Scalable and Accurate Algorithm for Graph Clustering

Hristo Djidjev
Los Alamos National Laboratory

Melih Onus
Cankaya University, Turkey

Networks

- **Network :** sets of nodes or verticies joined together in pairs by links or edges

Technological networks

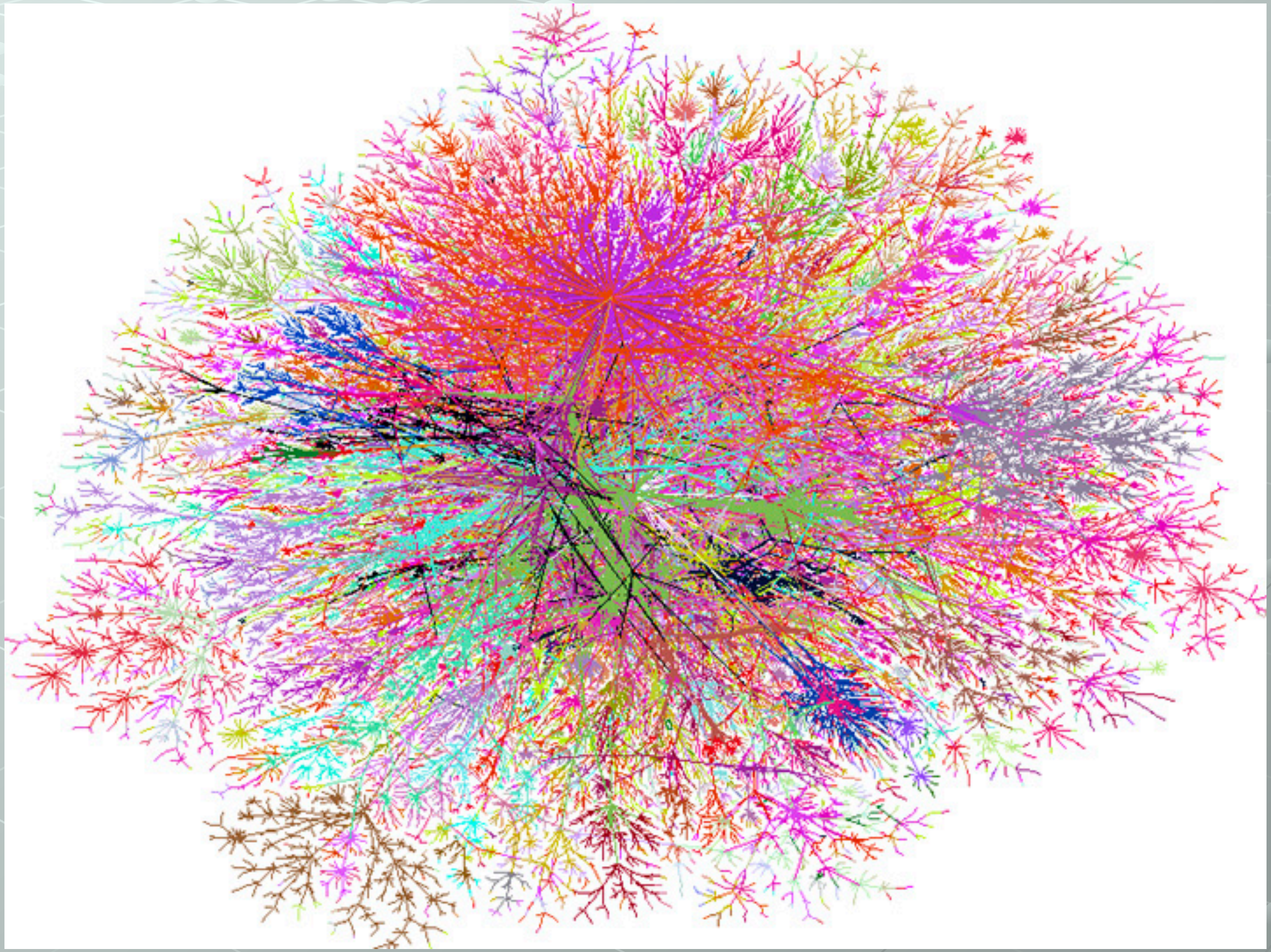
Transportation Networks

Biological networks

Semantic Networks

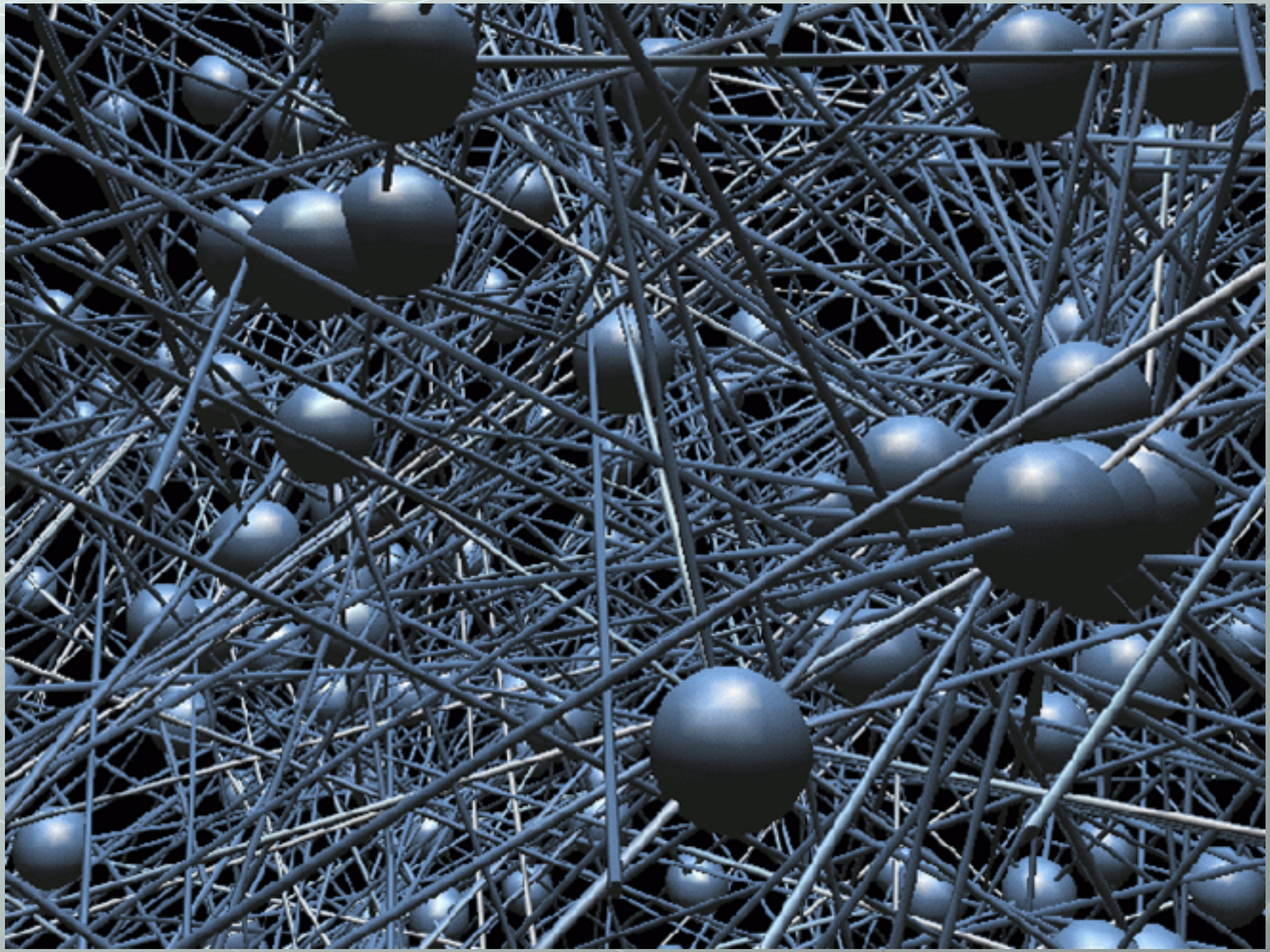
Social networks

The Internet and the WWW



credit: B. Cheswick

