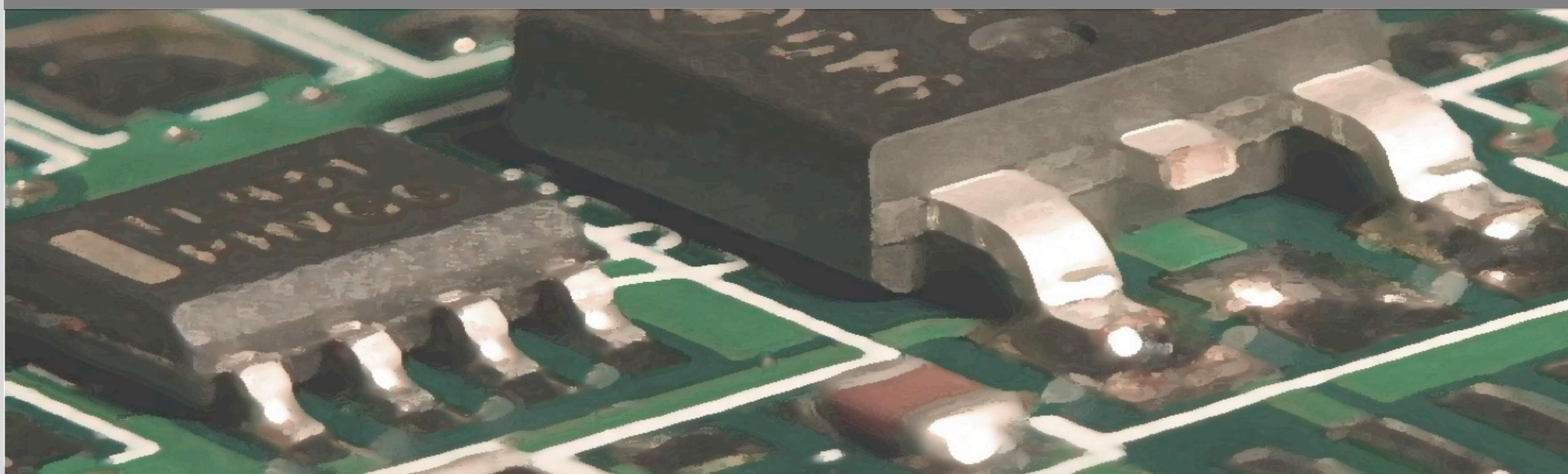
Karlsruhe Institute of Technology

Combinatorial and Numerical Algorithms for Network Analysis

Henning Meyerhenke and Christian Staudt

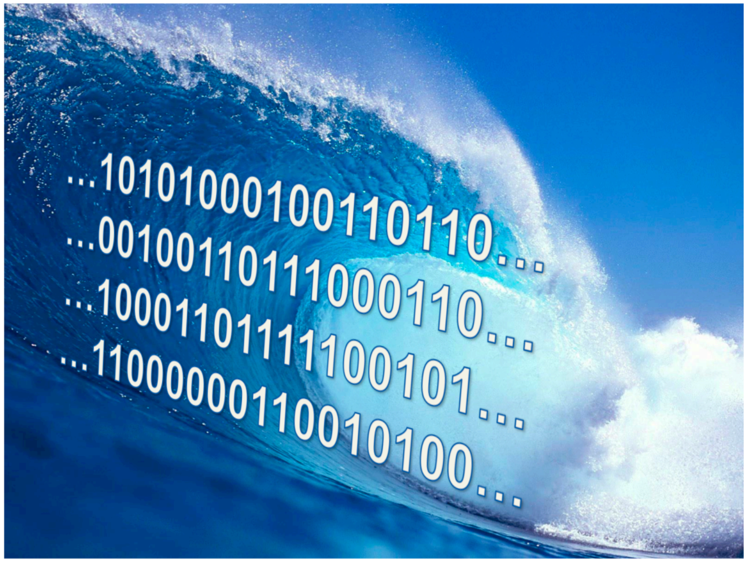
Institute of Theoretical Informatics
Karlsruhe Institute of Technology (KIT), Germany

Boston, MA · February 2013 · SIAM CSE 2013



Introduction

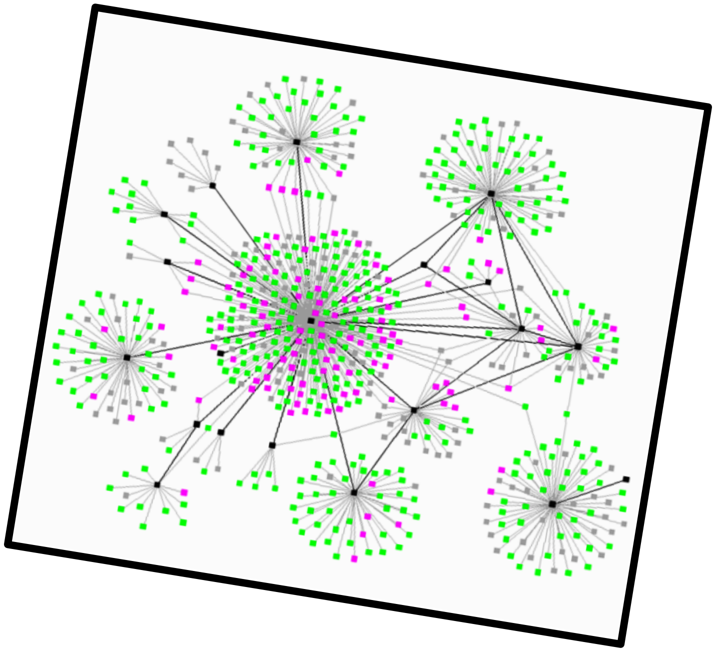
- Buzzword „Big Data“ everywhere
- Rapid growth of irregularly structured data:
  - Physics/astronomy: Accelerators, telescopes: Terabytes / day
  - Facebook: 1G+ members, 1G+ actions / day
  - Web graph, log files, smartphone actions, ...
- Big data: Not only graph data, but also graphs!



