Mapping Flows on Complex Networks with Graph Neural Networks

NetSciX 2023, Venice, Italy

Christopher Blöcker[1,2], **Chester Tan**[1], Ingo Scholtes[1,2]

christopher.bloecker@uzh.ch

[1]Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

[2]Data Analytics Group
Department of Informatics
University of Zurich
Zurich, Switzerland

Universität
Zürich[UZH]

too long; didn't listen

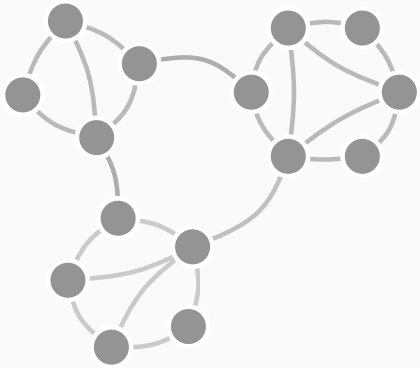
map equation + graph neural networks

- Community-detection methods typically rely on optimising objective functions without leveraging recent advances in deep learning
- We express the map equation in differentiable tensor form for optimisation with different (G)NN architectures and gradient descent on GPUs
- It works well for community detection in synthetic and real networks, naturally detects overlapping communities, and can incorporate node features

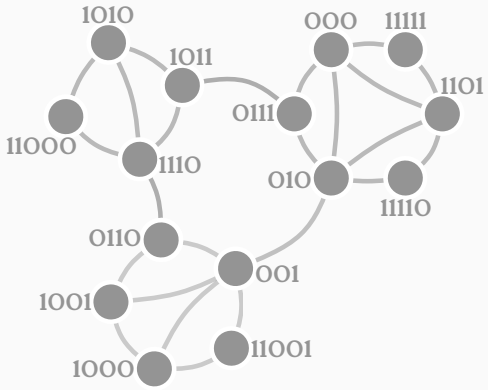
The Map Equation

→ Rosvall and Bergstrom; PNAS 2008

The Map Equation

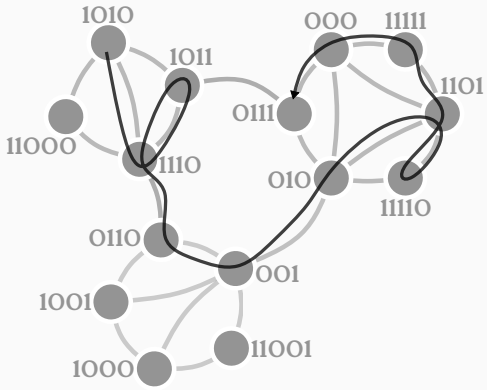


The Map Equation



1010 1110 1011 1110 0110 001 010
1101 11110 1101 11111 000 0111

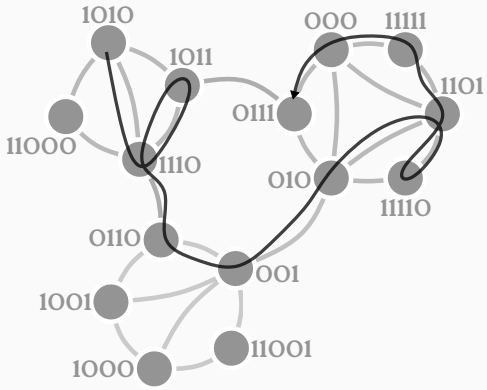
The Map Equation



1010 1110 1011 1110 0110 001 010
1101 11110 1101 11111 000 0111

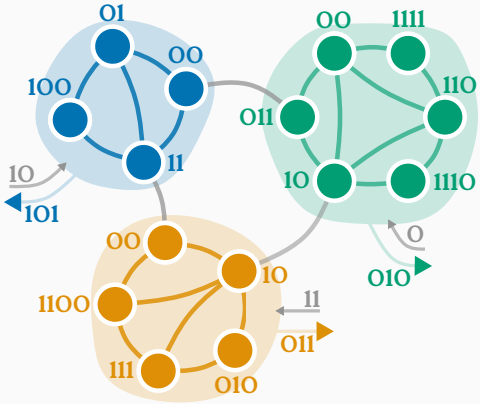
$$L(\mathsf{M}) = H(P) = -\sum_{p} p \log_2 p \text{ [bits]}$$