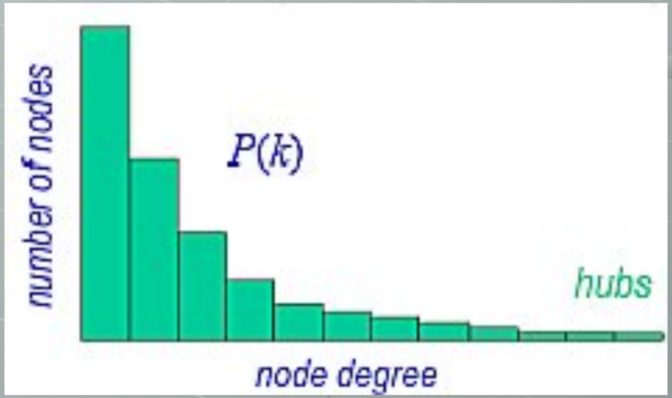
Social networks



credit: A. Klovdahl

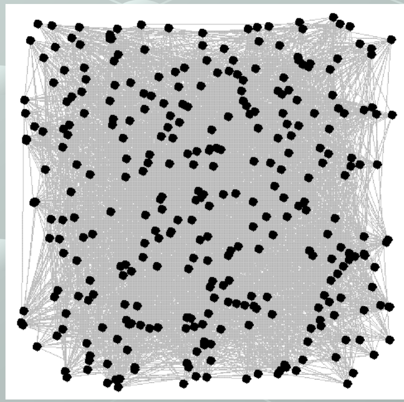
Common network properties

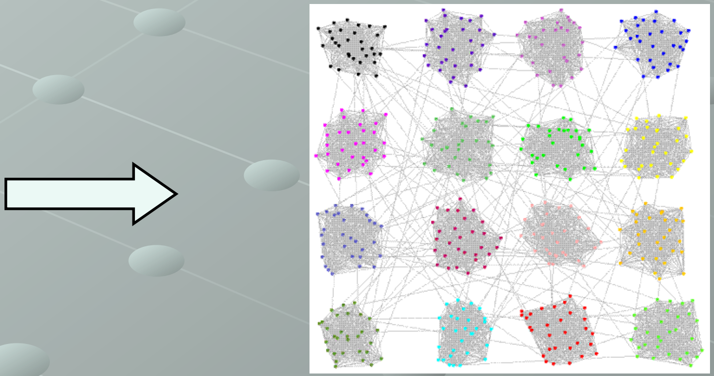
- <u>Small world effect:</u> Short distances between nodes
- <u>Power law distribution:</u> Non-uniform degree distribution $P(k)$ ☒ $\alpha k^{-☒}$
  - Many low degree nodes
  - Few very high degree nodes
- <u>Community structure</u>



Community structure

- Communities: subsets of nodes within which there are dense links, but between which connections are sparser.
- Community detection problem: given a network *N*, find a partition of *V(N)* into communities





Modularity

- A useful measure of clustering quality
  - Introduced by Newman, 2003
- *Modularity* of a partition

  = (fraction of edges within communities)

  – (expected fraction of such edges)
- *Community detection (graph clustering) problem*: Find a partition maximizing the modularity
- The optimization problem is NP-hard

Our goal:

