Christian Toth, BSc

Synthesizing Infomap

A Kullback-Leibler Divergence-Based Approach To Community Detection

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to
Graz University of Technology

Supervisor
Dr. Bernhard C. Geiger

Assessor
Univ.-Prof. Franz Pernkopf

Signal Processing and Speech Communication Laboratory

Graz, April 2020

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit, which we used for this thesis, is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Acknowledgments

I cherish the next few lines not only as an opportunity to express my gratitude, but also as a chance to pause and reflect on the road that led me to writing them.

For being born in Austria and being granted limitless opportunities in life, I feel *privileged*. For there is not always just one person smarter than you, but because there are many of them, I feel a strong sense of *humility*. For I have ever more questions despite intense studying, and for there is a vast world that I have seen so little of, I feel *curious*.

To the people I was allowed to meet along the way, I feel *indebted*. Without our many shared experiences, my life would have been dull and growth impossible. Since an exhaustive list of friends, mentors and family would fill too many pages, I want to thank a few selected people representative of the whole. Bernhard Geiger, for his boundless patience and kindness, for giving me a lot of freedom to explore things on my own, and for his guidance and support in supervising my thesis. Franz Pernkopf, for his willingness to assess and co-supervise this thesis without hesitation, as well as for supporting me in formal matters. Christian Knoll, for providing helpful advice and feedback on intermediate results and writing.

My stepmother Petra, for treating me like her her own child and taking such good care of me. My father Franz, because although we often disagree I can rely on him when thunderclouds hang heavy on the horizon. My deceased mother Anita, because her love and care can never exhaust. I feel endlessly *grateful*.

For the end of my current path is finally in sight, I feel *happy*.

Co-Authorship Note

The original idea for synthesizing Infomap as well as a complete derivation of the synthesizing Infomap objective, both serving as the basis of this thesis, have been provided by Bernhard C. Geiger. Given his passion for the subject, he contributed Appendix B. Furthermore, he provided parts of Section 2.3, Section 3.2, as well as Section C.1 and Section C.3 in Appendix C.

Abstract

Community detection is an essential tool for analyzing the organization of complex social, biological and information networks. Among the numerous community detection algorithms proposed so far, Infomap is a prominent and well-established framework. In this thesis, we propose a novel method for community detection inspired by Infomap. Whereas Infomap approaches community detection analytically by minimizing the average description length of a random walk on a network, our method minimizes dissimilarity, measured using Kullback-Leibler divergence, between a graph-induced and a synthetic random walker to arrive at a partition into communities. Hence, we call our method synthesizing Infomap. Specifically, we focus on community detection in undirected networks with non-overlapping communities and two-level hierarchies.

In this work, we provide a formalization and a rigorous derivation of the synthesizing Infomap objective function. By applying synthesizing Infomap on a set of toy graphs we explore its properties and qualitative behavior. Our experiments on artificially generated benchmark networks show that synthesizing Infomap outperforms original Infomap in terms of adjusted mutual information on networks with weak community structure. Both methods perform equally well on a selection of real-world networks, indicating that synthesizing Infomap produces reasonable results on real-world networks as well. The promising results of synthesizing Infomap encourage further evaluation on real-world networks, as well as extensions to multilevel hierarchies and overlapping community structures.

Contents

List of Figures

List of Tables

1. Introduction

Large complex systems are ubiquitous in our everyday environment, be they of social, biological or technological kind. Such systems can often be represented as networks [1, 2], where the actors of the system are described as nodes and the relationships between the actors are described as links between their respective nodes. Commonly, there exist groups of nodes, having strong relationships within the group and weak relationships to nodes outside of their group. Such groups are referred to as communities or modules [3, 4]. To know about the community structure of a complex network can greatly aid in comprehending its overall constitution as well as learning about their dynamics. Hence, numerous community detection algorithms have been proposed [1, 2, 5], aiming to extract the community structure of complex systems given their network representation.

A prominent method for community detection is the Infomap algorithm presented in [6, 7]. The authors exploit the duality between the community detection and a compression problem, to arrive at a compact and efficient cost function known as the map equation. The map equation embedded within a potent search algorithm lies at the core of the Infomap software package [8]. Originally, Infomap was designed to detect non-overlapping communities in a two-layer hierarchy. Many extensions have been developed in follow up research, e.g. for the detection of hierarchical community structures [9] or overlapping communities [10]. Further study [11] on the map equation revealed superior performance w.r.t the smallest detectable communities ("resolution limit") when compared to modularity-based methods (e.g. [12]). In contrast, in an analysis of the largest detectable communities the authors of [13] showed, that Infomap exhibits a "field-of-view limitation", potentially over-partitioning certain networks. However, this issue was conveniently solved by adapting Infomap to reflect also higher-order

Markov dynamics as proposed in [13, 14].

In a comparative study, the authors of [5] have shown that, albeit Infomap uncovers communities reliably when they are strongly connected internally, it fails to do so for weakly linked communities. Arguably the identification of strong communities in systems is of greater interest if one wants to learn about the mechanics of a system. Yet, densely connected communities may appear weak due to missing links in collected network data or a lack of information only allows for a sparse network representation.

To address this issue, in this thesis we propose a novel method for community detection, the synthesizing Infomap problem. The aim of this thesis is to conduct an initial investigation of the capabilities of synthesizing Infomap as a community detection method. Therefore, we provide a thorough derivation of synthesizing Infomap and explore its optimization properties and qualitative behavior by considering illustrative toy examples. Furthermore, we integrated the synthesizing Infomap objective into the existing Infomap framework in order to evaluate the method in experiments on artificially generated benchmark graphs, as well as on a set of real-world networks. Although we use Infomap as a performance baseline, our method is not supposed to be a substitute to the established Infomap method, but a useful supplement to its functionality. Moreover, in this work we focus on community detection in two-layer hierarchies with non-overlapping communities only.

The results of our experiments on generated benchmark graphs indicate, that synthesizing Infomap indeed outperforms its inspiring method on networks with weak community structure, and it performs on par with Infomap when strong community structures are present. In addition, a comparison on real-world networks shows that synthesizing Infomap delivers reasonable results, as it performs on par with Infomap. Overall, our evaluation shows promising results, calling for further study and experimentation. To this end, we propose several potential directions of future work around synthesizing Infomap.

The remainder of this thesis is structured as follows. In Chapter 2 we introduce the relevant notation and basic concepts used in the course of

this work. In Chapter 3 we present the synthesizing Infomap approach to community detection and derive its objective function. Moreover, we explore the optimization properties of the synthesizing Infomap objective by giving illustrative examples. In Chapter 4 we document the methods used in our experimental setup. Chapter 5 reports and discusses the results of our experiments. Finally, in Chapter 6 we summarize the main part of this thesis and give an outlook of potential future work.

Additionally, we provide three appendices, that supplement the main part of this thesis. In Appendix A we document the integration of the synthesizing Infomap objective into the Infomap software package. We connect synthesizing Infomap to related work in Appendix B. Lastly, in Appendix C we provide the proofs that we withheld in the main part.

2. Preliminaries

In this chapter we introduce concepts and notation that serve as a basis for the remainder of this thesis. We cover the notation for graphs, clusterings, Markov chains and information theory in Section 2.1. Section 2.2 clarifies the definition of communities and in Section 2.3 we compactly describe the intuition underlying Infomap and the map equation.

2.1. Notation

2.1.1. Graphs and Graph Properties

Let $\mathcal{G} := (\mathcal{X}, E, W)$ be a weighted, directed graph with vertices $\mathcal{X} = \{1, \ldots, N\}$, edges $E \subseteq \mathcal{X}^2$ and weight matrix W. The weight matrix is given by $W := [w_{\alpha\to\beta}]_{\alpha,\beta\in\mathcal{X}}$ where $w_{\alpha\to\beta} \geq 0$ denotes the weight of the edge $(\alpha, \beta) \in E$ starting at vertex α and pointing at vertex β. The adjacency matrix $A := [a_{\alpha\to\beta}]_{\alpha,\beta\in\mathcal{X}}$ of $\mathcal{G}$ is defined by $a_{\alpha\to\beta} = \mathbb{I}_E((\alpha, \beta))$, where $\mathbb{I}_E(\cdot)$ is the indicator function

$$\mathbb{I}_E((\alpha, \beta)) = \begin{cases} 1 & \text{if} \quad (\alpha, \beta) \in E \\ 0 & \text{if} \quad (\alpha, \beta) \notin E. \end{cases} \tag{2.1}$$

For an unweighted graph the weight matrix equals the adjacency matrix of the graph, i.e. $W = A$. If a graph is undirected, then it holds that $A = A^\mathsf{T}$, where $(\cdot)^\mathsf{T}$ denotes matrix transposition. Hence, if $(\alpha, \beta) \in E$ then also

$(\beta, \alpha) \in E$. A set $C \subseteq \mathcal{X}$ is a clique if it is a complete subgraph of $\mathcal{G}$, i.e. its edgeset is given as $E = \{(\alpha, \beta) | \alpha \neq \beta\}$ where $\alpha, \beta \in C$. We say it is a maximal clique if it is not a subset of a larger clique. We call $S \subseteq \mathcal{X}$ an independent set if $(S \times S) \cap E = \{\}$ and we call it a maximal independent set if it is not a subset of a larger independent set. Lastly, we say that an undirected graph $\mathcal{G}$ is strongly connected if every possible pair of vertices in $\mathcal{G}$ are connected by a path. Specifically, for every pair (α, β) of vertices, there exists an ordered set of intermediate vertices $(\gamma_1, \ldots, \gamma_k)$ such that (α, γ_1), (γ_i, γ_{i+1}) for $i \in \{1, \ldots, k-1\}$, and (γ_k, β) are in E. In the course of this work we will use the terms graph and network, vertex and node, as well as edge and link interchangeably.

2.1.2. Clusterings

Consider a *clustering* $\mathcal{Y}$ of $\mathcal{X}$ into a set of M nonempty elements $\mathcal{Y} := \{\mathcal{Y}_1, \ldots, \mathcal{Y}_M\}$, where $\bigcup_{i=1}^{M} \mathcal{Y}_i = \mathcal{X}$. We refer to the elements of such a clustering as *modules* or *communities*. Furthermore, if the modules are disjoint, i.e. $\mathcal{Y}_i \cap \mathcal{Y}_j = \{\}$ for all $i \neq j$ where $i, j \in \{1, \ldots, M\}$, then we call the modules *non-overlapping* and the clustering a *partition*. For the remainder of this thesis we assume that all clusterings are partitions, unless otherwise stated.

We call a partition $\mathcal{Y}^\bullet = \{\mathcal{Y}_1^\bullet, \ldots, \mathcal{Y}_M^\bullet\}$ a *coarsening* of another partition $\mathcal{Y} = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_K\}$ if each module in the coarsening $\mathcal{Y}_i^\bullet \in \mathcal{Y}^\bullet$ is comprised of one or more modules of the original partition $\mathcal{Y}$ and $M < K$.

Importantly, we indicate module indices by roman letters $i, j, \ldots$, contrasting them from Greek letters $\alpha, \beta, \ldots$ indicating the vertices of $\mathcal{G}$. Additionally, we define a mapping function m: $\mathcal{X} \to \mathcal{Y}$, uniquely mapping each vertex of $\mathcal{G}$ to the index of its containing module, i.e.

$$\forall \alpha \in \mathcal{X}: \quad m(\alpha) = i \quad \Longleftrightarrow \quad \alpha \in \mathcal{Y}_i, \quad \mathcal{Y}_i \in \mathcal{Y}. \tag{2.2}$$

Abusing notation, we write $\alpha \in i$ instead of $\alpha \in \mathcal{Y}_i$ and $i \in \mathcal{Y}$ instead of $\mathcal{Y}_i \in \mathcal{Y}$.

2.1.3. Markov Chains

We consider random walks on the graph $\mathcal{G}$. Specifically, let $\{X_t\}$ where $t \in \mathbb{N}$ be a first-order homogeneous Markov chain with alphabet $\mathcal{X}$. We assume that its stationary transition probability matrix $P := [p_{\alpha\to\beta}]_{\alpha,\beta\in\mathcal{X}}$ is derived from the graph's weight matrix W where its elements are given as

$$\forall t \in \mathbb{N}, \forall \alpha, \beta \in \mathcal{X}: \quad p_{\alpha\to\beta} := \mathbb{P}(X_{t+1} = \beta | X_t = \alpha) = const. \tag{2.3}$$

Then, for a state distribution $p := [p_\alpha]_{\alpha\in\mathcal{X}}$ where $p_\alpha := \mathbb{P}(X_t = \alpha)$, it follows that

$$p_\beta = \mathbb{P}(X_{t+1} = \beta) = \sum_\alpha p_\alpha p_{\alpha\to\beta}. \tag{2.4}$$

If it further holds that the state distribution at a given time step is the same as in the previous time step, i.e. if $p = P \cdot p$, then we call p a stationary distribution.