The Map Equation

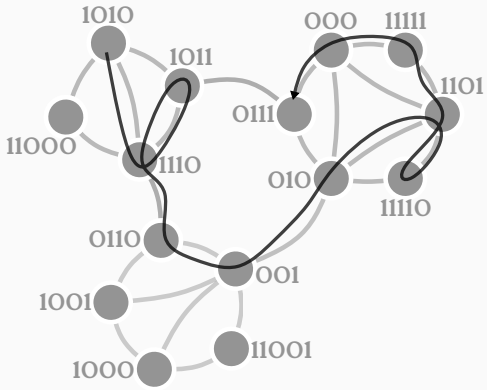


1010 1110 1011 1110 0110 001 010
1101 11110 1101 11111 000 0111

$$L(\mathrm{M}) = H(P) = -\sum_{p} p \log_2 p \ [\text{bits}]$$

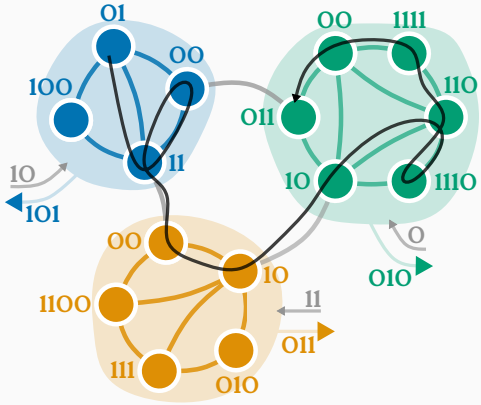


The Map Equation



1010 1110 1011 1110 0110 001 010
1101 11110 1101 11111 000 0111

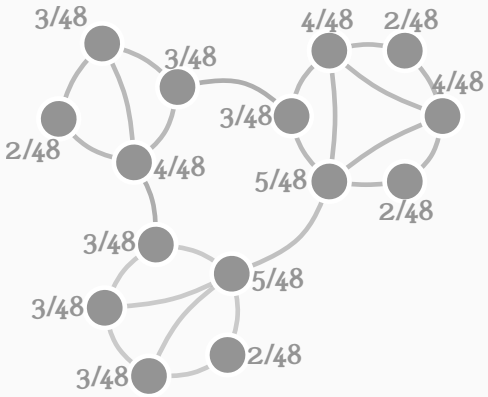
$$L(\mathrm{M}) = H(P) = -\sum_{p} p \log_2 p \text{ [bits]}$$



1001 11 00 11 10111001 10 011010
110 1110 110 1111 00 011

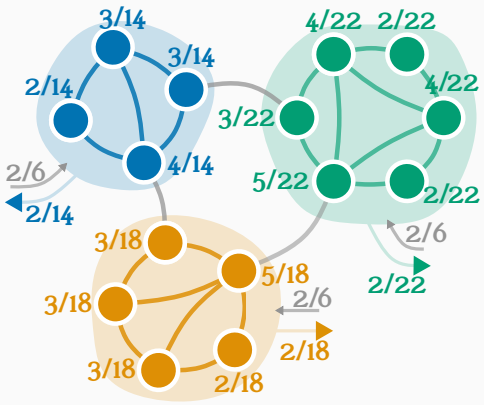
$$L(\mathrm{M}) = qH(Q) + \sum_{\mathrm{m}\in\mathrm{M}} p_{\mathrm{m}} H(P_{\mathrm{m}}) \text{ [bits]}$$

The Map Equation



3.84 bits

$$L(\mathsf{M}) = H(P) = -\sum_{p} p \log_2 p \text{ [bits]}$$



3.05 bits

$$L(\mathsf{M}) = qH(Q) + \sum_{\mathsf{m}\in\mathsf{M}} p_{\mathsf{m}} H(P_{\mathsf{m}}) \text{ [bits]}$$

The Map Equation Goes Neural

The Map Equation Goes Neural

- We introduce a soft cluster assignment matrix, making node assignments partial
  $\rightarrow$ differentiable map equation
  $\rightarrow$ overlapping modules
- We implement the differentiable map equation in tensor form
  $\rightarrow$ loss term for (G)NNs
- Optimisation with gradient descent means differentiating the codelength with respect to soft cluster assignments
  $\rightarrow$ update all assignments a bit instead of changing one binary assignment
- No regularisation beyond using the map equation

$$S = \mathrm{softmax}\left(\mathrm{GNN}\left(A, X\right)\right) \qquad \underset{S}{\mathrm{argmin}}\, L\left(S, F\right)$$