Algorithm that is

accurate **AND** scalable

Approach

- Reduce

Graph clustering Problem:
Find a partition of *G* of maximum modularity

to →

Min-Cut Problem:
Find a minimum cut in a complete, edge-weighted graph *G'*

- Solve

Min-Cut Problem:
Find a minimum cut in a complete weighted graph

by →

Graph partitioning:
Finding a minimum cut that produces a balanced partition

Reduction:
max modularity -> min cutsize

$$Q(\mathcal{P}) = \frac{1}{m}\sum_{i=1}^{k}(|E(V_i)| - \mathrm{Ex}(V_i,\mathcal{G}))$$

$$\max_{\mathcal{P}}\{\sum_{i=1}^{k}(\ |E(V_i)| - \mathrm{Ex}(V_i,\mathcal{G})\ )\}$$

$$= -\min_{\mathcal{P}}\{\ |\mathrm{Cut}(\mathcal{P})| - \mathrm{ExCut}(\mathcal{P},\mathcal{G})\}$$

$$\mathrm{weight}(i,j) = \begin{cases} 1 - p_{ij}, & \text{if } (i,j) \in E(G) \\ -p_{ij}, & \text{if } (i,j) \notin E(G). \end{cases}$$

Choice of random graph models

$$\text{weight}(i,j) = \begin{cases} 1 - p_{ij}, & \text{if } (i,j) \in E(G) \\ -p_{ij}, & \text{if } (i,j) \notin E(G) \end{cases}$$

p_{ij} : the probability that there is an edge between vertices i and j in a random graph from a given distribution

Erdos - Renyi Model:

$$p_{ij} = p = \frac{m}{\binom{n}{2}}$$

Chung - Lu Model:

$$p_{ij} = \frac{d_i d_j}{\sum_{k=1}^{n} d_k}$$

Reduction phase of algorithm

Community Detection Problem:
Maximize modularity

⇓

make complete & define weights

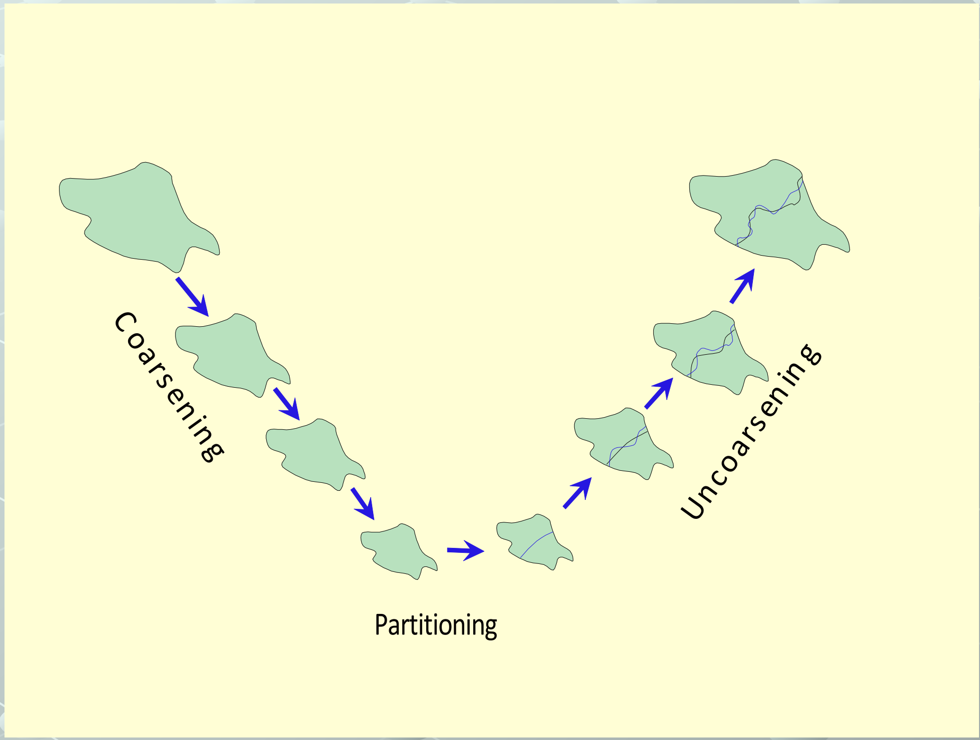
$$\text{weight}(i,j) = \begin{cases} 1 - p_{ij}, & \text{if } (i,j) \in E(G) \\ -p_{ij}, & \text{if } (i,j) \notin E(G) \end{cases}$$