Network Analysis

- Possible questions for analysts:
  - Who has high influence in a network?
  - How does the network decompose into “natural” groups?
  - How does the network evolve in the future?
  - …

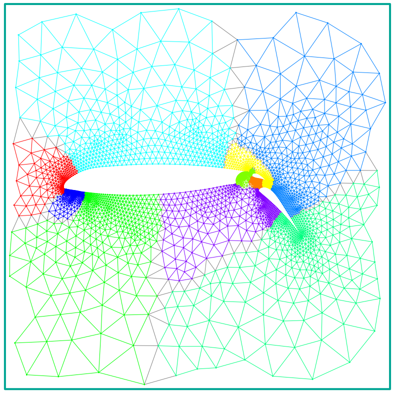
- Commercial interests:
  - Online marketing and ads
  - Recommendation systems (“Customers also bought”)
  - Cyber- and homeland security
  - Changes to technical infrastructure
  - …



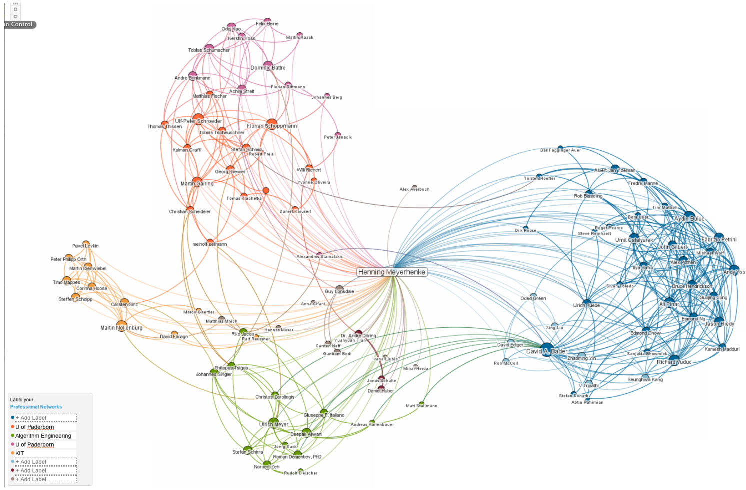
Challenges

Analysts need information from the piles of data…

… **but**:



- Data different from graphs in scientific computing!
  - Power-law degree distribution
  - Small world property – no long paths
  - Limited locality, highly unstructured
  - Difficult to partition
  - Vertices and edges can have types
- Graph analytics tax current hardware:
  - Classical numbercrunchers better on structured problems
  - Many analytics algorithms scale to a few million edges, but not billions

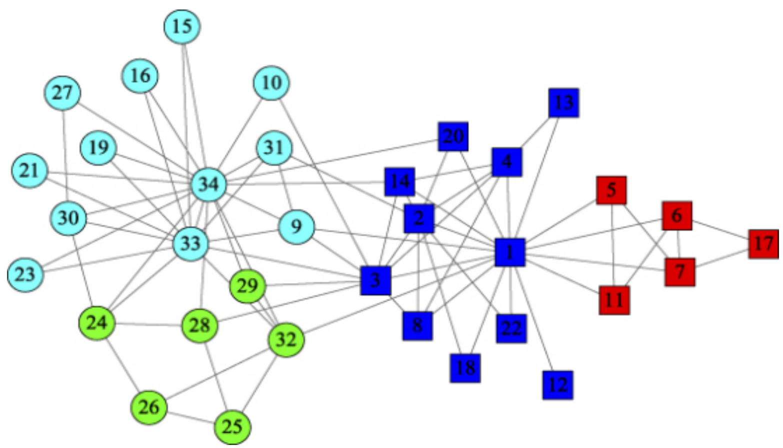


Analytics Tool: Graph Clustering

- Divide vertices into groups (clusters, communities) s.t.
  - vertices of the same group are “similar” to each other
  - vertices of different groups are “dissimilar” to each other
- Graphs: Well-connected vs not well-connected