Going Neural

The map equation can be rewritten as

→ Rosvall et al.; Eur. Phys. J. 2009

$$\Big(\sum_{m\in M} q_m\Big)\log_2\Big(\sum_{m\in M} q_m\Big) - 2\sum_{m\in M} q_m \log_2 q_m - \sum_{u\in V} p_u \log_2 p_u + \sum_{m\in M}\Big(m_{exit} + \sum_{u\in m} p_u\Big)\log_2\Big(m_{exit} + \sum_{u\in m} p_u\Big)$$

We implement the four parts

$$e_1 = \sum_{a,b} E_{ab} - \sum_{a,b} \delta_{ab} E_{ab} \quad (e_2)_a = \sum_b (E_{ab} - \delta_{ab} E_{ab}) \quad (e_3)_i = p_i \quad (e_4)_b = \sum_a (E_{ab} - \delta_{ab} E_{ab}) + \sum_a E_{ab}$$

where

- S is the soft cluster assignment matrix,
- p is the vector of node visit rates.
- F is the flow matrix,
- $E_{ab} = \sum_{i,j} S^T_{ia} F_{ij} S_{jb}$ summarises the flow from module a to b,

And put them back together,

$$L(S,F) = e_1 \log_2 e_1 - 2\sum_{a,b}\left((e_2)_a \delta_{ab} \log_2 (e_2)_b\right) - \sum_{i,j}\left((e_3)_i \delta_{ij} \log_2 (e_3)_j\right) + \sum_{a,b}\left((e_4)_a \delta_{ab} \log_2 (e_4)_b\right)$$

Evaluation

Evaluation – Setup

Community Detection

We use Infomap and different (G)NN architectures

- Simple linear layer and MLP
- GCN_t with $t \in \{1, 2\}$ layers → Kipf and Welling; arXiv 2016
- GIN_t with $t \in \{1, 2, 3\}$ layers → Xu et al.; arXiv 2018
- GNN_t with $t \in \{1, 2\}$ layers: message passing + skip-connections between MLP layers
- learning rate $\{10^{-1}, 10^{-2}, \ldots, 10^{-6}\}$

Datasets

25 LFR networks with planted (hard) communities for each combination of

- $n \in \{100, 1000\}$ nodes, $\boldsymbol{X} = \boldsymbol{I}$
- average degree $k \in \{\ln n, 2 \ln n\}$
- maximum degree $k_{\max} = 2\sqrt{n}$
- mixing $\mu \in \{0.1, 0.2, \ldots, 0.8\}$

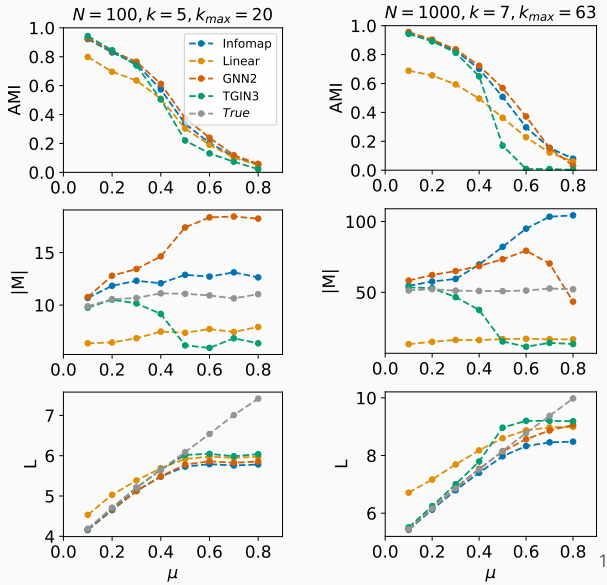
Results – undirected

We measure

- AMI against ground truth
- Number of modules |M|
- Codelength L

Main points

- Similar performance to Infomap, slighly outperforming it (AMI)
- performance $\sim$ (G)NN expressivity $\rightarrow$ over/underfitting (|M|)
- Infomap achieves lowest L
- consider all three measures!
  $\rightarrow$ similar AMI but different |M|
  $\rightarrow$ Smaller L always better?



