ALGORITHMS AND SOFTWARE FOR THE ANALYSIS OF LARGE COMPLEX NETWORKS

CHRISTIAN STAUDT

Fakultät für Informatik des Karlsruher Instituts für Technologie (KIT)
OUTLINE & CONTRIBUTIONS

main algorithmic contributions

- parallel heuristics for community detection [Staudt, Meyerhenke ‘16, in TPDS]
- network sparsification via edge centrality rating [Hamann, Lindner, Meyerhenke, Staudt, Wagner ‘16, in SNAM]
- realistic synthetic networks via generative models [with Gutfraind, Safro, Hamann, Meyerhenke, unpublished]

software package

- NetworKit [Staudt, Sazonovs, Meyerhenke ‘16, in Network Science]

known and novel algorithms by numerous contributors

data analysis
- inspection
- transformation
- modeling

network science
- analysis of relational data

algorithmics

graph algorithmics

software engineering

this thesis

network science / data science
Part I

INTRODUCTION
network model formation [Brandes et al. 13]

\[ G = (V, E, \omega) \]
| \[ |V| = n, |E| = m \] |

\[ V = N \]
\[ E = \{(u, v) \in D : x(u, v) \notin \{0, \infty\}\} \]
\[ \omega(u, v) = x(u, v) \text{ if } (u, v) \in E \]

example: Dolphin social network study [Lusseau et al. ‘03]
Part II

NETWORKKIT: A TOOL SUITE FOR THE ANALYSIS OF LARGE COMPLEX NETWORKS
NETWORKIT

state of the art & targeted improvement
• various graph processing or network analysis libraries exist (e.g. Boost Graph, JUNG, NetworkX…) 
• increase performance, parallelism, focus on network science, …
• provide short path from algorithms research to data analysis applications

contribution
• NetworKit, an open-source tool suite for the analysis of large networks
  [Staudt, Sazonovs, Meyerhenke ’16 in Network Science, to appear]
PRINCIPLES AND ARCHITECTURE

**design goals**
- performance
- usability and integration

**algorithm and implementation patterns**
- parallelism
- heuristics and approximation algorithms
- modular design

---

**Data Structures**

- C++ / OpenMP
  - Algorithms
  - I/O

**Python**

- Task-oriented Interface
- Pythonized Classes
- Additional Functionality

**Cython**

- Wrapper Classes

**Python shell / program**

- NetworKit

**ext. Python modules**

- pandas
- numpy
- matplotlib

**Additional Functionality**

- Pythonized Classes
- Task-oriented Interface

---

Cython

**Wrapper Classes**

**Python shell / program**

**NetworKit**
USE CASES

as algorithm library

as interactive data analysis tool (e.g. via Jupyter Notebook)

e.g. as component in a protein network analysis pipeline [Flick ‘14]

e.g. network profiles for explorative network analysis, here: connectome network [Staudt et al. ‘16]
COMPARISON AND EVALUATION

**comparative benchmark**

- consistently fastest average processing rate for typical analysis kernels on varied set of networks
Part III

COMMUNITY DETECTION IN COMPLEX NETWORKS
COMMUNITY DETECTION

**community detection**
- reveal modular composition of network by finding *internally dense, externally sparse subgraphs* (communities)
- community: vague concept, formalized via e.g. objective function *modularity*

**modularity**
- optimization NP-hard [Brandes et al. ‘08]
- parameter-free, well understood -> commonly applied for explorative network analysis

\[
M_{\gamma}(G, \zeta, \gamma) := \sum_{C \in \zeta} \frac{|E(C)|}{|E|} - \gamma \cdot \sum_{C \in \zeta} \frac{\left(\sum_{v \in C} deg(v)\right)^2}{(2 \cdot |E|)^2}
\]

(Dolphins social network - node color by modularity-based communities)

(multi-resolution) modularity ([Lambiotte ‘10]) [Girvan & Newman ‘02]
ENGINEERING PARALLEL ALGORITHMS FOR COMMUNITY DETECTION

state of the art & targeted improvement
• O(m)-time heuristics for modularity maximization exist, but few parallel algorithms (10th DIMACS Challenge 2012 [Bader et al. ’13])
• desired: implementation that robustly processes billion-edge graphs on typical multicore computer in minutes

contribution
• two fast, robust & scalable parallel heuristics: PLP & PLM
• PLM: best quality/running time tradeoff in experimental comparison with current competitors [Staudt & Meyerhenke ’13 at ICPP] [Staudt & Meyerhenke ’16 in TPDS]
PLM: PARALLEL LOUVAIN METHOD

predecessor

• sequential locally greedy multilevel algorithm [Blondel et al. ’08]
• optional extension: additional refinement phase (PLMR) [Rotta & Noack ’11]

our improvement

• first parallelization

initialize to singletons
repeat
   while communities not stable do
      parallel for $v \in V$
         move $v$ to neighbor community for max. $\Delta mod$
      endfor
   end
   coarsen graph
until no change in communities
return communities induced by coarsest graph

move phase
coarsening phase
prolongation (+ refinement phase)
parallelization issues and solutions

