Data Science Methods and Statistical Learning

Week 7

Prof. Samin Aref

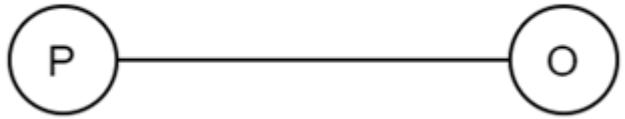
Part 1

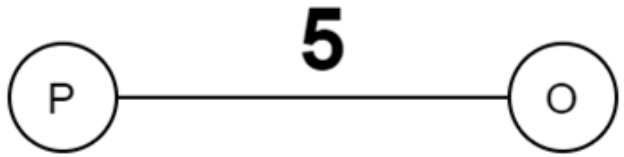
Network Science Fundamentals

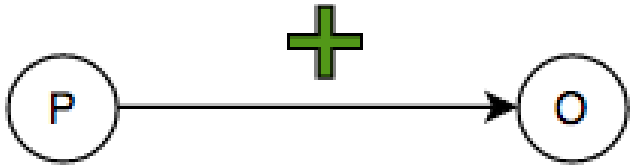
Network Basics: Nodes and Edges

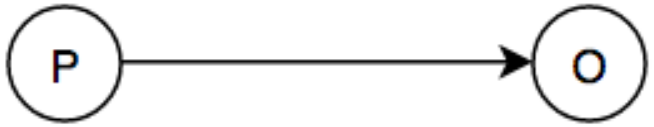
- **Nodes** (or points, vertices, agents)

- **Edges** (or connections, lines, links, ties)
  - Unweighted or weighted (frequency, intensity)
  - Unsigned or signed
  - Undirected or directed





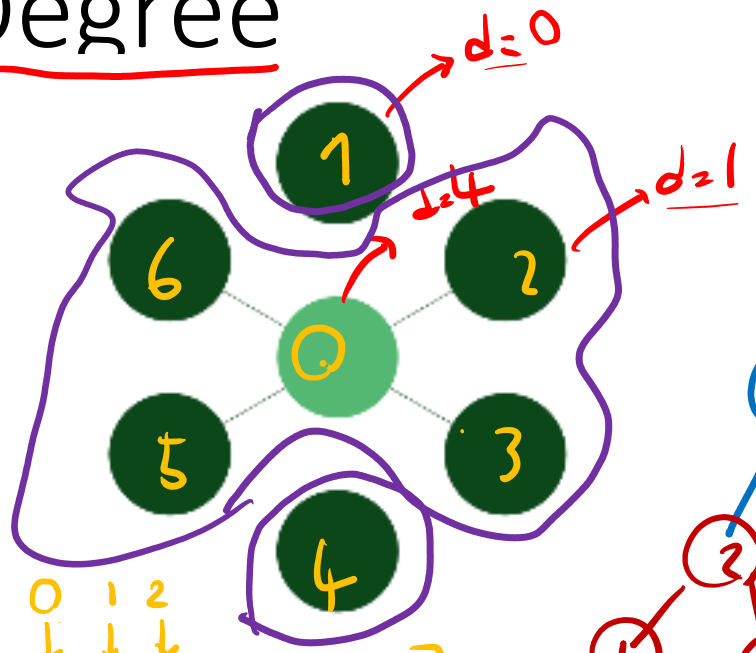


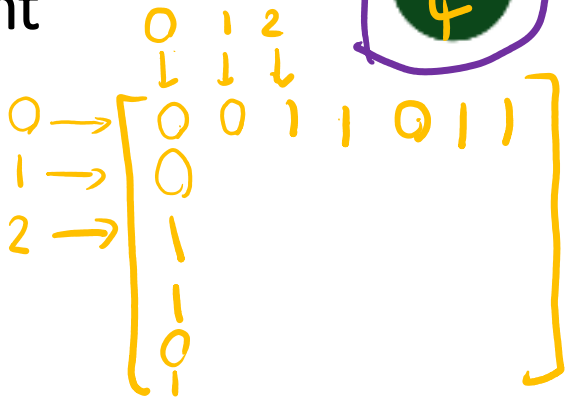


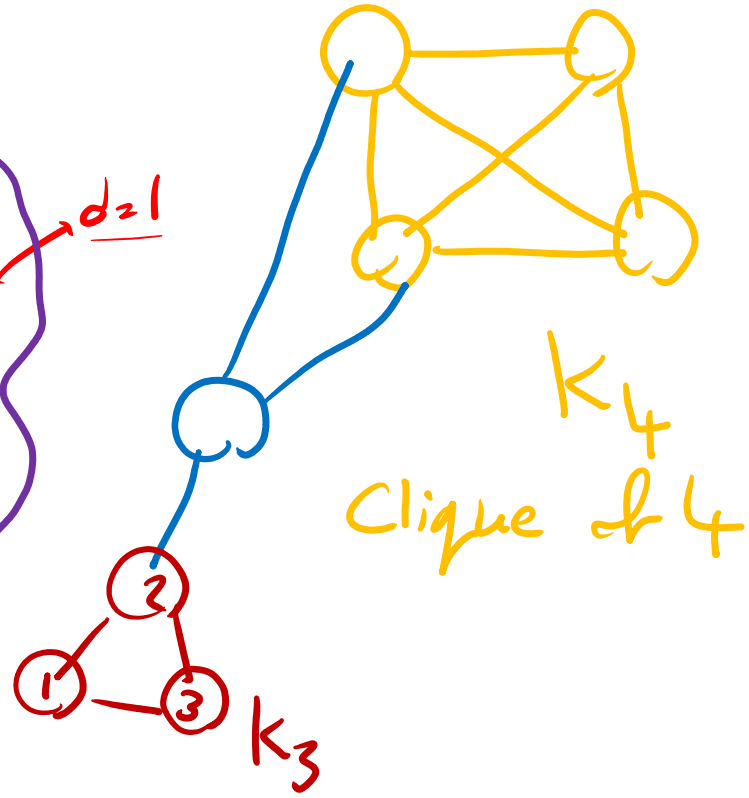
Network Basics: Degree

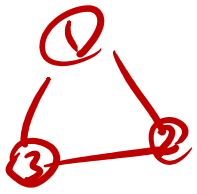
- Degree of a node

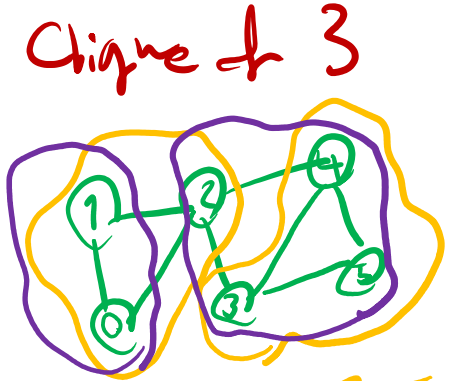
d_i degree of node i











P_1 P_2

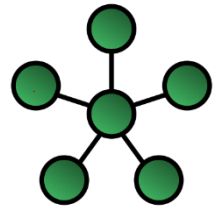
- Connected component
- Isolated node
- Pendant node
- Clique
- Adjacency matrix
- Partition (non-overlapping communities)
- Partition similarity measures: NMI, AMI, and RMI

$AMI(P_1, P_2) = 0$ & close to 1

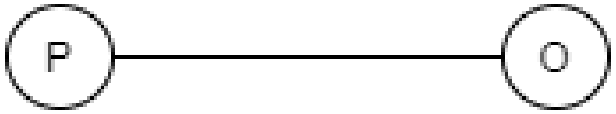
$P_2 = [[0,1],[2,3,4,5]]$

$P_1 = [[0,1,2],[3,4,5]]$

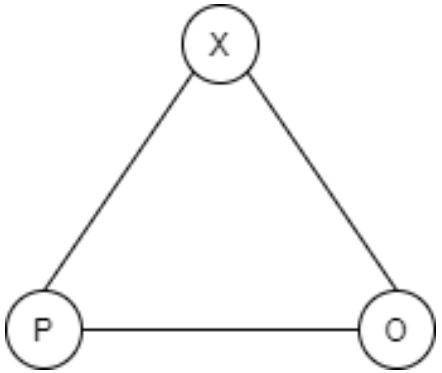
Network Basics: Units of analysis



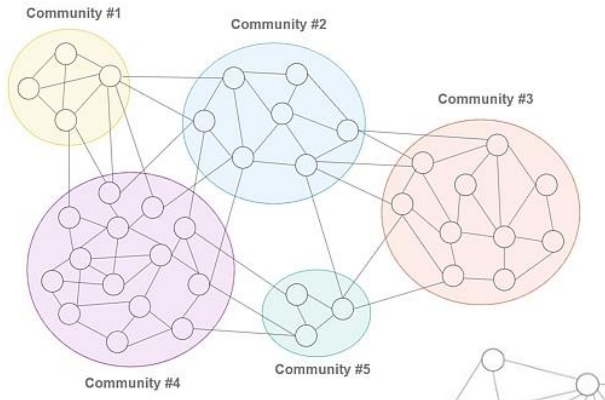
- **Ego:** egocentric analysis (e.g. centrality measures)
- **Dyad:** Two nodes and edge(s) that connect them



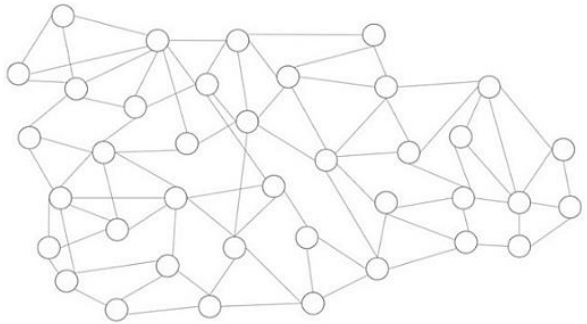
- **Motifs (e.g. triad):** triad is the smallest meaningful social unit



- **Cluster:** community, groups



- **Complete network:** *G(V,E)*



Part 2

Community Detection
(Clustering network data)

Zachary Karate Club

Nodes: club members (n)

Edges: interactions outside club (m)

Communities: densely connected groups of nodes with sparser between-group connections

Community assignments are predicted by a maximum flow algorithm.

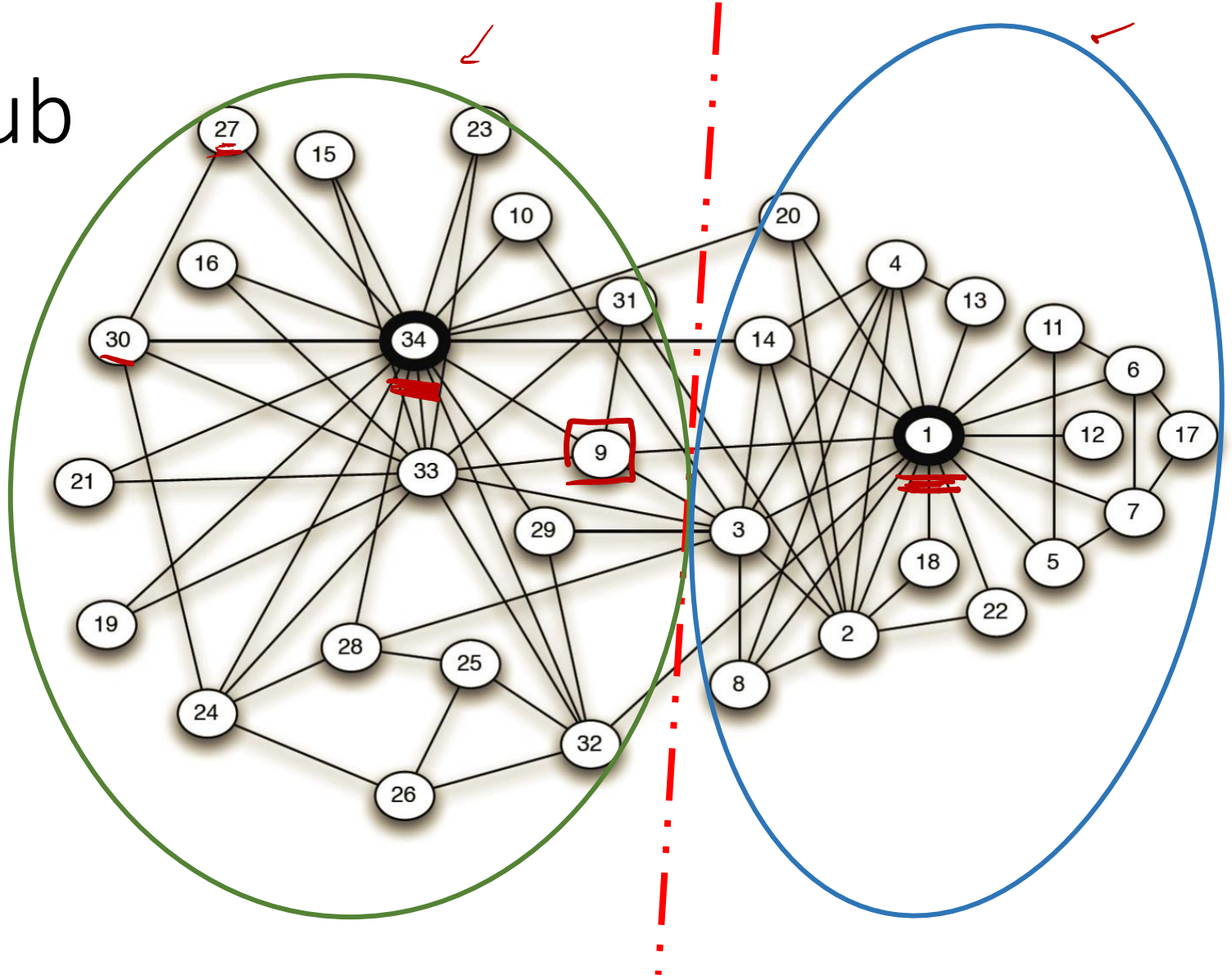
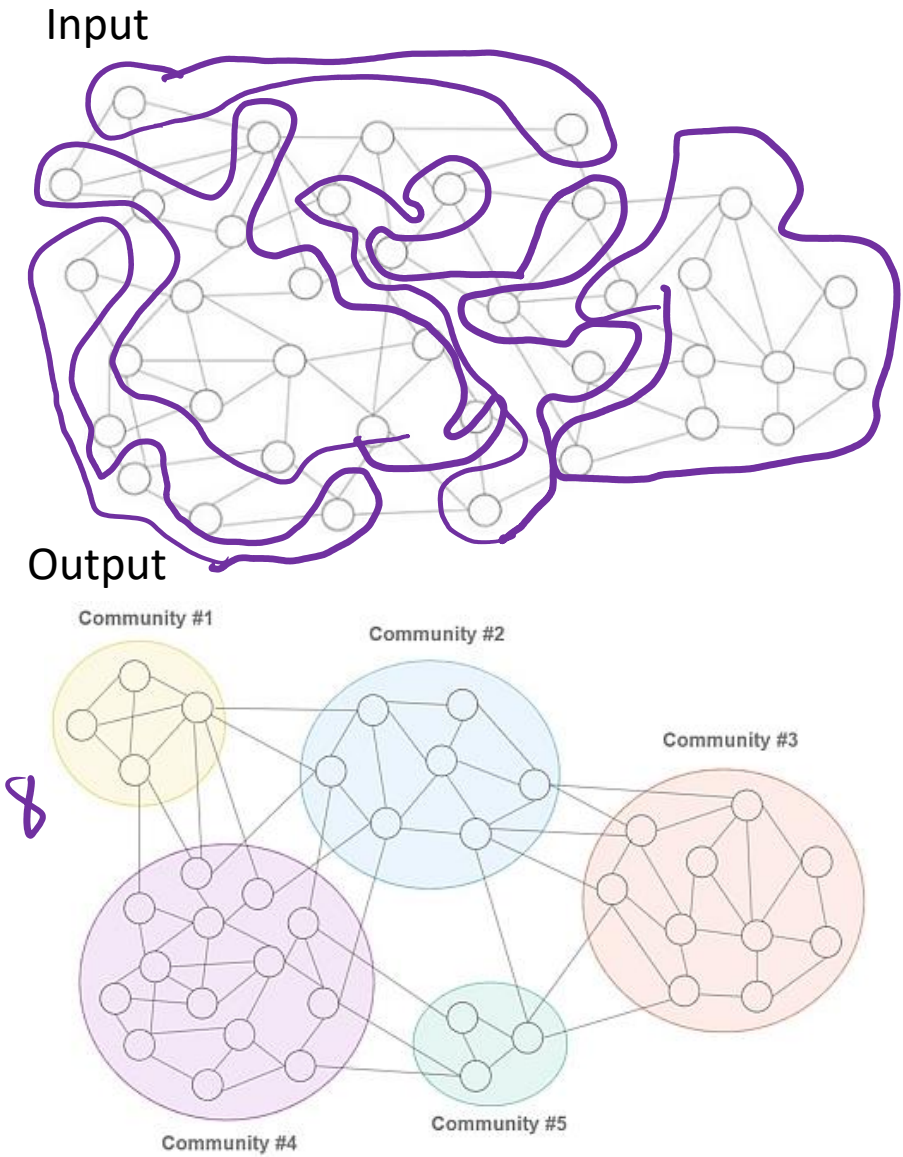


