Introduction | Complex Networks

- non-trivial topological features that do not occur in simple networks (lattices, random graphs) but often occur in reality
  - social networks
  - web graphs
  - internet topology
  - protein interaction networks
  - neural networks
"statistics of relational data"

often

- exploratory in nature
- requires data preprocessing to extract graph
- creates large datasets easily
- requires domain-specific post-processing for interpretation
Introduction | Design Goals

Performance
- implementation with efficiency and parallelism in mind

Interface
- exploratory workflows → freely combinable functions and interactive interface

Integration
- seamless integration with Python ecosystem for scientific computing and data analysis

Target Platforms
- shared-memory parallel computers
- multicore PCs, workstations, compute servers . . .
### Introduction | Overview

<table>
<thead>
<tr>
<th></th>
<th>NetworKit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>language</strong></td>
<td>C++, Python</td>
</tr>
<tr>
<td><strong>interface</strong></td>
<td>object-oriented, functional</td>
</tr>
<tr>
<td><strong>platform</strong></td>
<td>cross-platform</td>
</tr>
<tr>
<td><strong>parallelism</strong></td>
<td>shared memory (OpenMP)</td>
</tr>
<tr>
<td><strong>license</strong></td>
<td>MIT</td>
</tr>
<tr>
<td><strong>first release</strong></td>
<td>1.0 (Mar 2013)</td>
</tr>
<tr>
<td><strong>latest release</strong></td>
<td>3.1 (Apr 2014)</td>
</tr>
<tr>
<td><strong>web</strong></td>
<td><a href="http://parco.iti.kit.edu/software/networkkit.shtml">http://parco.iti.kit.edu/software/networkkit.shtml</a></td>
</tr>
</tbody>
</table>
**Introduction**

**Architecture**

![Architecture Diagram]

NetworKit

**Python**
- Task-oriented Interface
- Pythonized Classes
- Additional Functionality

**Cython**
- Wrapper Classes

**C++ / OpenMP**
- Data Structures
- Algorithms
- I/O
- Tests

Dependencies:
- matplotlib
- scipy
- numpy
- pandas
- networkx
Degree Distribution

Concept

- distribution of node degrees
- typically heavy-tailed (especially power law \( p(k) \sim k^{-\gamma} \))

Algorithm

- `powerlaw` Python module determines whether distribution fits power law and estimates exponent \( \gamma \)

[Alstott et al.2014: powerlaw: a python package for analysis of heavy-tailed distributions.]
[Clauset et al.2009: Power-law distributions in empirical data]
Degree Assortativity

Concept
- prevalence of connections between nodes with similar degree
- expressed as correlation coefficient

Algorithm
- linear ($O(m)$) time and constant memory

Analytics | Diameter

Concept
- longest shortest path between any two nodes

Exact Algorithm
- all pairs shortest path using BFS or Dijkstra

Approximation
- lower and upper bound within an error $\epsilon$

[Magnien et al.2009: Fast computation of empirically tight bounds for the diameter of massive graphs]
Components

Concept

- maximal subgraphs in which all nodes are reachable from each other

Algorithm

- parallel label propagation, accelerated by multi-level technique
Analytics | Cores

Concept

- iteratively peeling away nodes of degree $k$ reveals the $k$-cores

Algorithm

- sequential, $O(m)$ time
Analytics | Clustering Coefficients

Concept

- ratio of closed triangles

Exact Algorithm

- parallel node iterator: $O(nd_{\text{max}}^2)$ time

Approximation

- wedge sampling: linear to constant time approximation with bounded error

[Schank, Wagner 2005: Approximating clustering coefficient and transitivity]
### Eigenvector Centrality / PageRank

#### Concept

- A node’s centrality is proportional to the centrality of its neighbors.
- PageRank theory: probability of a random web surfer arriving at a page.

#### Algorithm

- Parallel power iteration.

[Page et al. 1999: The PageRank citation ranking]
Analytics | Betweenness Centrality

Concept
- a central node lies on many shortest paths

Exact Algorithm
- Brandes’ algorithm: $O(nm + n^2 \log n)$ time

Approximation
- parallel path sampling with probabilistic error guarantee (additive constant)

[Brandes 2001: A faster algorithm for betweenness centrality]
[Riondato, Kornaropoulos 2013: Fast approximation of betweenness centrality through sampling]
Community Detection

- find **internally dense, externally sparse subgraphs**
- goals: uncover community structure, prepartition network

[survey: Schaeffer 07, Fortunato 10]

Modularity

- fraction of intra-community edges minus expected value

[Girvan, Newman 2002: Community structure in social and biological networks]
Community Detection

PLP
- parallel label propagation
- very fast, scalable, low modularity

PLM
- parallel Louvain method
- fast, high modularity

PLMR
- PLM with multi-level refinement
- slightly slower and better than PLM

[Staudt, Meyerhenke 2013: Engineering High-Performance Community Detection Heuristics for Massive Graphs]
etc | Generators

Erdős-Renyi
- random graph, efficient generator

Barabasi-Albert
- power law degree distribution

Chung-Lu & Havel-Hakimi
- replicate input degree distributions

R-MAT
- power law degree distribution, small world-ness, self-similarity
> Demo
Conclusion
Conclusion | Call for Participation

Case studies?
- apply NetworKit to study large complex networks

Working with networks?
- use NetworKit to characterize data sets structurally

Wheel reinvention planned?
- integrate implementations into NetworKit

Teaching graph algorithms?
- use NetworKit as a hands-on teaching tool
Conclusion | Info & Support

Sources

- technical report: arxiv.org/abs/1403.3005
- package documentation
  - Readme
  - User Guide (IPython Notebook)
  - docstrings, Doxygen comments
- e-mail list: networkkit@ira.uni-karlsruhe.de
  - ask us anything (related to NetworKit )
  - stay up to date
Conclusion | Credits

Responsible Developers
- Christian L. Staudt - christian.staudt @ kit.edu
- Henning Meyerhenke - meyerhenke @ kit.edu

Co-Maintainer
- Maximilian Vogel - maximilian.vogel @ student.kit.edu

Contributors
- Miriam Beddig
- Stefan Bertsch
- Andreas Bilke
- Guido Brückner
- Patrick Flick
- Lukas Hartmann
- Daniel Hoske
- Yassine Marrakchi
- Aleksejs Sazonovs
- Florian Weber
- Jörg Weisbarth
- Michael Wegner

Acknowledgements
This work was partially funded through the project Parallel Analysis of Dynamic Networks - Algorithm Engineering of Efficient Combinatorial and Numerical Methods by the Ministry of Science, Research and Arts Baden-Württemberg. A. S. acknowledges support by the RISE program of the German Academic Exchange Service (DAAD).
Thank you for your attention
# NetworKit: An Interactive Tool Suite for High-Performance Network Analysis

## Architecture

### Graph Implementation

```cpp
template<typename L> inline void NetworKit::Graph::parallelForNodes(L handle) {
    #pragma omp parallel for
    for (node v = 0; v < z; ++v) {
        if (exists[v]) {
            handle(v);
        }
    }
}
```

### Graph API

```cpp
std::vector<node> tempMap(G.upperNodeIdBound());
G.parallelForNodes([&](node v){
    tempMap[v] = v; // initialize to identity
});
```

---

*Image: algs4.cs.princeton.edu*