Setting $Y_t := m(X_t)$ defines a stationary process $\{Y_t\}$ on the modules. Specifically, the marginal and joint probabilities describing $\{Y_t\}$ are obtained as

$$p_i := \mathbb{P}(Y_t = i) = \sum_{\alpha\in i} p_\alpha, \quad i \in \{1, \dots, M\} \tag{2.5a}$$

and

$$p_{i,j} := \mathbb{P}(Y_{t+1} = j, Y_t = i) = \sum_{\alpha\in i}\sum_{\beta\in j} p_\alpha p_{\alpha\to\beta}, \quad i, j \in \{1, \dots, M\}. \tag{2.5b}$$

From the product rule of probability it follows that

$$p_{i\to j} := \mathbb{P}(Y_{t+1} = j | Y_t = i) = \frac{p_{i,j}}{p_i} \quad \text{if} \quad p_i > 0, \quad i, j \in \{1, \dots, M\}. \tag{2.5c}$$

We further abbreviate

$$p_{\neg i} := 1 - p_i = \mathbb{P}(Y_t \neq i), \tag{2.6a}$$

$$p_{i\not\to j} := 1 - p_{i\to j} = \mathbb{P}(Y_{t+1} \neq j | Y_t = i) \tag{2.6b}$$

and

$$p_{i,\neg j} := p_i - p_{i,j} = \mathbb{P}(Y_{t+1} \neq j, Y_t = i). \tag{2.6c}$$

2.1.4. Information Theory

In the course of our work we make use of the following concepts from the field of information theory that are well-described in [15]. Let Y, Z denote random variables (RV), then we call $H(Z)$ the entropy of Z, $H(Y|Z)$ the conditional entropy of Y given Z and $I(Y;Z)$ the mutual information between Y and Z. Furthermore, let p and q denote discrete probability distributions over the same alphabet. Then we call $D(p\|q)$ the Kullback-Leibler divergence between p and q. If p, q are Bernoulli distributions i.e., $p = [p_1, 1-p_1]$ and $q = [q_1, 1-q_1]$, then we abbreviate

$$D(p_1\|q_1) := p_1 \log \frac{p_1}{q_1} + (1-p_1) \log \frac{(1-p_1)}{(1-q_1)}. \tag{2.7}$$

Let $\overline{H}(\{Z_t\})$ denote the entropy rate of the stationary stochastic process $\{Z_t\}$. If $\{Z_t\}$ is an irreducible, stationary Markov chain with alphabet $\mathcal{Z}$, transition probability matrix P and invariant distribution $[p_\alpha]_{\alpha\in\mathcal{Z}}$, then its entropy rate is

$$\overline{H}(\{Z_t\}) := -\sum_{\alpha\in\mathcal{Z}} \sum_{\beta\in\mathcal{Z}} p_\alpha p_{\alpha\to\beta} \log p_{\alpha\to\beta} \tag{2.8}$$

where we define $0 \log 0 := 0$ by continuous extension of the logarithm. Finally, let $Q := [q_{\alpha\to\beta}]_{\alpha,\beta\in\mathcal{Z}}$ be another transition probability matrix of the same size as P. The Kullback-Leibler divergence rate $\overline{D}(\cdot\|\cdot)$ between two stationary Markov chains governed by P and Q is

$$\overline{D}(P\|Q) := \sum_{\alpha\in\mathcal{Z}} \sum_{\beta\in\mathcal{Z}} p_\alpha p_{\alpha\to\beta} \log \frac{p_{\alpha\to\beta}}{q_{\alpha\to\beta}} \tag{2.9}$$

given that the Markov chains are irreducible [16, Th. 1].

2.2. Defining Communities

In order to approach the problem of community detection and to evaluate the results of any detection method, a more precise definition of a com-

munity is very helpful, if not necessary. Several definitions of communities have been proposed in the literature. The authors of a recent survey [2] give a thorough overview of the evolution of community definitions. Here, we give a compact summary, introducing the necessary concepts appearing in the rest of this thesis.

In a classical view, communities are densely connected subgraphs in a network that are well separated. A primal example fitting to this intuition is a clique, that comes, however, with several implications. First, communities seldom appear as cliques and secondly, connections to other subgraphs are not taken into account. Thirdly, in a clique every node has the same degree and hence appears to be equally important, which also does not reflect the structure of real-world networks satisfactorily (e.g. in (online) social networks). Yet another complication arises when we consider overlapping communities, as communities may not be well separated at all.

An evolved approach to community definition takes the internal and external node degrees w.r.t a community into account. As such, the mixing parameter [17, 18] of a node is the ratio of the number of links it shares with nodes outside of its community to its total number of links. It is a key tuning parameter [19] for the LFR benchmark and a measure for the existence and strength of communities [18]. Assume a graph $\mathcal{G} = (\mathcal{X}, E)$ with community structure $\mathcal{Y}$. The mixing parameter of a node $\alpha \in \mathcal{Y}_i$ is given as

$$\mu(\alpha) = \frac{k_\alpha^{ext}}{k_\alpha^{int} + k_\alpha^{ext}} \tag{2.10}$$

where k_α^{int} is the internal degree and k_α^{ext} is the external degree of α with respect to its community $\mathcal{Y}_i$. Assuming a common global mixing parameter μ for all nodes in $\mathcal{G}$, clearly, the average mixing parameter across any subset of nodes is again μ. Thus, for any community $\mathcal{Y}_i$ it holds that

$$\mu = \frac{k_\beta^{ext}}{k_\beta^{int} + k_\beta^{ext}}, \quad \forall \beta \in \mathcal{Y}_i \tag{2.11}$$

$$= \frac{1}{|\mathcal{Y}_i|} \sum_{\alpha \in \mathcal{Y}_i} \frac{k_\alpha^{ext}}{k_\alpha^{int} + k_\alpha^{ext}} \tag{2.12}$$

$$= \frac{\sum_{\alpha \in \mathcal{Y}_i} k_\alpha^{ext}}{\sum_{\alpha \in \mathcal{Y}_i} k_\alpha^{int} + k_\alpha^{ext}}. \tag{2.13}$$

Strong communities [4] satisfy $\mu > 0.5$, i.e. each node has more connections to nodes inside its community than to external nodes. According to [18], weaker communities are still well defined for $\mu < \frac{N - N_c^{max}}{N}$, where N is the number of nodes in the network and N_c^{max} is the size of the largest community.

A more modern view on community definition employs a probabilistic description. There, for each node within a strong community its linking probability to every other node in the same community is higher than the linking probability to any external node. For weak communities this constraint is relaxed such that only the average internal edge probability of a node has to be greater than the average external edge probability. However, also this probabilistic view does not come without practical issues, as it is not straight forward to compute the necessary linking probabilities. Thus, usually one assumes an underlying probabilistic model describing the network formation process. An example of such a model is the popular stochastic block model.

Whereas the above definitions mainly focus on the topology of a network, alternative perspectives also take dynamic processes occurring on the network into account. Random walks provide a simple proxy for diffusion processes describing the dynamics of a network. Here, the notion of a community is related to the average time a random walker spends within subgraph of a network. In the next section we will see that Infomap is conceptually based on this idea.

2.3. Infomap and the Map Equation

The map equation, as presented in [7], was proposed as a cost function for community detection. Suppose we have a graph $\mathcal{G}$ and an arbitrary partition $\mathcal{Y}$ of its vertex set $\mathcal{X}$ into M modules $\mathcal{Y} = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_M\}$. Additionally, suppose we have a random walker traversing the edges of the graph according to some transition probability matrix P derived from its weight matrix W. We can then describe the path of the random walker on the graph using a two-level coding scheme. Firstly, we have an *index codebook* $\mathcal{Q}$ with one codeword for each module. Secondly, each module $\mathcal{Y}_i$ has its own *module codebook* $\mathcal{P}^i$ with one codeword for each vertex in the module and an additional *exit codeword*. We can then encode the movements of the random walker by considering the following two cases.

1. If the random walker moves from some vertex α to some vertex β within the *same module* $\mathcal{Y}_i$, then its movement is encoded using the codeword for vertex β from the module codebook $\mathcal{P}^i$.
2. If the random walker moves from some vertex α in module $\mathcal{Y}_i$ to some vertex β in a *different module* $\mathcal{Y}_j$, then the exit codeword from module codebook $\mathcal{P}^i$ is used together with the codeword for module $\mathcal{Y}_j$ from the index codebook $\mathcal{Q}$ and the codeword for vertex β from the module codebook $\mathcal{P}^j$.

The key idea behind the map equation assumes that by finding an efficient coding scheme (and hence a partition into modules) one can compactly describe the community structure of a network. Since intuitively a shorter description length of the random walk should correspond to a better, more compact representation, we measure the efficiency of such a coding scheme by considering the average description length of a single step of the random walker. The map equation thus provides us with a lower bound on this average description length as follows.

Shannon's source coding theorem [20] states that the average codeword length for each of our codebooks is lower bounded by their entropy, given

their usage probabilities. Specifically, the codebook entropies are given as

$$H(\mathcal{Q}) = -\sum_{i \in \mathcal{Y}} \frac{p_{i,\neg i}}{\sum_{i \in \mathcal{Y}} p_{i,\neg i}} \log \frac{p_{i,\neg i}}{\sum_{i \in \mathcal{Y}} p_{i,\neg i}} \tag{2.14}$$

and

$$H(\mathcal{P}^i) = -\frac{p_{i,\neg i}}{p_{i,\neg i} + p_i} \log \frac{p_{i,\neg i}}{p_{i,\neg i} + p_i} - \sum_{\alpha \in i} \frac{p_\alpha}{p_{i,\neg i} + p_i} \log \frac{p_\alpha}{p_{i,\neg i} + p_i}, \tag{2.15}$$

according to the probabilities (cf. Equations (2.5)) governing the random walker. The index codebook is used with a rate $\sum_{i \in \mathcal{Y}} p_{i,\neg i}$, i.e. the cumulative exit rates of all modules. In contrast, any module codebook $\mathcal{P}^i$ is used at a rate $p_{i,\neg i} + p_i$, which reflects the exit rate of the module plus the time we spend in it. Hence, the map equation

$$\mathcal{L}(\mathcal{Y}) := \sum_{i \in \mathcal{Y}} p_{i,\neg i} H(\mathcal{Q}) + \sum_{i \in \mathcal{Y}} (p_i + p_{i,\neg i}) H(\mathcal{P}^i) \tag{2.16}$$

gives a lower bound of the *expected* codeword length required to encode a single step of the random walker. As a consequence, the Infomap community detection method proposes to minimize the map equation (2.16) over all possible partitions $\mathcal{Y}$. A minimizer $\mathcal{Y}^* \in \arg\min_{\mathcal{Y}} \mathcal{L}(\mathcal{Y})$ is supposed to represent the community structure of the graph very well.

In conclusion, we can say that the map equation *analytically* describes a networks' dynamics by deriving a lower bound on the average description length of a random walker for a predefined coding scheme. In contrast, we will explore a synthetic approach to describing the network dynamics in the next chapter.

3. A Synthesis-based Map Equation

In this chapter we outline a synthetic approach to arrive at an alternative map equation, contrasting the analytical minimum description length idea underlying the map equation in Section 3.1. Furthermore, we formalize our model into an optimization problem and derive a novel utility function, the synthesizing Infomap objective, which we investigate with respect to its applicability to community detection tasks in the remainder of this thesis. In Section 3.2 we explore fundamental properties of the optimization landscape of synthesizing Infomap and provide illustrative examples.

3.1. Derivation of Synthesizing Infomap

Assume a graph $\mathcal{G} = (\mathcal{X}, E, W)$ and an inherent partition $\mathcal{Y}_{true}$ of its nodes into modules. Consider further a random walker moving on $\mathcal{G}$ governed by the transition probability matrix P, which is derived from the weight matrix W. We refer to this random walker as the *graph-induced* random walker, as its movements depend on the topology of $\mathcal{G}$ (i.e. implicitly its community structure). In the next step we design a *synthetic* random walker, governed by some transition probability matrix $Q^{\mathcal{Y}}$ which, in contrast to P, explicitly depends on some arbitrary partition $\mathcal{Y}$. The fundamental idea behind our approach is then to find a partition $\mathcal{Y}$ such that the graph-induced and the synthetic random walker behave as similarly as possible (regarding their stochastic behavior). Intuitively, such a partition will resemble the intrinsic partition $\mathcal{Y}_{true}$ very closely. We formalize this concept in the following.

At any given time step, our synthetic random walker is visiting some node $\alpha \in i$ where $i \in \mathcal{Y}$. We decide whether to leave or to stay in the current module i in the next time step, based on a module-specific Bernoulli distribution $[s_i, 1 - s_i]$. In case of a module change, we choose a new module $j \neq i$ according to a distribution over modules $[u_i]_{i \in \mathcal{Y}}$. Indifferent to whether we stay in or leave the current module, we choose the next visited node β lying in the new module j by a module-specific distribution over nodes $[r_\beta^j]_{\beta \in j}$ (note that $j = i$ if we stay in the current module). We can now assemble the transition probability matrix $Q^{\mathcal{Y}} = [q_{\alpha \to \beta}]_{\alpha, \beta \in \mathcal{X}}$ where

$$q_{\alpha \to \beta} = \begin{cases} r_\beta^{m(\beta)} \quad (1 - s_{m(\alpha)}) & \text{for} \quad m(\alpha) = m(\beta), \\ r_\beta^{m(\beta)} \quad s_{m(\alpha)} \quad \frac{u_{m(\beta)}}{1 - u_{m(\alpha)}} & \text{for} \quad m(\alpha) \neq m(\beta). \end{cases} \tag{3.1}$$

Note that when changing modules we have to normalize the distribution over modules by $1 - u_{m(\alpha)} = \sum_{k \neq m(\alpha)} u_k$ since we exclude the current module $m(\alpha)$ as a choice.

We quantify the similarity between the graph-induced and the synthetic random walker by utilizing the Kullback-Leibler divergence rate $\overline{D}(P \| Q^{\mathcal{Y}})$, i.e. the lower $\overline{D}(P \| Q^{\mathcal{Y}})$, the more similar are P and $Q^{\mathcal{Y}}$. Hence, we minimize $\overline{D}(P \| Q^{\mathcal{Y}})$ with respect to the distribution parameters of $Q^{\mathcal{Y}}$ on the one hand, and regarding the partition $\mathcal{Y}$ on the other hand. This leads to the following optimization problem.

Problem 1. Let $\mathcal{G} = (\mathcal{X}, E, W)$ be a graph and let $P \sim W$ be a transition probability matrix derived from the weight matrix W. Moreover, let $Q^{\mathcal{Y}}$ denote a transition probability matrix dependent on partition $\mathcal{Y}$ as described in (3.1). We then want to find an optimal distribution $\mathcal{Y}^*$, subject to [a]

$$\mathcal{Y}^* \in \arg\min_{\mathcal{Y}} \left[\min_{\{[r_\alpha^i]_{\alpha \in i},\, s_i,\, u_i\}_{i \in \mathcal{Y}}} \overline{D}(P \| Q^{\mathcal{Y}}) \right]. \tag{3.2}$$

[a] In general, multiple optimal partitions w.r.t the inner minimization problem may

exist, hence the set notation.