⇓

Min-Cut Problem:
Minimize cut size

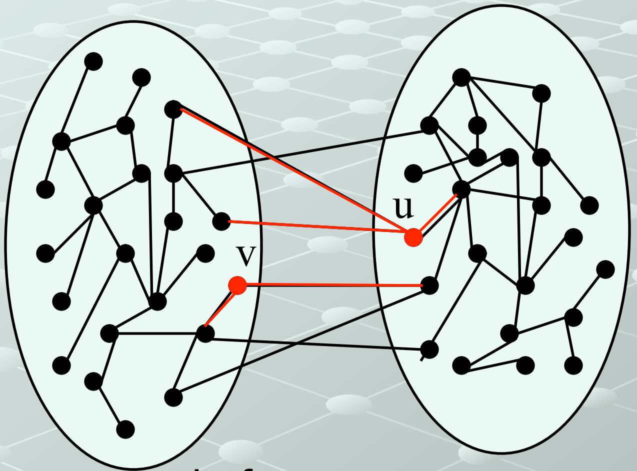
Phase 2: Solving the mincut problem

- Use multi-level graph partitioning method

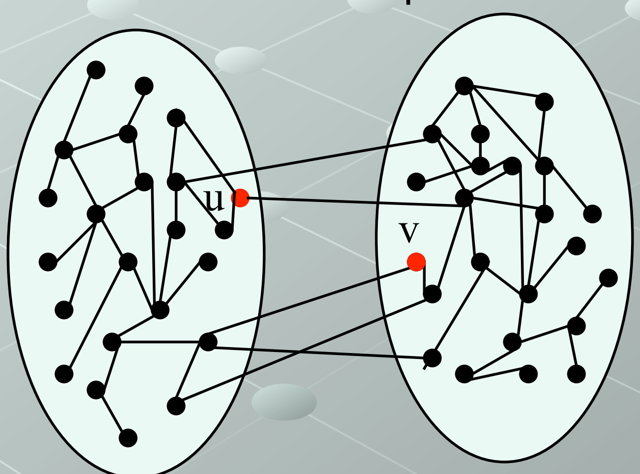


- Consists of the three phases:
  - Coarsening phase
  - Partitioning phase
  - Uncoarsening and refinement phase

Refinement: Kernighan-Lin procedure



before swap



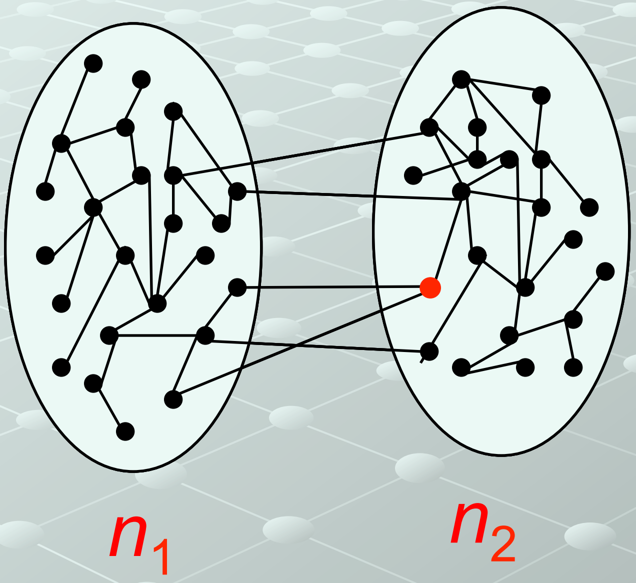
after swap

- Find an initial raw partition
- Improve by a greedy procedure that swaps pairs of vertices from different partitions
- Continue until no further improvement possible

Implementation issues

- GP always produces *balanced* partitions.
  - Ignore the restrictions on the sizes of the parts.
- The number of the parts in the optimal clustering is not known.
  - Employ a recursive bisection procedure.
- The original graph G might be sparse, while the transformed one G' is complete.
  - Do not *explicitly* generate G'.

