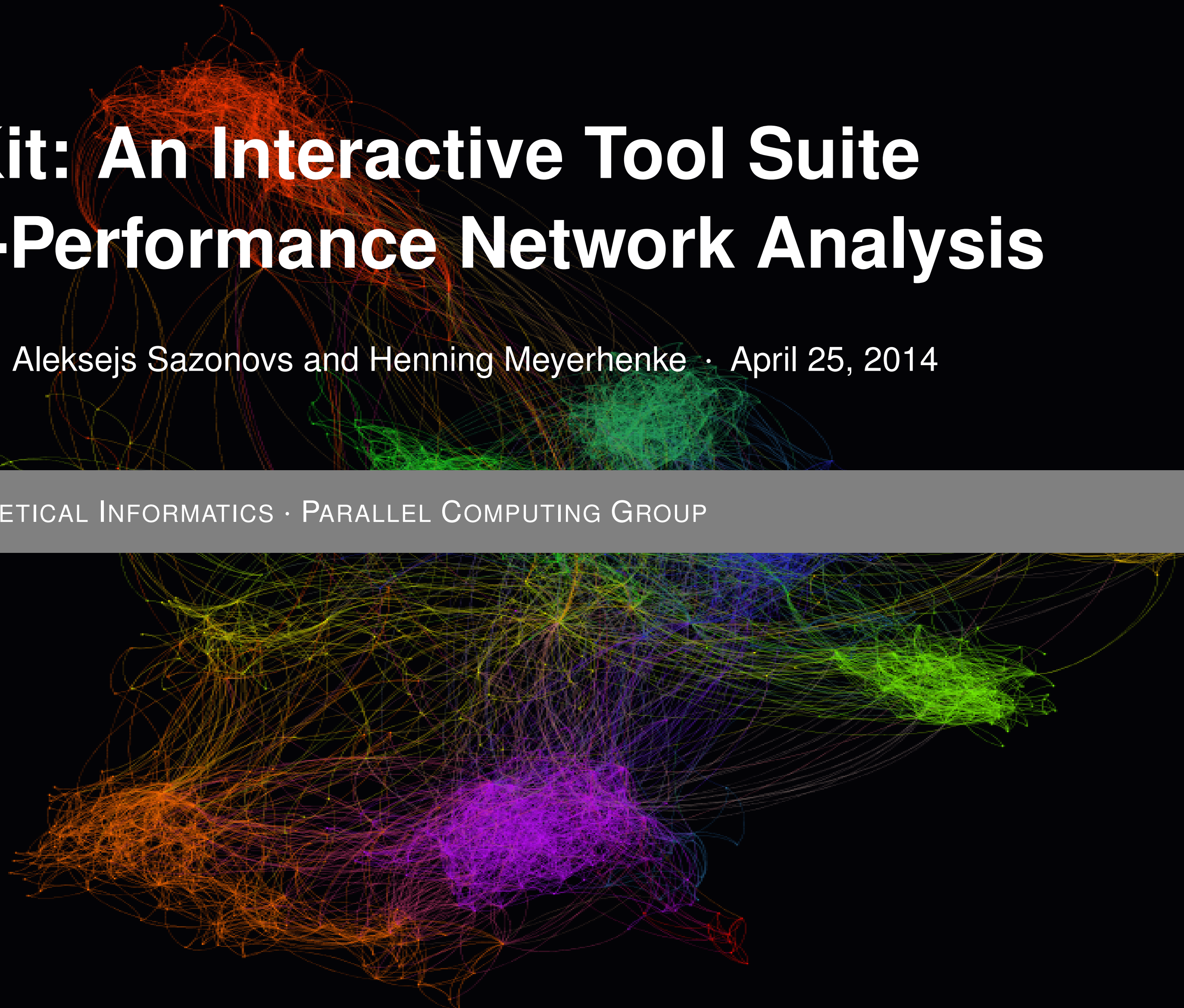
NetworKit: An Interactive Tool Suite for High-Performance Network Analysis

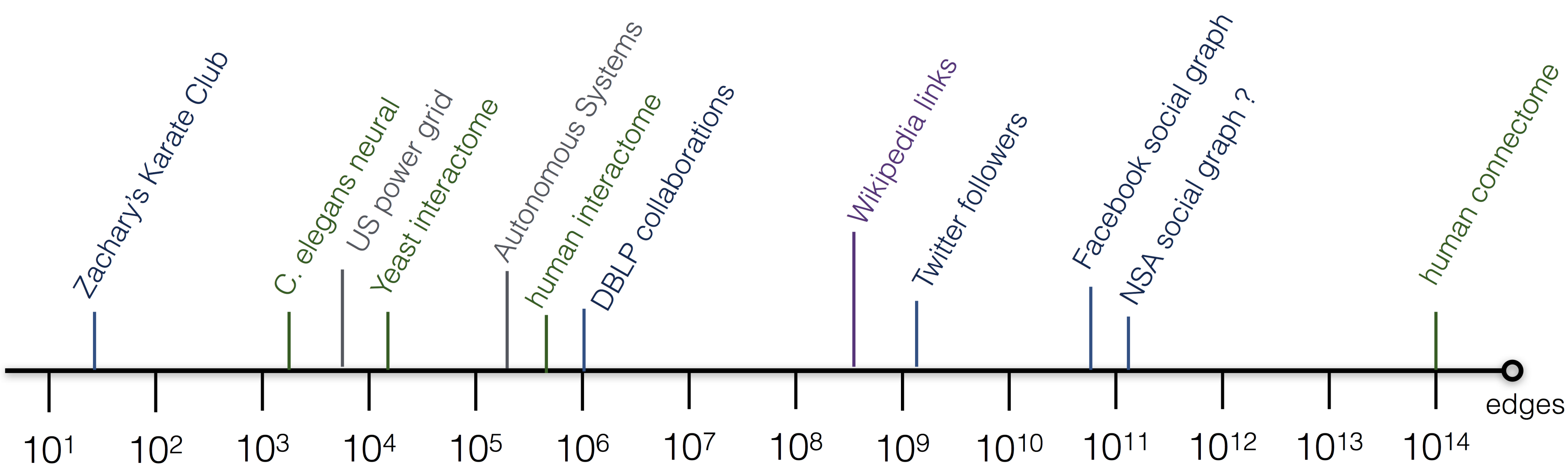
Christian L. Staudt, Aleksejs Sazonovs and Henning Meyerhenke · April 25, 2014