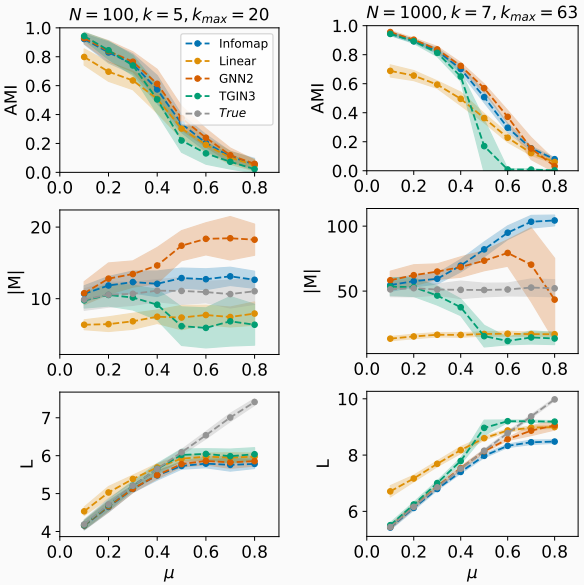
Results – undirected

We measure

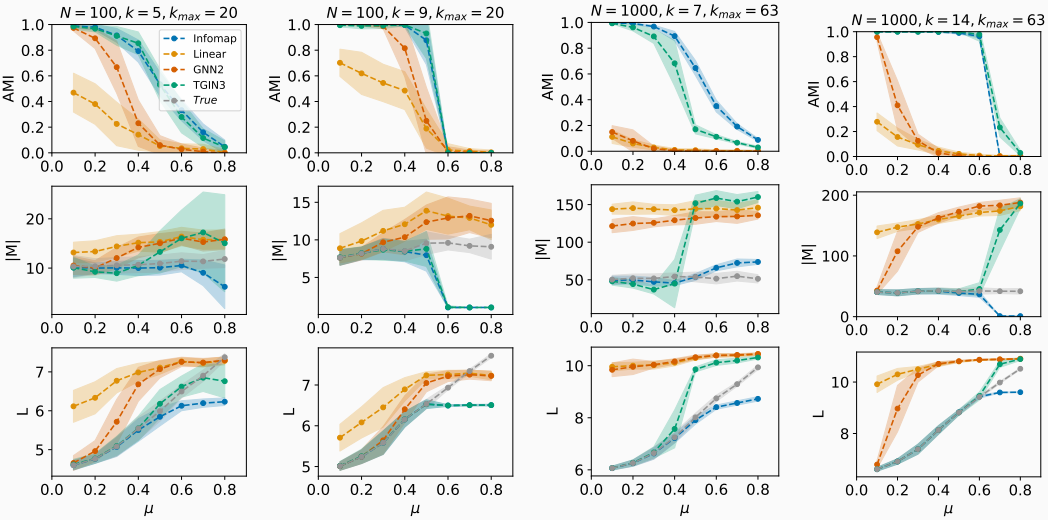
- AMI against ground truth
- Number of modules |M|
- Codelength L

Main points

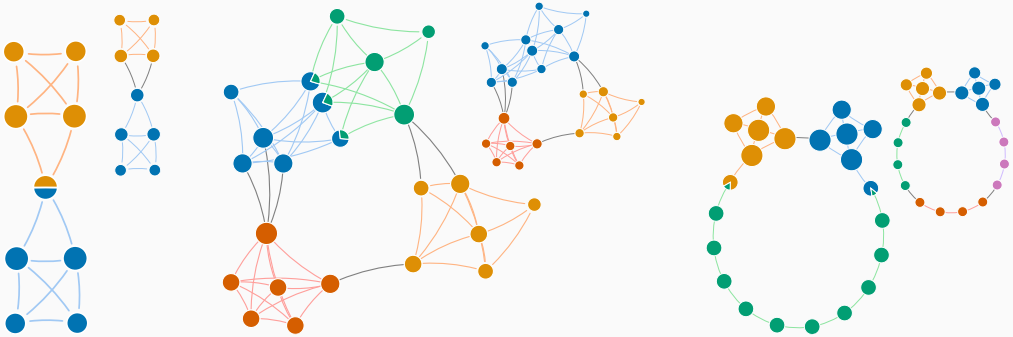
- Similar performance to Infomap, slighly outperforming it (AMI)
- performance $\sim$ (G)NN expressivity $\rightarrow$ over/underfitting (|M|)
- Infomap achieves lowest L
- consider all three measures!
  $\rightarrow$ similar AMI but different |M|
  $\rightarrow$ Smaller L always better?



Results – directed



Results



Conclusion

Conclusion

Motivation

Leverage advances from deep learning for community detection (GPUs!)

Our Approach

We make the map equation differentiable by introducing soft cluster assignments, and optimise it with (G)NNs and gradient descent.

Results

- Community-detection performance, overfitting, and underfitting depend on the chosen (G)NN architecture
- We naturally detect overlapping communities
- Benefit: enables fast experimentation with map equation adaptions!

Thank you for your attention!

Check out our preprint: arXiv:2310.01144

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

ML 4
MACHINE LEARNING FOR
COMPLEX NETWORKS
NETS

Universität
Zürich[UZH]