Figure from Leto Peel

Community Detection

- Input: a graph
- Method: a network clustering algorithm
- Output: a partition of nodes into communities (node colours)

- Goal: grouping tightly connected nodes into communities (clusters) which can be gently connected to the rest of the network (Schaub et al. 2017).



Figures from timbr.ai

Schaub et al. (2017) The many facets of community detection in complex networks

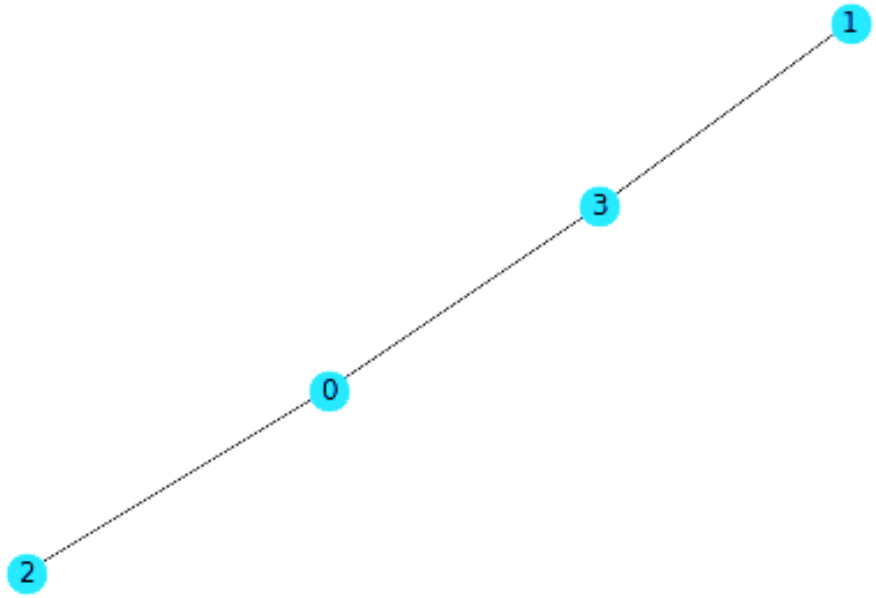
Optimization-based community detection

Given a network (graph) $G(V,E)$, find a partition X of V into V_1, V_2, …, V_c such that the objective function $U(G,X)$ is maximized.

Objective functions to optimize:

Maximizing flow (Zachary 1977)

Maximizing ***modularity*** (Newman 2006)

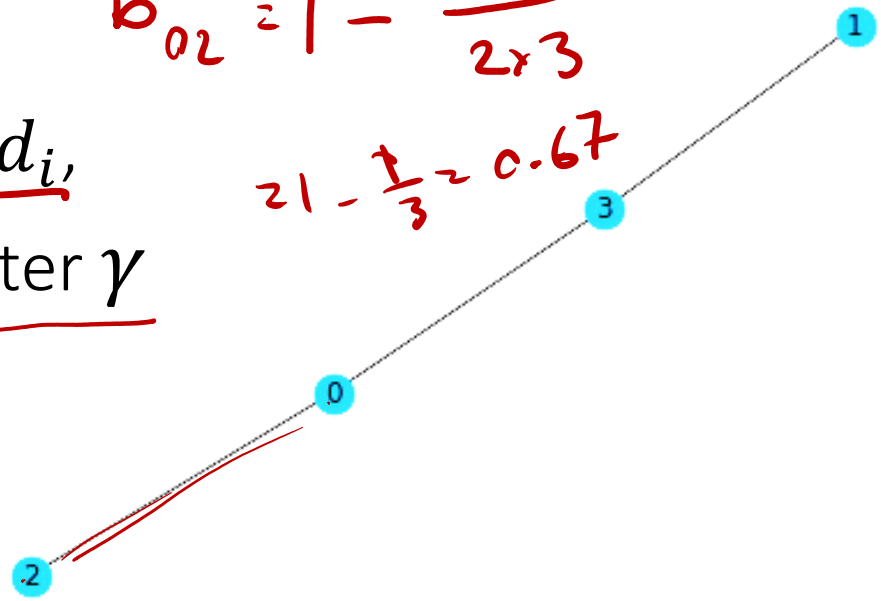


Modularity

Given a network (graph) $G(V,E)$, find a partition of V into V_1, V_2, …, V_c such that the modularity is maximized.

Modularity entry b_{ij} : a function of degrees d_i, connections a_{ij}, and the resolution parameter γ

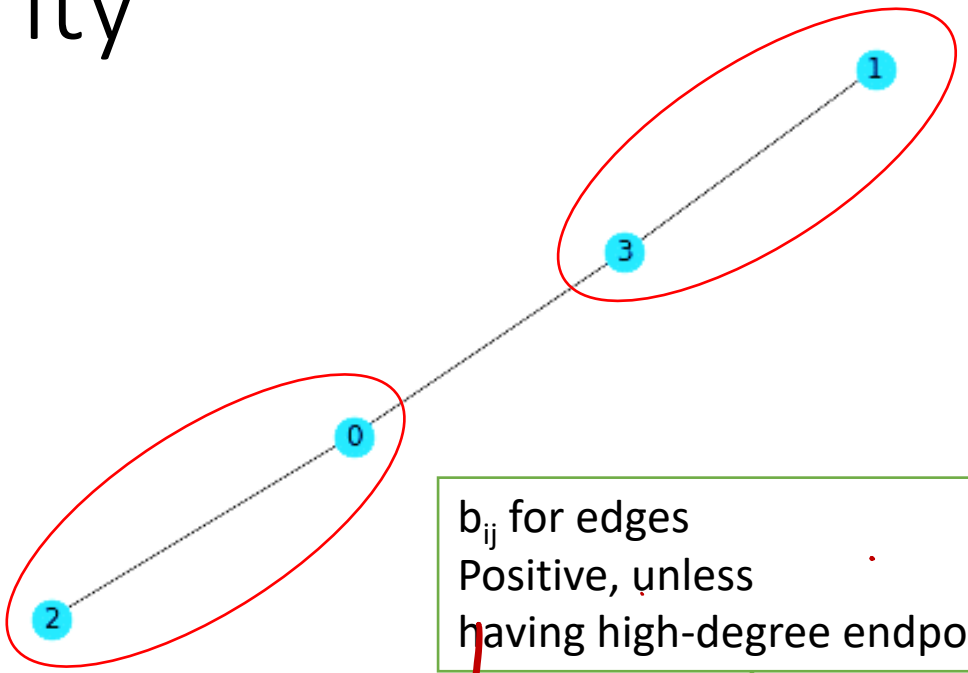
$$b_{ij} = a_{ij} - \gamma \frac{d_i d_j}{2m}$$



Objective function: Modularity

Given a graph *G(V,E)*, find a partition *X* of the nodes that maximizes the modularity function *Q(G,X)*.

$$Q_{(G,X)} = \frac{1}{2m} \sum_{(i,j) \in V^2} \left(a_{ij} - \gamma \frac{d_i d_j}{2m} \right) \delta(i,j)$$



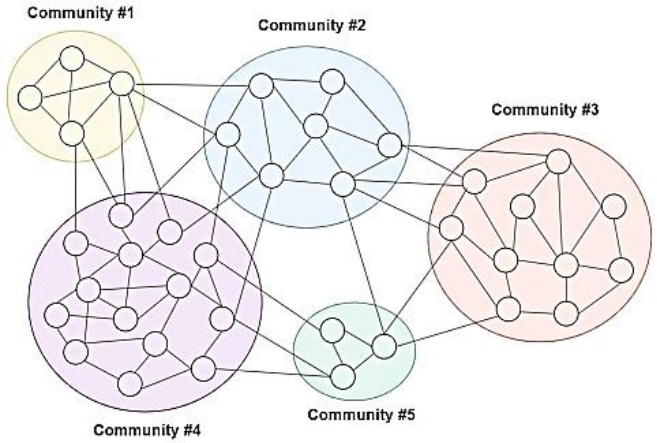
b_{ij} for edges
Positive, unless
having high-degree endpoints

B=
[[-0.67, -0.33, 0.67, 0.33],
[-0.33, -0.17, **-0.17**, **0.67**],
[0.67, -0.17, -0.17, -0.33],
[0.33, 0.67, -0.33, -0.67]]

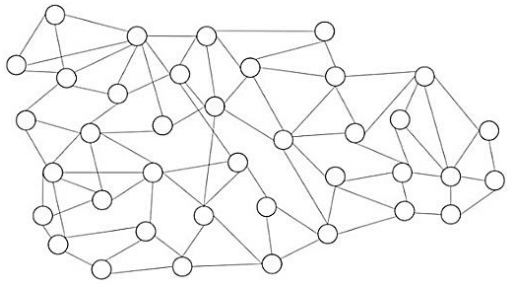
b_{ij} for non-edge pairs
Always negative

Detecting communities via Modularity Maximization (MM)

Output: communities



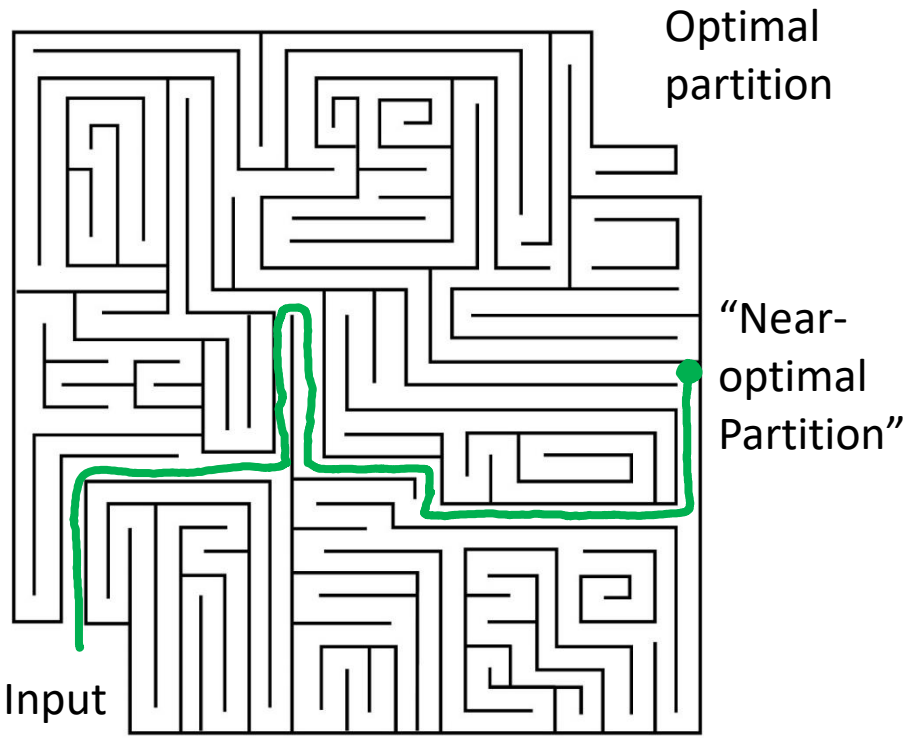
Input: a simple graph



Figures from timbr.ai

Approach 1: Modularity Maximization Heuristics

1. Edge Motif (EdMot) (Li et al. 2019)
2. Leiden (Traag et al. 2019)
3. Paris (Bonald et al. 2018)
4. Belief (Zhang & Moore 2014)
5. Combo (Sobolevsky et al. 2014)
6. Leicht-Newman (LN) (Leicht & Newman 2008)
7. Louvain (Blondel et al. 2008)
8. Greedy (CNM) (Clauset et al. 2004)
9. Graph Neural Network (GNN) (Sobolevsky et al. 2022)