- Also called *community detection*

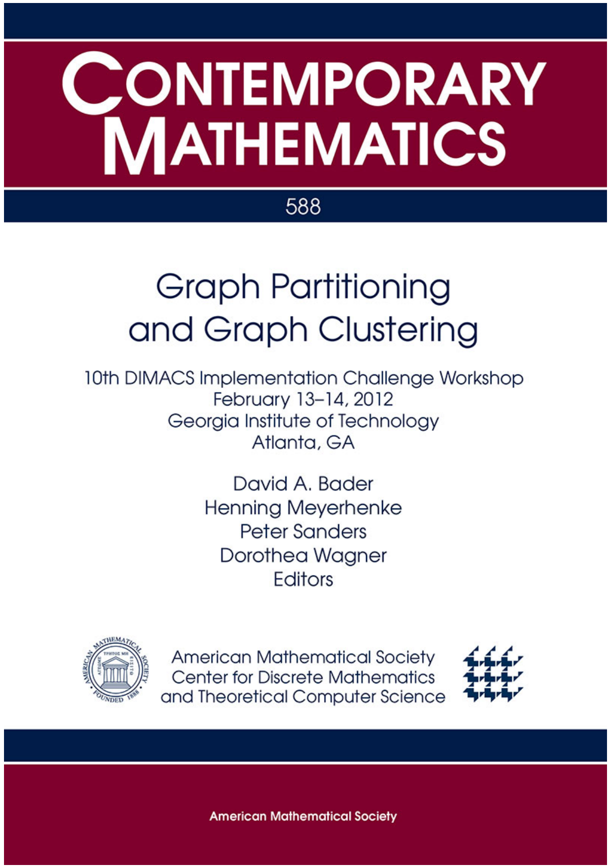
- Applications:
  - Complexity reduction
  - Classifying related genes
  - Distributed data storage, computations
  - Visualization



[Shen and Cheng, J. Stat. Mech. 2010]

10^{th} DIMACS Implementation Challenge

- Scientific competitions on “best” algorithm implementations
- Topics of 10^{th} Challenge: Graph partitioning and **graph clustering**
- Two different graph clustering challenges:
  - Modularity maximization
  - Mix challenge: Four different objectives combined
- Graph archive from various sources and applications: http://www.cc.gatech.edu/dimacs10/downloads.shtml
- Participants submitted their results for specified test set



Modularity Challenge

- (Still) Very popular objective: Modularity

$$\mathsf{q}(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{m} - \left(\frac{\sum_{v \in C} \deg(v)}{2m} \right)^2 \right]$$

- Expected deviation from random graph with same degree sequence

- NP-hard to optimize for modularity [Brandes et al., IEEE TKDE 2008]
- Modularity has some known issues [Berry et al., Phys Rev E 2011], [Good, de Montoye, Clauset, Phys. Rev. E 2010] and [Lancichinetti and S. Fortunato, Phys. Rev. E 2011]

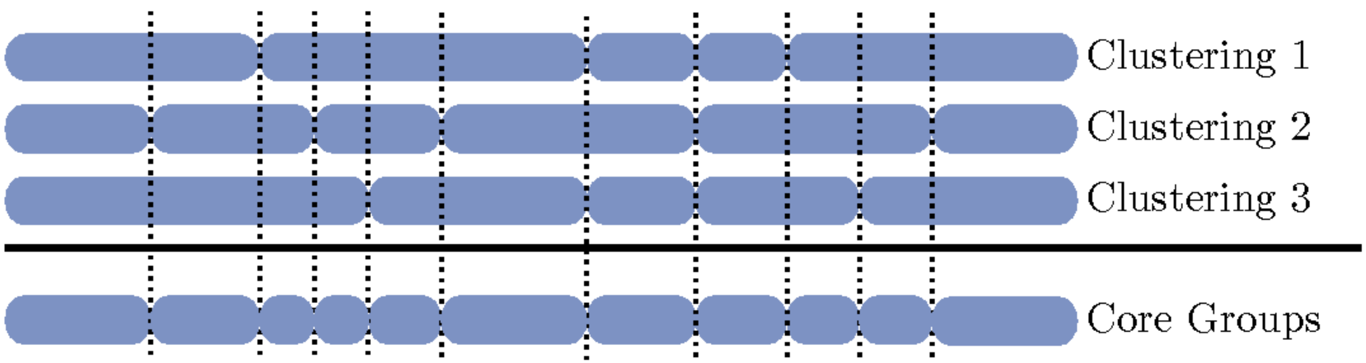
- **Quality competition:** Find solution with highest modularity value!
- Pareto competition: Trade-off quality and running time!

Summary

Modularity Challenge

- Winner CGGCi based on core groups:
  - Compute several solutions quickly with reasonable quality
  - **Base algorithm:** Randomized greedy agglomerative
  - Determine core groups, i. e., groups of nodes that belong together in all previous solutions
  - Refine solution, starting from core groups, iterate/ recurse with “multilevel-ish” approach
  - **Intriguing:** Different base algorithms yield similar quality

(Ovelgönne and Geyer-Schulz, KIT)