- lock-free parallelisation where possible
- move/refinement phase
  - race conditions? yes, but mitigated by self-correcting iterative algorithm
  - locks only on update of the volume of communities

\[
\Delta mod(u, C \rightarrow D) = \frac{\omega(u, D \setminus \{u\}) - \omega(u, C \setminus \{u\})}{\omega(E)} + \frac{(vol(C \setminus \{u\}) - vol(D \setminus \{u\})) \cdot vol(u)}{2 \cdot \omega(E)^2}
\]
实验评估
- 运行时间和解的质量（模块性）在不同网络集上测量。
- 性能和扩展行为通过算法工程优化。

PLM：300亿边网络图的强大扩展。

Pareto评估与来自第10届DIMACS挑战的代码比较[Bader et al. ’13]和其他。
Part IV

EDGE CENTRALITY MEASURES FOR NETWORK SPARSIFICATION
EXAMPLES

(a) original  (b) Random Edge  (c) Edge Forest Fire

(d) Local Similarity  (e) Triangular Simmelian Backbone  (f) Local Degree
SPARSIFICATION

(edge) sparsification
• reduce the edge set of a network while preserving important structural properties

state of the art & targeted improvement
• numerous sparsification methods proposed, but lack of comparative work and unifying concepts

contributions
• conceptual framework: network sparsification as edge centrality rating and filtering
• first comparative study of various edge centrality measures for the purpose of structure-preserving network sparsification
• an effective novel method
• (parallelized) efficient implementations
  [Lindner, Staudt, Hamann, Meyerhenke, Wagner ’15 at ASONAM]
  [Hamann, Lindner, Meyerhenke, Staudt, Wagner ’16 in SNAM]

\[ G = (V, E) \rightarrow \text{edge score calculation} \rightarrow \text{edge filtering} \rightarrow G' = (V, E') \]
EVALUATION

**conclusions**
- class of methods (Simmelian Backbones, Jaccard Similarity, Algebraic Distance) that effectively preserves community structures
- novel method Local Degree preserves shortest paths, connectivity, many centralities
- local filtering improves the preservation of almost all properties (diameter, centralities, …)

**experimental evaluation**
- quantify how structural properties vary with decreasing edge ratio
- network set: >100 social graphs (Facebook and others)
Part V

GENERATIVE MODELS FOR REALISTIC SYNTHETIC NETWORKS
GENERATING SCALED REPLICA OF REAL-WORLD NETWORKS

state of the art
• large variety of generative models with varying degrees of (claimed) realism

motivation
• algorithm engineering: given a small real network, generate realistic (scaled) replicas to enable representative experiments on larger data sets

contributions
• fitting schemes for a variety of generative models
• experimental study clarifying the degree of realism of various models
• LFR+ generator, an effective tool for creating realistic (scaled) replicas of networks
  [current joint work with Gutfraind, Safro, Meyerhenke, Hamann, unpublished]
EXAMPLE REPLICATION

original dolphins social network

consider key structural features...

- degrees
- connectedness
- clustering
- community structure
- ...

models (& generator algorithms)

- [Erdös & Rényi '60]([Batagelj & Brandes ‘05])
- [Barabasi & Albert ‘02]
- [Chung et al. ‘00]
- Edge-Switching Markov Chain
  [Milo et al. ‘03]
- RMAT [Chakrabarti et al. ‘04]
- Hyperbolic Unit-Disk Graph
  [Krioukov et al. ‘10][Looz et al. ‘15]
- BTER [Kolda et al. ‘13]
- LFR [Lancichinetti et al. ‘08]
LFR+ GENERATOR: EXAMPLE REPLICATION

epidemiological contact network used in HIV research [Potterat et al. '02]

scale-2 replica produced by LFR+ generator

sample from scale-200k replica produced by LFR+ generator
SUMMARY

all contributions

• Part II
  • NetworKit
    [Staudt, Sazonovs, Meyerhenke ‘16, in Network Science, to appear]
    • network analysis on distributed systems
      [Koch, Staudt, Vogel, Meyerhenke ’15 at FAB]

• Part III
  • parallel heuristics for community detection
    [Staudt, Meyerhenke ‘13, at ICPP]
    [Staudt, Meyerhenke ‘16, in TPDS]
  • heuristics for selective community detection
    [Staudt, Marrakchi, Meyerhenke ’14 at IEEE BigData]