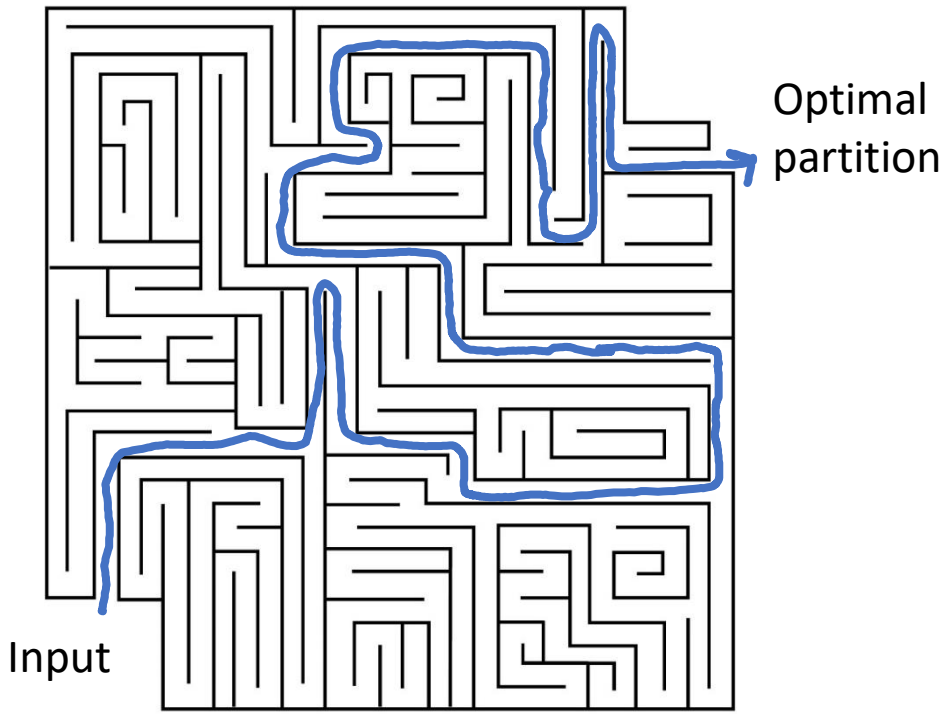
Approach 2: Exact/Approximate Modularity Maximization

- Integer Programming - IP (Brandes et al. 2007)
- IP and LP rounding (Agarwal & Kempe 2008)
- Column generation (Aloise et al. 2010)
- Sparse IP and LP rounding (Dinh & Thai 2015)
- Approximation (Kawase et al. 2021)
- Bayan algorithm (Aref et al. 2022)

$$\max_{x_{ij}} Q = \frac{1}{2m}\left(\sum_{(i,j)\in V^2, i<j} 2b_{ij}(1-x_{ij}) + \sum_{i\in V} b_{ii}\right)$$

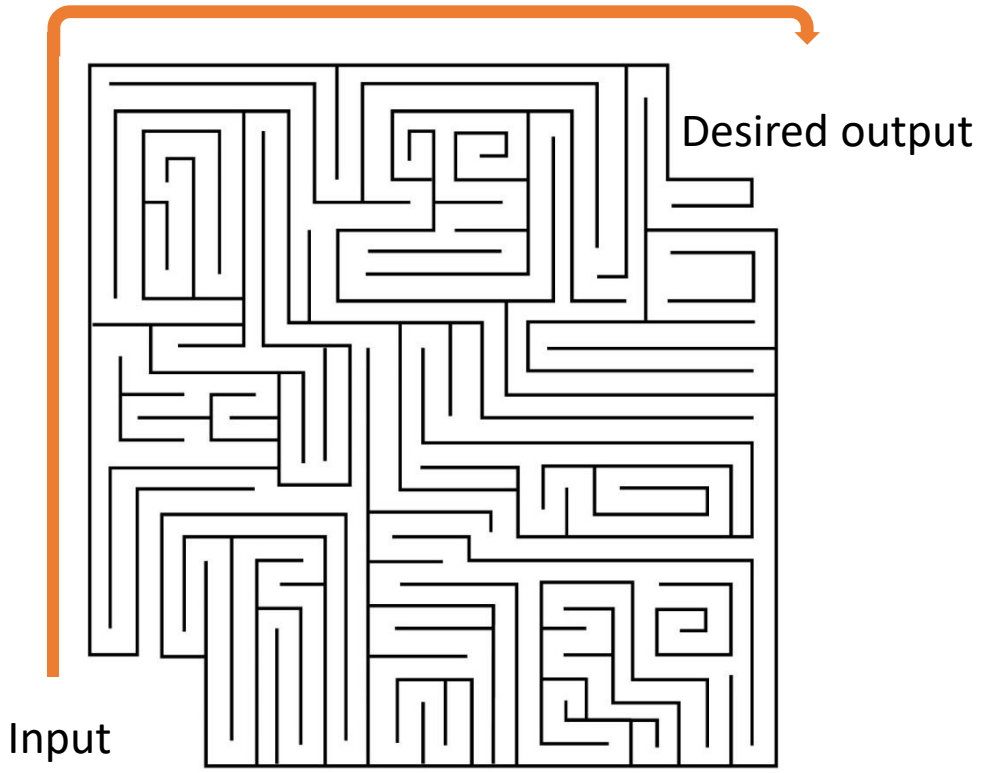
$$\text{s.t.} \quad x_{ik} + x_{jk} \geq x_{ij} \quad \forall (i,j) \in V^2, i < j, k \in K(i,j)$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in V^2, i < j$$

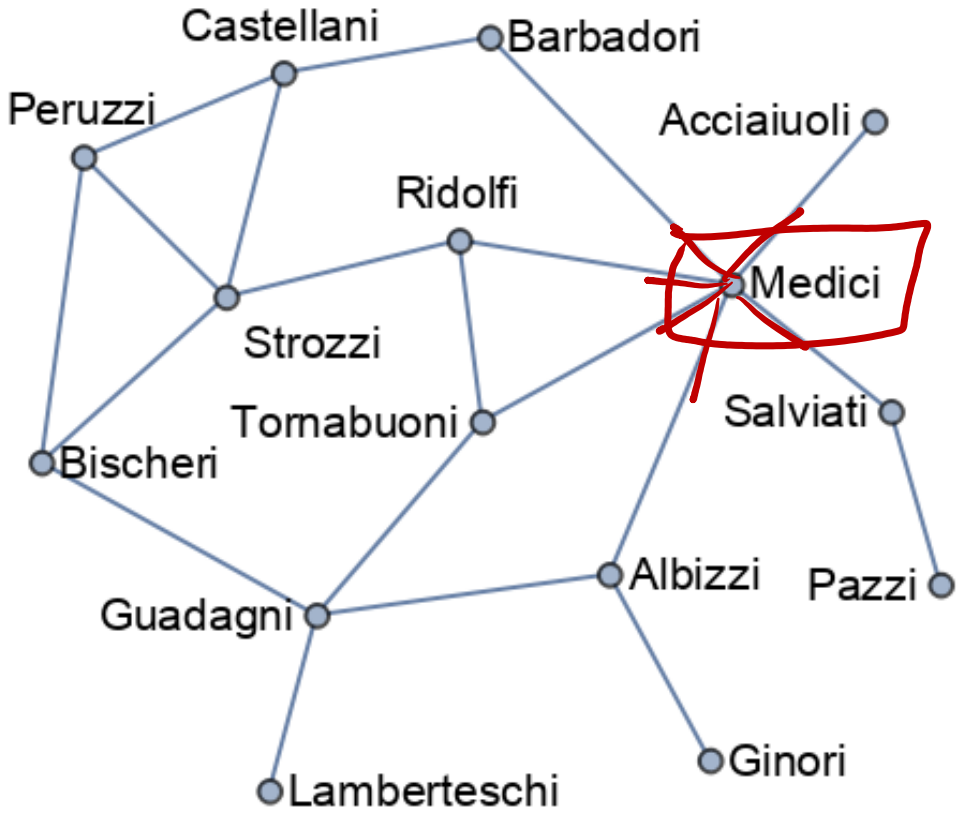


Approach 3: Other Methods

1. Kernighan-Lin bisection (Kernighan and Lin 1970)
2. RB Potts model with Erdős–Rényi as null (Reichardt & Bornholdt 2006)
3. Chinese whispers (Biemann et al. 2006)
4. Walktrap (Pons & Latapy 2006)
5. k-cut (Ruan & Zhang 2007)
6. Asynchronous label propagation (Raghavan et al. 2007)
7. Infomap (Rosvall & Bergstrom 2008)
8. Genetic Algorithm (Pizzuti 2008)
9. Semi-synchronous Label propagation (Cordasco & Gargano 2010)
10. Constant Potts Model (CPM) (Traag et al. 2011)
11. Significant scales (Traag et al. 2013)
12. Stochastic Block Model (SBM) (Peixoto 2014a)
13. SBM with Monte Carlo Markov Chain (MCMC) (Peixoto 2014b)
14. WCC (Prat-Pérez et al. 2014)
15. Surprise (Traag et al. 2015)
16. Diffusion Entropy Reducer (DER) (Kozdoba and Mannor 2015)
17. GemSec (Rozemberczki et al. 2019)
18. Bayesian Planted Partition (BPP) (Zhang and Peixoto, 2020)
19. Markov Stability (PyGenStability) (Arnaudon et al., 2023)



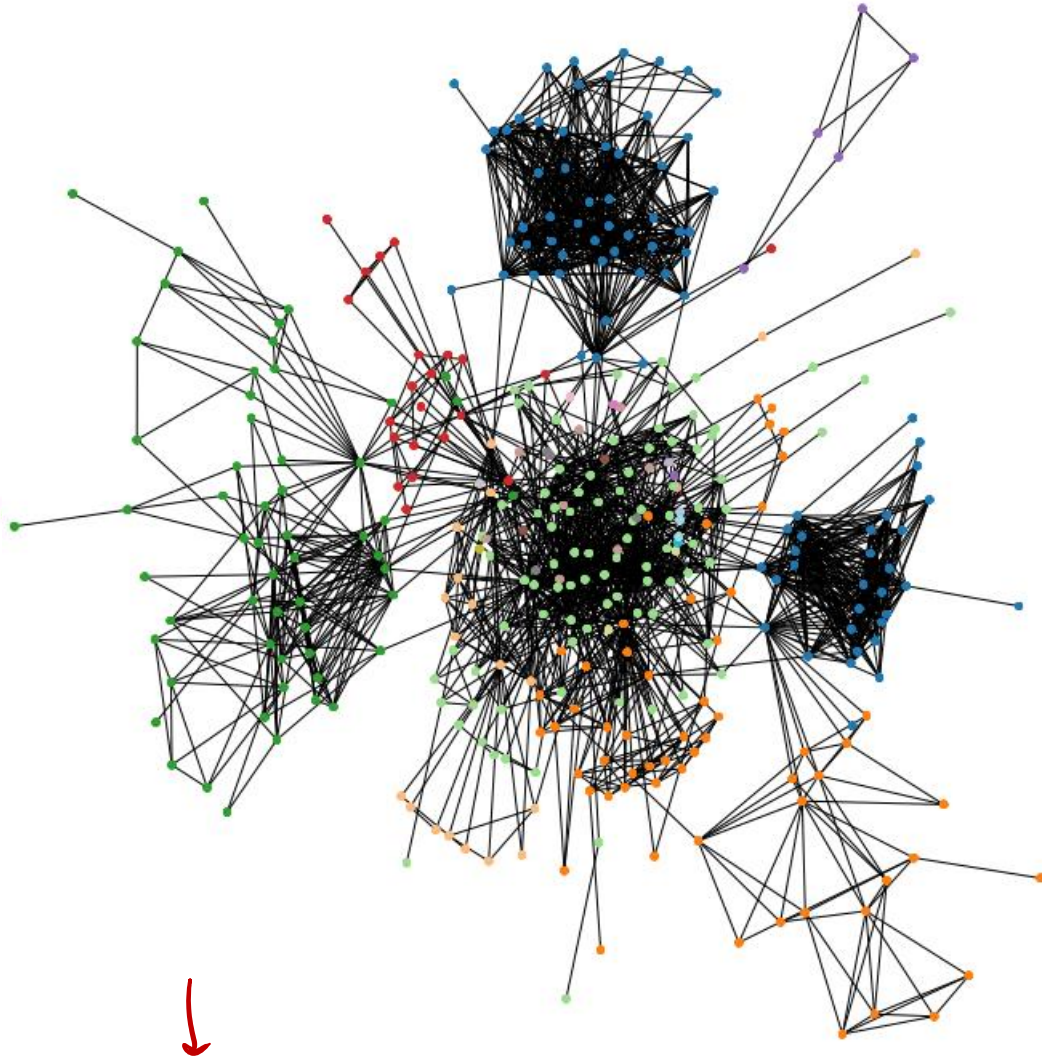
What is the maximum modularity of the Florentine families network? (resolution parameter γ=1)



1 network
10 modularity maximization algorithms
9 sub-optimal partitions
1 optimal partition

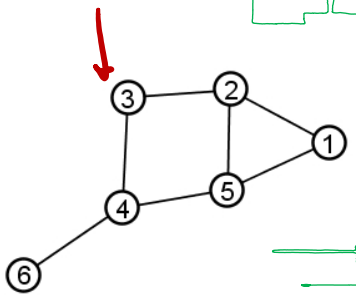
Dataset:
facebook_friends
m=1988

Q: modularity for a partition
Q*: maximum modularity of the graph
k: number of communities
AMI: adjusted mutual information
(similarity to an optimal partition)