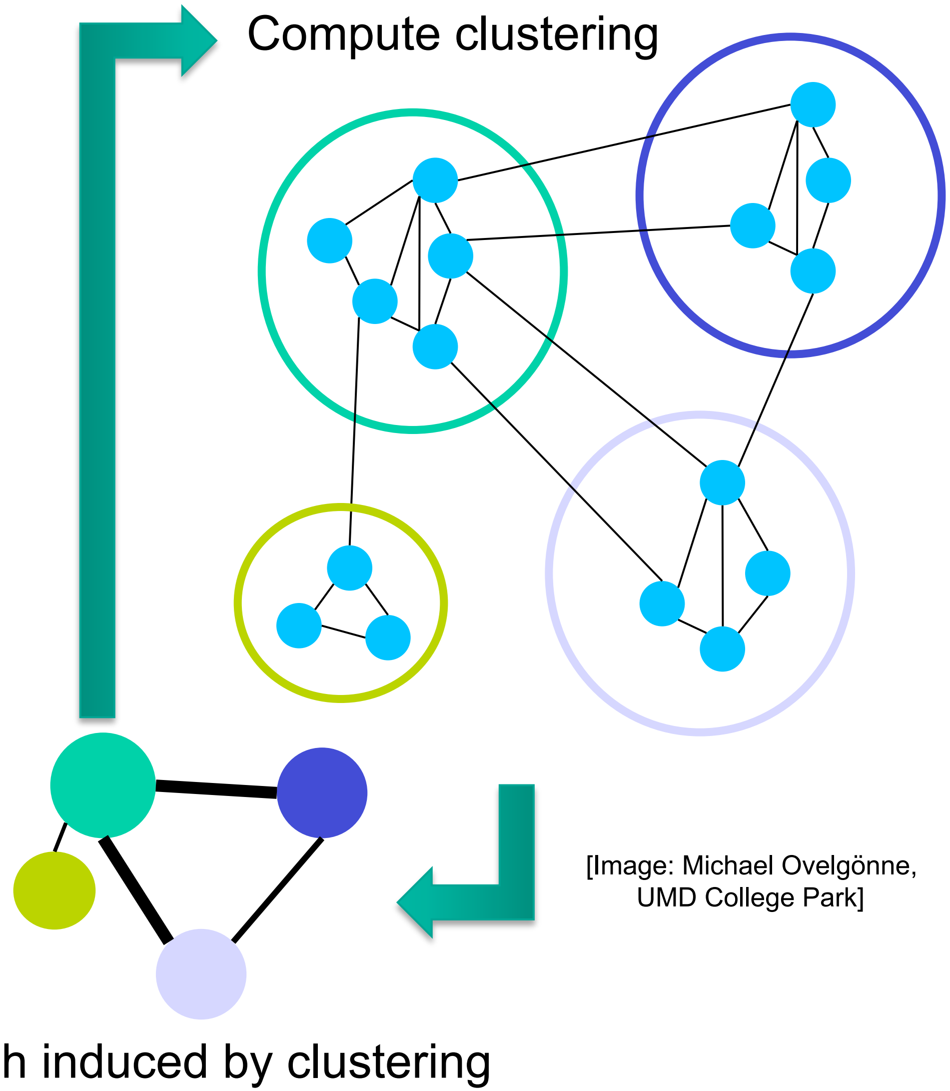
[Image: Michael Ovelgönne, UMD College Park]

OUR APPROACH

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

Our Approach based on core groups by O-G

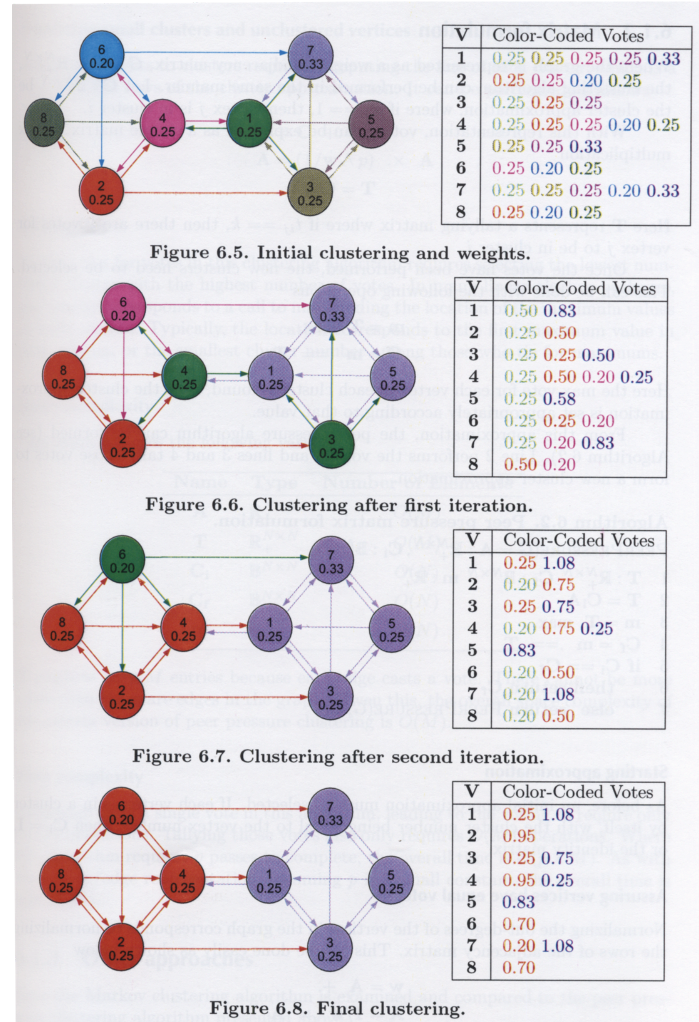
- **Motivation:**
  - Obtain high parallel performance...
  - ...while retaining a high solution quality
- **Idea:** Combine core group approach with highly parallel base algorithm
- **Base Algorithm:** Label propagation, also known as peer pressure clustering



[Image: Michael Ovelgönne, UMD College Park]

Base Algorithm: Label Propagation...
...and some of our insights

- Label Propagation: [Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Physical Review E 76(3), 036106 (2007)]
  - Start with singleton clustering
  - While labels can still change
    - For each vertex v in random order do
      - Label v with label of majority of its neighbors, arbitrary tie-breaking
- Some minor experimental insights:
  - Randomization does not really help with quality
  - Running time improvement for some instances (hardly any quality penalty): Stop when only few labels remain
  - Huge running time improvement: Active vs inactive vertices



V	Color-Coded Votes
1	0.25 0.25 0.25 0.25 0.33
2	0.25 0.25 0.20 0.25
3	0.25 0.25 0.25
4	0.25 0.25 0.25 0.20 0.25
5	0.25 0.25 0.33
6	0.25 0.20 0.25
7	0.25 0.25 0.25 0.20 0.33
8	0.25 0.20 0.25

Figure 6.5. Initial clustering and weights.