Efficiently updating modularity



cut = cut(vis)– $n_1 n_2 p$

⇓

cut' = cut(vis)' – $(n_1+1)(n_2-1)p$

$$\text{weight}(i,j) = \begin{cases} 1 - p_{ij}, & \text{if } (i,j) \in E(G) \\ -p_{ij}, & \text{if } (i,j) \notin E(G). \end{cases}$$

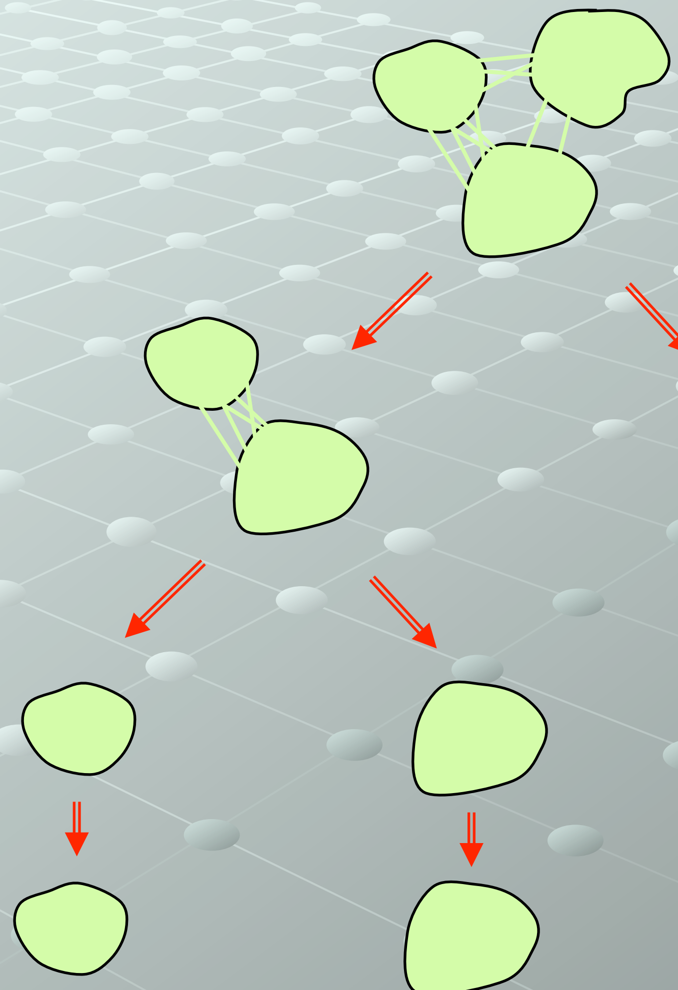
Finding the optimal number of communities

- Assign weights to the edges of *G*.
- Partition using the bisection algorithm
- **If** the number of resulting parts is one, then **done**;

  Else run recursively on each subset.