Solving the inner optimization problem with respect to the parameters $\{[r^i_\beta]_{\beta\in i},\ s_i\}_{i\in\mathcal{Y}}$ yields (see Appendix C.1)

$$r^i_\alpha = \frac{p_\alpha}{p_i} = \mathbb{P}(X_t = \alpha, Y_t = i) \tag{3.3}$$

$$s_i = p_{i\not\to i} = \mathbb{P}(Y_{t+1} \neq i | Y_t = i). \tag{3.4}$$

Regarding the distribution over modules $[u_i]_{i\in\mathcal{Y}}$ there exists no closed-form solution to the best of our knowledge. Nevertheless, by choosing

$$u_i = p_i = \mathbb{P}(Y_t = i) \tag{3.5}$$

as a sub-optimal solution we can relax (see Appendix C.1) Problem 1 to arrive at the following.

Problem 2 (Synthesizing Infomap)**.** Let $\mathcal{G} = (\mathcal{X}, E, W)$ be a graph and let $P \sim W$ be a transition probability matrix derived from the weight matrix W. Moreover, let $Q^{\mathcal{Y}}$ denote a transition probability matrix dependent on some partition $\mathcal{Y}$ as described in (3.1). The *synthesizing Infomap* problem is concerned with finding an optimal partition $\mathcal{Y}^*$ of $\mathcal{X}$ into modules, subject to

$$\mathcal{Y}^* \in \arg\max_{\mathcal{Y}} \sum_{i\in\mathcal{Y}} p_i D(p_{i\to i} \| p_i) \tag{3.6}$$

where p_i and $p_{i\to i}$ are defined as in (2.5).

Hence, we define the *synthesizing Infomap objective* for a given partition $\mathcal{Y}$ as

$$\mathcal{J}(\mathcal{Y}) := \sum_{i\in\mathcal{Y}} p_i D(p_{i\to i} \| p_i). \tag{3.7}$$

3.2. Bounds and Characterization of the Optimization Landscape

In Appendix C.1 we show that fundamental bounds of the synthesizing Infomap objective $\mathcal{J}(\mathcal{Y})$ are given by (cf. Corollary 1)

$$0 \leq \mathcal{J}(\mathcal{Y}) \leq I(X_t; X_{t-1}). \tag{3.8}$$

In the following we construct examples where the synthesizing Infomap objective achieves these bounds for particular graphs and partitions in Section 3.2.1 and Section 3.2.2 respectively. In Section 3.2.3 we will explore the potential existence of local optima for partitions into maximal independent sets.

3.2.1. Achieving the lower bound

Example 1. Let $\mathcal{G}$ be an arbitrary graph and let $\mathcal{Y} = \{\mathcal{Y}_1\} = \{\mathcal{X}\}$, i.e. all vertices of $\mathcal{G}$ are placed in a single module. Then, $p_1 = 1$ and $p_{1\to 1} = 1$, from which we obtain $\mathcal{J}(\mathcal{Y}) = 0$.

Example 1 indicates, that the lower bound $\mathcal{J}(\mathcal{Y}) = 0$ is tight, as it can be achieved on every possible graph. More importantly, Example 1 indicates that a single module will never be a solution of the Problem 2, at least not in non-trivial cases.

3.2.2. Achieving the upper bound

In the following examples we achieve the upper bound $\mathcal{J}(\mathcal{Y}) = I(X_t; X_{t-1})$ for selected graphs and partitions. Anyways, there is no guarantee that the upper bound is achievable by any possible partition of an arbitrary given graph.

Example 2. Let $\mathcal{G} = (\mathcal{X}, E)$ be an unweighted and undirected graph of disconnected cliques, i.e. there exists a partition $\mathcal{Y} = \{\mathcal{Y}_1, \dots, \mathcal{Y}_M\}$ such that the edge set E of $\mathcal{G}$ is given as $E = \bigcup_{i=1}^{M} \mathcal{Y}_i \times \mathcal{Y}_i$. Note that here we include self-loops in the edge set. Furthermore, let $P \sim A$ be the transition probability matrix derived from the graph's adjacency matrix A.

Since the graph is unweighted, the movement within cliques and the invariant distribution within each clique are uniform. The resulting Markov chain is not irreducible and therefore infinitely many invariant distributions exist for the entire alphabet $\mathcal{X}$. Specifically, for every distribution over the modules $[p_i]_{i \in \mathcal{Y}}$ the distribution over vertices $[p_\alpha]_{\alpha \in \mathcal{X}}$ given by $p_\alpha = \frac{p_{m(\alpha)}}{|\mathcal{Y}_{m(\alpha)}|}$ is invariant under P. Thus, we have that

$$H(X_t) = -\sum_{\alpha \in \mathcal{X}} \frac{p_{m(\alpha)}}{|\mathcal{Y}_{m(\alpha)}|} \log \frac{p_{m(\alpha)}}{|\mathcal{Y}_{m(\alpha)}|} = -\sum_{\mathcal{Y}_i \in \mathcal{Y}} p_i \log \frac{p_i}{|\mathcal{Y}_i|}. \tag{3.9}$$

Given that the random walker is currently at vertex α in module i, all vertices in this module (including α) are equally likely to be visited in the next step. Thus, it follows that $H(X_t | X_{t-1} = \alpha) = \log |\mathcal{Y}_{m(\alpha)}|$. Combining this with $H(X_t)$ yields $I(X_t; X_{t-1}) = H(Y_t)$.

Since the modules in $\mathcal{Y}$ represent the cliques in $\mathcal{G}$ and since consequently there are no edges connecting one module with another, we have $p_{i \to i} = 1$ for every module. Thus, it follows that $\mathcal{J}(\mathcal{Y}) = H(Y_t) = I(X_t; X_{t-1})$, i.e., the upper bound of the synthesizing Infomap objective is achieved.

Note that a coarsening $\mathcal{Y}^\bullet = \{\mathcal{Y}_1^\bullet, \dots, \mathcal{Y}_M^\bullet\}$ of the partition $\mathcal{Y}$ does not achieve this optimum. Although for $\mathcal{Y}^\bullet$ being a coarsening of $\mathcal{Y}$ we still have $p_{i \to i} = 1$, for the distribution over modules $[p_i^\bullet]_{i \in \mathcal{Y}^\bullet}$ we get

$$p_i^\bullet = \sum_{\mathcal{Y}_j \in \mathcal{Y}_i^\bullet} p_j \tag{3.10}$$

and thus $H(Y_t^\bullet) < H(Y_t)$.

In Example 2 we show that the utility function $\mathcal{J}(\mathcal{Y})$ achieves its upper bound for a partition $\mathcal{Y}$ that splits the graph $\mathcal{G}$ into sets of disconnected cliques. This suggests that our proposed synthesis-based map equation is a theoretically valid approach to community detection and, at least in this trivial example, has its global optimum at the desired partition.

Example 3. Let $\mathcal{G}$ be a complete graph (unweighted and undirected) and consider a partition $\mathcal{Y} = \{\{\alpha\} | \alpha \in \mathcal{X}\}$ where each vertex of $\mathcal{G}$ is a module. Then for each module $\mathcal{Y}_i$ obviously $p_{i\to i} = 0$ and hence

$$\mathcal{J}(\mathcal{Y}) = -\sum_{i\in\mathcal{Y}} p_i \log(1 - p_i) \tag{3.11}$$

$$= -\sum_{\alpha\in\mathcal{X}} p_\alpha \log(1 - p_\alpha). \tag{3.12}$$

Furthermore, since $\mathcal{G}$ is a complete graph, it follows that $p_\alpha = \frac{1}{N}$ for all $\alpha \in \mathcal{X}$ and therefore

$$\mathcal{J}(\mathcal{Y}) = \log N - \log(N - 1). \tag{3.13}$$

It is straightforward to show that $I(X_t; X_{t-1}) = \log N - \log(N - 1)$ and thus the partition $\mathcal{Y}$ achieves the upper bound of the synthesizing Infomap objective for a complete graph.

Example 3 demonstrates an interesting behavior of synthesizing Infomap. Although a clique can be seen as an exemplary proxy for a community, synthesizing Infomap achieves its minimum objective for such a partition on a single clique (cf. Example 1). Instead, we achieve the maximum objective for a partition into single vertices, which arguably is not a desirable behavior as it over-partitions the given graph. However, we would argue that a partition into a single module (in general) provides little to no information about a given network and hence is an unusable result if provided by

any community detection algorithm. From this perspective, the behavior of synthesizing Infomap in Example 3 does seem sensible in community detection tasks (cf. the experiments in Section 5.2).

Example 4. Let $\mathcal{G} = (\mathcal{X}, E)$ be an unweighted and undirected graph with a star topology where one central node is connected to $N-1$ satellite nodes by a single edge and satellite nodes are not connected with each other. The edge set is given as $E = \{(1, \alpha) | \alpha \in \{2, \dots, N\}\}$ and stationary distribution over nodes $[p_\alpha]_{\alpha \in \mathcal{X}}$ is given by

$$p_\alpha = \begin{cases} \frac{1}{2} & \text{if} \quad \alpha = 1 \\ \frac{1}{2(N-1)} & \text{otherwise.} \end{cases} \tag{3.14}$$

Now consider a partition $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2\}$ into two modules where the central node represents the first module and the satellite nodes represent the second module, i.e. $\mathcal{Y}_1 = \{1\}$ and $\mathcal{Y}_2 = \{\alpha | \alpha \in \{2, \dots, N\}\}$. Clearly, $p1 = p2 = \frac{1}{2}$ and since both partitions are independent sets it follows that $p_{1\to1} = p_{2\to2} = 0$. For this partition the objective attains a value of

$$\mathcal{J}(\mathcal{Y}) = -\sum_{i \in \mathcal{Y}} p_i \log(1 - p_i) = 1. \tag{3.15}$$

It is straightforward to show that for the given graph $I(X_t; X_{t-1}) = 1$ and thus the partition $\mathcal{Y}$ achieves the upper bound of the synthesizing Infomap objective for a complete graph.

The optimal partition in Example 4 can be interpreted as a classification into master and slave nodes e.g. in a computer network. Thus, the partition appears to be a reasonable result for the given graph, although it may not fit an intuitive understanding of a community at a first glance.

3.2.3. Local Optimality of Maximal Independent Sets

Interestingly, the globally optimal solutions in Example 3 and Example 4 are partitions into maximal independent sets. Indeed, partitions into maximal independent sets appear to be local or even global optima of the synthesizing Infomap objective, depending on the graph at hand. Actually, one can show (see Appendix C.2), that for any given partition $\mathcal{Y}$ of a graph into (not exclusively maximal) independent sets, a coarsening $\mathcal{Y}^\bullet$ of $\mathcal{Y}$ into solely maximal independent sets will always achieve a better objective $\mathcal{J}(\mathcal{Y}^\bullet) > \mathcal{J}(\mathcal{Y})$, if $\mathcal{Y}^\bullet$ has at least two modules. Consider further Example 5.

Example 5. In this example we consider a graph of k disconnected cliques $\mathcal{Y} = \{\mathcal{Y}_1, \dots, \mathcal{Y}_k\}$ as in Example 2. This time, however, we assume that the graph does not contain any self-loops. This does not affect the cliques having uniform invariant distributions. We further assume that all cliques have the same size, i.e. we set $|\mathcal{Y}_i| = n_c$ for every $\mathcal{Y}_i \in \mathcal{Y}$. Lastly, we assume an invariant distribution such that $[p_i = \frac{1}{k}]_{i \in \mathcal{Y}}$ and hence $[p_\alpha = \frac{1}{k \cdot n_c}]_{\alpha \in \mathcal{X}}$.

Let $\mathcal{Y}^\bullet$ be a partition of $\mathcal{X}$ into n_c maximal independent sets of size k, i.e. each set $\mathcal{Y}^\bullet_1, \dots, \mathcal{Y}^\bullet_{n_c}$ contains exactly one element of each clique. Therefore, $p^\bullet_i = \frac{1}{n_c}$ and since the graph has no self-loops, it follows immediately that $p^\bullet_{i \to i} = 0$ for every $i \in \mathcal{Y}^\bullet$. Thus, $\mathcal{J}(\mathcal{Y}^\bullet) = \log n_c - \log(n_c - 1)$.

Now we define a partition $\mathcal{Y}^\circ$ that coincides with $\mathcal{Y}^\bullet$ except that one vertex α from module $\mathcal{Y}^\bullet_i$ is moved to module $\mathcal{Y}^\bullet_j$, i.e. $\mathcal{Y}^\circ_i = \mathcal{Y}^\bullet_i \setminus \{\alpha\}$ and $\mathcal{Y}^\circ_j = \mathcal{Y}^\bullet_j \cup \{\alpha\}$. This changes the marginal distributions of both modules to $p^\circ_i = \frac{k-1}{k \cdot n_c}$ and $p^\circ_j = \frac{k+1}{k \cdot n_c}$. Module $\mathcal{Y}^\circ_i$ remains an independent set and thus $p^\circ_{i \to i} = 0$. However, module $\mathcal{Y}^\circ_j$ now has one internal connection to some node $\beta \in \mathcal{Y}^\circ_j$ where $p_{\alpha \to \beta} = p_{\beta \to \alpha} = \frac{1}{n_c - 1}$. It

follows that

$$p^{\circ}_{j\to j} = \frac{2}{(n_c - 1)(k+1)}. \tag{3.16}$$

A numerical evaluation (see Figure 3.1) indicates that for every possible clique size and every possible number of cliques it holds, that $\mathcal{J}(\mathcal{Y}^{\bullet}) \geq \mathcal{J}(\mathcal{Y}^{\circ})$. Thus, the independent set is a local optimum of Problem 2 for the considered graph $\mathcal{G}$.

The existence of local optima in the form of maximal independent sets has implications for the choice and parameterization of the optimization algorithm. On the one hand, they can be a potential nuisance when one wants to detect densely connected communities. However, the Infomap search algorithm and hence our synthesizing Infomap implementation do not identify independent sets *by design* (cf. Appendix A), as the search routine merges only adjacent nodes/modules, thus violating the independent set condition. The only exception is the initial partition of the algorithm, where each node serves as a module as in Example 3. To avoid getting stuck in this initialization in ill-posed scenarios (cf. Section 5.1.1) we propose an alternative initialization in Section 4.3.