V	Color-Coded Votes
1	0.50 0.83
2	0.25 0.50
3	0.25 0.25 0.50
4	0.25 0.50 0.20 0.25
5	0.25 0.58
6	0.25 0.25 0.20
7	0.25 0.20 0.83
8	0.50 0.20

Figure 6.6. Clustering after first iteration.

V	Color-Coded Votes
1	0.25 1.08
2	0.20 0.75
3	0.25 0.75
4	0.20 0.75 0.25
5	0.83
6	0.20 0.50
7	0.20 1.08
8	0.20 0.50

Figure 6.7. Clustering after second iteration.

V	Color-Coded Votes
1	0.25 1.08
2	0.95
3	0.25 0.75
4	0.95 0.25
5	0.83
6	0.70
7	0.20 1.08
8	0.70

Figure 6.8. Final clustering.

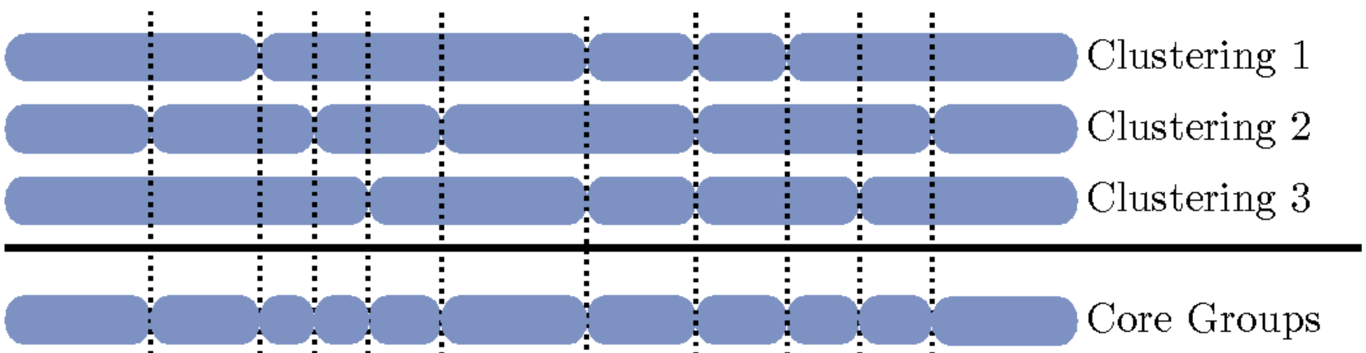
[Eric Robinson: Complex Graph Algorithms. In: Graph Algorithms in the Language of Linear Algebra. SIAM, 2011.]

Computing Core Groups in Parallel

- Core Groups: Maximal overlaps of clusters
- Core group: All vertices that agree on all base clusterings
- Problem: Cluster IDs are different in different base clusterings

- Variant 1: BFS (not maximal, but connected core groups)
  - Do not expand BFS at vertex v if v does not agree with source on all base clusterings

- Variant 2: Hashing (highly parallel!)
  - Use k-way hash function for k base clusterings



[Image: Michael Ovelgönne, UMD College Park]