• Part IV
  • network sparsification via edge centrality rating
    [Lindner, Staudt, Hamann, Meyerhenke, Wagner ‘15, at ASONAM]
    [Hamann, Lindner, Meyerhenke, Staudt, Wagner ‘16, in SNAM]

• Part V
  • realistic synthetic networks via generative models
    [with Gutfraind, Safro, Hamann, Meyerhenke, unpublished]
Appendix
CHARACTERIZING THE STRUCTURE OF NETWORKS

**distance**
- e.g. diameter, algebraic distance

**node centrality**
- e.g. degree, betweenness, PageRank

**edge centrality**
- e.g. edge betweenness, -> sparsification

**partitioning**
- e.g. components, k-cores, communities

**correlations**
- e.g. assortativity

**emergent properties**
- e.g. epidemic spreading

Dolphins social network - betweenness
Dolphins social network - k-core decomposition
std::vector<double> oldLoads(loads.size());

for (index iter = 0; iter < numIters; ++iter) {
    loads.swap(oldLoads); // store previous iteration

    G.balancedParallelForNodes([&](node u) {
        std::vector<double> val(numSystems, 0.0);
        double weightedDeg = 0;
        // step 1
        G.forNeighborsOf(u, [&](node v, edgewidth weight) {
            for (index i = 0; i < numSystems; ++i) {
                val[i] += weight * oldLoads[v*numSystems + i];
            }
            weightedDeg += weight;
        });

        for (index i = 0; i < numSystems; ++i) {
            val[i] /= weightedDeg;
            // step 2
            loads[u*numSystems + i] = (1 - omega) * oldLoads[u*numSystems + i] + omega * val[i];
        }
    });
}

Figure 16: Code example: Parallel computation of algebraic distances.
## FUNCTIONALITY

<table>
<thead>
<tr>
<th>module</th>
<th>description</th>
<th>example components</th>
</tr>
</thead>
<tbody>
<tr>
<td>centrality</td>
<td>node centrality measures</td>
<td>centrality.Betweenness</td>
</tr>
<tr>
<td>community</td>
<td>community detection</td>
<td>community.PLM</td>
</tr>
<tr>
<td>components</td>
<td>connected components</td>
<td>components.ConnectedComponents</td>
</tr>
<tr>
<td>correlation</td>
<td>correlations</td>
<td>correlations.Assortativity</td>
</tr>
<tr>
<td>distance</td>
<td>distance measures</td>
<td>distance.Diameter</td>
</tr>
<tr>
<td>generators</td>
<td>generative models</td>
<td>generators.ErdosRenyiGenerator</td>
</tr>
<tr>
<td>graph</td>
<td>graph API</td>
<td>graph.Graph</td>
</tr>
<tr>
<td>linkprediction</td>
<td>link prediction algorithms</td>
<td>linkprediction.JaccardIndex</td>
</tr>
<tr>
<td>profiling</td>
<td>network profiling tool</td>
<td>profiling.Profile</td>
</tr>
<tr>
<td>simulation</td>
<td>simulations</td>
<td>simulation.EpidemicSimulationSEIR</td>
</tr>
</tbody>
</table>

examples of NetworKit’s modules and components
**conclusions**

- good scaling can be due to distributing own overheads [McSherry et al '15]
- distributed frameworks have significant overheads, so their application should be motivated by lack of memory on a single node
PLP: PARALLEL LABEL PROPAGATION

**predecessor**
- sequential label propagation algorithm [Raghavan et al. ‘07]
- a local coverage maximizer (implicitly maximizes modularity by getting stuck in local optima)

**our improvement**
- parallelization & optimizing heuristics

```plaintext
initialize nodes with unique labels
while labels not stable do
    parallel for \( v \in V \)
    | adopt dominant label in \( N(v) \)
endfor
return communities from labels
```
**SELECTIVE COMMUNITY DETECTION**

**task**
- given a set of seed nodes, detect the communities that contain them

**state of the art & targeted improvement**
- plethora of objective functions and heuristics proposed
- lack of comparative work

**contributions**
- comparative study clarifying the real-world performance of existing and novel algorithms
- Greedy Community Expansion: generic greedy algorithm for the incremental maximization of different objective functions, subsuming several previous efforts
- SelSCAN: application of density-based clustering to SCD

density-based community
**EDGE CENTRALITY MEASURES FOR SPARSIFICATION**

**methods**
- Random Edge (RE)
- Triange Count (Tri)
- Jaccard Similarity (JS) [Satuluri et al. ’11]
- (Quadrilateral or Triadic) Simmelian Backbones (TS, QLS) [Nick et al. ’13][Nocaj et al. ’14]
- Edge Forest Fire (EFF) [Leskovec et al. ’06]
- Algebraic Distance (AD) [Chen & Safro ’11]
- Local Degree (LD)