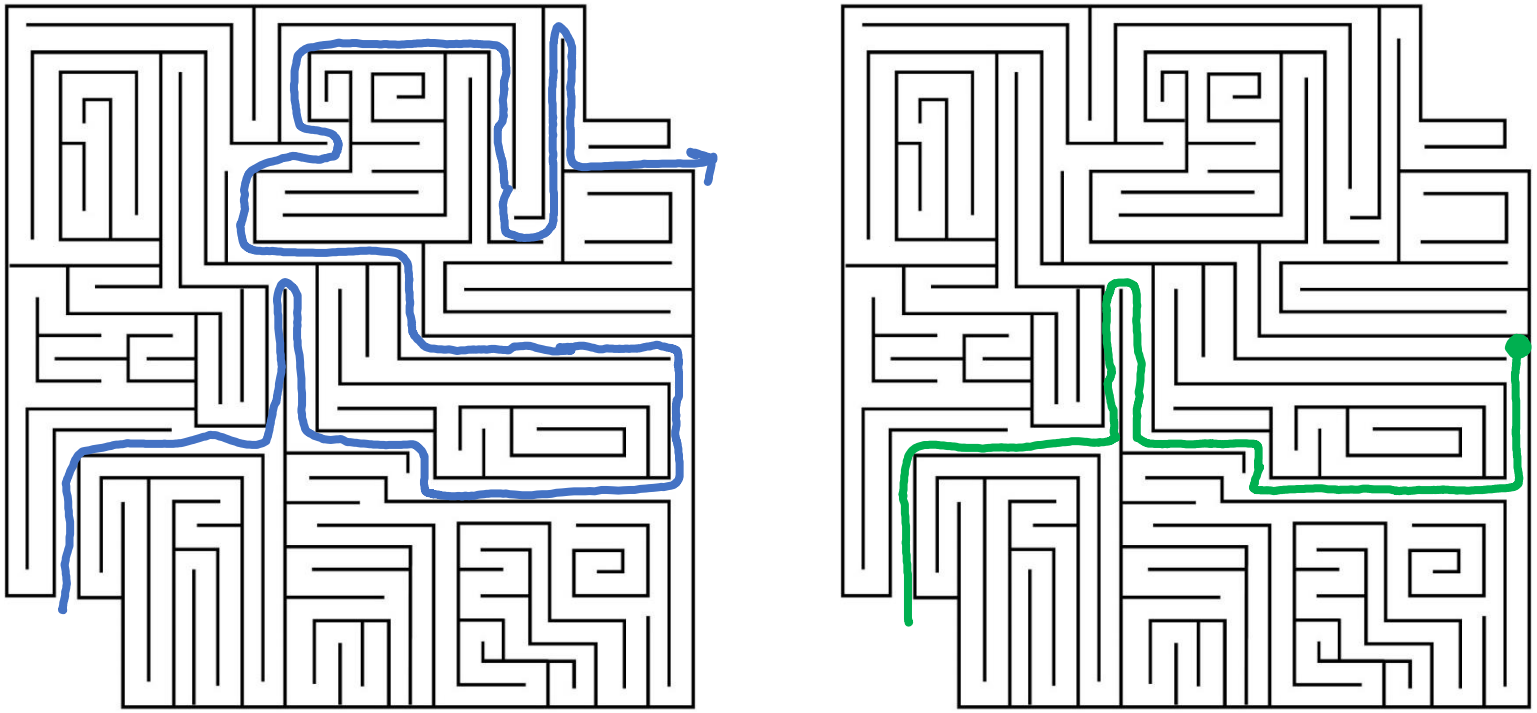
(a) Bayan, $Q^* = 0.7157714$, $k = 28$, AMI $= 1$

Evaluating modularity maximization heuristics



Algorithm	Partition (communities)	Modularity	Optimality percentage	Similarity to optimal partition (adjusted mutual information)
Heuristic method 1	[1,2,3] [4,5,6]	Q=0.4	Q/Q*=80%	90%
Heuristic method 2	[4,5,6] [1,2,3]	Q=0.4	Q/Q*=80%	90%
…				
Heuristic method 8	[1,4] [3,5] [2,6]	Q=0.3	Q/Q*=60%	70%
Exact method (IP)	[1,2,3,5] [4,6] All optimal partitions	Q*=0.5	100%	100%
			Y-axis	X-axis

Assessing the solution quality (optimality) of several modularity-based methods

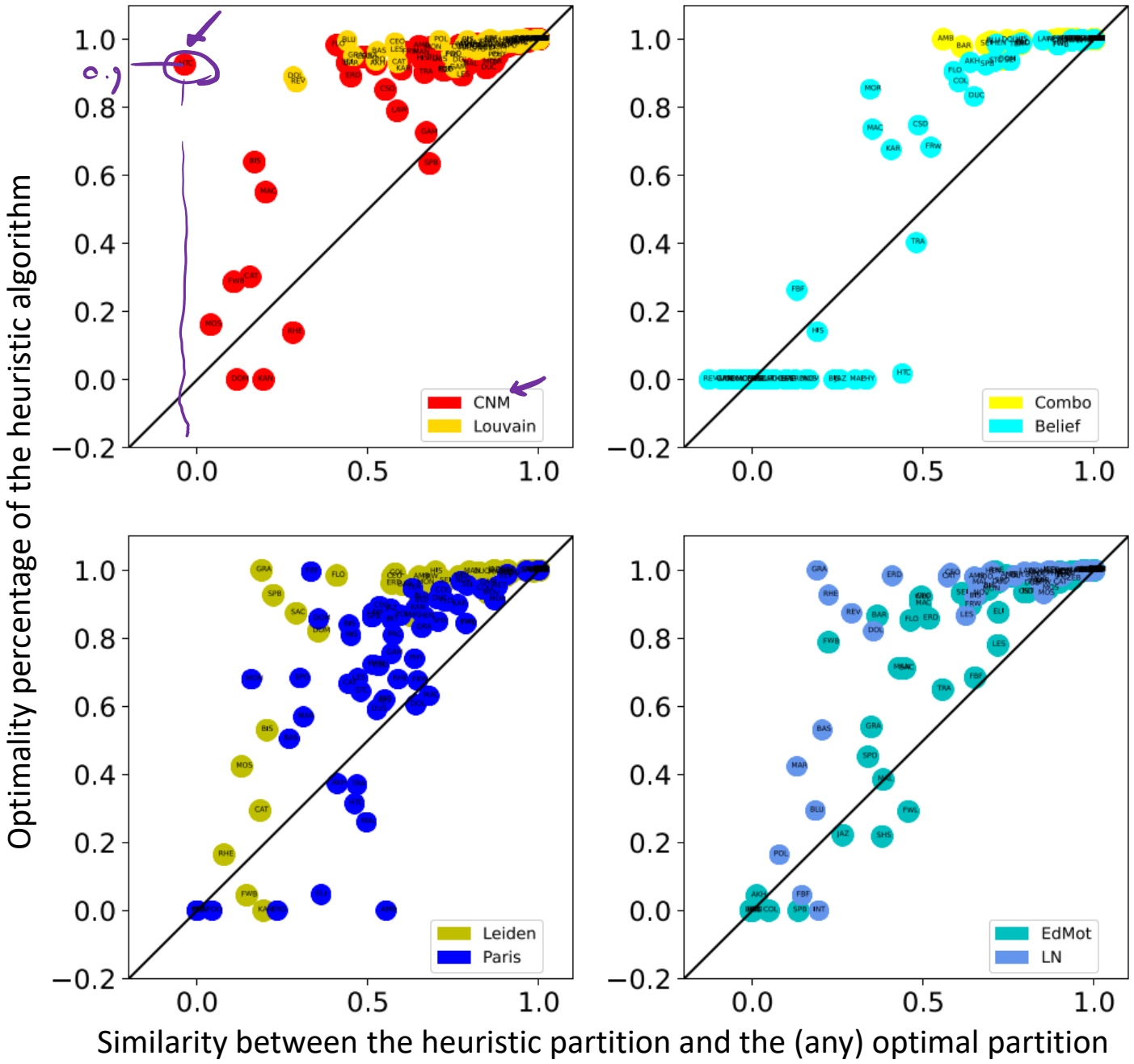


Test cases are 80 graphs with no more than 2812 edges:

- 60 real networks
- 10 Erdős–Rényi random graphs
- 10 Barabási–Albert random graphs

Each datapoint represents the performance of one algorithm on one test case.

1. Y-values: Many partitions are sub-optimal
2. X-values: Many partitions are dissimilar to any optimal partition
3. 45°-line: near-optimal partitions are not similar to any optimal partitions

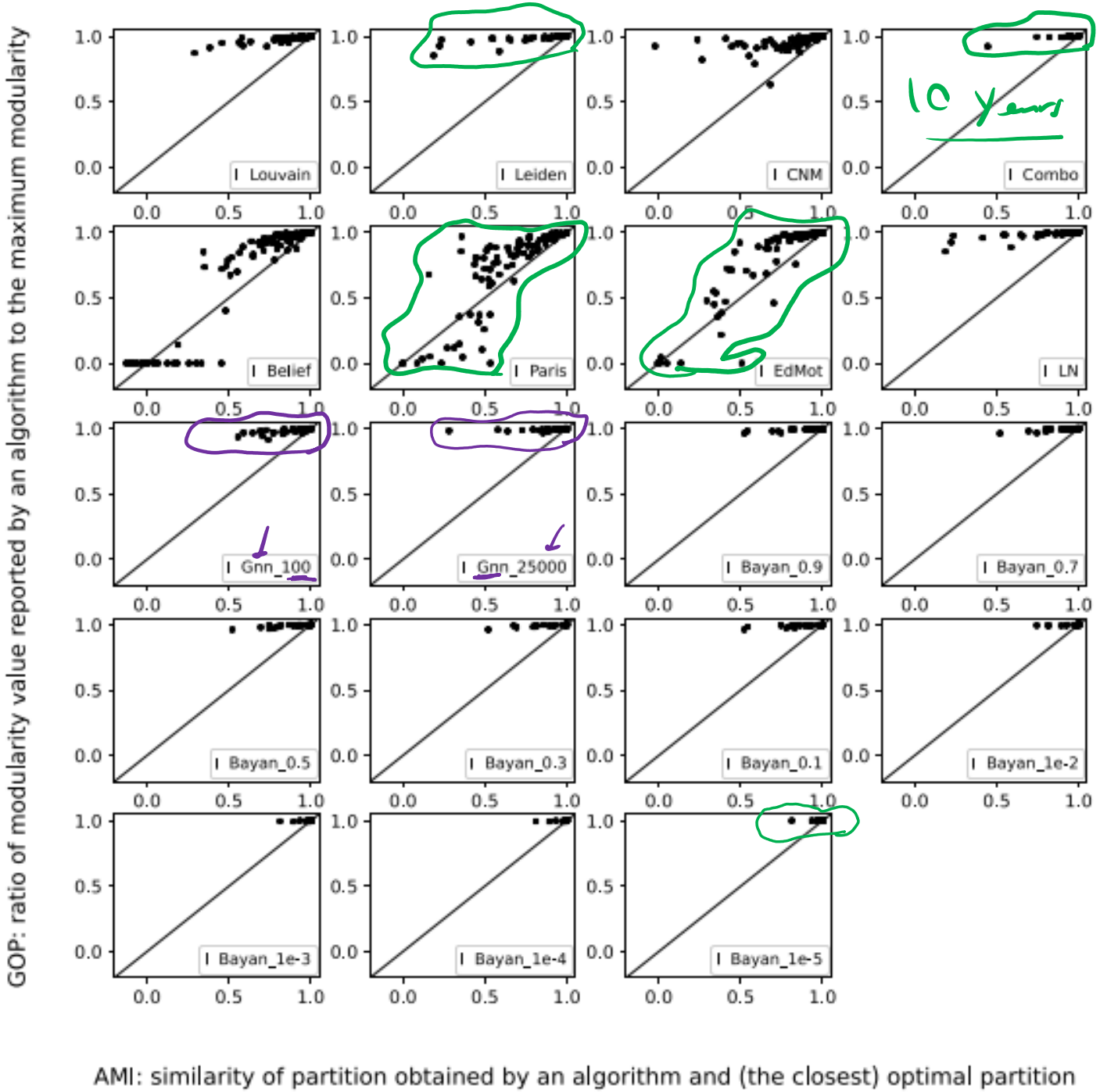


Test cases are 104 graphs with modular structure:

- 54 real networks
- 20 LFR random graphs
- 30 ABCD random graphs

Each datapoint represents the performance of one algorithm on one test case.

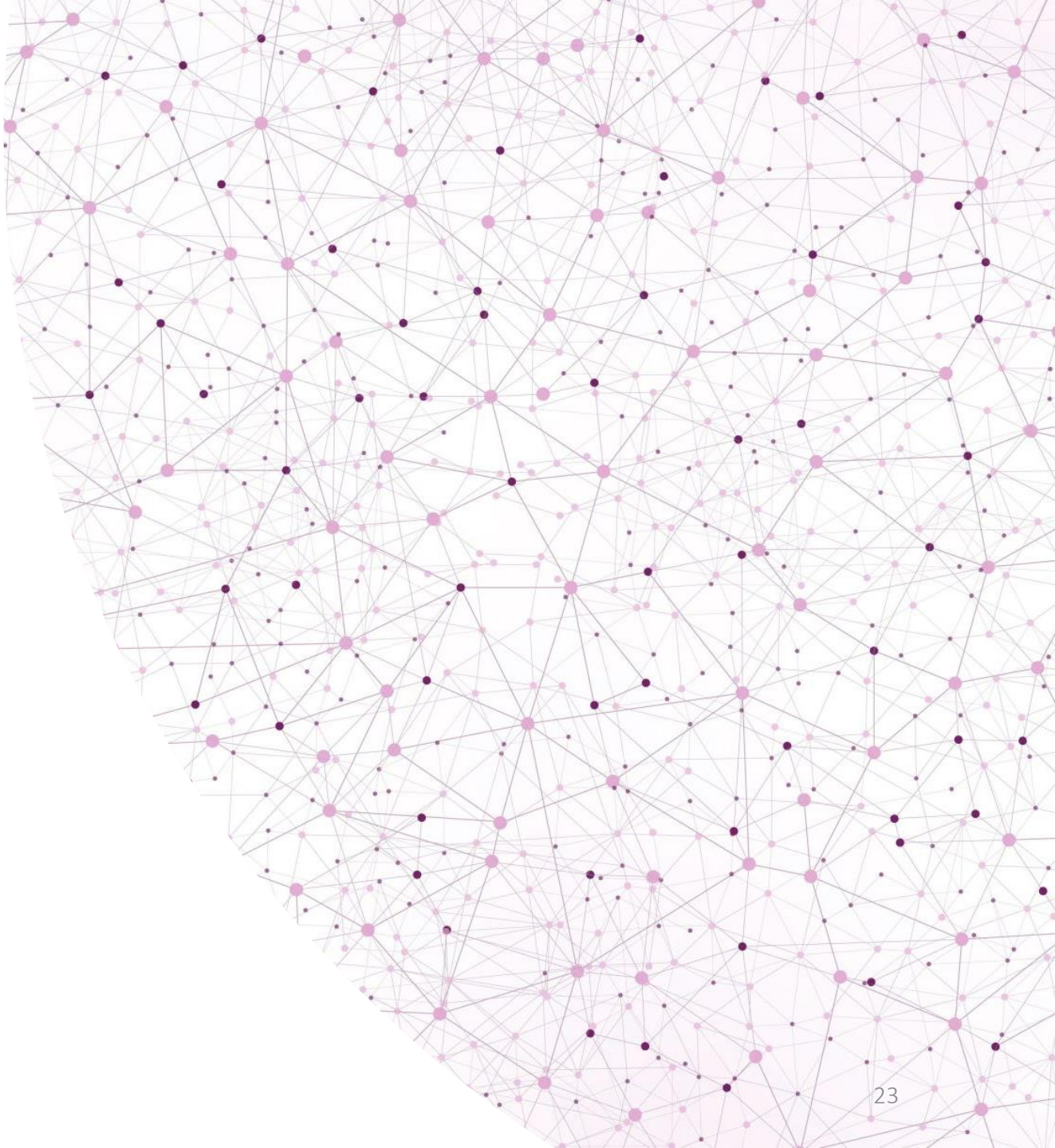
1. Y-values: Many partitions are sub-optimal
2. X-values: Many partitions are dissimilar to any optimal partition
3. 45°-line: near-optimal partitions are not similar to any optimal partitions



1

Heuristic modularity maximization algorithms rarely* maximize modularity.