INSTITUTE OF THEORETICAL INFORMATICS · PARALLEL COMPUTING GROUP



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



Network Science

"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

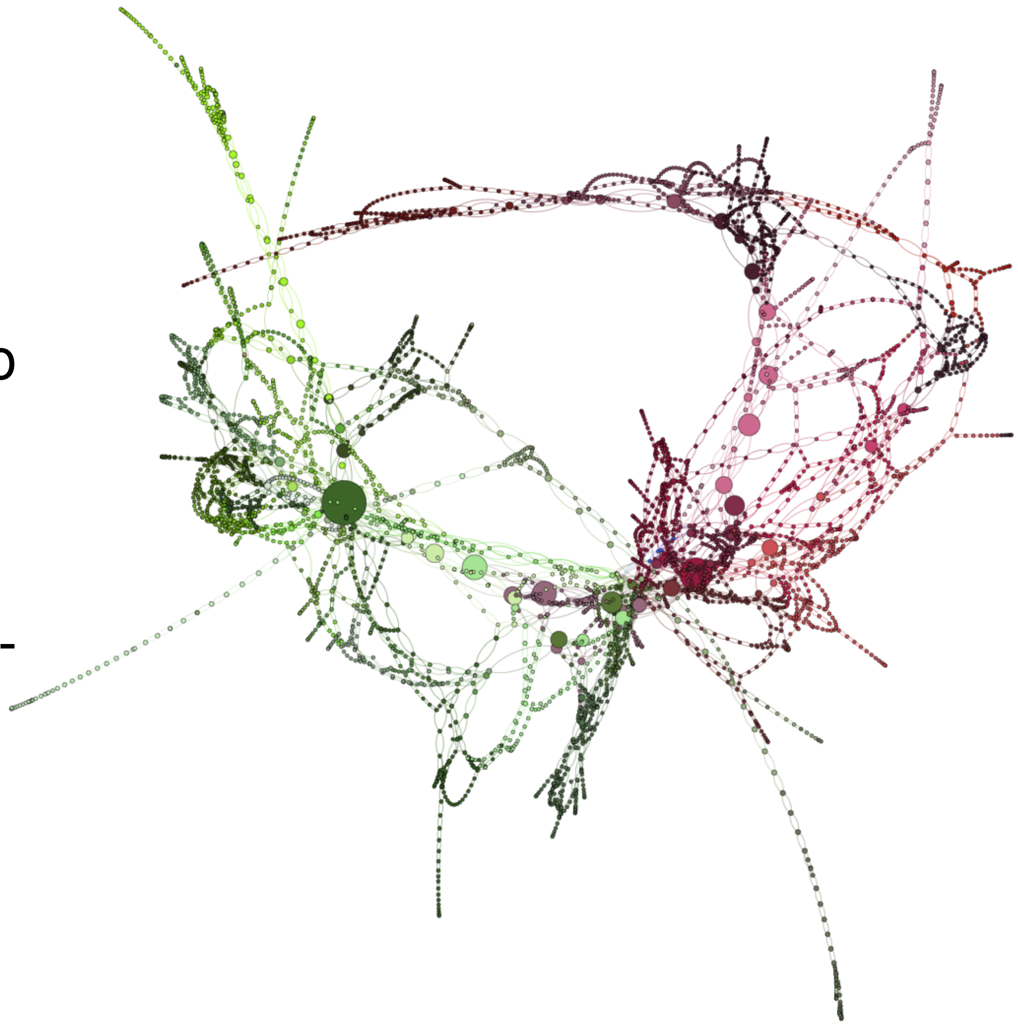


image: sayasaya2011.wordpress.com/

Introduction | Design Goals

Performance

- implementation with efficiency and parallelism in mind

Interface

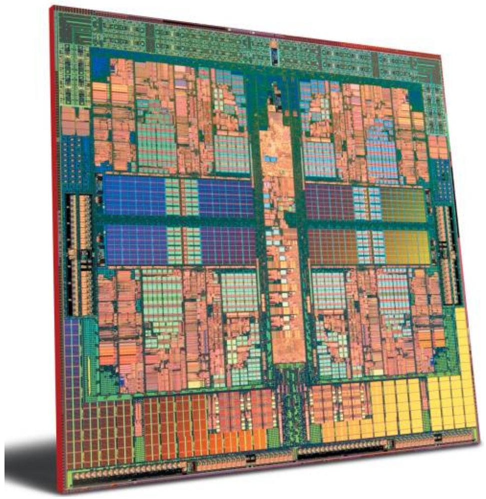
- exploratory workflows $\rightarrow$ freely combinable functions and interactive interface