On the other hand, a partition into maximal independent sets might be of interest for example in multipartite graphs (cf. Example 6). In such graphs, nodes of the same type have no mutual connections at all, although they form a natural functional unit of interest. It is therefore remarkable, that synthesizing Infomap can potentially detect densely connected communities as well as independent sets. In spite of this fact, our synthesizing Infomap implementation does not yet provide a suitable search algorithm for independent sets. Hence, we will not investigate such cases extensively in the course of this thesis.

Example 6. Consider a bipartite graph $\mathcal{G}$ with $N = N_1 + N_2$ nodes in two groups of nodes $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2\}$, where N_1 nodes are in group $\mathcal{Y}_1$ and N_2 nodes are in group $\mathcal{Y}_2$. Every node from group $\mathcal{Y}_1$ is connected

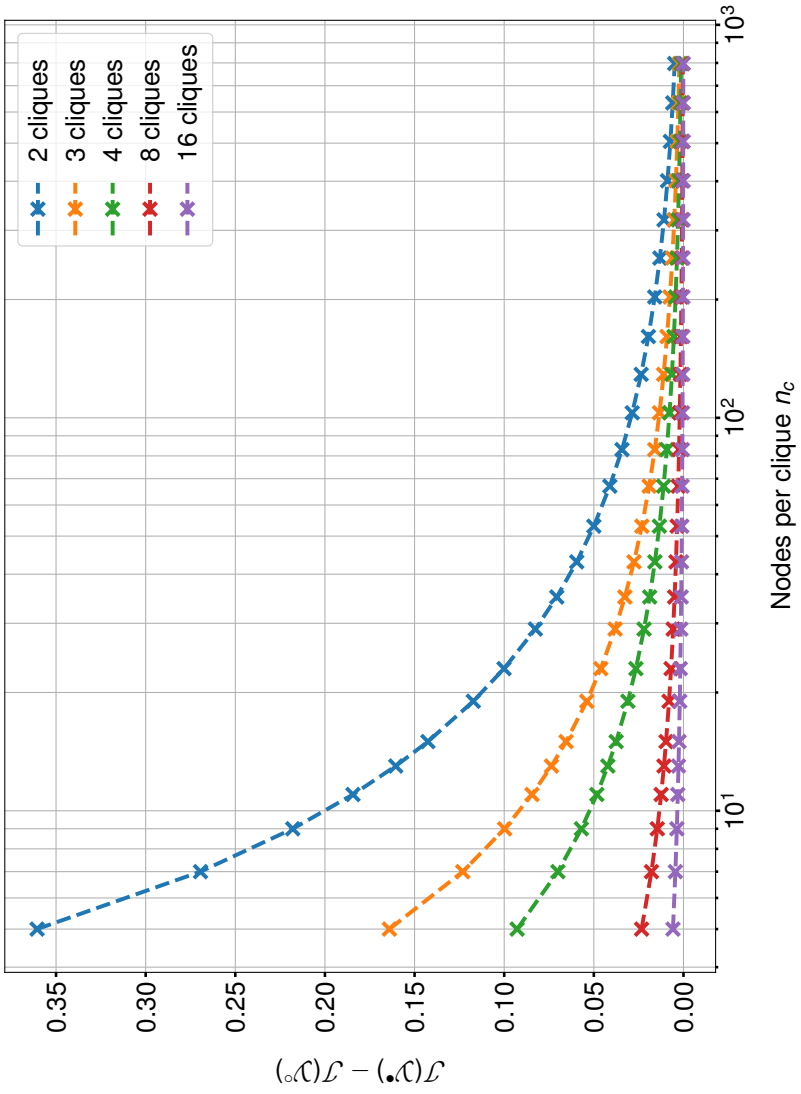


Figure 3.1.: Difference of the synthesizing Infomap objective for a partition into maximal independent sets $\mathcal{J}(\mathcal{Y}^\bullet)$ and a modified partition $\mathcal{J}(\mathcal{Y}^\circ)$ as defined in Example 5. We plot the difference as a function of clique size n_c for selected numbers of cliques k. As can be seen, the partition into independent sets always yields a higher objective. The difference vanishes for increasing clique size, independent of the number of cliques.

to every node from group $\mathcal{Y}_2$, however the groups have no internal connections, i.e. the edge set is given as $E = \{(\alpha, \beta) | \alpha \in \mathcal{Y}_1, \beta \in \mathcal{Y}_2\}$. Clearly, $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are maximal independent sets and $p_1 = p_2 = \frac{1}{2}$. Hence, the synthesizing Infomap objective for the given partition yields

$$\mathcal{J}(\mathcal{Y}) = -\sum_{i \in \mathcal{Y}} p_i \log(1 - p_i) = 1. \tag{3.17}$$

Furthermore, we get

$$\begin{aligned} H(X_t) &= -N_1 \cdot \frac{1}{2N_1} \log \frac{1}{2N_1} - N_2 \cdot \frac{1}{2N_2} \log \frac{1}{2N_2} \\ &= 1 + \frac{1}{2}(\log N_1 + \log N_2) \end{aligned} \tag{3.18}$$

and

$$\begin{aligned} H(X_t | X_{t-1}) &= -N_1 \cdot \frac{1}{2N_1} \cdot N_2 \cdot \frac{1}{N_2} \log \frac{1}{N_2} - N_2 \cdot \frac{1}{2N_2} \cdot N_1 \cdot \frac{1}{N_1} \log \frac{1}{N_1} \\ &= \frac{1}{2}(\log N_1 + \log N_2). \end{aligned} \tag{3.19}$$

Therefore, $I(X_t; X_{t-1}) = H(X_t) - H(X_t | X_{t-1}) = 1 = \mathcal{J}(\mathcal{Y})$, i.e. the synthesizing Infomap objective achieves its global optimum for correct partition of the given bipartite network.

4. Methods

In this chapter we briefly summarize how we implemented our cost function in Section 4.1 and give an overview of the software packages we utilized in the course of our work in Section 4.2. In Section 4.3 we describe different initialization strategies for the search algorithm before we explain in Section 4.4 the benchmark selection and generation for the experiments. Finally, we define metrics used to analyze and compare the examined community detection algorithms in Section 4.5.

4.1. Synthesizing Infomap Implementation

To compare the performance of the original Infomap and synthesizing Infomap we aimed for a common optimization algorithm to provide comparability. Hence, we integrated the synthesizing Infomap objective into the existing Infomap framework [8] since it is available open-source and proven to be a practical framework. The modified framework used in the course of this work can be found at `https://github.com/chritoth/infomap`. Further details regarding the implementation are given in Appendix A.

4.2. Utilized Software Packages

In the course of our work we utilized the software packages listed in Table 4.1 to conduct our experiments.

Package Name	Version	Reference
`networkx`	`2.4`	[21]
`scikit-learn`	`0.22.1`	[22]
`clusim`	`0.3.7`	[23]
`matplotlib`	`3.1.2`	[24]
`numpy`	`1.18.1`	[25]
`pandas`	`0.25.3`	[26]

Table 4.1.: Utilized software packages and versions.

4.3. Initialization

The standard initialization used in the Infomap framework puts each node of a given network in its own module, i.e. $\mathcal{Y} = \mathcal{X}$ and $m_{init}(\alpha) = \alpha, \forall \alpha \in \mathcal{X}$ for a given graph $\mathcal{G} = (\mathcal{X}, E)$. This provenly works well for the map equation as a cost function. Additionally, we propose an alternative initialization for synthesizing Infomap based on spectral clustering [27, 28, 29] and hence refer to it as spectral clustering initialization (SCI). We precluster a given network with N nodes into $\sqrt{N}$ clusters by employing the well known *scikit-learn* software package. From there we utilize `SpectralClustering()` with parameters as given in Table 4.2. As the similarity matrix we take the graphs adjacency matrix. In our experiments we report results for both initialization strategies used with synthesizing Infomap whenever it provides valuable insight.

4.4. Benchmark Selection and Generation

In order to validate and compare the results of community detection methods it is common practice to evaluate their performance on benchmark graphs [5, 2, 3, 17, 19] where the ground truth community structure is

Parameter	Value
`n_clusters`	$\sqrt{N}$
`n_init`	`1`
`affinity`	`'precomputed'`
`assign_labels`	`'kmeans'`

Table 4.2.: Parameter setup of the *scikit-learn* spectral clustering method. Any not listed parameters assume default values. N denotes the number of nodes in the network.

known. Such benchmark graphs can either be real-world networks or artificially generated. The former seems to be the favorable choice, since eventually any community detection algorithm is to be applied on real-world networks. However, there are several limitations to this approach. It is often difficult and time-consuming to obtain ground truth for real-world networks. Hence, available benchmark networks are either too small (e.g. Zachary's karate club [30]) or ground truth is either missing or extracted from node metadata (cf. the network databases [31, 32, 33]). Albeit the use of node metadata as ground truth can conveniently provide labeled networks at scale, the authors of [34] show that this approach gives rise to several practical problems when testing community detection algorithms. First, particular metadata can be irrelevant to the network structure or it may capture different aspects than the communities detected by an algorithm. Secondly, communities might not exist or be non-detectable in a network, although metadata raises such an impression. Moreover, the authors prove that no single algorithm can behave optimally for all possible community detection scenarios which raises another issue: the network parameters of real-world networks are not controllable, which limits the possible test cases to certain types of networks.

To this regard, artificially generated graphs provide (high) flexibility in tuning network parameters. In addition, arbitrarily many graph realizations can be generated, allowing for a more robust evaluation of algorithm behavior. The two prevalent benchmarks found in the literature are the GN benchmark [3] (after Girvan and Newman) and the LFR benchmark [17,

Parameter	Description	Value
`n`	Number of nodes in the network N	$500 \ldots 10000$
`mu`	Mixing parameter μ	$0.1 \ldots 0.75$
`max_community`	Maximum community size N_c^{max}	$0.2N$
`min_community`	Minimum community size N_c^{min}	$0.25N_c^{max}$
`max_degree`	Maximum node degree	$0.3N_c^{max}$
`min_degree`	Minimum node degree	$0.4N_c^{min}$
`tau1`	Degree distribution exponent	3.5
`tau2`	Community size distribution exponent	1.1

Table 4.3.: Parameter setup of the LFR benchmark generation using *networkx*. Any not listed parameters assume default values.

35] proposed by Lancichinetti, Fortunato and Radicchi. The former was found [17] to be a bad proxy for the community structure in real-world networks, as all its communities are equally sized and all nodes approximately have the same degree. As a result, the LFR benchmark more and more replaces the GN benchmark in more recent studies [5, 19]. Therefore, we use the LFR benchmark to evaluate the performance of synthesizing Infomap and to compare it to the results achieved by standard Infomap in Section 5.2. For the generation of the LFR benchmark graphs we use the *networkx* software package. We utilize `LFR_benchmark_graph()` with parameters as given in Table 4.3. A sample benchmark graph is visualized in Figure 4.1. Nevertheless, in order to prove the practical applicability of synthesizing Infomap we also evaluate its performance on a set of real-world networks taken from [31] in Section 5.3.

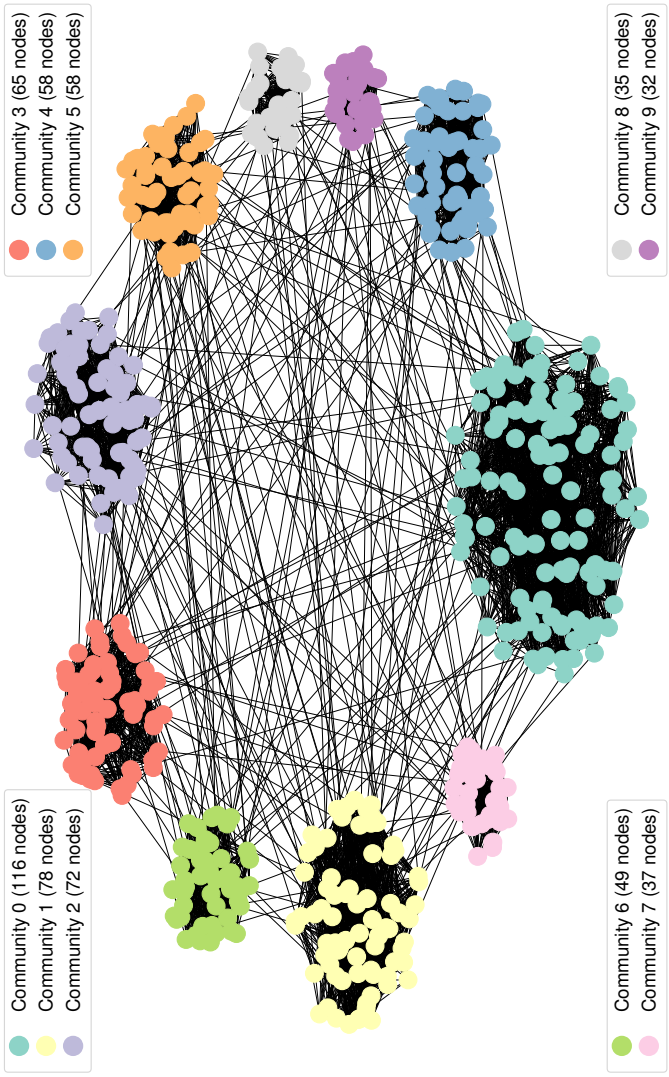


Figure 4.1.: Sample LFR benchmark graph with ground truth communities generated with `LFR_benchmark_graph()`. The graph has $N = 600$ nodes and a mixing parameter of $\mu = 0.03$. Nodes with the same color belong to the same community.

4.5. Metrics and Performance Measures

For analyzing and comparing the behavior of Infomap and synthesizing Infomap we consider the following metrics.