*Only 19.4%-43.9% of the times according to our experiments on synthetic and real networks.

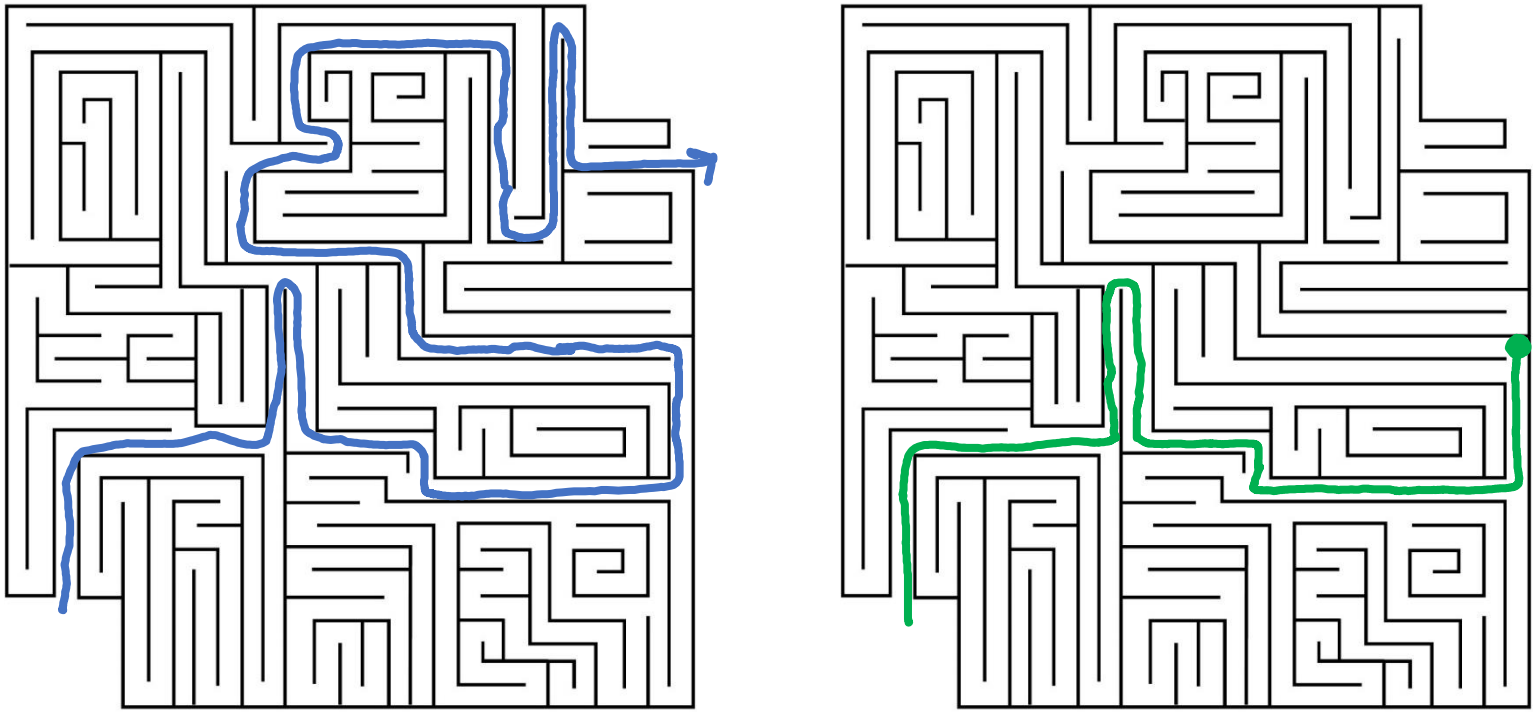


2

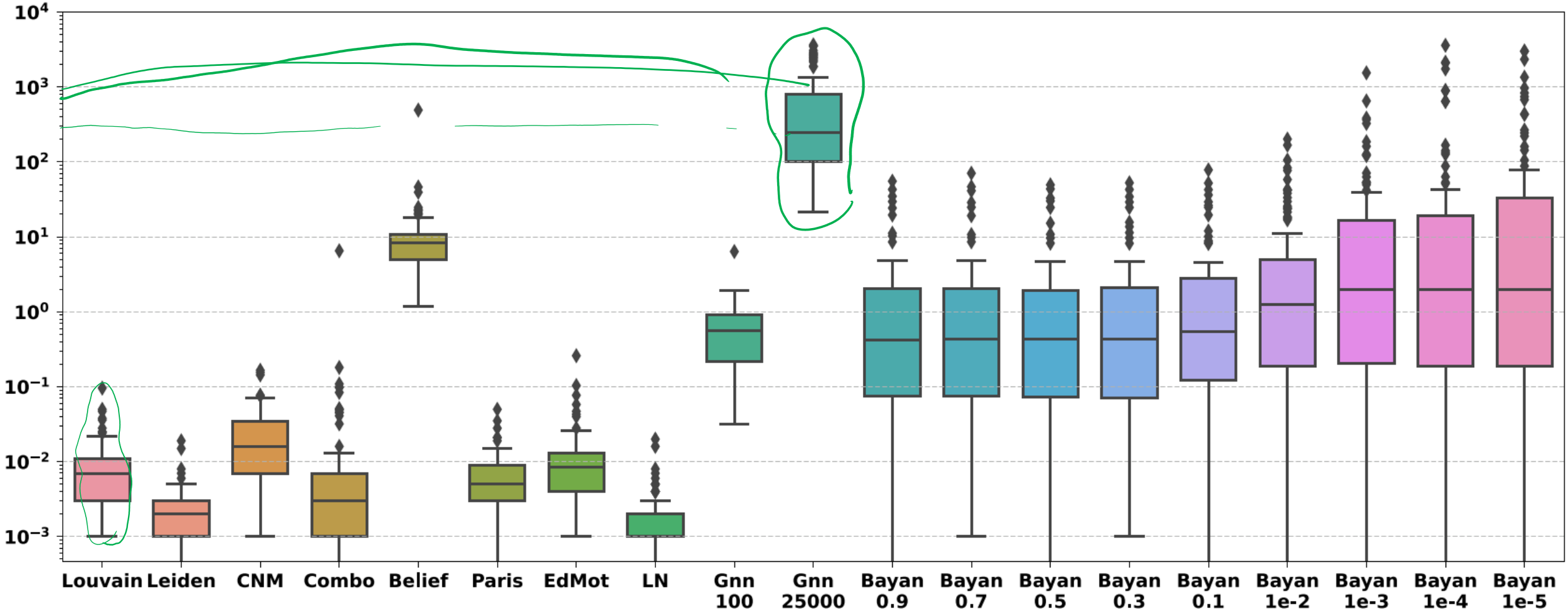
Suboptimal partitions of heuristic algorithms are disproportionately dissimilar to any optimal partition.

An x% suboptimality is often associated with a dissimilarity much larger than x% from any optimal partition.

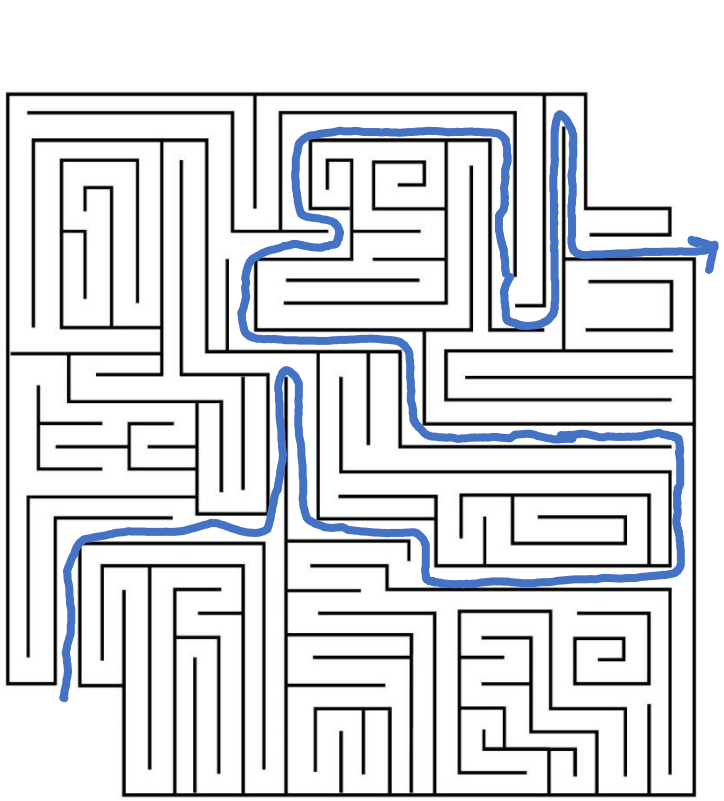
Assessing the time performance of several modularity-based methods

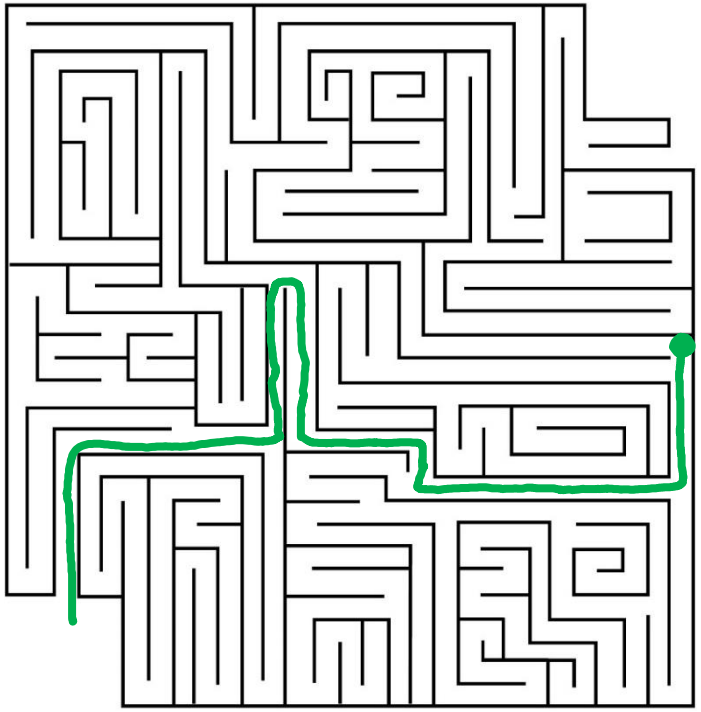


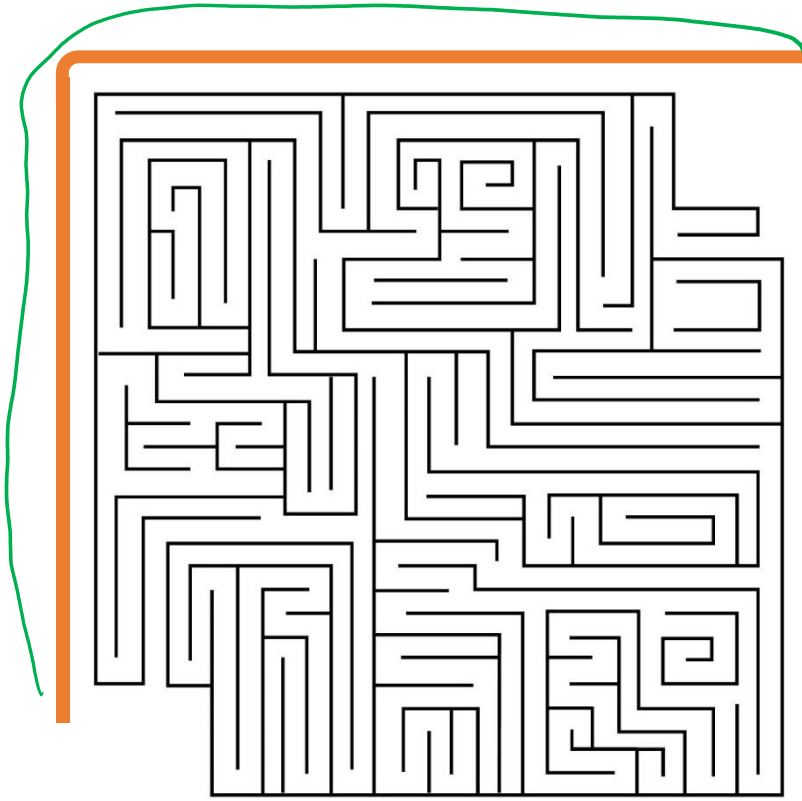
Solve time of ten modularity-based methods



Comparing 30 community detection algorithms based on retrieving ground-truth communities







Comparing 30 detection algorithms (on random ABCD graphs)

Test cases:

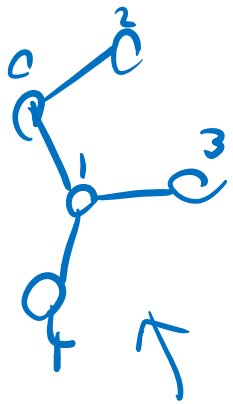
- 500 ABCD random graphs with up to 1000 edges (generated with planted communities)
- Noise ξ (mixing parameter)