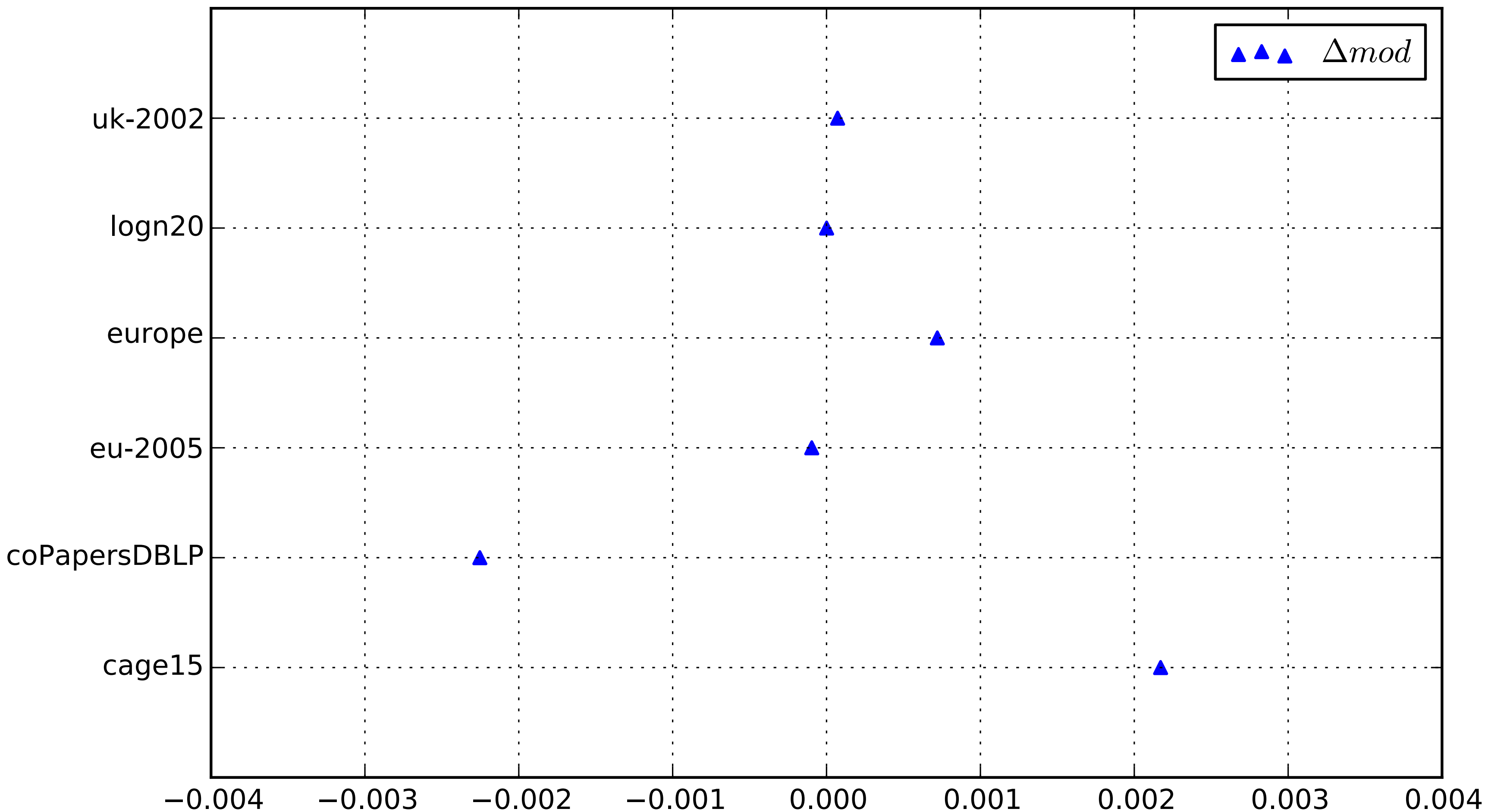
Implementation and Experimental Settings

- C++ with STL containers and OpenMP
  - Support for dynamic graphs easy
  - High-level interfaces, easy to switch

- Experimental platform (so far):
  - 2x 8-core Intel Xeon™ E5-2670 CPU, 2.6 GHz
  - 64 GB RAM
  - GCC 4.7.1 C++ compiler

- Test graphs from DIMACS Implementation Challenge archive:
  - Different applications: Web graphs, citation networks, optimization matrices, street networks, R-MAT
  - Focus on large graphs: Millions of vertices, millions/billions of edges

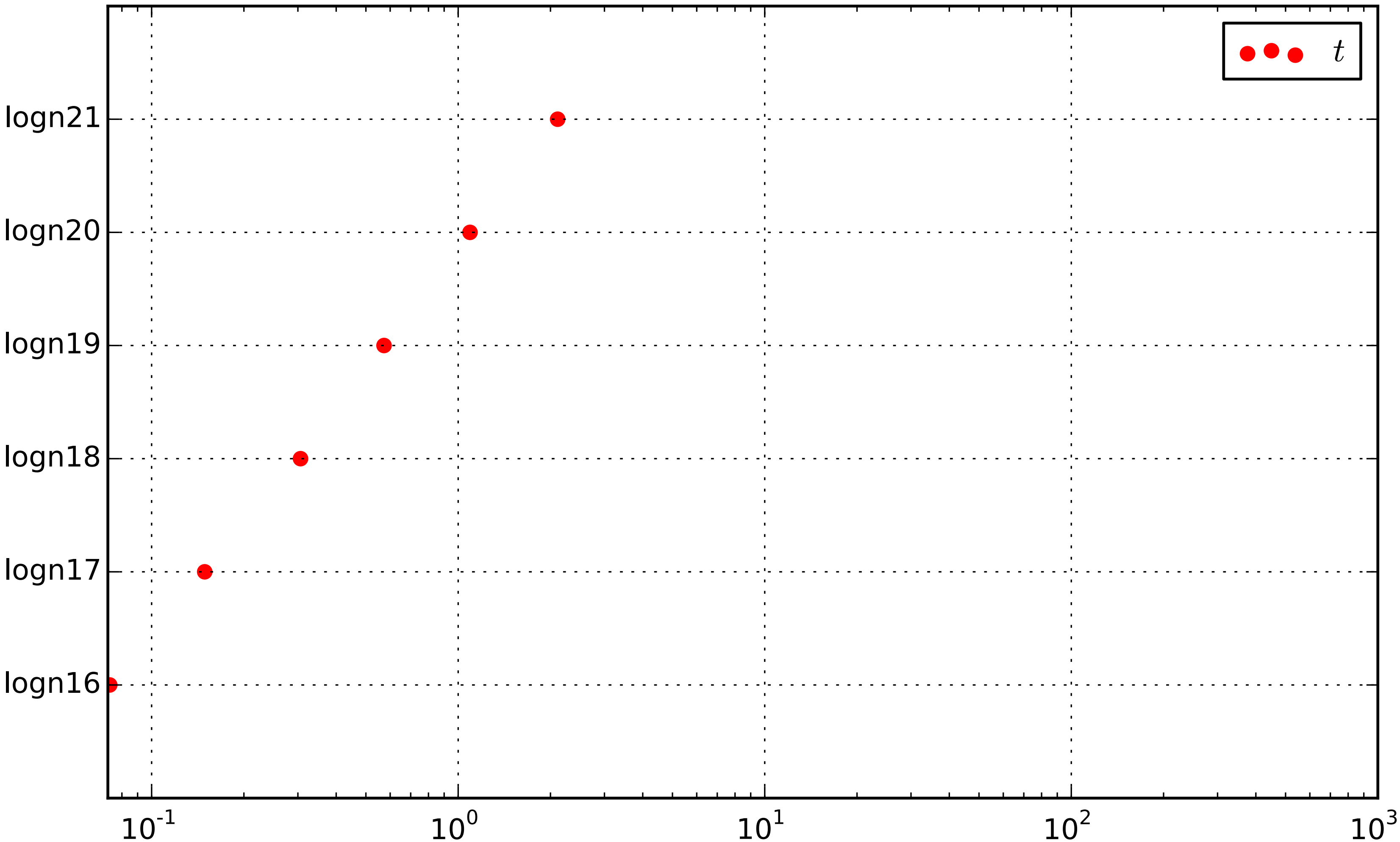
Quality
Difference between 1xLP and core group recursion