4.5.1. Adjusted Mutual Information

The adjusted mutual information (AMI) is a similarity measure for comparing partitions proposed in [36]. AMI values close to 1 indicate high similarity whereas a values around 0 reflect low similarity. We consider two partitions $\mathcal{U}, \mathcal{V}$ as described in Section 2.1.2. Then we define the AMI as

$$\mathrm{AMI}(\mathcal{U}, \mathcal{V}) = \frac{I(\mathcal{U}, \mathcal{V}) - \mathrm{E}\{I(\mathcal{U}, \mathcal{V})\}}{\frac{1}{2}[H(\mathcal{U}) + H(\mathcal{V})] - \mathrm{E}\{I(\mathcal{U}, \mathcal{V})\}}, \tag{4.1}$$

where $\mathrm{E}\{\cdot\}$ denotes the expectation operator with respect to a chosen permutation model. For a thorough discussion on how to compute the information theoretic measures necessary to compute $\mathrm{AMI}(\mathcal{U}, \mathcal{V})$ we refer the interested reader to [36]. Although there are several variants of the AMI in terms of normalization, we stick to the AMI normalized by the arithmetic mean as in Equation (4.1).

We note that further similarity measures based on the mutual information have been proposed in [37, 38]. However, we chose AMI over variants of the normalized mutual information [37] (NMI), because the adjustment for chance in AMI provides us with a constant baseline when comparing the different community detection algorithms, which NMI does not. We did not consider the standardized mutual information [38] (SMI) as to the best of our knowledge there is no easy-to-integrate implementation publicly available. Hence, for the sake of reproducibility we use the AMI implementation `adjusted_mutual_info_score` provided by *scikit-learn*.

4.5.2. Average Community Clustering

In our experiments we employ the average community clustering (ACC) as a measure of „connectedness "within the communities of a network. Assume an undirected and unweighted graph $\mathcal{G}(\mathcal{X}, E)$ and an arbitrary clustering $\mathcal{Y}$ on $\mathcal{G}$ as described in 2.1.2. The clustering coefficient [39] $c_{\mathcal{G}}(\alpha)$ of a node $\alpha \in \mathcal{X}$ with respect to the graph $\mathcal{G}$ is given as the number of triangles $T(\alpha)$ including α normalized by the number of triangles if $\mathcal{G}$ was fully connected, i.e.

$$c_{\mathcal{G}}(\alpha) = \frac{2 \cdot T(\alpha)}{\deg(\alpha)(\deg(\alpha) - 1)}. \tag{4.2}$$

The average clustering coefficient for the graph $\mathcal{G}$ is then given as

$$\bar{c}_{\mathcal{G}} = \frac{1}{|\mathcal{X}|} \sum_{\alpha \in \mathcal{X}} c_{\mathcal{G}}(\alpha). \tag{4.3}$$

Now let $\mathcal{G}_i$ denote the sub-graph

$$\mathcal{G}_i = (\mathcal{Y}_i, E_i) \quad \text{where} \quad E_i = E \cap \{(\alpha, \beta) | \alpha \neq \beta;\ \alpha, \beta \in \mathcal{Y}_i\} \tag{4.4}$$

induced by the community $\mathcal{Y}_i \in \mathcal{Y}$. Then we define the community clustering coefficient of a module $\mathcal{Y}_i$ as

$$\bar{c}_{\mathcal{G}}(\mathcal{Y}_i) = \frac{1}{|\mathcal{Y}_i|} \sum_{\alpha \in \mathcal{Y}_i} c_{\mathcal{G}_i}(\alpha) \tag{4.5}$$

and finally the average community clustering of a graph $\mathcal{G}$ given the clustering $\mathcal{Y}$ as

$$\mathrm{ACC}(\mathcal{Y}) = \frac{1}{|\mathcal{Y}|} \sum_{\mathcal{Y}_i \in \mathcal{Y}} \bar{c}_{\mathcal{G}}(\mathcal{Y}_i). \tag{4.6}$$

One can see that the ACC is bounded by $0 \leq \mathrm{ACC}(\mathcal{Y}) \leq 1$, where a higher ACC value indicates more densely connected communities (w.r.t internal connections) and $\mathrm{ACC}(\mathcal{Y}) = 1$ if all communities in $\mathcal{Y}$ are cliques.

In contrast to the mixing parameter and the modularity measure [12], the ACC is *not* a global measure in the sense that it evaluates the existence of communities and how well they are defined. Instead, we consider each community as an independent entity to see how well it is connected internally, regardless of its external connections.

4.5.3. Mean Relative Error in the Number of Detected Communities

We compute the mean relative error in the number of detected communities (MRENDC) over a set of K realizations of benchmark graphs for a given parameter set θ (i.e. parameters according to Table 4.3). Let $C^{\theta}_{true,i}$ denote the true number of communities of the i-th benchmark graph realization and let C^{θ}_{i} denote the number of communities detected by any of the examined methods. Then the mean relative error is given as

$$\bar{e}_{\theta} = \frac{1}{K} \sum_{k=1}^{K} \frac{C^{\theta}_{i} - C^{\theta}_{true,i}}{C^{\theta}_{true,i}}. \tag{4.7}$$

Note that this error is lower bounded by $\bar{e}_{\theta} > -1$ since any number of (detected) communities will be a positive integer, i.e. $C^{\theta}_{i} \geq 1$ and $C^{\theta}_{true,i} \geq 1$ $\forall i, \theta$. Thus, an algorithm detecting always one community will stay within an error bound of $|\bar{e}_{\theta}| \leq 100\%$.

4.5.4. Omega Index

The Omega index is a similarity measure for comparing two clusterings with overlapping modules proposed in [40]. An Omega index close to 1 indicates high similarity whereas a index close to 0 reflects low similarity. We consider two clusterings $\mathcal{U}$, $\mathcal{V}$ as described in 2.1.2. Let $N_k^{\mathcal{U},\mathcal{V}}$ denote the number of node pairs that occur exactly k times within a common cluster in

both clusterings $\mathcal{U}$ and $\mathcal{V}$. Then the non-adjusted Omega index writes

$$\omega(\mathcal{U},\mathcal{V}) = \frac{1}{\binom{N}{2}} \sum_k N_k^{\mathcal{U},\mathcal{V}}. \tag{4.8}$$

Adjusting for chance yields the Omega index

$$\Omega(\mathcal{U},\mathcal{V}) = \frac{\omega(\mathcal{U},\mathcal{V}) - \mathrm{E}\{\omega(\mathcal{U},\mathcal{V})\}}{1 - \mathrm{E}\{\omega(\mathcal{U},\mathcal{V})\}}, \tag{4.9}$$

where $\mathrm{E}\{\cdot\}$ denotes the expectation operator with respect to a chosen permutation model. For a detailed derivation we refer the interested reader to [40]. In this work we use the Omega index implementation `omega_index` provided by *clusim*.

4.5.5. Overlapping Normalized Mutual Information

The overlapping normalized mutual information (ONMI) is a similarity measure for comparing two clusterings with overlapping modules proposed in [41]. ONMI values close to 1 indicate high similarity whereas a values close to 0 reflect low similarity. We consider two clusterings $\mathcal{U}$, $\mathcal{V}$ as described in 2.1.2. Then we define the ONMI as

$$\mathrm{ONMI}(\mathcal{U},\mathcal{V}) = 1 - \frac{1}{2}[H_{norm}(\mathcal{U}|\mathcal{V}) + H_{norm}(\mathcal{V}|\mathcal{U})], \tag{4.10}$$

where $H_{norm}(\cdot|\cdot)$ indicates a normalization of the conditional entropy such that $H_{norm}(\cdot|\cdot) \in [0,1]$. For a thorough discussion on how to compute $\mathrm{ONMI}(\mathcal{U},\mathcal{V})$ we refer the interested reader to [41]. In this work we use the OMNI implementation `onmi` provided by *clusim*.

5. Experiments

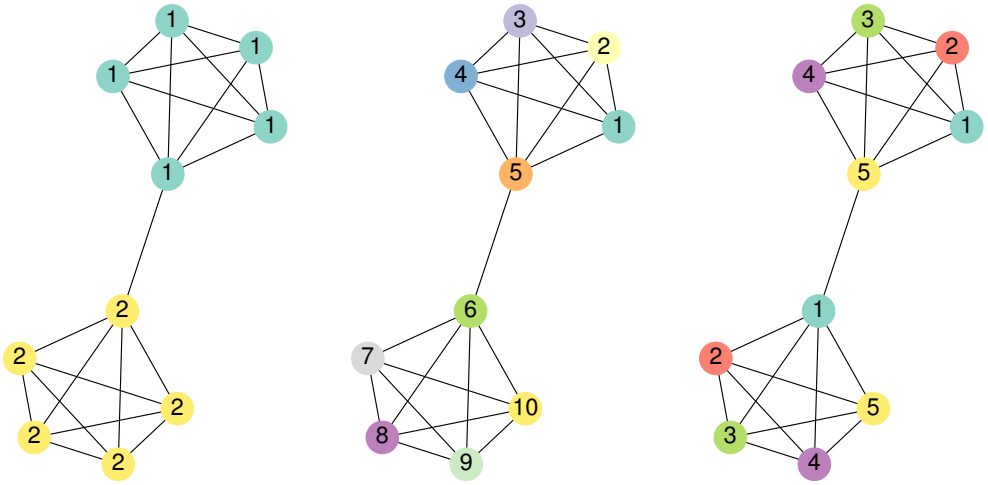
In this chapter we explore the behavior of synthesizing Infomap on a set of toy graphs in Section 5.1. Furthermore, we compare the results of standard Infomap and synthesizing Infomap with and without SCI on the LFR benchmark in Section 5.2 and on a selection of real-world networks in Section 5.3. We do so by utilizing the metrics described in Section 4.5.

5.1. Behavior on Toy Graphs

In this section we test our implementation of the synthesizing Infomap objective by examining a selection of prototypical toy graphs. In particular we consider the barbell graph in Section 5.1.1 and as a natural extension a ring of cliques in Section 5.1.2. Lastly, we discuss a network of two connected rings in Section 5.1.3. We restrict all graphs to be unweighted and undirected. Moreover, we compare the observed behavior with the results achieved by standard Infomap where insightful.

5.1.1. The Barbell Network

The barbell network is an undirected and unweighted graph $\mathcal{G} = (\mathcal{X}, E)$ consisting of two maximal cliques $\mathcal{C}_1, \mathcal{C}_2 \subset \mathcal{X}$ that are connected by a single edge, i.e., $|(\mathcal{C}_1 \times \mathcal{C}_2) \cap E| = 1$. We further assume that the cliques are of equal size, i.e. $n_c := |\mathcal{C}_1| = |\mathcal{C}_2|$.



(a) $\mathcal{Y}_{true}$ - ground truth. (b) $\mathcal{Y}_{init}$ - each node a module. (c) $\mathcal{Y}_{ind}$ - maximal independent sets.

Figure 5.1.: Sample barbell networks with $n_c = 5$ nodes per clique. Nodes with the same color and numbering belong to the same module. For ground truth we have two modules, i.e. each clique is a module. The initial partition has $2 \cdot n_c = 10$ modules (each node) and the maximal independent sets partition has $n_c = 5$ modules. Synthesizing Infomap cannot escape the initial partition whereas initialization with SCI always detects the ground truth communities.

We analyze the behavior of synthesizing Infomap by examining three partitions of interest (see Figure 5.1). Firstly, we define ground truth as the partition $\mathcal{Y}_{true}$ where each clique is a module, i.e. $\mathcal{Y}_{true} = \{\mathcal{C}_1, \mathcal{C}_2\}$. Secondly, a partition $\mathcal{Y}_{init} = \mathcal{X}$ where each vertex is a separate module. We note that Infomap uses this partitioning as the initialization for optimization, hence the subscript. Thirdly, a partition $\mathcal{Y}_{ind}$ into n_c maximal independent sets of cardinality two, i.e. $\mathcal{Y}_{ind} = \{S_i\}$ where $|S_i| = 2$, $(S_i \times S_i) \cap E = \{\}$ and $S_i \cap S_j = \{\}$ for $i, j \in \{1, \ldots, n_c\}$.

Figure 5.2 shows the analytical objective as well as the implementation results for the three examined partitions as a function of the clique size n_c. Apparently, the analytical objective for all three partitions approaches some limit as n_c increases. Indeed, for $\mathcal{Y}_{init}$ and $\mathcal{Y}_{ind}$ it is easy to see that

$p_{1\to1} = p_{2\to2} = 0$ and therefore

$$\mathcal{J}(\mathcal{Y}_{ind}) = \mathcal{J}(\mathcal{Y}_{init}) = -\sum_{i\in\mathcal{Y}} p_i \log(1 - p_i).$$

Now as $n_c \to \infty$ it follows that $p_i = p_\alpha \to \frac{1}{N}$ for $\mathcal{Y}_{init}$ and thus

$$\lim_{n_c\to\infty} \mathcal{J}(\mathcal{Y}_{init}) = \lim_{n_c\to\infty} -\sum_{i\in\mathcal{Y}} \frac{1}{N} \log\left(1 - \frac{1}{N}\right) = \lim_{n_c\to\infty} -\log\left(1 - \frac{1}{N}\right) = 0.$$

The limit for $\mathcal{Y}_{ind}$ can be reasoned along similar lines for $p_i \to \frac{1}{n_c}$ as $n_c \to \infty$.

For the ground truth partition $\mathcal{Y}_{true}$ we have $p_{1\to1}, p_{2\to2} \to 1$ as $n_c \to \infty$. Therefore,

$$\lim_{n_c\to\infty} \mathcal{J}(\mathcal{Y}_{true}) = -\sum_{i\in\mathcal{Y}} p_i \log(p_i)$$

and since $p_1 = p_2 = \frac{1}{2}$ we get

$$\lim_{n_c\to\infty} \mathcal{J}(\mathcal{Y}_{true}) = \lim_{n_c\to\infty} -2 \cdot \frac{1}{2} \log\left(\frac{1}{2}\right) = 1.$$

Given these observations, using each node as a module initially appears to be a reasonable starting point for optimization, as it tends to the minimum achievable value of a function we aim to maximize. However, practical experiments with synthesizing Infomap (see Figure 5.2) reveal that, for the barbell network, our standard implementation cannot escape the initial partition even for larger networks ($n_c > 1000$). By utilizing the spectral clustering initialization (SCI) as described in Section 4.3 our implementation can correctly identify ground truth for all clique sizes.