Integration

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

Target Platforms

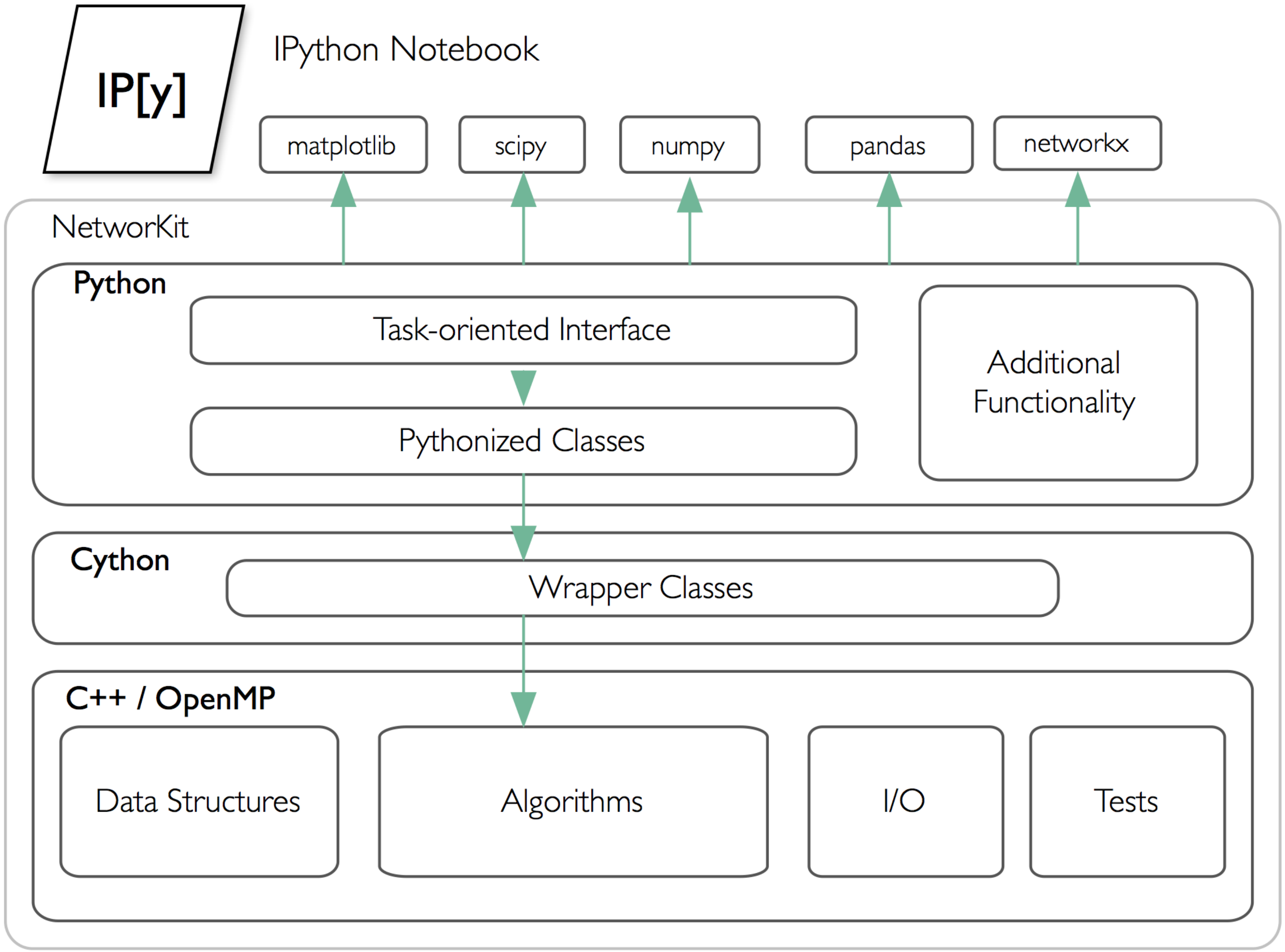
- shared-memory parallel computers
- multicore PCs, workstations, compute servers . . .

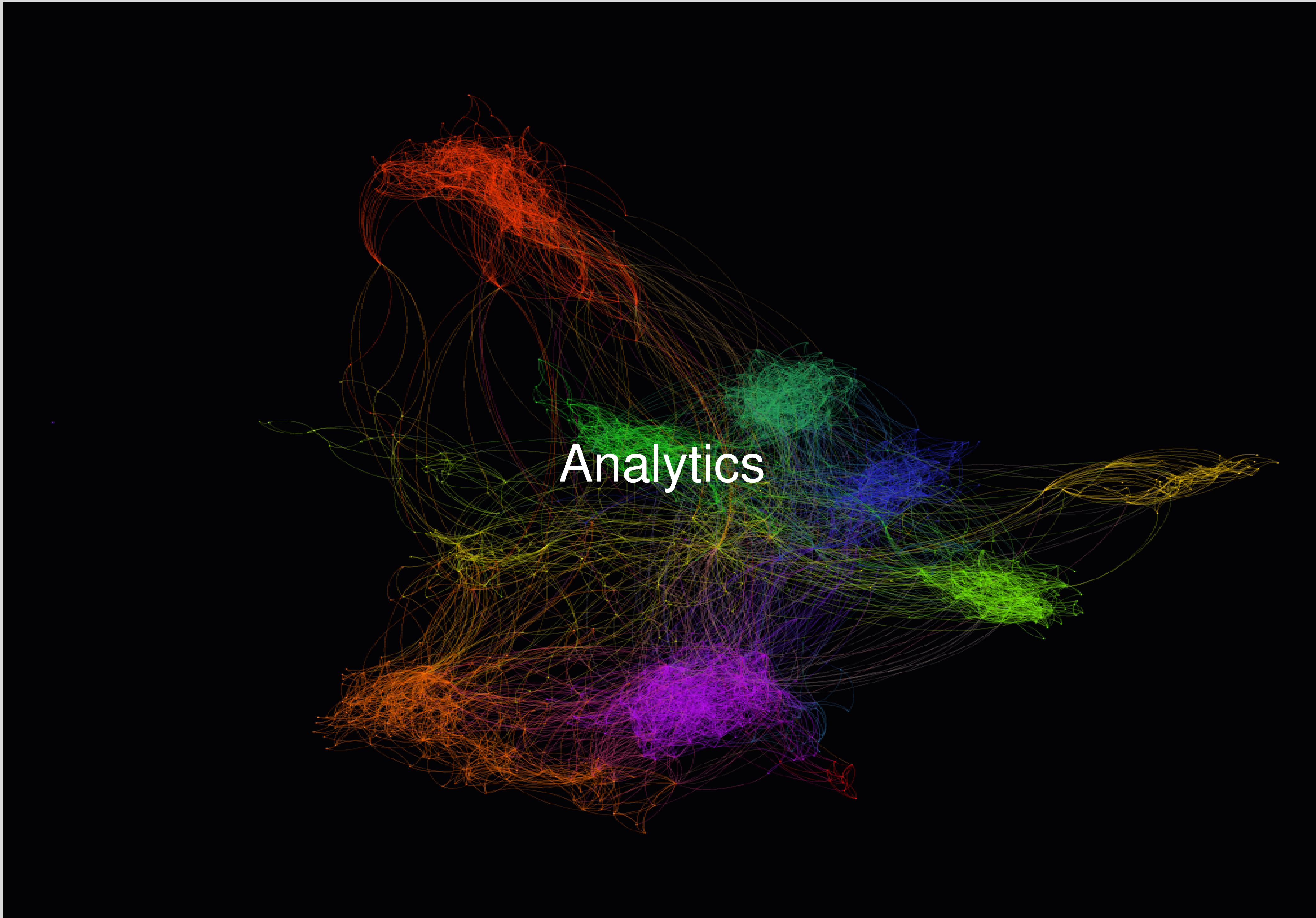


Introduction | Overview

	NetworKit
language	C++, Python
interface	object-oriented, functional
platform	cross-platform
parallelism	shared memory (OpenMP)
license	MIT
first release	1.0 (Mar 2013)
latest release	3.1 (Apr 2014)
web	http://parco.iti.kit.edu/software/networkit.shtml