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

Running Time

Linear scaling on R-MAT graphs (1x LP)



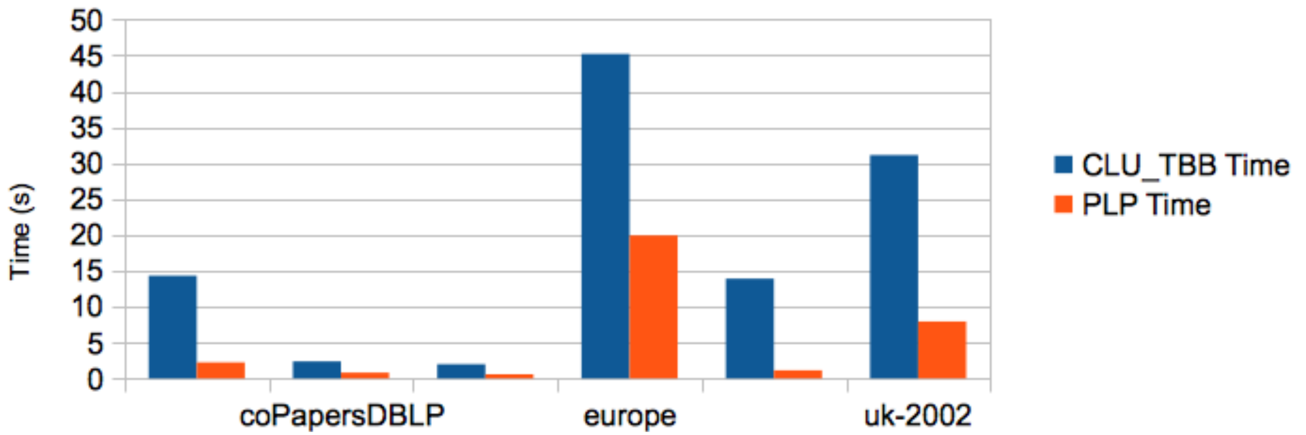
Institute of Theoretical Informatics,
Karlsruhe Institute of Technology (KIT)

Comparison to Challenge Results
Running time and modularity (1x LP)

- RG [Ovelgönne and Geyer-Schulz] was Pareto Challenge winner
  - **PLP is about 2 orders of magnitude faster**
  - RG about 0.06 better in modularity

- CLU_TBB was one of two parallel codes
  - Second best group in Pareto challenge
  - PLP is slightly better in quality

(unnormalized)



Comparison to Agglomerative Algorithm

- Similar to CLU_TBB: Agglomerative algorithm by Riedy et al.
- Later improved concerning running time:
  - [Riedy, Meyerhenke, Bader, MTAAP 2012]

- Parallelism in agglomeration: Merge matching edges concurrently
- Problem: Matchings can become small

- **Comparison:**
  - PLP is faster with fewer cores
  - PLP‘s quality is in general better
  - PLP offers more parallelism

- Web graph uk-2007 (n = 105,896,555, m = 3,301,876,564):
  - Agglomerative: Modularity around 0.48,
    took in Challenge 8 minutes (on faster system with 40 cores)
  - PLP: Modularity > 0.97 in less than two minutes

Summary

- Community detection one of the main network analysis kernels
- Parallel implementation of label propagation clustering algorithm (PLP)
- **Objective:** Fast running time with high quality due to core group approach
- **Insight:** Recursive core groups not helping unless large number of base algorithms used