5.1.2. A Ring of Cliques

The ring of cliques network is an undirected and unweighted graph $\mathcal{G} = (\mathcal{X}, E)$ consisting of c maximal cliques $\mathcal{C}_i \subset \mathcal{X}$ for $i \in \{1, \ldots, c\}$. Additionally, each clique is connected to two neighboring cliques by exactly one edge

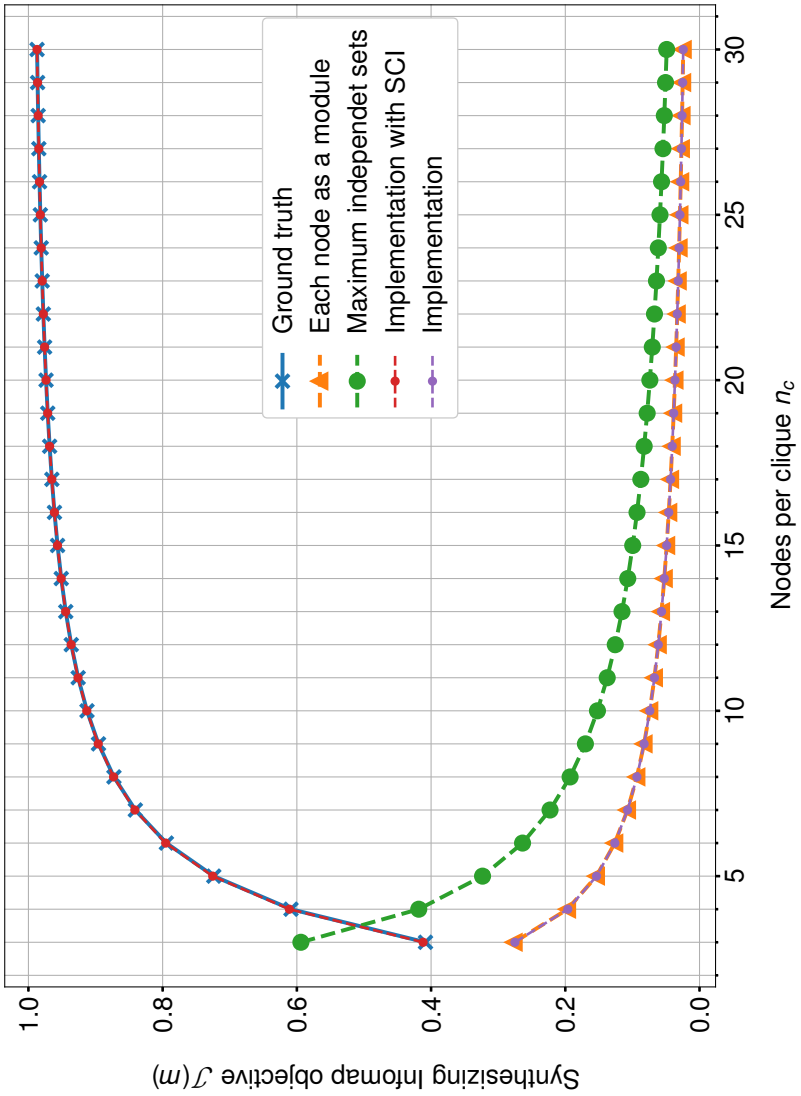
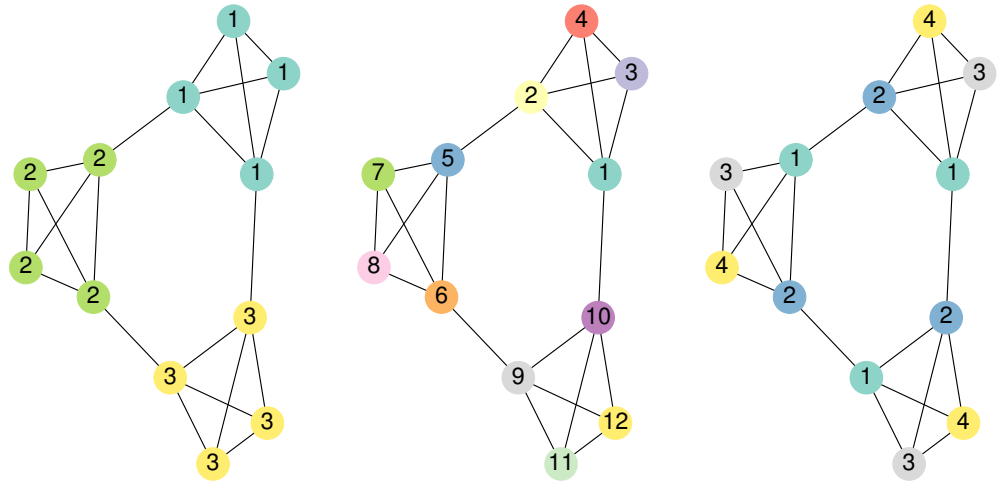


Figure 5.2.: Synthesizing Infomap objective as a function of clique size $n_c \in \{3, \ldots, 30\}$ for a barbell network. For sufficiently large cliques, the ground truth partition yields the best objective whereas the objectives for the initial partition and the independent sets partition vanish with increasing clique size.



(a) $\mathcal{Y}_{true}$ - ground truth. (b) $\mathcal{Y}_{init}$ - each node a module. (c) $\mathcal{Y}_{ind}$ - maximal independent sets.

Figure 5.3.: Sample ring of cliques networks with $c = 3$ cliques $n_c = 4$ nodes per clique. Nodes with the same color and numbering belong to the same community. For ground truth we have three communities, i.e., each clique is a community. The initial partition has $c \cdot n_c = 12$ communities (each node) and the maximal independent sets partition has $n_c = 4$ communities.

between any of their nodes, i.e. $|(\mathcal{C}_1 \times \mathcal{C}_c) \cap E| = |(\mathcal{C}_i \times \mathcal{C}_{i+1}) \cap E| = 1$ for $i \in \{1, \dots, c-1\}$. Furthermore, the cliques are of equal size, i.e. $n_c := |\mathcal{C}_i| = |\mathcal{C}_j|$ for all $i, j \in \{1, \dots, c\}$.

We again analyze the behavior of synthesizing Infomap by examining three partitions of interest (see Figure 5.3). Firstly, we define ground truth as the partition $\mathcal{Y}_{true}$ where each clique is a module, i.e. $\mathcal{Y}_{true} = \{\mathcal{C}_i\}$ for $i \in \{1, \dots, c\}$. Secondly, the partition $\mathcal{Y}_{init} = \mathcal{X}$. Thirdly, a partition $\mathcal{Y}_{ind}$ into n_c maximal independent sets of cardinality c, i.e. $\mathcal{Y}_{ind} = \{S_i\}$ where $|S_i| = c$, $(S_i \times S_i) \cap E = \{\}$ and $S_i \cap S_j = \{\}$ for $i, j \in \{1, \dots, n_c\}$.

Figures 5.4 and 5.5 show the analytical objective as well as the implementation results for the three examined partitions as a function of the clique size n_c. Again, the objective for all three partitions approaches specific limits. The vanishing limits for $\mathcal{J}(\mathcal{Y}_{init})$ and $\mathcal{J}(\mathcal{Y}_{ind})$ can be obtained as in Section 5.1.1.

For $\mathcal{J}(\mathcal{Y}_{true})$ we again assume

$$\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_{true}) = -\sum_{i \in \mathcal{Y}} p_i \log(p_i).$$

However, now $p_i = \frac{1}{c}$ for all $i \in \mathcal{Y}$ is independent of the clique size and therefore

$$\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_{true}) = \lim_{n_c \to \infty} -c \cdot \frac{1}{c} \log\left(\frac{1}{c}\right) = \log c.$$

This matches the observed behavior in Figures 5.4 and 5.5.

Conducting experiments with varying parameter pairs $\{c, n_c\}$ present us with the following observations. For networks with few cliques and sufficiently many nodes per clique (e.g., $\{c = 4, n_c = 30\}$), synthesizing Infomap cannot escape the initial partition $\mathcal{Y}_{init}$. Conversely, for networks with a large number of small-sized cliques (e.g. $\{c = 32, n_c = 3\}$) we do not arrive at the ground truth partition either. Although synthesizing Infomap can escape the initial partition, it still under-partitions the network as shown in Figure 5.6. Since Infomap cannot identify the correct partitioning either, we argue that for such edge cases the problem is ill-posed.

For balanced parameter sets $\{c, n_c\}$ synthesizing Infomap identifies the cliques correctly. Using synthesizing Infomap with SCI we can uncover the cliques as long as $c \lessapprox n_c$ (cf. Figure 5.7). Otherwise, the variant with SCI under-partitions the network and yields similar partitions as the variant without SCI (e.g. Figure 5.6).

5.1.3. A Two Rings Network

The two rings network is an undirected and unweighted graph $\mathcal{G} = (\mathcal{X}, E)$ consisting of two rings of nodes $\mathcal{C}_1, \mathcal{C}_2 \subset \mathcal{X}$ of equal size, i.e. $n_c := |\mathcal{C}_1| = |\mathcal{C}_2|$. The rings are connected by a single edge, i.e. $|(\mathcal{C}_1 \times \mathcal{C}_2) \cap E| = 1$. Within each ring a node is adjacent to exactly two neighboring nodes, i.e. the set of edges is given as $E = \{(\alpha, \alpha + 1), \alpha \in \{1, \ldots, N - 1\}\} \cup \{(1, n_c), (n_c + 1, N)\}$.

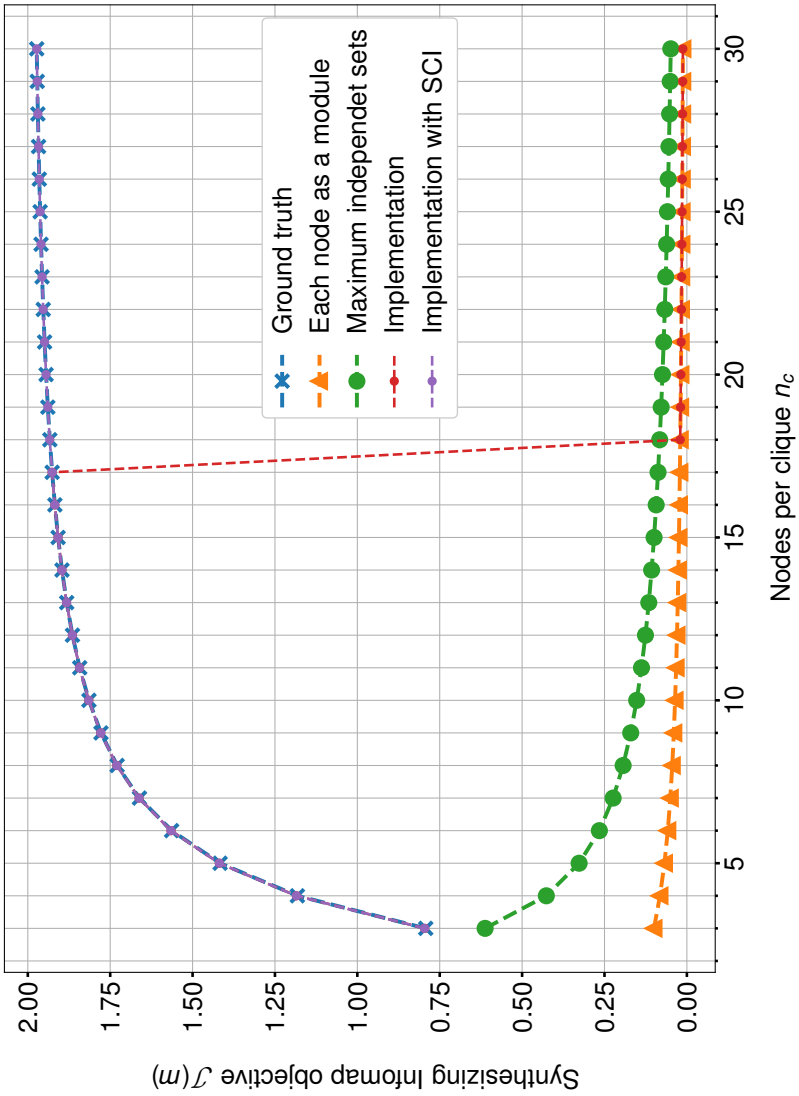


Figure 5.4.: Synthesizing Infomap objective as a function of clique size $n_c \in \{3, \ldots, 30\}$ for a ring of $c = 4$ cliques. The ground truth partition always yields the best objective for the three partitions in comparison. The objectives of the initial partition and the independent sets partition vanish with increasing clique size. Synthesizing Infomap with SCI always detects the ground truth modules whereas the variant without SCI fails to do so for $n_c \geq 18$ nodes per clique.