Karlsruhe Institute of Technology

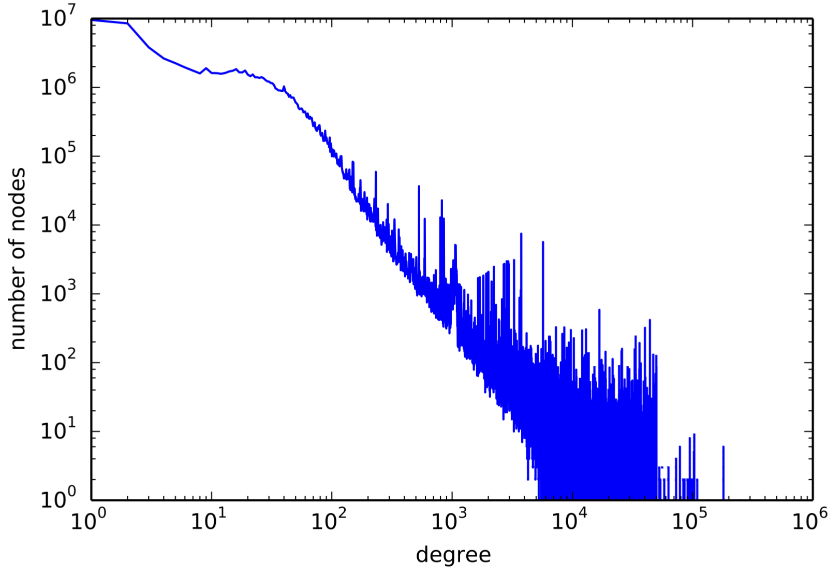



Analytics

Analytics | Degree Distribution

Concept

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



Algorithm

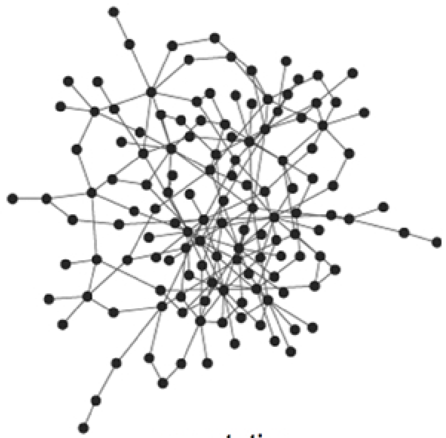
- `powerlaw` Python module determines whether distribution fits power law and estimates exponent γ

[Alstott et al.2014: powerlaw: a python package for analysis of heavy-tailed distributions.]

[Clauset et al.2009: Power-law distributions in empirical data]

Concept

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



assortative

disassortative

Algorithm

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

[Newman 2002: Assortative mixing in networks.]

Analytics | Diameter

Concept

- longest shortest path between any two nodes

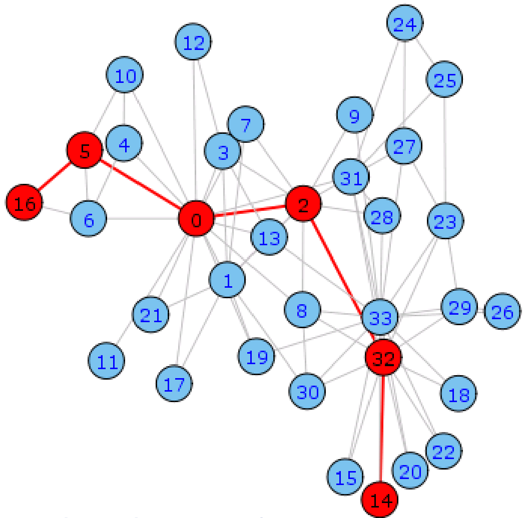


image: igraph.sourceforge.net

Exact Algorithm

- all pairs shortest path using BFS or Dijkstra

Approximation

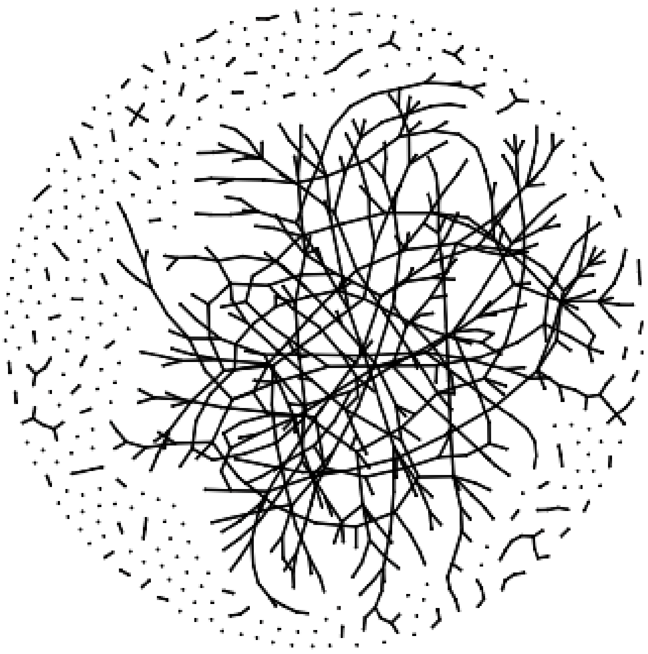
- lower and upper bound within an error ϵ