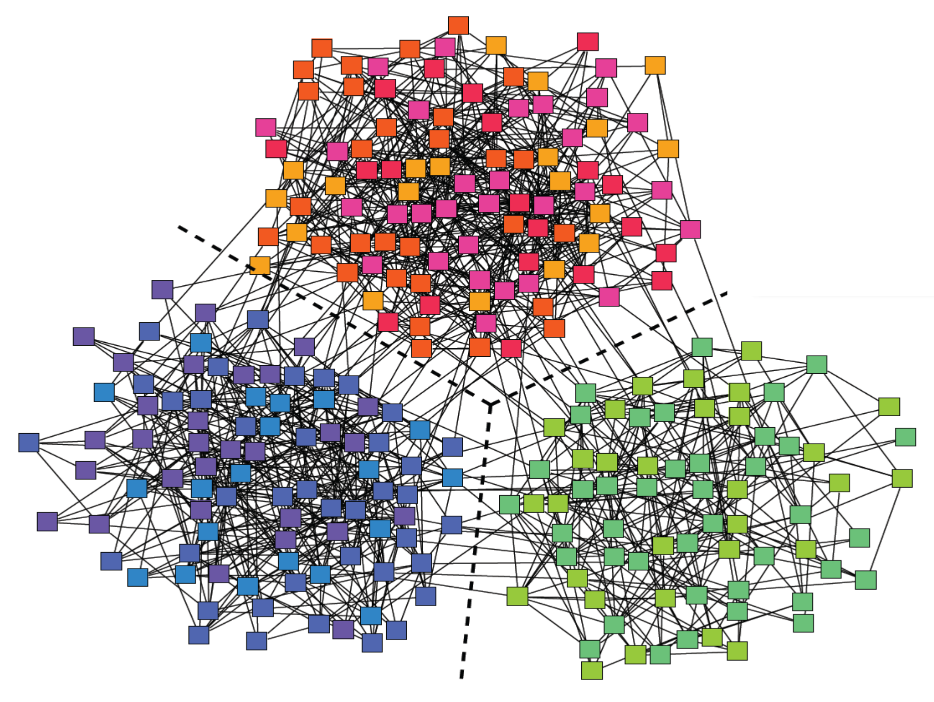
- Time Complexity: $O((n + m)\, k)$

Example



Experiments

- Test graphs: clustered, random



credit: Aaron Clauset

Comparison with other algorithms

	Exp #	# vert.	# edges	# clust	Q_{orig}	Q_{CNM} / t_{CNM}	Q_N / t_N	Q_{GA} / t_{GA}	Q_{RB} / t_{RB}	Q_{here} / t_{here}
# communities	1	200	8934	2	0.388	0.387	0.388	0.387	0.386	0.388
						1.15	0.70	88.45	35.55	0.00
	2	400	21811	4	0.476	0.474	0.476	0.472	0.473	0.476
						2.45	3.35	335.50	102.40	0.15
	3	600	38743	6	0.447	0.445	0.447	0.445	0.445	0.447
						4.15	9.95	928.20	189.95	0.30
	4	900	71654	9	0.386	0.370	0.386	0.385	0.384	0.386
						7.85	23.05	2539.15	388.25	0.50
sparsity	5	200	9919	2	0.298	0.296	0.298	0.296	0.296	0.298
						1.05	0.65	98.60	38.70	0.10
	6	200	4958	2	0.299	0.297	0.299	0.297	0.297	0.299
						0.95	0.30	37.85	21.25	0.05
	7	200	2483	2	0.300	0.299	0.300	0.300	0.299	0.300
						0.95	0.40	27.50	22.40	0.05
sensitivity	8	400	38783	4	0.209	0.206	0.209	0.208	0.208	0.209
						3.00	3.40	716.65	184.80	0.10
	9	400	47775	4	0.123	0.113	0.123	0.122	0.122	0.122
						3.45	3.30	819.90	229.85	0.05
	10	400	53864	4	0.081	0.060	0.081	0.081	0.080	0.081
						3.50	3.80	1242.90	248.15	0.35

CNM = [Clauset, Newman, and Moore, 2004]

N = [Newman, 2007]

GA = [Guimera and Amaral, 2005]

RB = [Reichardt and Bornholdt, 2004]

Comparison (cont.)

	Exp #	# vert.	# edges	# clust	Q_{orig}	Q_{CNM} / t_{CNM}	Q_N / t_N	Q_{GA} / t_{GA}	Q_{RB} / t_{RB}	Q_{here} / t_{here}
scalability	1	1000	174990	2	0.357	0.357	0.357	0.356	0.358	0.357
						10.33	17.00	15808.67	1333.67	0.47
	2	5000	3749007	2	0.333	0.332	0.333	–	0.333	0.333
						329.50	2973.00	–	53119.50	8.00
	3	20000	24995617	2	0.300	0.297	0.300	–	–	0.300
						2199.33	18234.67	–	–	76.33

CNM = [Clauset, Newman, and Moore, 2004]

N = [Newman, 2006]

GA = [Guimera and Amaral, 2005]

RB = [Reichardt and Bornholdt, 2004]

Conclusions

- Community structure detection reduced to *mincut*
- *mincut* solved efficiently by multilevel graph partitioning
- The resulting algorithm is highly scalable and accurate

Thank you!