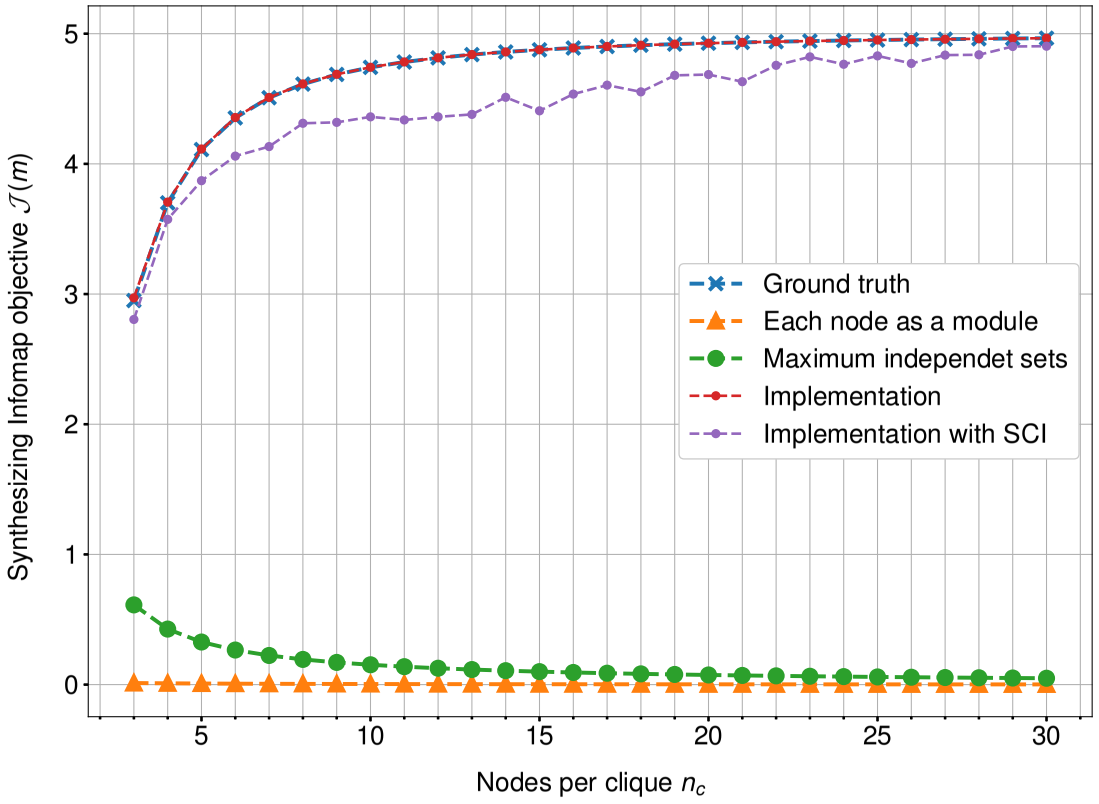


Figure 5.5.: Synthesizing Infomap objective as a function of clique size $n_c \in \{3, \ldots, 30\}$ for a ring of $c = 32$ cliques. The ground truth partition always yields the best objective for the three partitions in comparison. The objectives of the initial partition and the independent sets partition vanish with increasing clique size. For sufficiently large clique sizes (here for $n_c \geq 4$) our synthesizing Infomap implementation always detects the ground truth modules whereas the variant with SCI fails to do so. We note that for $n_c = 3$ a partition as depicted in Figure 5.6 yields a better objective than the ground truth partition.

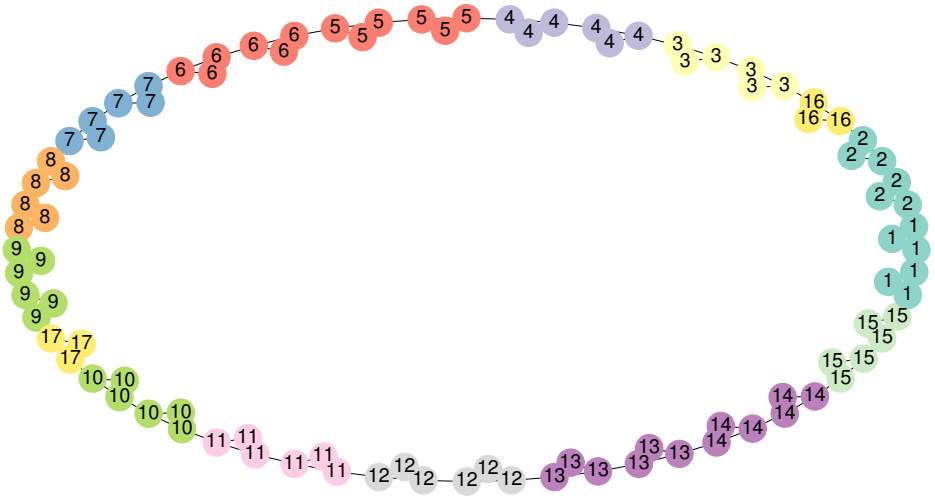


Figure 5.6.: Ring of $c = 32$ cliques with $n_c = 3$ nodes per clique classified by synthesizing Infomap with standard initialization. Nodes with the same color and numbering belong to the same community.

We analyze the behavior of synthesizing Infomap by examining four partitions of interest (see Figure 5.8). Firstly, we define ground truth as the partition $\mathcal{Y}_{true} = \{\mathcal{C}_1, \mathcal{C}_2\}$ where each ring is a module (Figure 5.8a. Secondly, again the partition $\mathcal{Y} = \mathcal{X}$ (Figure 5.8b). Thirdly, a partition into maximal independent sets. Here we distinguish between the cases n_c even (Figure 5.8c) and n_c odd (Figure 5.8d). For even n_c, we have a partition $\mathcal{Y}_{ind,even}$ into two maximal independent sets of cardinality n_c. For odd n_c, we have a partition $\mathcal{Y}_{ind,odd}$ into two independent sets of cardinality $n_c - 1$ and an independent set of cardinality two. Fourthly, a partition $\mathcal{Y}_c$ into c modules per ring ($2 \cdot c$ modules in total) as depicted in Figure 5.8e, i.e. $\mathcal{Y}_c = \{S_i\}$ where $i \in \{1, \ldots, c\}$. The number of nodes in each module are balanced, i.e., for all i it holds that $|S_i| \in \{\lfloor(\frac{n_c}{c})\rfloor, \lfloor(\frac{n_c}{c})\rfloor + 1\}$.

Figure 5.9 shows the analytical objective for the four examined partitions as a function of the ring size n_c. Again, the analytical objective for all four partitions approaches specific limits. The limit $\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_{init}) = 0$ can be obtained as in Section 5.1.1. The objective for the maximal independent sets

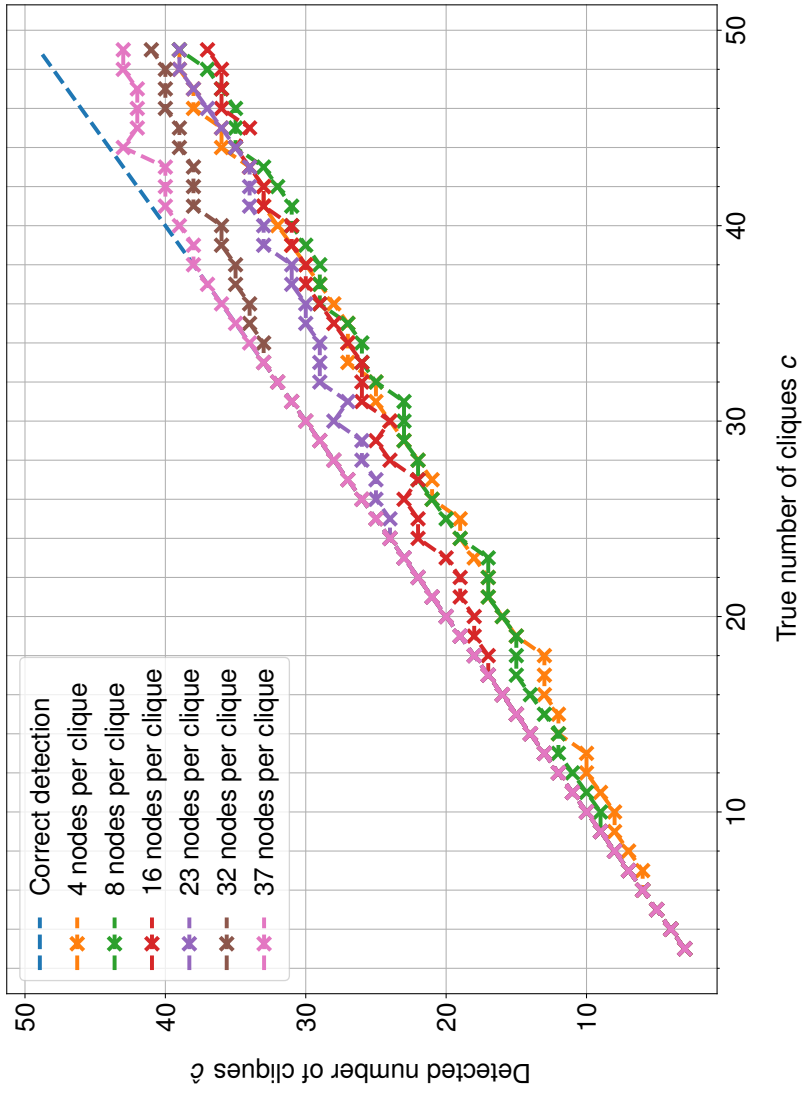
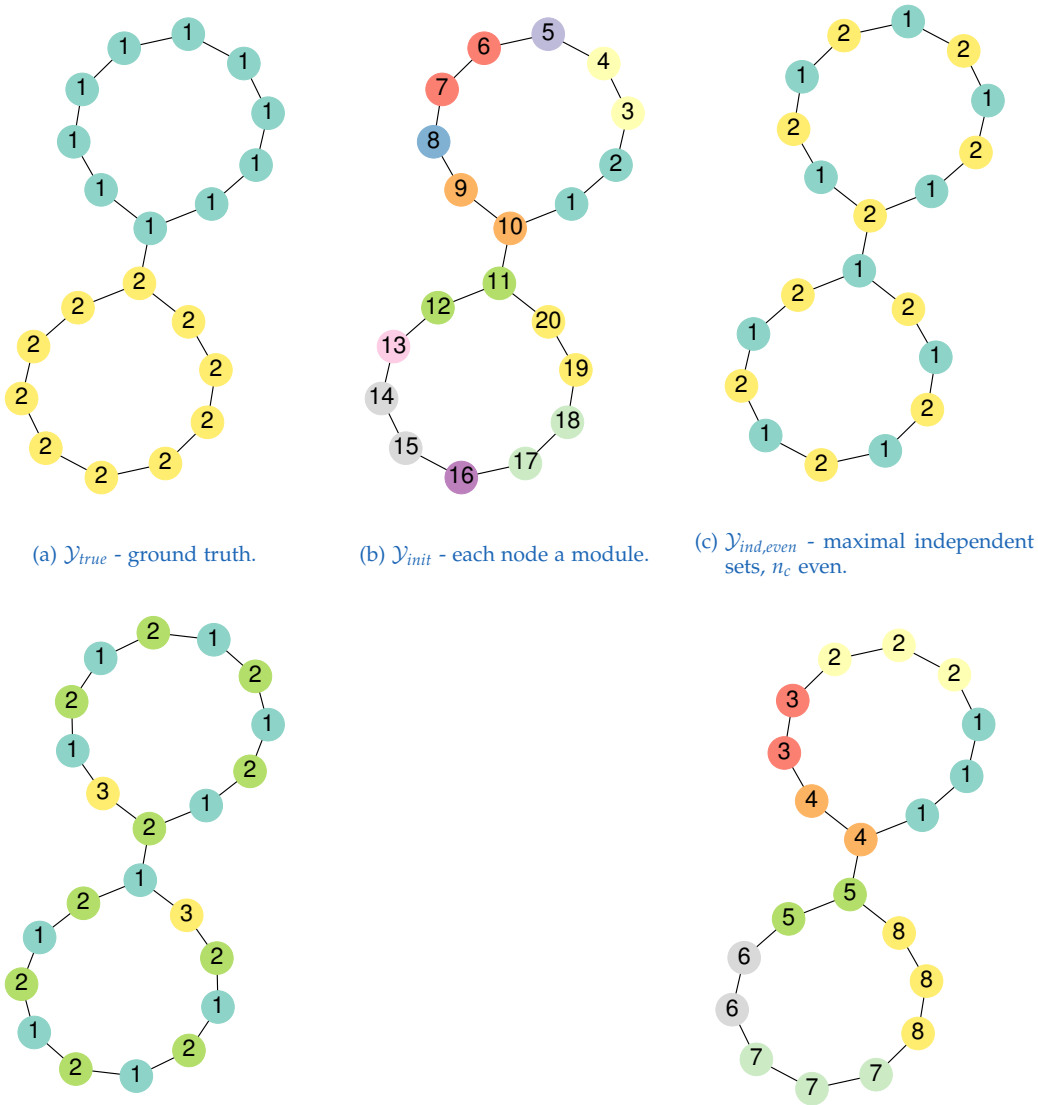


Figure 5.7.: Detected number of communities as a function of the true number of cliques for synthesizing Infomap with SCI on a ring of cliques network. We plot results for a set of fixed clique sizes $n_c \in \{4, 8, 16, 23, 32, 37\}$. Synthesizing Infomap with SCI detects the correct number of communities as long as $c \lessapprox n_c$.



(a) $\mathcal{Y}_{true}$ - ground truth.

(b) $\mathcal{Y}_{init}$ - each node a module.

(c) $\mathcal{Y}_{ind,even}$ - maximal independent sets, n_c even.

(d) $\mathcal{Y}_{ind,odd}$ - maximal independent sets, n_c odd.

(e) $\mathcal{Y}_c$ - $c = 4$ modules per ring.

Figure 5.8.: Sample two rings networks with different partitions. All shown networks have $n_c = 10$ nodes per ring except for (d) where $n_c = 11$. Nodes with the same color and numbering belong to the same module.

can be computed as

$$\mathcal{J}(\mathcal{Y}_{ind,\alpha}) = 1 - \log\left(1 + \frac{2\alpha}{|E|}\right) - \frac{2\alpha}{|E|}\log\left(\frac{|E| - 2\alpha}{|E| + 2\alpha}\right) - \frac{2\alpha}{|E|}$$

where

$$|E| = 2n_c + 1 \text{ and } \alpha = \begin{cases} 0 \text{ if } n_c \text{ even,} \\ 1 \text{ if } n_c \text{ odd.} \end{cases}$$

It is clear to see that $\mathcal{J}(\mathcal{Y}_{ind,even}) = 1$ and it is straightforward to show that $\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_{ind,odd}) = 1$. For any partition $\mathcal{Y}_c$ into a finite number c of modules per ring, one can show that

$$\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_c) = 1 + \log(c).$$

For $c = 2$ and $c = 4$, the objective approaches the limits $\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_2) = 2$ and $\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_8) = 4$, respectively, as depicted in Figure 5.10. The ground truth can be seen as a special case of a partition into $c = 1$ module per ring. Thus, $\lim_{n_c \to \infty} \mathcal{J}(\mathcal{Y}_{true}) = 1$.