  $\xi \in \{10\%, 30\%, 50\%, 70\%, 90\%\}$

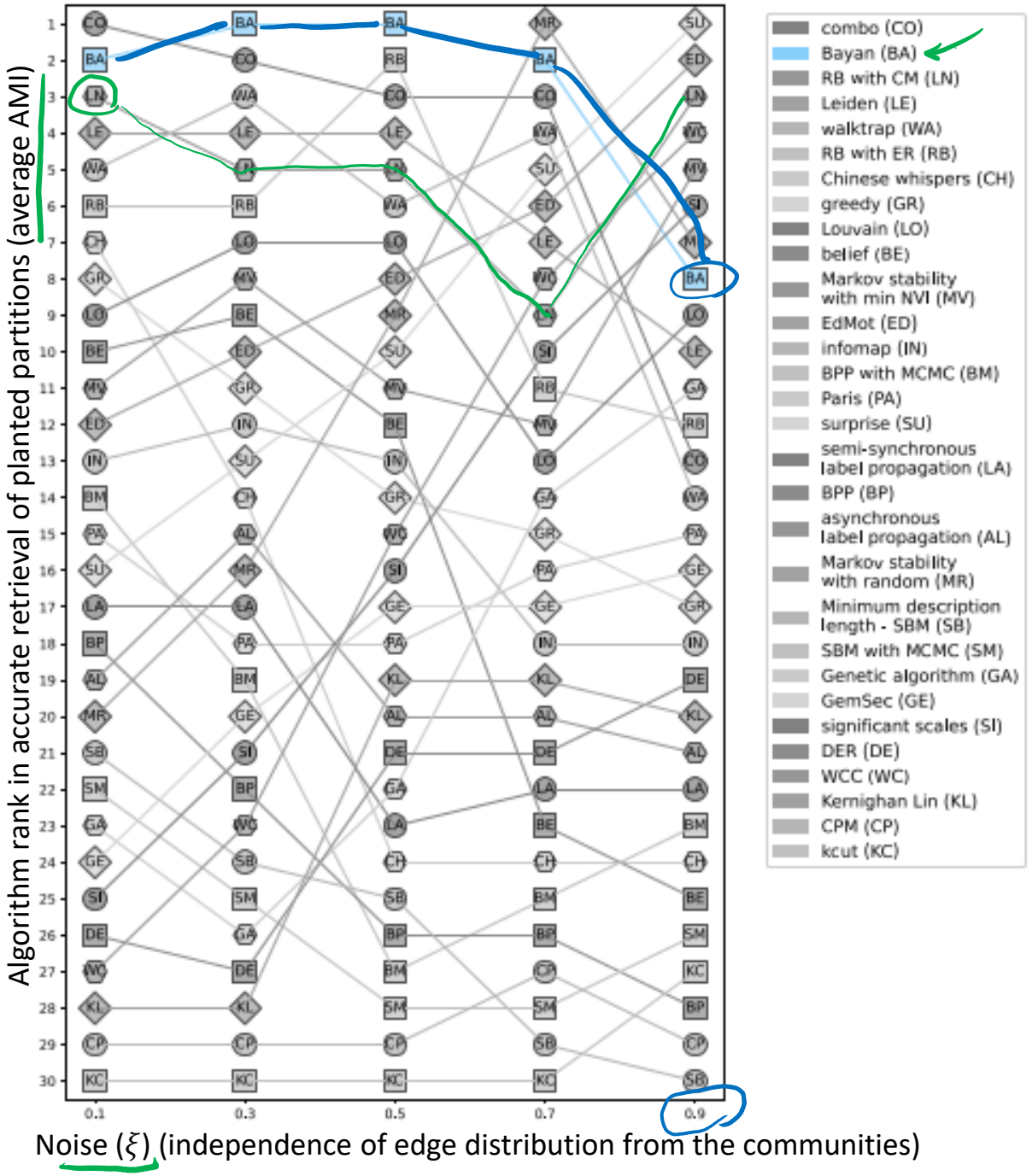
Performance measure:

- Similarity with the ground-truth communities (adjusted mutual information averaged over 100 graphs)

Comparing 30 community detection algorithms (based on retrieving ground-truth communities of random ABCD graphs)



[[0,2,1],[4,3]]



Comparing 30 community detection algorithms (on random LFR graphs)

Test cases:

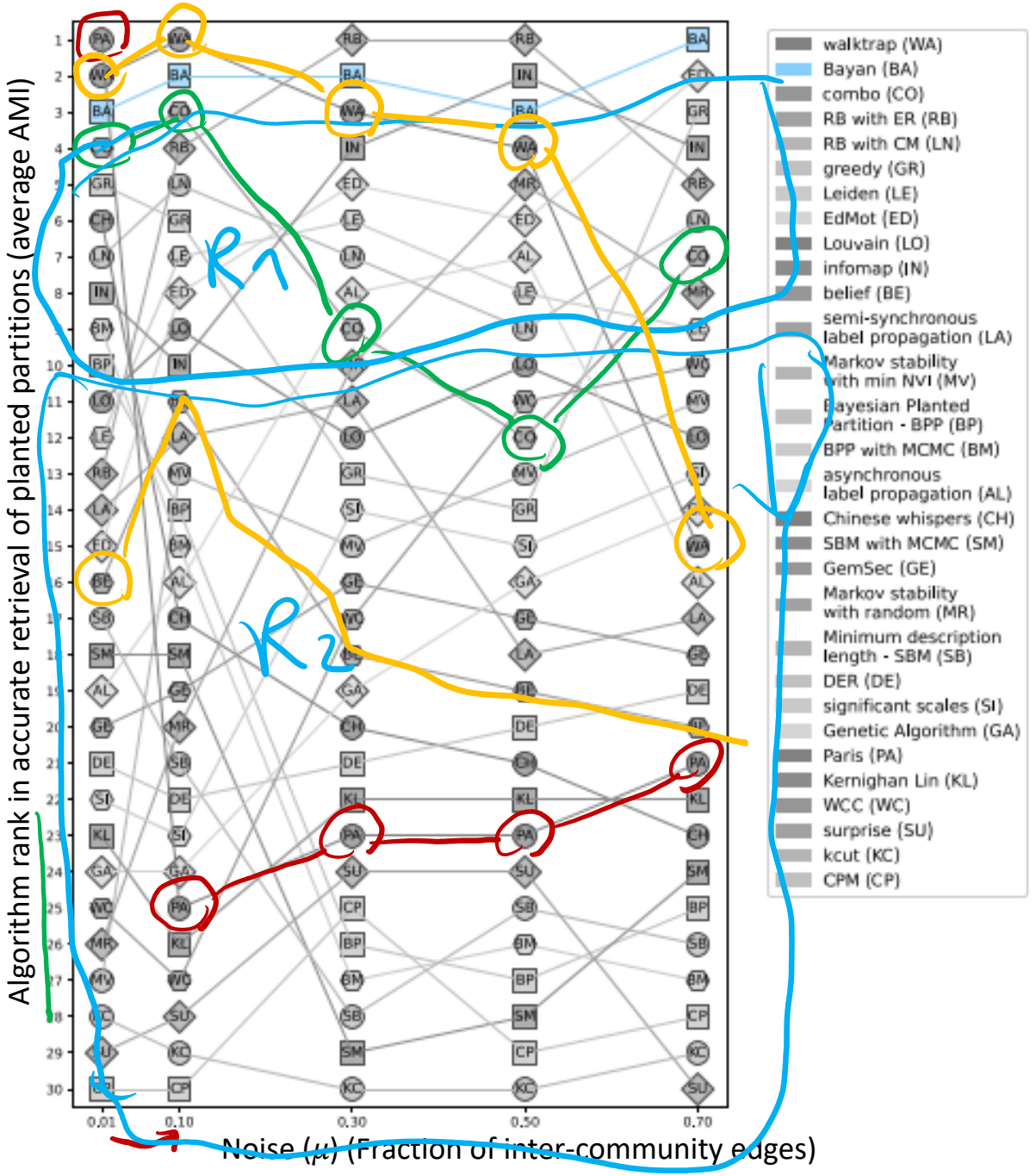
- 500 LFR random graphs with up to 1000 edges (generated with planted communities)
- Noise μ (fraction of inter-community edges)

$$\mu \in \{1\%, 10\%, 30\%, 50\%, 70\%\}$$

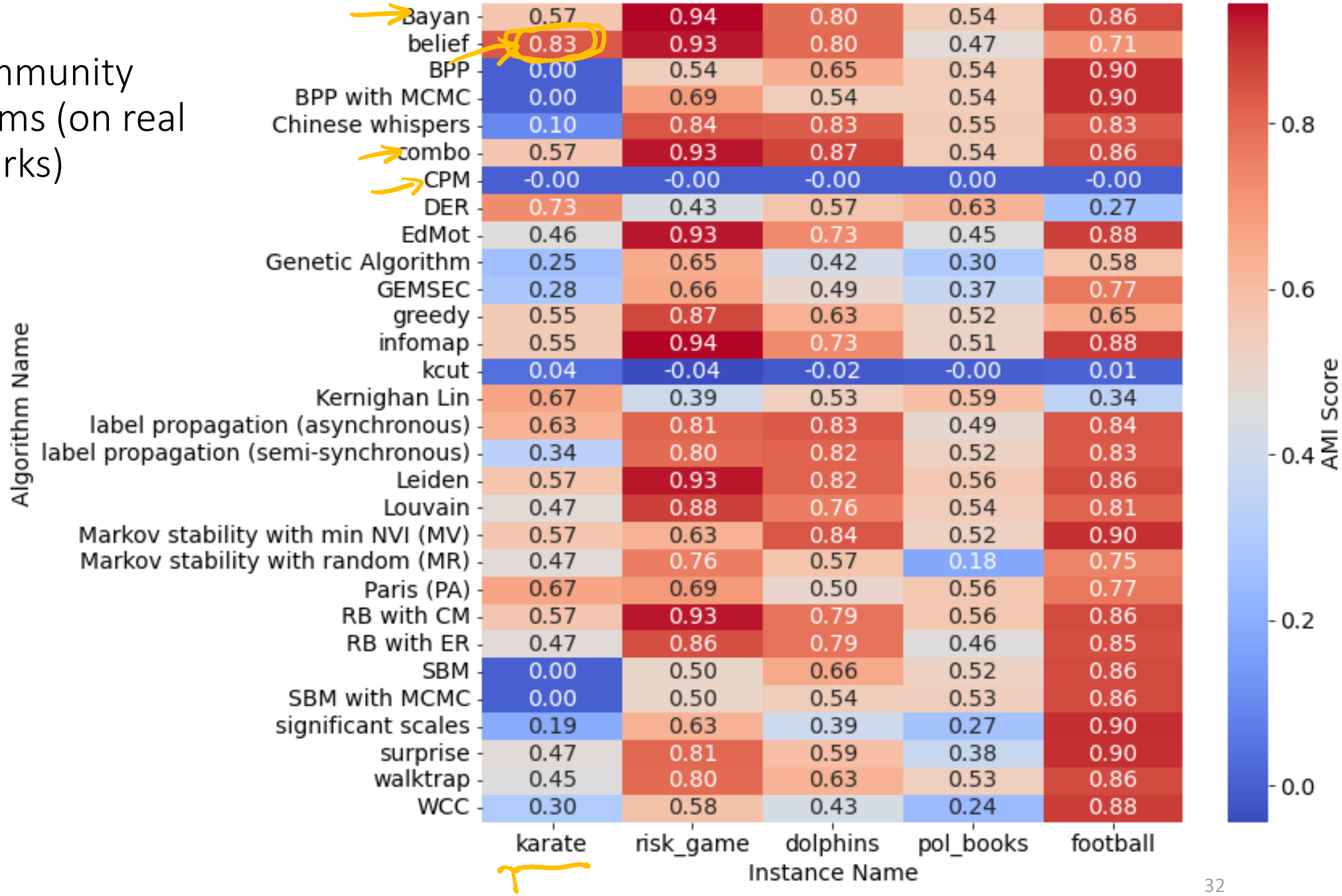
Performance measure:

- Similarity with the ground-truth communities (adjusted mutual information averaged over 100 graphs)

Comparing 30 community detection algorithms (based on retrieving ground-truth communities of random LFR graphs)



Comparing 30 community detection algorithms (on real network benchmarks)



Solve time: Bayan vs. Gurobi IP

Dataset: baseball

Nodes: baseball players (green) and steroid suppliers (red)

Edges: supply of steroid (illegal performance-enhancing substances)

Gurobi IP: 0.569 (s)
Bayan: 0.002 (s)

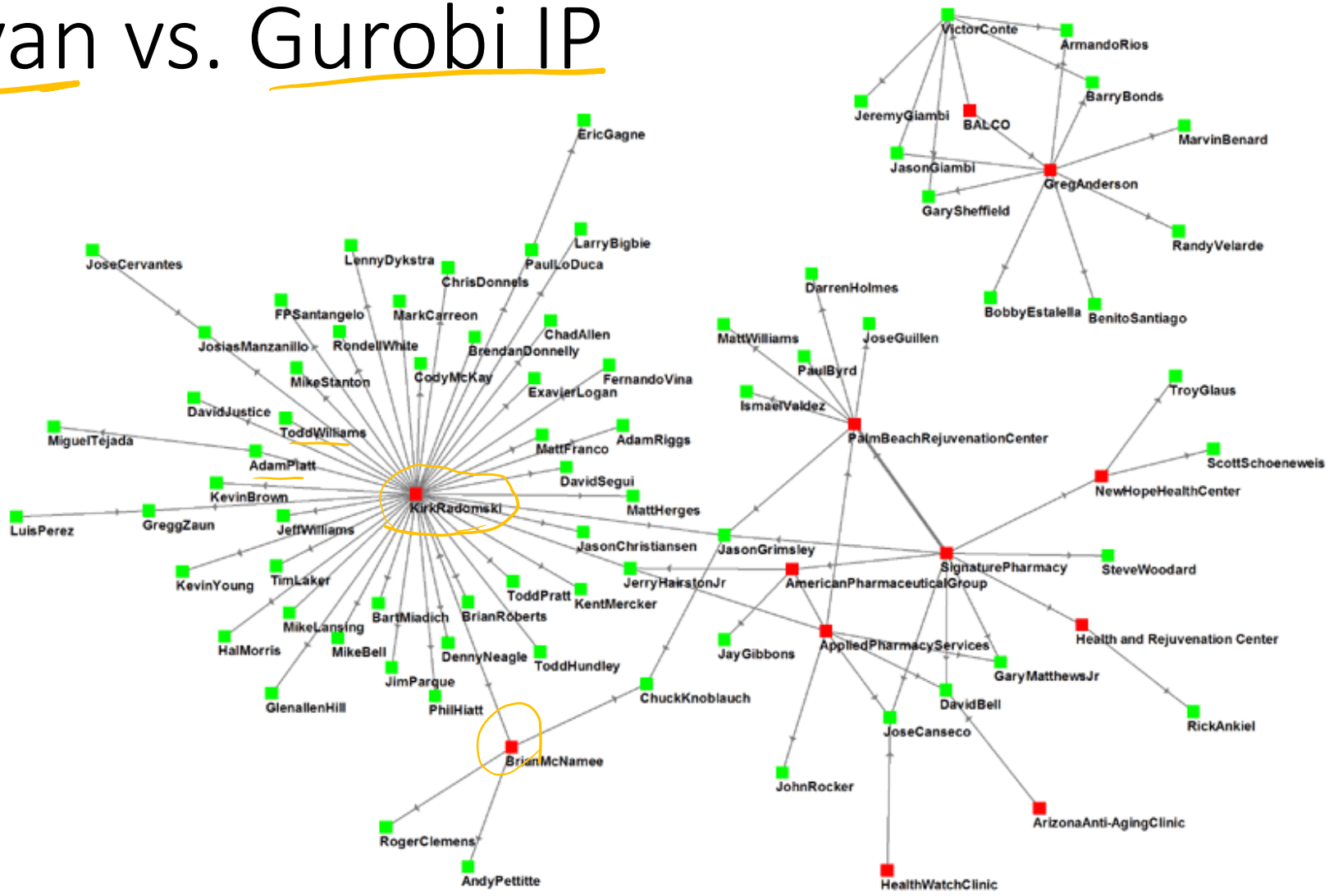


Figure from orgnet.com/steroids.html

Solve time: Bayan vs. Gurobi IP

Dataset: game_thrones

m=352

Nodes: characters from the tv-series Game of Thrones

Edges: Co-appearances within 15 words of each other in the text

Gurobi IP: 4.39 (s)
Bayan: 0.23 (s)

