

NetworKit

An Interactive Tool Suite for High-Performance Network Analysis

<u>Christian L. Staudt</u>, Aleksejs Sazonovs, Henning Meyerhenke Parallel Computing Group - Institute of Theoretical Informatics - Karlsruhe Institute of Technology (KIT)

NetworKit

is an open-source software package for high-performance analysis of large complex networks uses shared-memory parallelism and scales from notebooks to compute servers \bullet combines kernels written in C++ with a convenient interactive interface written in Python.

Design Goals performance interface integration

- scalable algorithms, employing parallelism approximation performance-aware implementation
- modular design
- interactive usage via Python
- Python ecosystem for scientific computing & data analysis additional network analysis software (e.g. Gephi, NetworkX)

www.network-analysis.info



Analytics

Community Detection

Parallel Community Detection Heuristics [Staudt, Meyerhenke, ICPP 2013]

- PLM: modularity-driven multi-level technique, based on sequential Louvain method [Blondel et al. 2008]
 - high modularity
 - fast, scales to billions of edges



- fastest community detection heuristic
- scales well with the number of processors

EPP: ensemble technique, combining several weak classifiers into a strong one



Core Decomposition

k-cores result from iteratively

[Batagelj, Zaversnik 2003]

 \bullet O(m) algorithm

peeling away nodes of degree k



powerlaw module [Alstott et al. 2014] tests statistically for powerlaw distribution

Degree Assortativity



degree assortativity coefficient: correlation of node degrees among neighbors \blacksquare O(m) time algorithm

exact calculation (BFS/Dijkstra)

bounded error [Magien et al. 2009]

fast approximation with

Diameter

Clustering Coefficients

local and global exact parallel computation in $O(nd_{\max}^2)$ time very fast approximation with error guarantee

[Schank, Wagner 2005]

Additional Algorithms

NetworKit architecture

breadth-first and depth-first search Dijkstra's algorithm

approximate maximum weight matching algebraic distance maximum flows

Centrality

PageRank

- eigenvector centrality
- betweenness centrality [Brandes 2001]
- betweenness approximation
 - fast heuristic [Geisberger, Sanders, Schultes 2008]
 - approximation with maximum error guarantee
 - [Riondato, Kornaropoulos 2014]

Connected Components



computed using parallel label propagation scheme

Performance



Degree assortativity – soc-LiveJournal (43M edges) Average local clustering coefficient - uk-2002 (261M edges) timeout (2h) Network) NetworkX 216s igraph 789s graph-tool graph-tool 464s Networ NetworKit NetworKi 0.1s max. error = 0.1 (approxima Running time (s) Running time (s)



<pre>iie Edit View Inset Cell Kernel Help // Cell Toolbar None // B</pre>	Р[у	I: Notebook NetworKit Example Last Checkpoint: Jun 25 13:18 (autosaved)
<pre>import numpy as np import numpy as np if (a = readGraph("input/PCPgiantcompo.graph") (c = centrality.Betweenness(G1) (c = cen</pre>	ile	Edit View Insert Cell Kernel Help
<pre>1]: from NetworKit import * import seaborn as sns import numpy as np 3]: G1 = readGraph("input/PGPgiantoompo.graph") G2 = readGraph("input/astro-ph.graph") 2]: sns.set_style("white") dd1 = properties.degreeDistribution(G1) dd2 = properties.degreeDistribution(G2) xscale("log"); xscale("log") xlabel("degree", fontsize=16); ylabel("number of nodes", fontsize=16) plot(dd1); plot(dd2) 2]: [<matpletlib.lines.line2d 0x10f110f50="" at="">]</matpletlib.lines.line2d></pre>		
<pre>3]: G1 = readGraph("input/PGPgiantcompo.graph") G2 = readGraph("input/astro-ph.graph") 2]: sns.set_style("white") dd1 = properties.degreeDistribution(G1) dd2 = properties.degreeDistribution(G2) xscale("log"); yscale("log") xlabel("degree", fontsize=16); ylabel("number of nodes", fontsize=16) plot(dd1); plot(dd2) 2]: [<matplotlib.lines.line2d 0x10f110f50="" at="">]</matplotlib.lines.line2d></pre>	1]:	from NetworKit import * import seaborn as sns import numpy as np
<pre>22: sns.set_style("white") dd1 = properties.degreeDistribution(G1) dd2 = properties.degreeDistribution(G2) xscale("log"); yscale("log") xlabel("degree", fontsize=16); ylabel("number of nodes", fontsize=16) plot(d1); plot(dd2) 22: [<matplotlb.lines.line2d 0x10f110f50="" at="">]</matplotlb.lines.line2d></pre>	3]:	<pre>G1 = readGraph("input/PGPgiantcompo.graph") G2 = readGraph("input/astro-ph.graph")</pre>
<pre>i2]: [<matplotlib.lines.line2d 0x10f110f50="" at="">]</matplotlib.lines.line2d></pre>	2]:	<pre>sns.set_style("white") dd1 = properties.degreeDistribution(G1) dd2 = properties.degreeDistribution(G2) xscale("log"); yscale("log") xlabel("degree", fontsize=16); ylabel("number of nodes", fontsize=16) plot(dd1); plot(dd2)</pre>
7]: bc = centrality.Betweenness(G1)	2]:	[<matplotlib.lines.line2d 0x10f110f50="" at="">]</matplotlib.lines.line2d>
7]: bc = centrality.Betweenness(G1)		sepure degree to the second se
	7]:	<pre>bc = centrality.Betweenness(G1) bc = run()</pre>

Performance measurements on a shared-memory server with 256 GB RAM and 2x8 Intel(R) Xeon(R) E5-2680 cores (max. 32 threads)

Graph Generators

Erdős-Renyi model classic random graph model ■ fast generator

Barabasi-Albert Chung-Lu model produces networks distribution with powerlaw degree distribution

static and dynamic generator

R-MAT generator replicates any given degree graph generator power-law degree

PubWeb generator popular high-performance geometric disk graph model simulates P2P network distribution, small-world static and dynamic property and self-similarity generator



Interactive network analysis using IPython Notebook

Open Source

By publishing NetworKit under the permissive open-source MIT license, we encourage usage and contributions by a community of algorithm engineers and data analysts. We thank all previous contributors. **Get NetworKit:** http://www.network-analysis.info

Future

improved support for dynamic and

attributed graphs

- dynamic network analysis algorithms
- sparsification, filtering and compression

new generative models



KIT – University of the State of Baden- Württemberg and National Research Center of the Helmholtz Association

model