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

Analytics | Components

Concept

- maximal subgraphs in which all nodes are reachable from eachother



Algorithm

- parallel label propagation, accelerated by multi-level technique

Concept

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

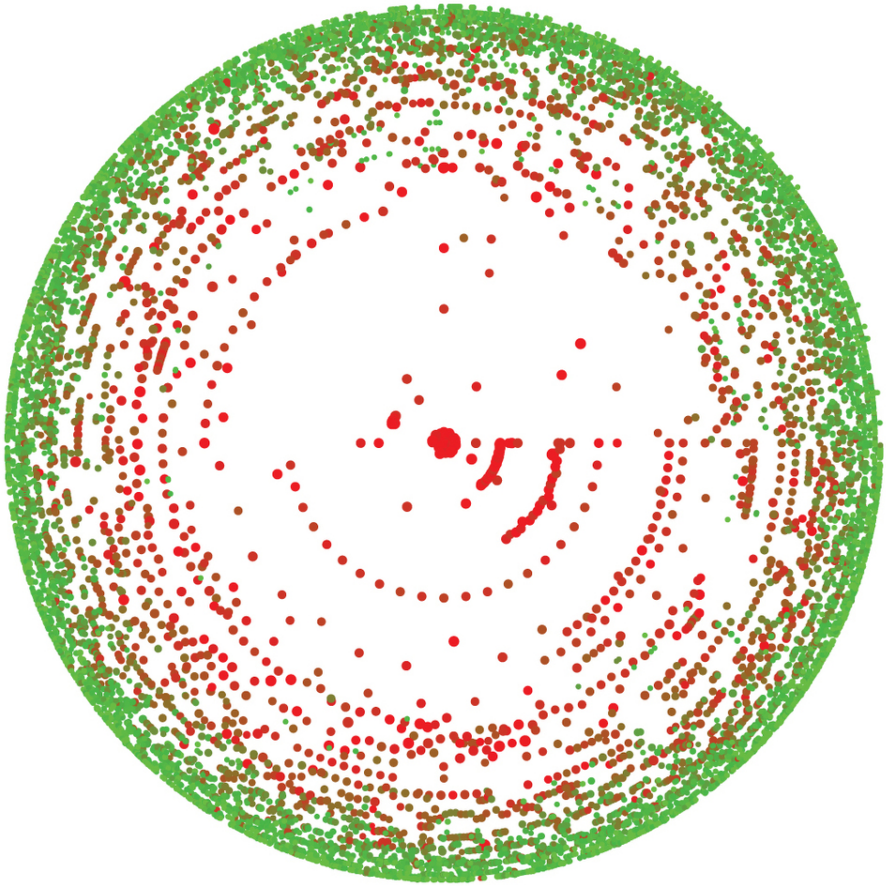


image: Hébert-Dufresne et al.2013

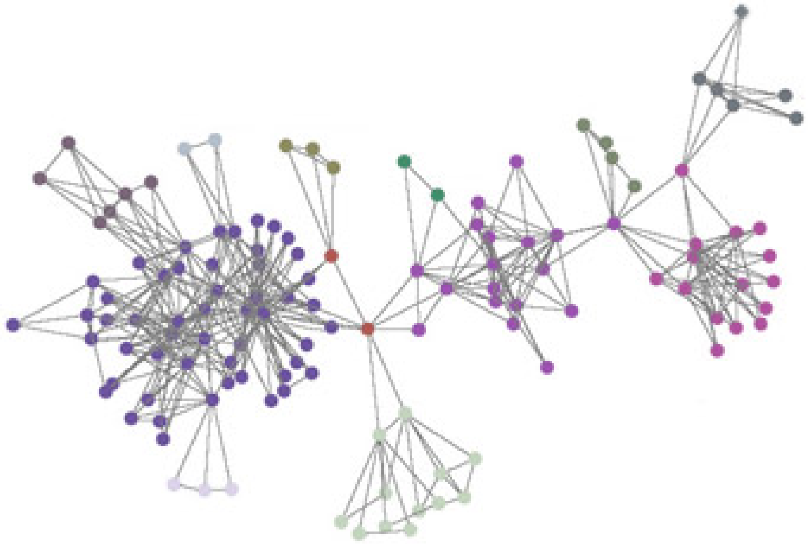
Algorithm

- sequential, $O(m)$ time

Analytics | Clustering Coefficients

Concept

- ratio of closed triangles



Exact Algorithm

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

Approximation

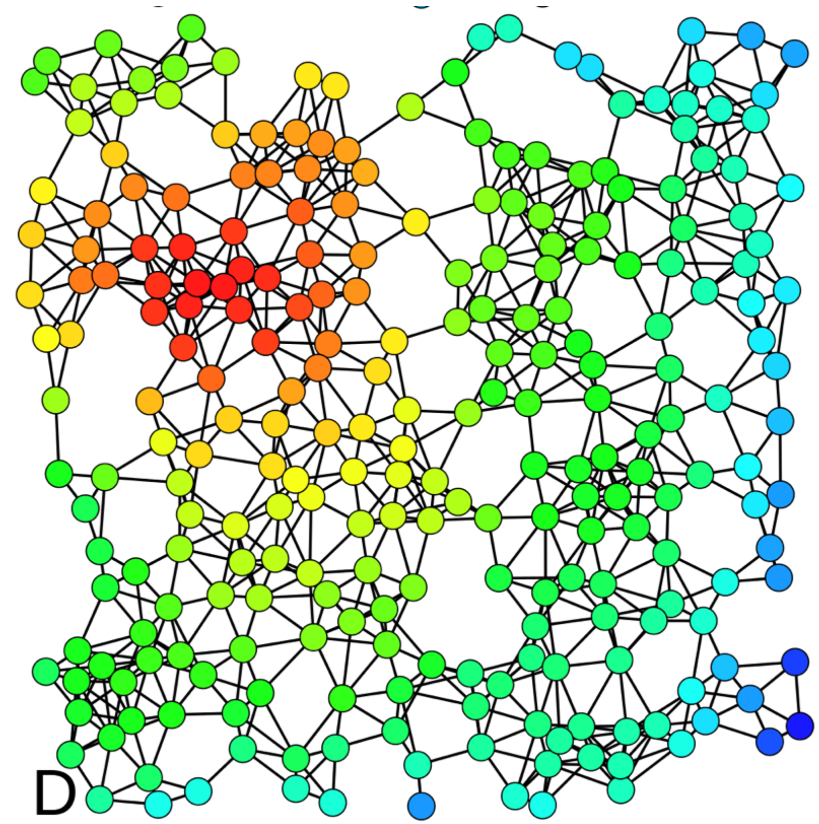
- wedge sampling: linear to constant time approximation with bounded error

[Schank, Wagner 2005: Approximating clustering coefficient and transitivity]