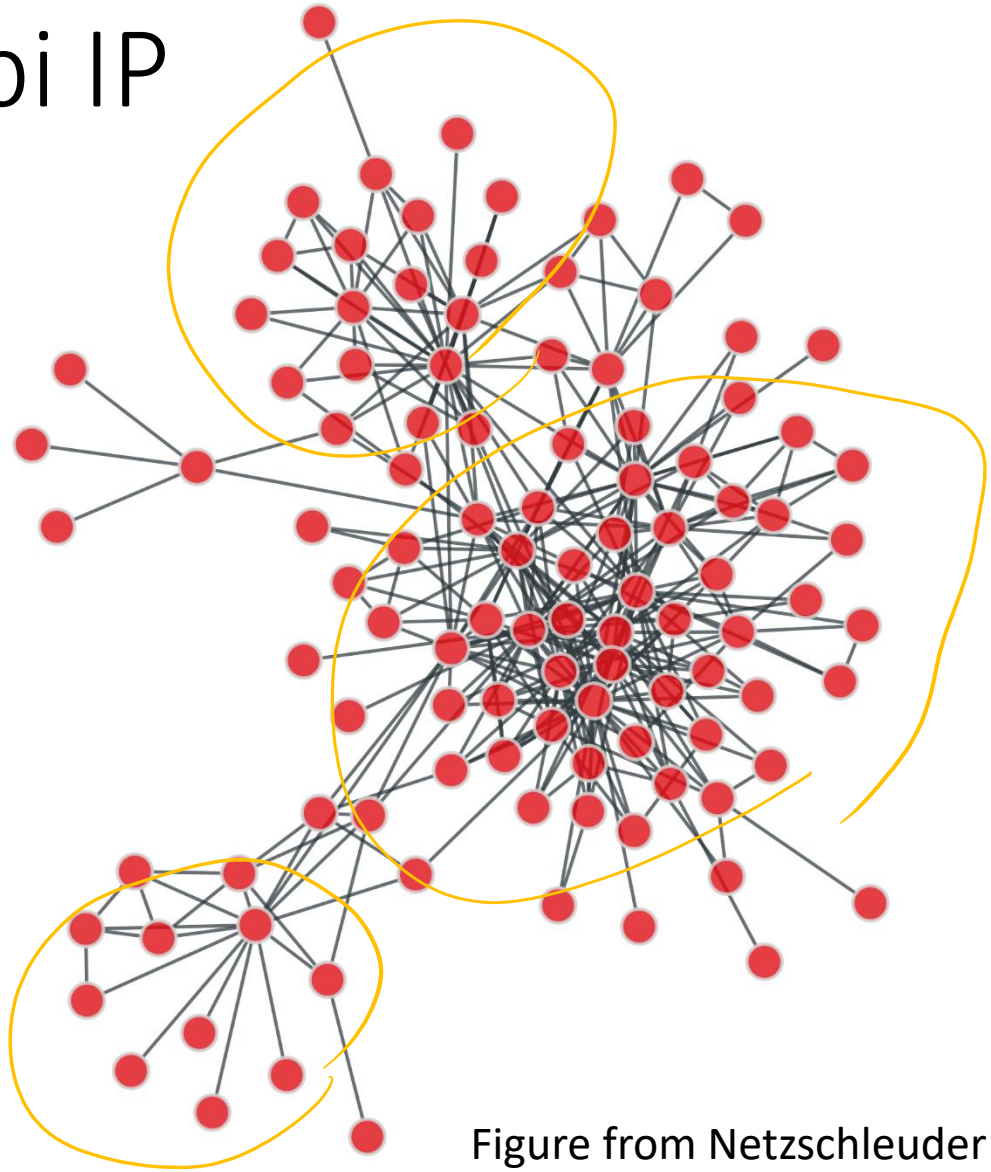


Figure from Netzschleuder

Solve time: Bayan vs. Gurobi IP

Dataset:
facebook_friends

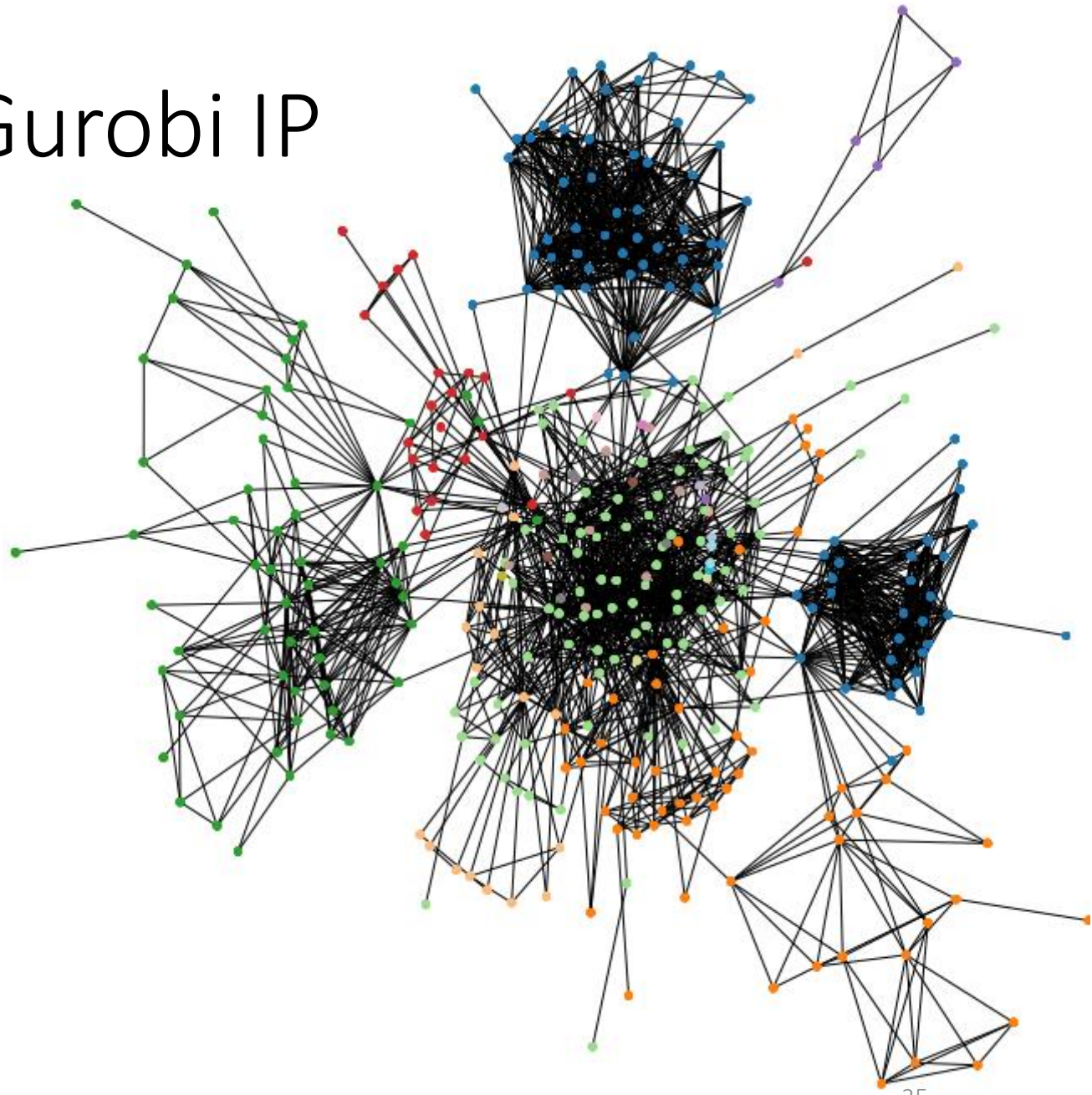
m=1988

Nodes: Facebook users

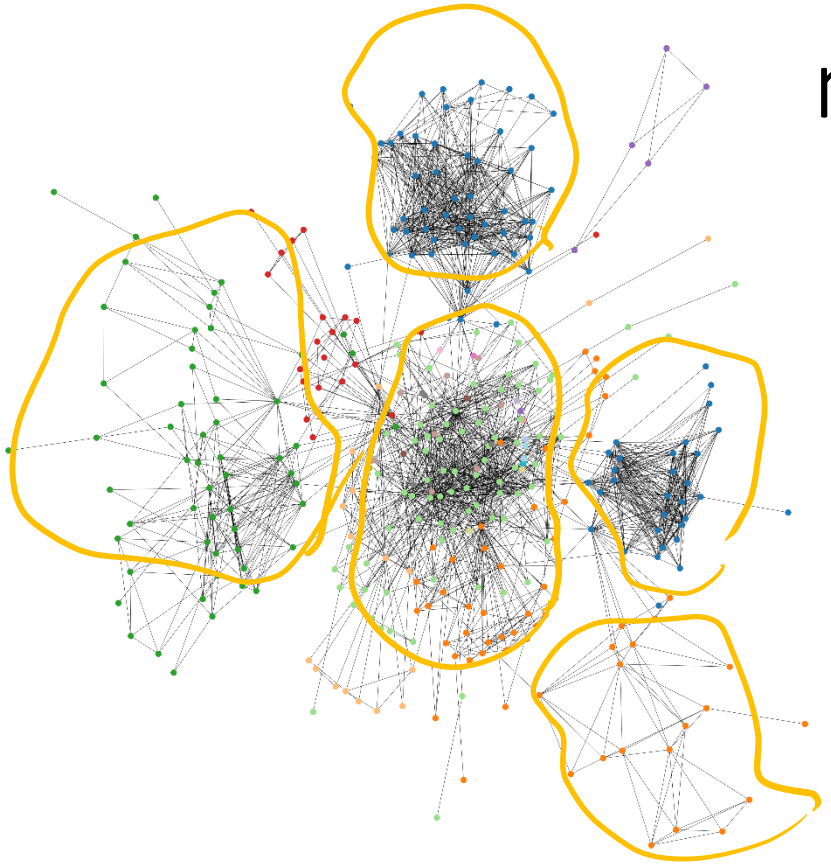
Edges: friendship on Facebook

Gurobi IP: 42.23 (s)

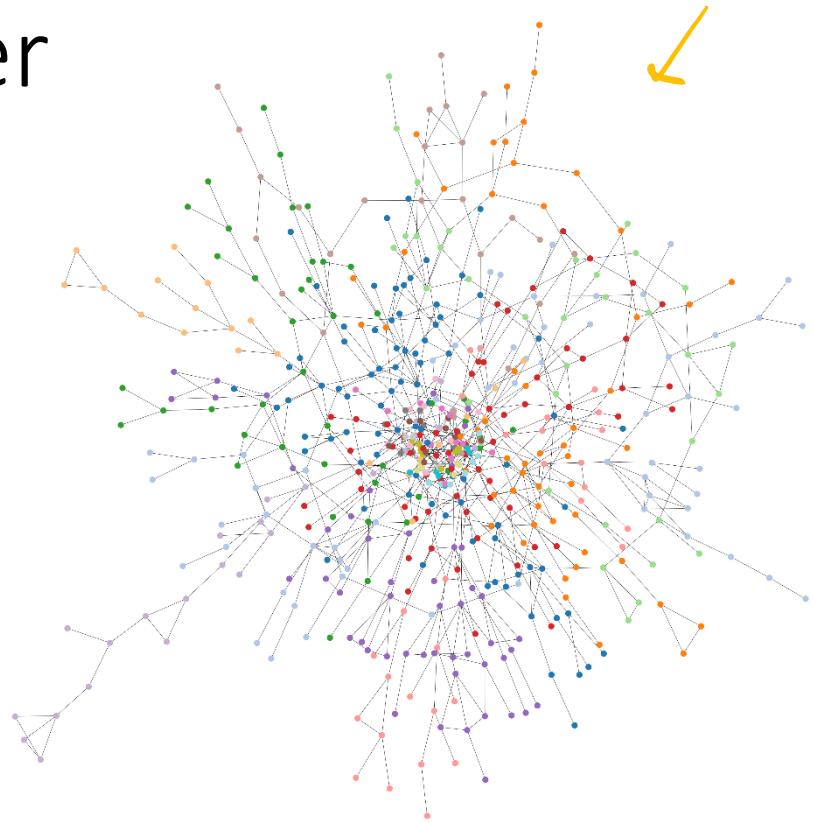
Bayan: 20.84 (s)