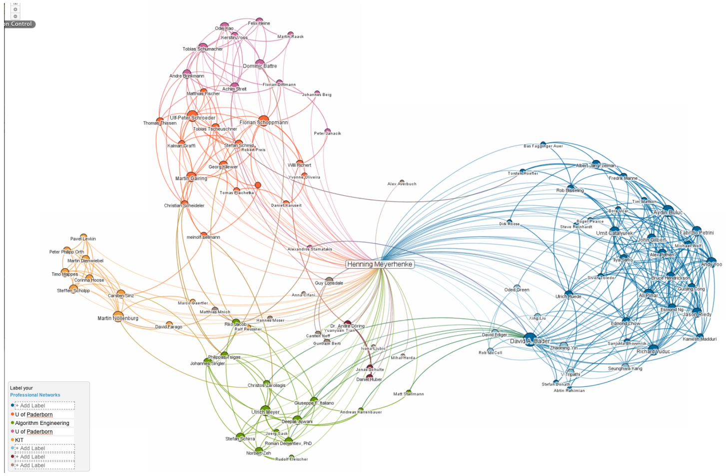
- **Ongoing improvement:** Apply Louvain-type local search to core groups without recursion (with PLP base clusterings)

FURTHER ONGOING WORK

Seed Set Expansion

Based on thesis by Jonathan Dimond (KIT / Georgia Tech)

- Useful to find community around specified vertices
- Use Cases:
  - Reduce cost for expensive analysis
  - Selection for visualization
  - Find possible collaborators

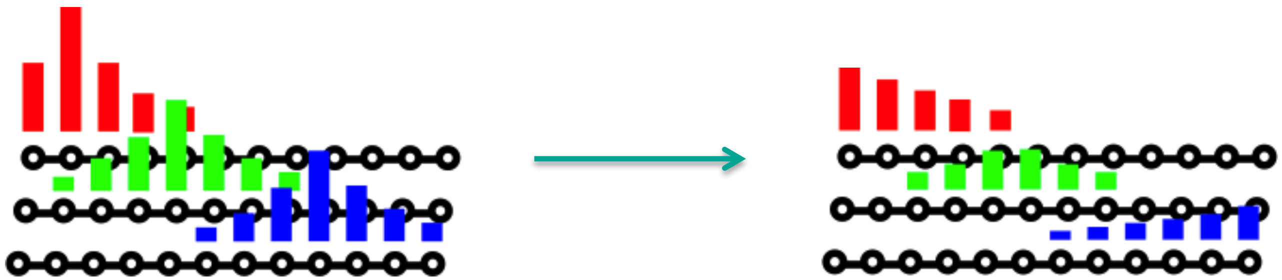


- Results so far:
  - Algorithms related to random walks work well in static setting
  - Agglomerative approach worse
  - Dynamic setting: No clear picture

Algebraic Distance

Following [Chen and Safro, SIAM J. on SC, 2011]

- Useful as a preprocessing tool
- Perform a few Jacobi or Gauss-Seidel iterations on r random vectors

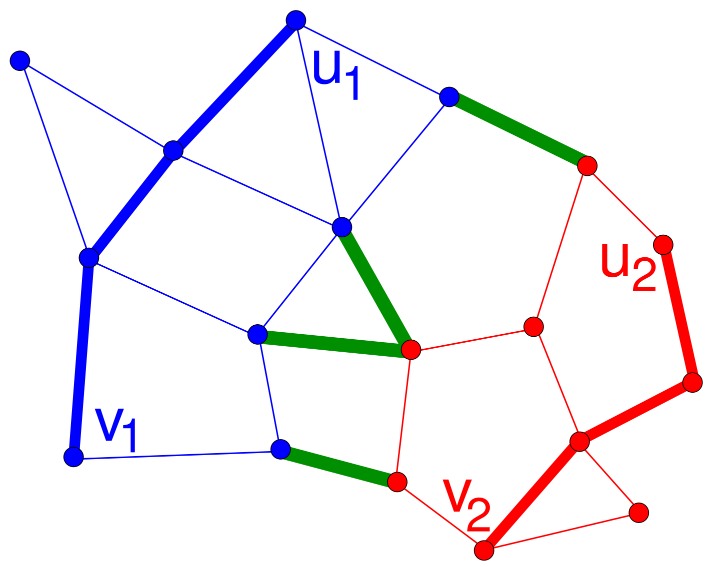


- Vertices that are well connected will have similar values
- Interpret vector entries as coordinates of r-dimensional points
- Distance between vertices = distance between their r-dim. Points

- **Ongoing:** Investigate if useful for seed set expansion

Djokovic Relation

- Djokovic relation: Relation based on shortest paths
- Finds edges whose endpoints divide the graph w.r.t. graph distance to source edge
- Useful for finding convex cuts (or similar)
- Convex cuts yield nicely shaped parts



R. Glantz, H. Meyerhenke: Finding all Convex Cuts of a Plane Graph in Cubic Time. Accepted at 8th International Conference on Algorithms and Complexity (CIAC'13).

Lean Algebraic Multigrid

Based on [Livne, Brandt; SIAM J. Scientific Computing, to appear]

- Algebraic Multigrid (AMG): Solver for linear systems from PDEs
- Designed for sparse matrices with “PDE structure”
- Does not work well on complex networks

- Lean AMG: Extension to Laplacian matrices of complex networks
- Motivation for solving Laplacian problems:
  - Machine learning
  - Spectral clustering (images, genes, ...)
  - Network flows, electrical circuits

- **Ongoing work:**
  Python frontend, C/C++ parallel backend for performance-critical parts

- **Objective:** Solve Laplacian problems within larger graph framework

Thank you!

http://parco.iti.kit.edu

Acknowledgement: This work is partly funded by the Ministry of Science, Research and the Arts Baden-Württemberg.