Analytics | 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

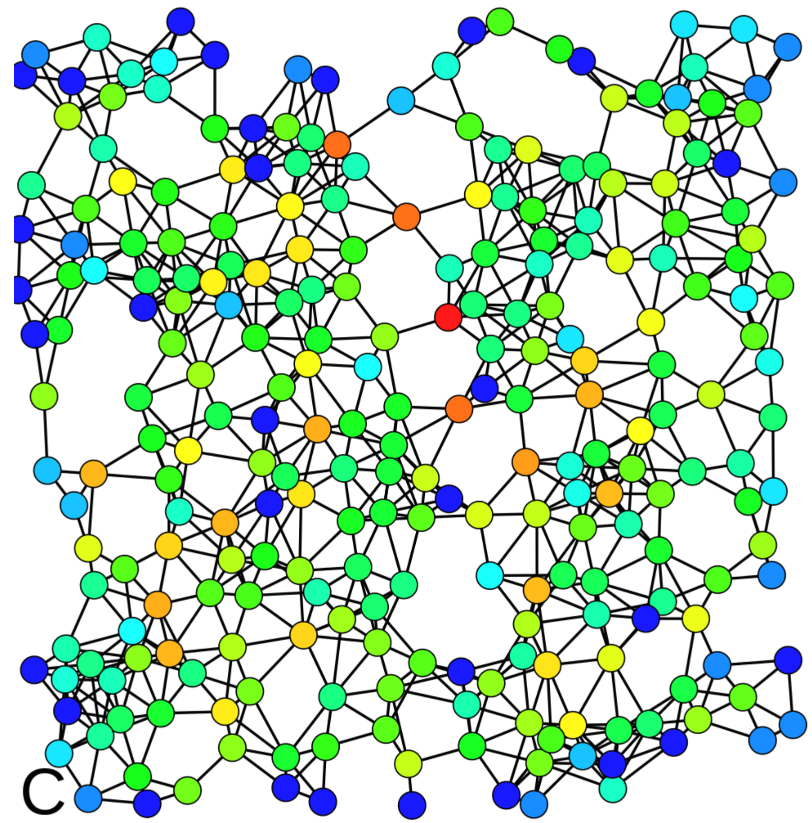
- parallel power iteration

[Page et al.1999: The PageRank citation ranking]

Analytics | Betweenness Centrality

Concept

- a central nodes 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]

Analytics | Community Detection

Community Detection

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

[survey: Schaeffer 07, Fortunato 10]

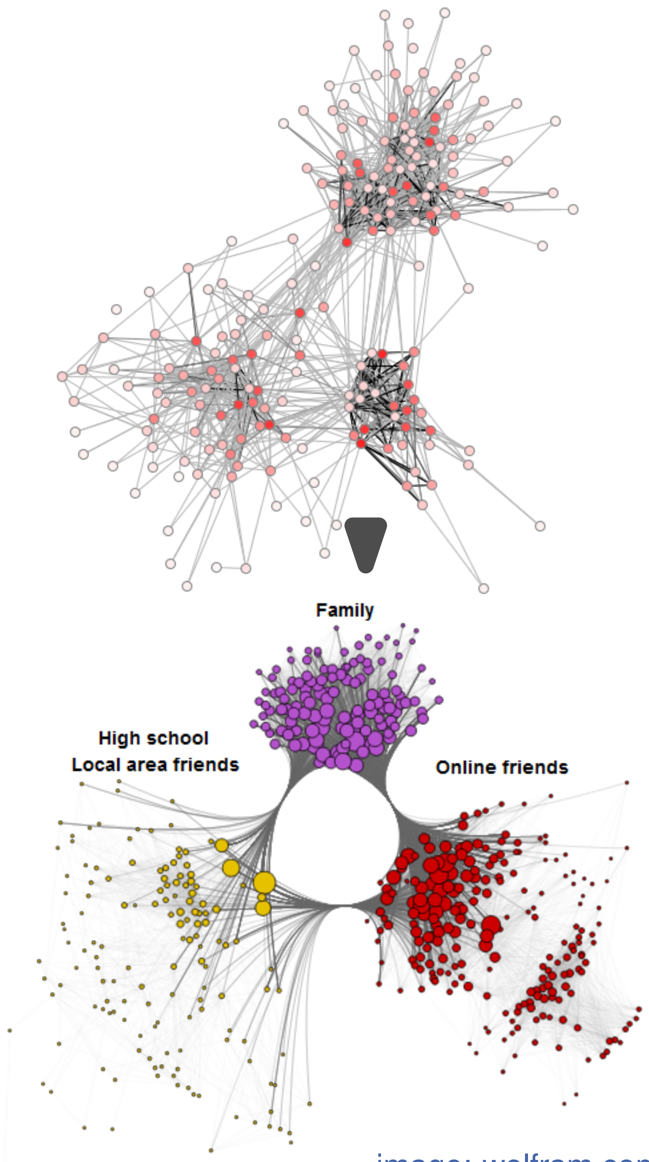


image: wolfram.com

Modularity

- fraction of intra-community edges minus expected value

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

Community Detection

PLP

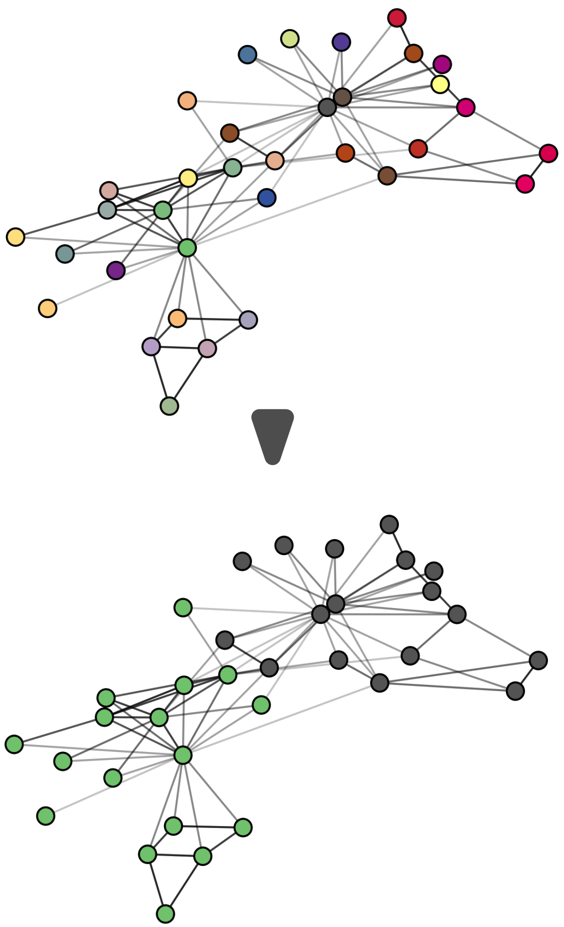
- parallel label propagation
- very fast, scalable, low modularity

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]