Graphs with high modularity are easier instances to solve

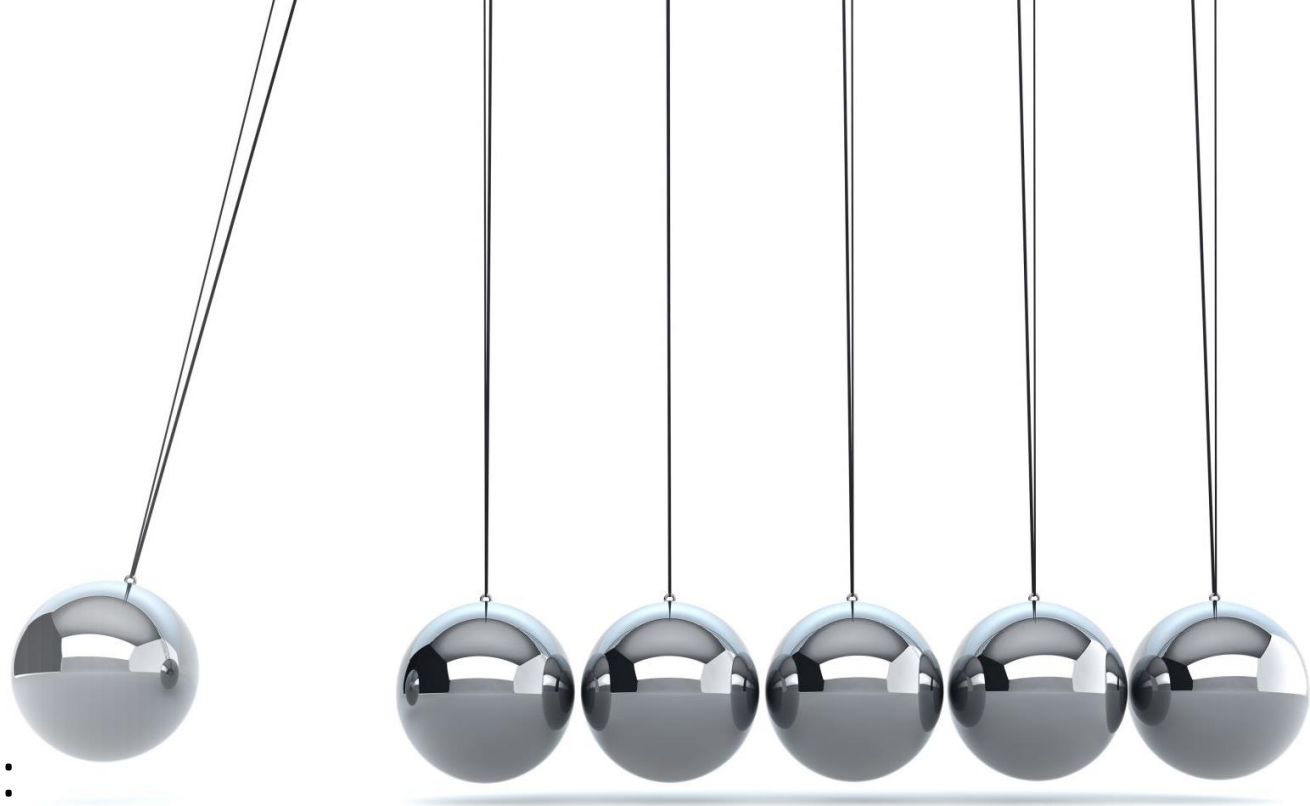


Facebook ego network
m=1988
Bayan: 20 seconds



Text messages of students
m=697
Bayan: 843.7 seconds

Pushing the computational limits



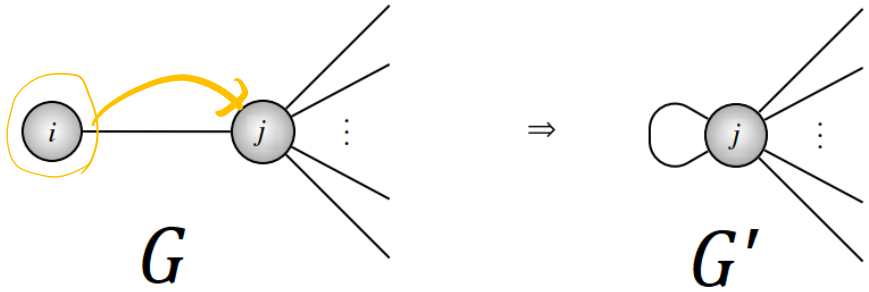
Simple idea:

1. Pendant clique filtering

Advanced ideas (check the references):

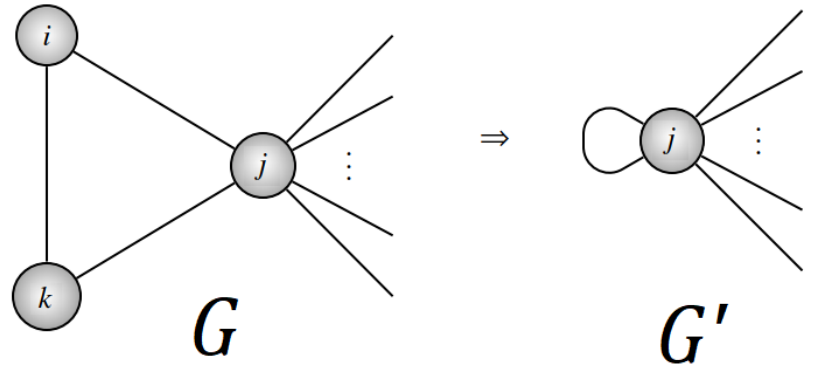
2. Lower bound in the branch-and-cut
3. Variable fixing
4. Implied branching
5. Parallel processing of the separating sets

1. Pendant nodes and cliques stay with neighbours



i stays with j

(Arenas et al. 2007)

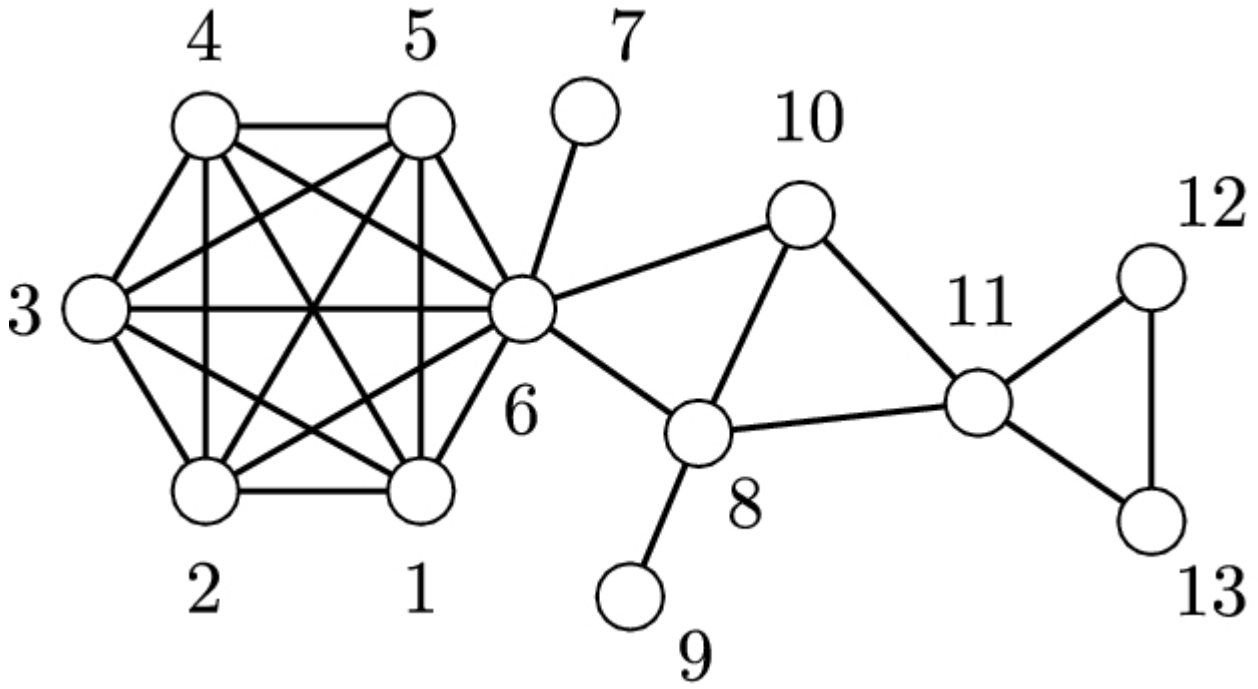


i and k stay with j

(Lorena et al. 2019)

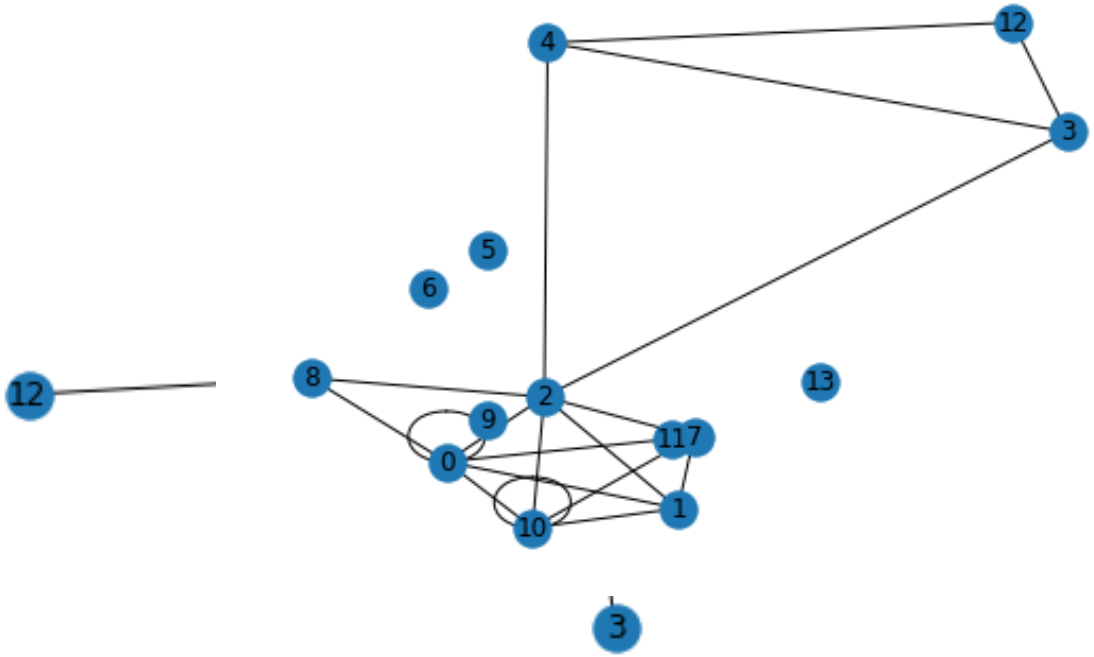
1. Pendant clique filtering

Reducing the input graph by replacing pendant cliques with self-loops. (Lorena et al. 2019)

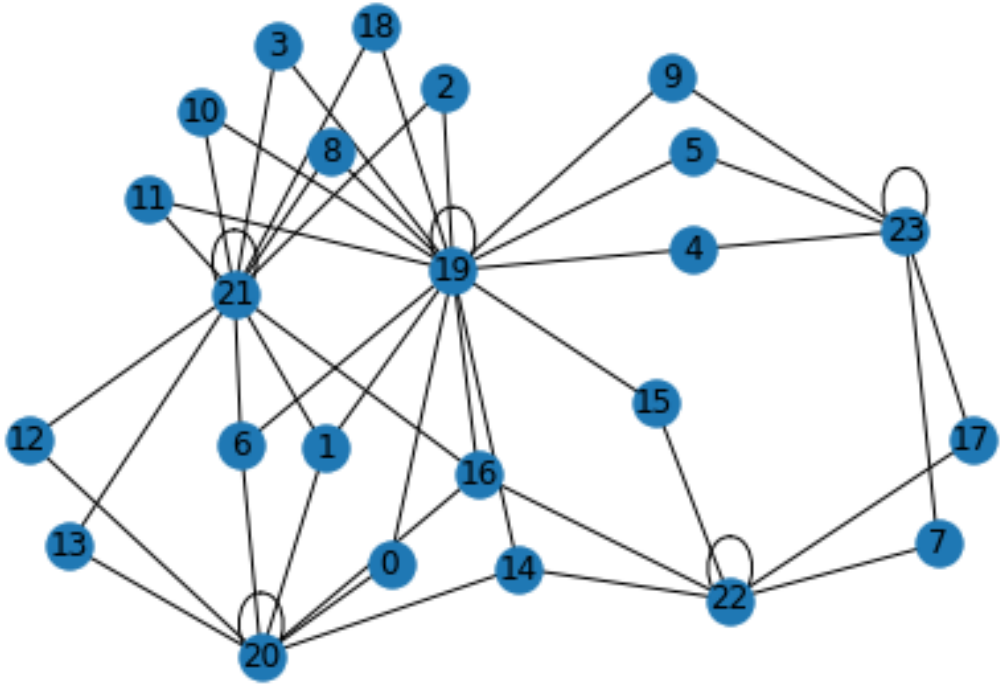


1. Pendant clique filtering

Philippines Ambassador bombing (1985-1989)



American Revolutionary groups (1765-1783)



Resources on Bayan

```
%pip install bayanpy
```

```
import networkx as nx
import bayanpy
```

```
G = nx.barbell_graph(5,2)
```

```
bayanpy.bayan(G)
```

Paper 1: doi.org/10.1007/978-3-031-36027-5_48
Paper 2: arxiv.org/pdf/2209.04562
Paper 3: arxiv.org/pdf/2310.10898
GitHub Repo: github.com/saref/bayan
Project website: bayanproject.github.io

Google Colab examples:
tinyurl.com/bayancolab

Additional references on community detection

Dinh, T.N., Thai, M.T.: Toward optimal community detection: From trees to general weighted networks. Internet Mathematics 11(3), 181–200 (2015)

Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. Physical review E 70(6), 066111 (2004)

Biemann, C.: Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In: Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing. TextGraphs-1, pp. 73–80. (2006)

Reichardt, J., Bornholdt, S.: Statistical mechanics of community detection. Physical Review E 74(1), 016110 (2006).

Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. Graph Algorithms Appl. 10(2), 191–218 (2006)

Ruan, J., Zhang, W.: An efficient spectral algorithm for network community discovery and its applications to biological and social networks. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), pp. 643–648. IEEE, (2007)

Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment 2008(10), 10008 (2008).

Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences 105(4), 1118–1123 (2008).

Leicht, E.A., Newman, M.E.J.: Community structure in directed networks. Physical Review Letters 100(11), 118703 (2008).

Pizzuti, C.: GA-Net: A Genetic Algorithm for Community Detection in Social Networks. In: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature — PPSN X - Volume 5199, pp. 1081–1090. Springer, Berlin, Heidelberg (2008)

Cordasco, G., Gargano, L.: Community detection via semi-synchronous label propagation algorithms. In: 2010 IEEE International Workshop On: Business Applications of Social Network Analysis (BASNA), pp. 1–8. IEEE, (2010)

Traag, V.A., Van Dooren, P., Nesterov, Y.: Narrow scope for resolution-limit-free community detection. Physical Review E 84(1), 016114 (2011)

Traag, V.A., Krings, G., Van Dooren, P.: Significant scales in community structure. Scientific reports 3(1), 1–10 (2013)

Peixoto, T.P.: Efficient monte carlo and greedy heuristic for the inference of stochastic block models. Physical Review E 89(1), 012804 (2014)

Prat-P´erez, A., Dominguez-Sal, D., Larriba-Pey, J.-L.: High quality, scalable and parallel community detection for large real graphs. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 225–236 (2014)

Sobolevsky, S., Campari, R., Belyi, A., Ratti, C.: General optimization technique for high-quality community detection in complex networks. Physical Review E 90(1), 012811 (2014)

Zhang, P., Moore, C.: Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. Proceedings of the National Academy of Sciences 111(51), 18144–18149 (2014)

Traag, V.A., Aldecoa, R., Delvenne, J.-C.: Detecting communities using asymptotical surprise. Physical Review E 92(2), 022816 (2015)

Bonald, T., Charpentier, B., Galland, A., Hollocou, A.: Hierarchical graph clustering using node pair sampling. In: MLG 2018 - 14th International Workshop on Mining and Learning with Graphs, London, UK (2018)

Traag, V.A., Waltman, L., van Eck, N.J.: From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports 9(1) (2019).

Li, P.-Z., Huang, L., Wang, C.-D., Lai, J.-H.: EdMot: An edge enhancement approach for motif-aware community detection. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 479–487 (2019)

Rozemberczki, B., Davies, R., Sarkar, R., Sutton, C.: Gemsec: Graph embedding with self clustering. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 65–72 (2019)