<table>
<thead>
<tr>
<th>MCD</th>
<th>0.4</th>
<th>0.46</th>
<th>0.39</th>
<th>0.38</th>
<th>0.42</th>
<th>0.39</th>
<th>0.44</th>
<th>0.41</th>
<th>0.24</th>
<th>-0.13</th>
<th>0.026</th>
<th>-0.025</th>
<th>-0.00022</th>
<th>0.013</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD</td>
<td>0.74</td>
<td>0.38</td>
<td>0.37</td>
<td>0.37</td>
<td>0.37</td>
<td>0.4</td>
<td>0.39</td>
<td>0.31</td>
<td>-0.14</td>
<td>-0.075</td>
<td>-0.087</td>
<td>0.0016</td>
<td>-0.0094</td>
<td></td>
</tr>
<tr>
<td>Tri</td>
<td>0.36</td>
<td>0.44</td>
<td>0.4</td>
<td>0.45</td>
<td>0.42</td>
<td>0.47</td>
<td>0.21</td>
<td>-0.17</td>
<td>0.046</td>
<td>-0.018</td>
<td>-0.00011</td>
<td>0.021</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>0.75</td>
<td>0.83</td>
<td>0.84</td>
<td>0.7</td>
<td>0.93</td>
<td>0.77</td>
<td>0.81</td>
<td>-0.19</td>
<td>-0.15</td>
<td>-0.18</td>
<td>0.0002</td>
<td>-0.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QLS</td>
<td>0.88</td>
<td>0.85</td>
<td>0.76</td>
<td>0.68</td>
<td>-0.13</td>
<td>-0.11</td>
<td>-0.14</td>
<td>3.2e-05</td>
<td>-0.017</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LTS</td>
<td>0.76</td>
<td>0.84</td>
<td>0.48</td>
<td>-0.19</td>
<td>0.034</td>
<td>-0.028</td>
<td>-3.4e-05</td>
<td>0.015</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLS</td>
<td>0.88</td>
<td>0.71</td>
<td>-0.18</td>
<td>-0.059</td>
<td>-0.11</td>
<td>9.2e-05</td>
<td>-0.011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LQS</td>
<td>0.53</td>
<td>-0.19</td>
<td>0.05</td>
<td>-0.017</td>
<td>-9.5e-05</td>
<td>0.017</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tri</td>
<td>0.21</td>
<td>-0.51</td>
<td>-0.4</td>
<td>6.5e-05</td>
<td>-0.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

edge rank correlations
**EDGE CENTRALITY: LOCAL DEGREE**

**motivation**
- hierarchy of hub nodes (with high degree) is important for real-world network's connectivity (esp. shortest paths)

**measure**
- retain edges to top \( \lfloor \text{deg}(u)^\alpha \rfloor \) neighbors by degree
- time: \( O(m \log(d_{\text{max}})) \)

**experimental results**
- preserves wide range of properties (connectivity, shortest paths, centralities, epidemic simulation behavior)
- can be strongly correlated with edge betweenness - depending on the network
- (edge betweenness: \( O(nm) \) time)
SPARSIFICATION BY LOCAL FILTERING

\[ G = (V, E) \rightarrow \text{edge score calculation} \]
\[ \text{score transformation} \rightarrow G' = (V, E') \]

edge centrality measure \( c\{u, v\} \)

keep top \([\text{deg}(u)^\alpha], \quad \alpha \in [0, 1]\)
edges incident to every node…

... by applying global threshold to transformed scores \( l_c\{u, v\}\)

disadvantages

improves preservation of all properties (by preserving connectivity)

accommodates structurally heterogeneous regions in networks

advantages

Quadrilateral Simmelian Backbone (QLS) [Nocaj et al. ’14]

Quadrilateral Simmelian Backbone with local filtering (LQLS)
REALISTIC SCALING OF NETWORKS

Figure 69: Scaling behavior of 100 Facebook networks
**LFR+ ALGORITHM**

**predecessor**
- LFR generator for community detection benchmarking [Lancichinetti, Fortunato, Radicchi ’08]

**our modification**
- utilize core algorithm but accept more general input, fitted to original graph
- -> increased flexibility and realism of the replica

---

1. assign degrees to nodes
2. assign sizes to communities
3. assign nodes randomly & iteratively to communities so that intra-community degrees are satisfied
4. connect graph using Edge-Switching Markov Chain Generator model (one graph per community, one global graph, rewiring step to remove additional intra-community edges)
RUNNING TIME REPLICATION

Are algorithm running times obtained on synthetic graphs representative for those on real-world inputs?