Generators

Erdös-Renyi

- random graph, efficient generator

Barabasi-Albert

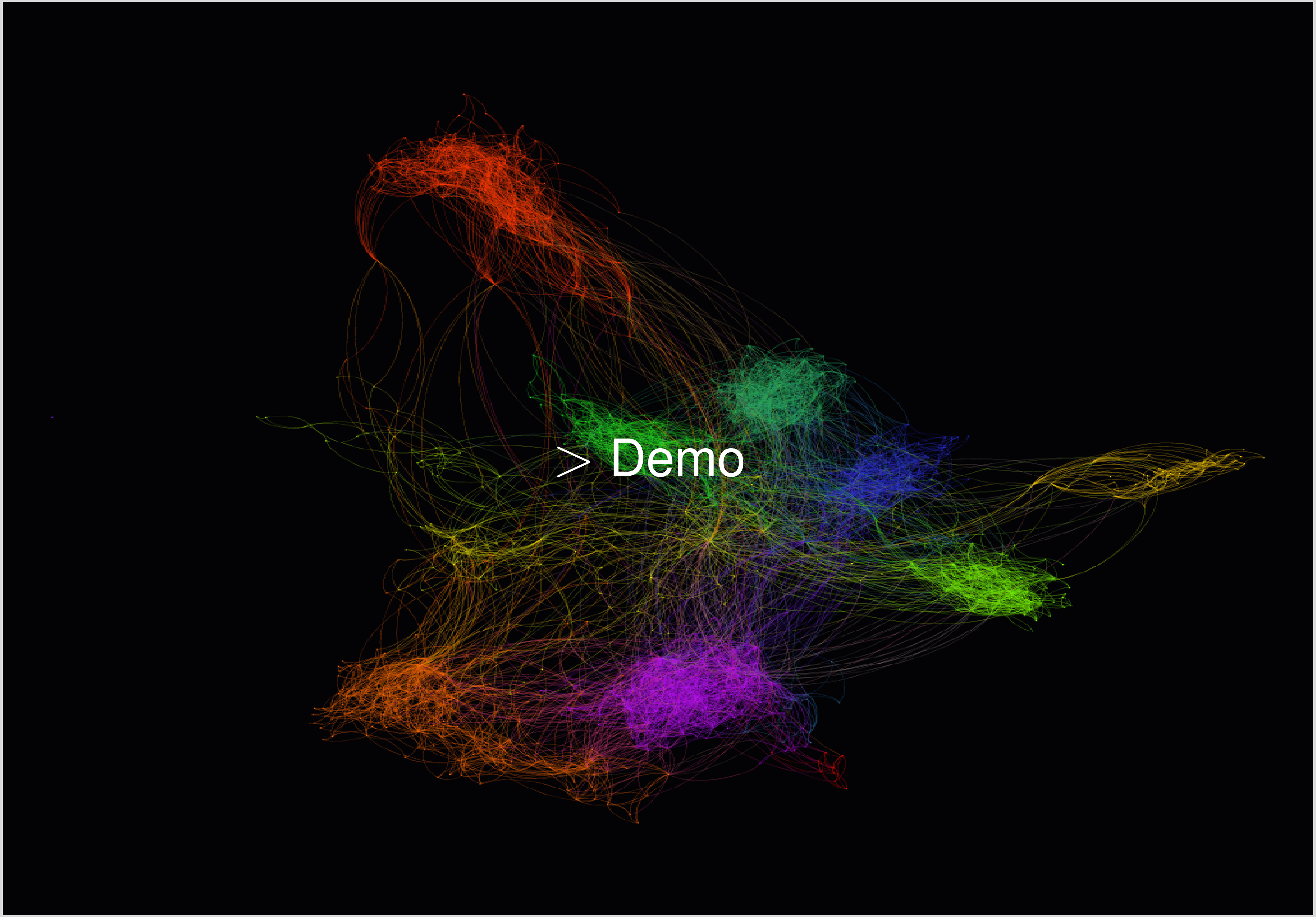
- power law degree distribution

Chung-Lu & Havel-Hakimi

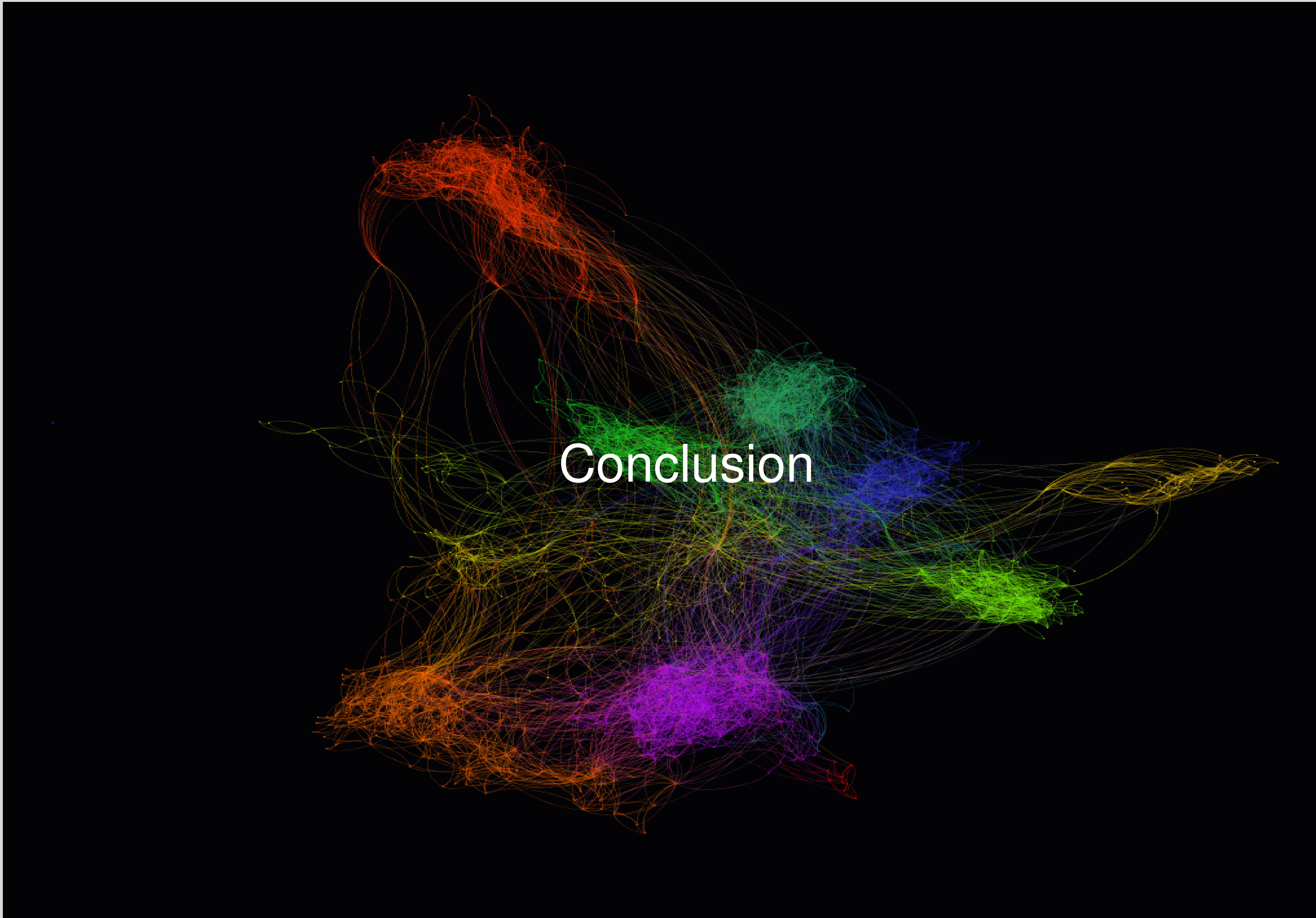
- replicate input degree distributions

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

Sources

- technical report: `arxiv.org/abs/1403.3005`
- package documentation
  - Readme
  - User Guide (IPython Notebook)
  - docstrings, Doxygen comments
- e-mail list: `networkit@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

