and edge sets using different load balancing schemes. Shared-memory parallelized is realized with OpenMP. Classes are then exposed to Python via the Cython toolchain [3]: Wrapper classes are converted to C++ code via the Cython compiler, then compiled and linked with the core into a native Python extension module. Additional functionality and a convenient interface is implemented in pure Python, yielding the final Python module.

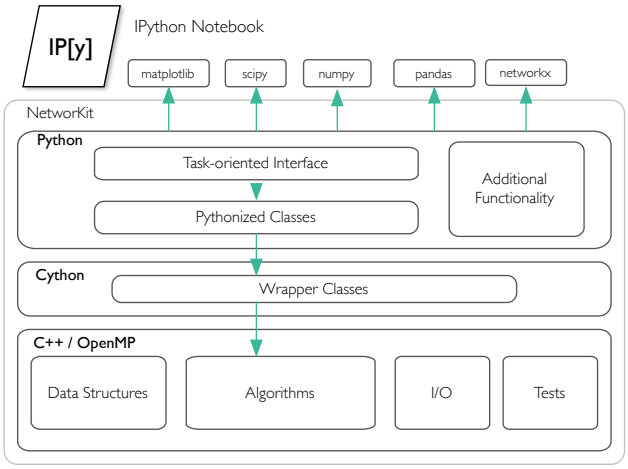

Figure 1: NetworKit architecture

Open Source

NetworKit is published[1] under the permissive MIT License to encourage review, reuse and extension by the community. We invite algorithm engineers and potential users from various research domains to benefit from and contribute to the development effort. ¿¿¿¿¿¿¿ other

References

[1] J. Alstott, E. Bullmore, and D. Plenz. powerlaw: a python package for analysis of heavy-tailed distributions. *PLOS ONE*, 9(1):e85777, 2014.

[2] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *ICWSM*, pages 361–362, 2009.

[3] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[4] U. Brandes and T. Erlebach. *Network analysis: methodological foundations*, volume 3418. Springer, 2005.

[5] M. Newman. *Networks: an introduction*. Oxford University Press, 2010.

[6] F. Perez, B. E. Granger, and C. Obispo. An open source framework for interactive, collaborative and reproducible scientific computing and education, 2013.

[7] C. L. Staudt and H. Meyerhenke. Engineering high-performance community detection heuristics for massive graphs. *arXiv preprint arXiv:1304.4453*.

[1] http://www.network-analysis.info