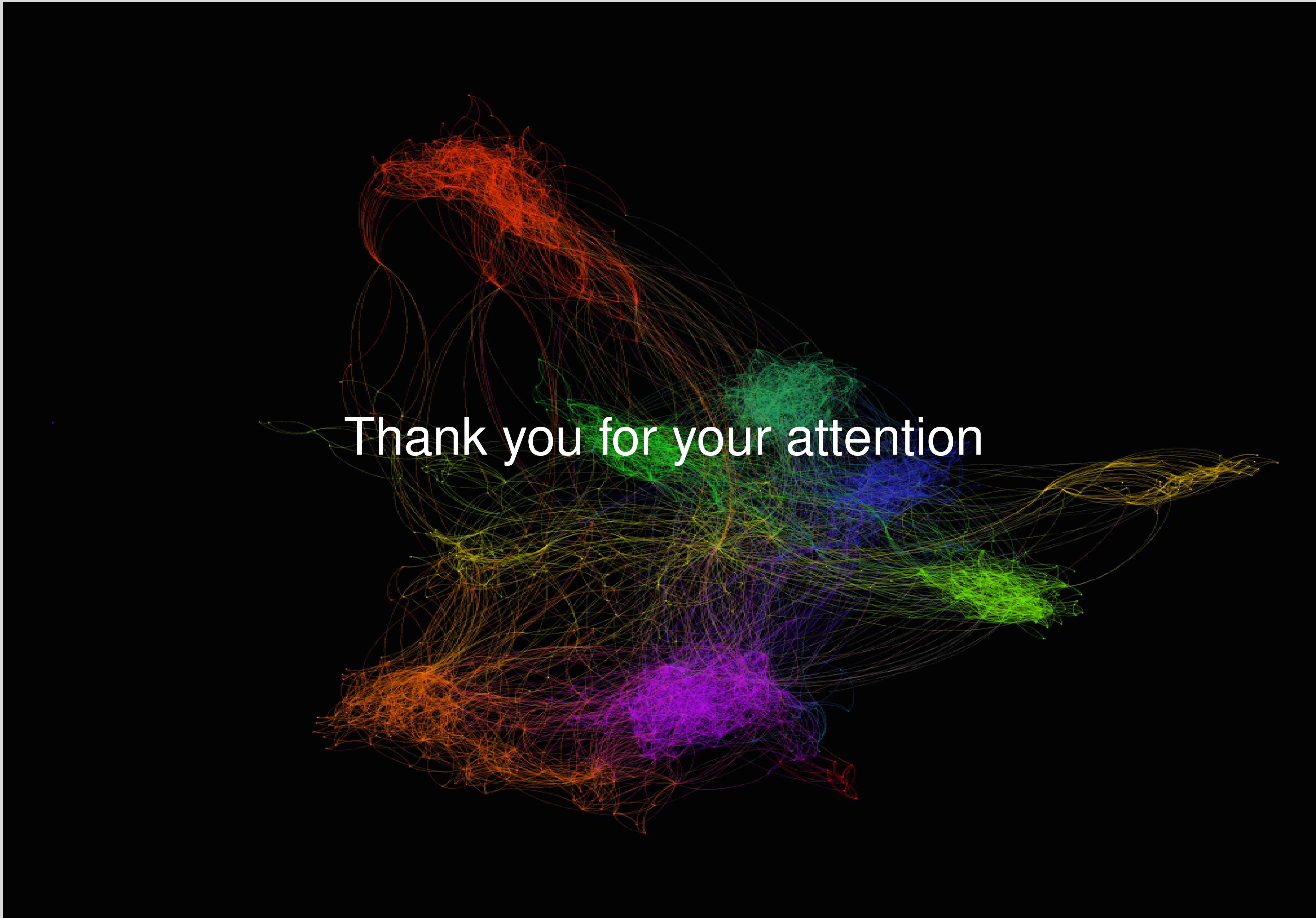
- Maximilian Vogel - `maximilian.vogel @ student.kit.edu`

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).



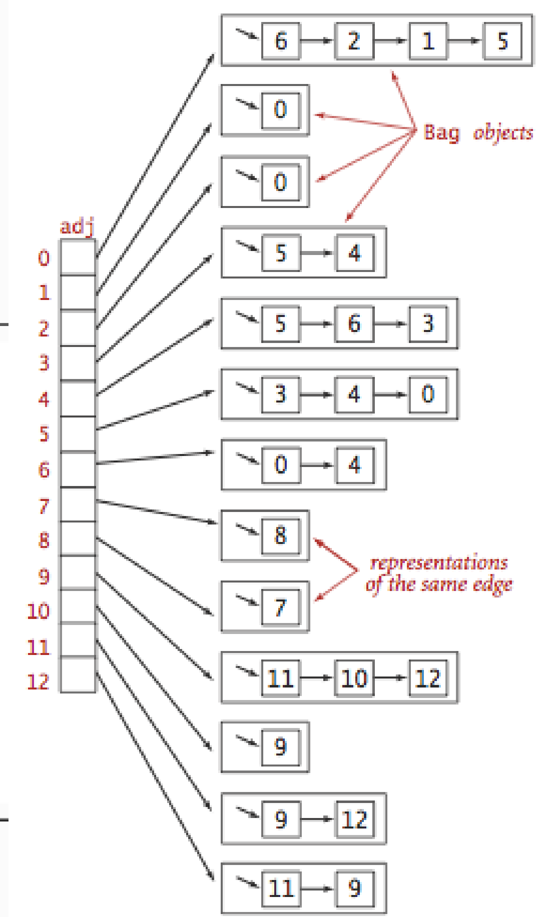
Thank you for your attention

```
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 implementation

graph API

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



Adjacency-lists representation (undirected graph)

image: algs4.cs.princeton.edu