Notably, for the two rings network the partition into maximal independent sets always yields a better objective than ground truth (cf. Figure 5.9). However, our implementation never returns such a partition even for small n_c, as the (synthesizing) Infomap optimization algorithm by design does not examine independent sets (cf. Section 3.2.3 and Appendix A.

For reasonably large rings (e.g. $n_c \geq 10$), synthesizing Infomap yields a partition into d connected ring segments (cf. Figure 5.10), i.e. the resulting partitions resemble our model description of a partition into c modules per ring. Varying the number of modules c as depicted in Figure 5.11, one can see that there is an optimal value c^* maximizing the objective for a given ring size n_c. Indeed, synthesizing Infomap finds $d \approx 2c^*$ partitions, closely approximating the optimal number of partitions for $\mathcal{Y}_c$. For example, $d = [6, 9, 18, 32]$ for $n_c = [10, 20, 50, 100]$ respectively (cf. Figure 5.11). Standard Infomap delivers the same type of partitioning with similar numbers of partitions, e.g. for $n_c = [10, 20, 50, 100]$ we get $d = [5, 8, 19, 31]$ respectively.

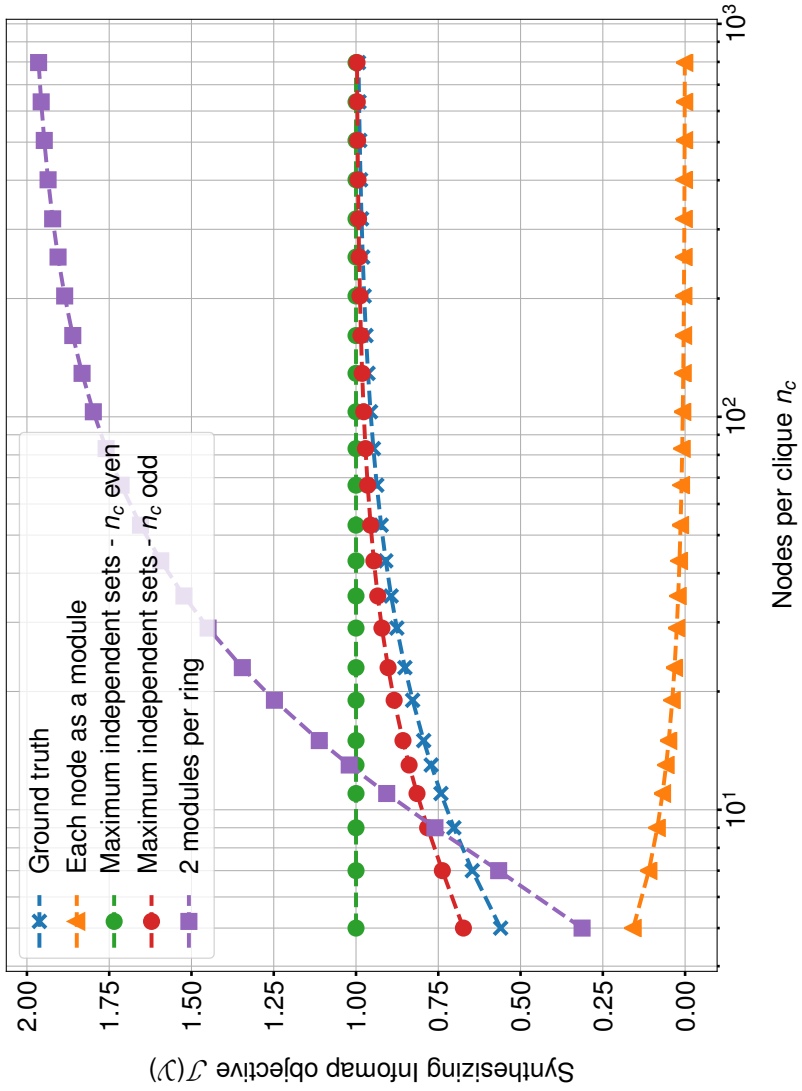


Figure 5.9.: Synthesizing Infomap objective as a function of the ring size $n_c \in \{3, \ldots, 1000\}$ on a semi-logarithmic scale. For sufficiently large rings any partition into $c \geq 2$ modules per ring yields a better objective than ground truth. Ground truth and maximal independent set partitions converge to $\mathcal{J}(\mathcal{Y}) = 1$.

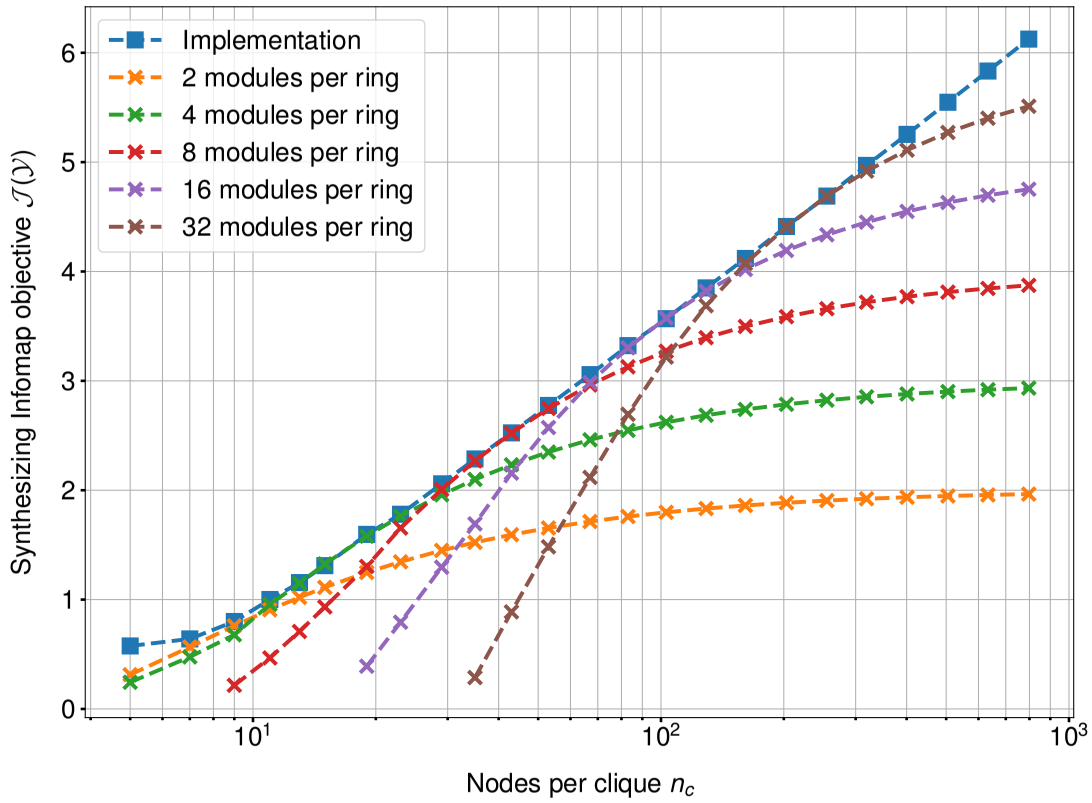


Figure 5.10.: Synthesizing Infomap implementation objective as a function of the ring size $n_c \in \{3, \ldots, 1000\}$ on a semi-logarithmic scale. Additionally, we plot the analytical objective for partitions into c modules per ring with varying values of c. Our implementation always achieves the optimal objective with respect to any partition into c modules per ring.

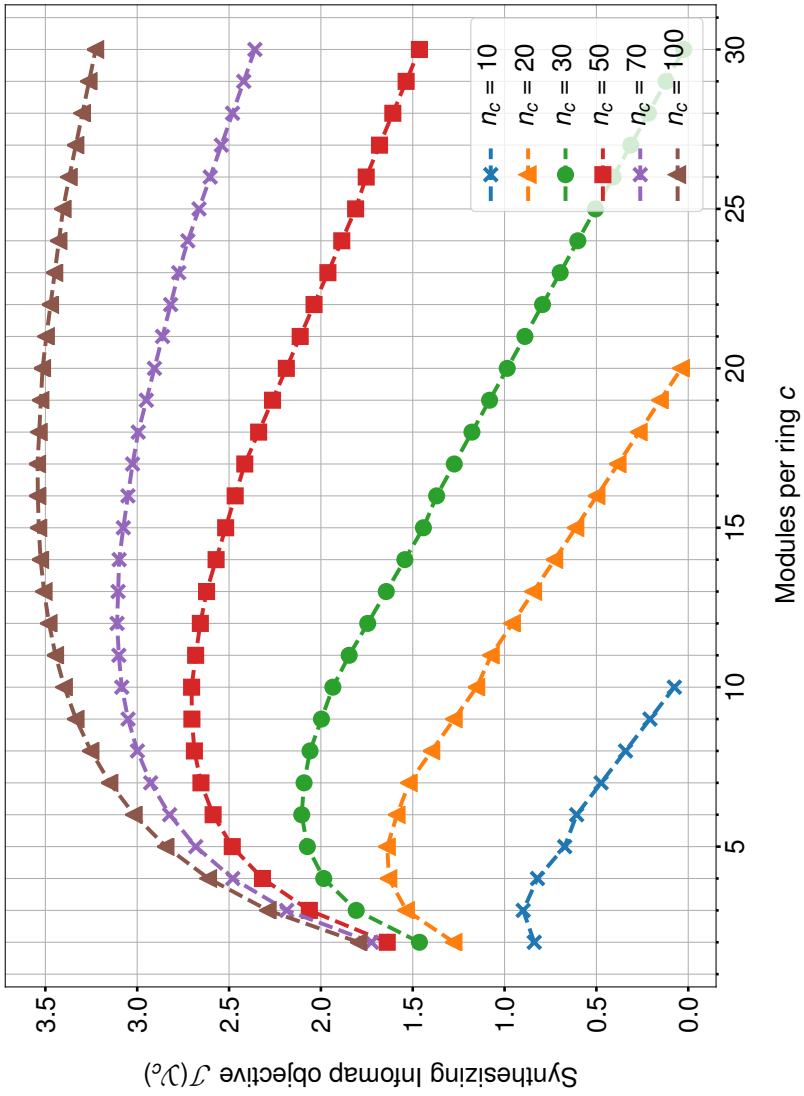


Figure 5.11.: Synthesizing Infomap objective for fixed ring sizes n_c as a function of the number of modules per ring $c \in \{2, \ldots, 30\}$. Notably, the optimal value for c is less pronounced the larger the number of nodes per ring.

5.2. LFR Benchmark

In this section we report the benchmark results for three types of experiments. First, we examine the AMI as a function of the mixing parameter in Section 5.2.1 and secondly the MRENDC as a function of the mixing parameter in Section 5.2.2. Thirdly, we look at the AMI as a function of the network size in Section 5.2.3. Additionally, in these three experiments we report the ACC for the generated benchmark networks. Finally, we compare the results visually by plotting a network with the communities detected by the different methods.

In our experiments we vary the mixing parameter μ in the range from 0.1 to 0.75. Lower values of μ provide no additional insight, as all detection algorithms manage to identify all modules accurately in our setup. Since we limit $N_c^{max} = 0.2N$ (see Table 4.3) in the benchmark graphs, theoretically communities exist up to $\mu < 0.8$ (cf. Section 2.2).

5.2.1. AMI as a Function of the Mixing Parameter

The results for AMI performances as a function of the mixing parameter are shown in Figures 5.12, 5.13, 5.14 and 5.15. As can be seen, Infomap correctly identifies the communities for sufficiently small values of μ, i.e. strong community structures, and transitions to vanishing AMI values as $\mu \to 0.75$ (cf. [5]). Interestingly, in contrast to [5], in our experiments the transition interval is dependent on the network size. Whereas AMI performance sharply drops for larger networks (e.g. Figure 5.15), it decreases gradually for smaller networks (e.g. Figure 5.12). Moreover, the turning point rather depends on the ACC than on the mixing parameter, occurring at a value of $\mathrm{ACC}(\mathcal{Y}_{true}) \approx 0.26$ regardless of the network size.

The synthesizing Infomap variants show similar behavior to Infomap. However, transition happens smoothly and turning points occur at increasing values of μ for increasing network sizes, i.e. for a fixed mixing parameter

AMI performance rises for larger network sizes (see Section 5.2.3). For sufficiently large networks (e.g. $N \geq 600$) both synthesizing Infomap variants outperform Infomap in terms of AMI, whereas synthesizing Infomap with SCI performs consistently better than the variant with standard initialization for all network sizes.

5.2.2. MRENDC as a Function of the Mixing Parameter

The MRENDC values as a function of the mixing parameter are compared in Figures 5.16, 5.17, 5.18 and 5.19. Infomap identifies the correct number of communities for sufficiently small values of μ. Transition phases are apparent in the MRENDC behavior as well, whereas the transition intervals coincide with the respective intervals of the AMI behavior described in Section 5.2.1. Again, the turning point much rather depends on the ACC than on the mixing parameter, occurring at a value of $\mathrm{ACC}(\mathcal{Y}_{true}) \approx 0.26$ regardless of the network size. For small networks (see Figure 5.16), Infomap transitions to MRENDC values in the vicinity of zero. In this case, Infomap either over- or underestimates the number of communities, yielding small MRENDC values on average. For the larger network sizes with $N \geq 600$ nodes, Infomap predicts only a single community for all nodes, transitioning to MRENDC values in range $[-0.8, -0.95]$ depending on the true number of communities.

The synthesizing Infomap variants correctly identify the number of communities for sufficiently small values of μ as well. Smooth transitions occur for increasing values of μ where, as for Infomap, the transition intervals coincide with the ones in Section 5.2.1. During and after the transition phases, both variants overestimate the number of communities, whereas synthesizing Infomap with SCI consistently performs better. Interestingly, MRENDC values after the transition approximately double with doubling network size.

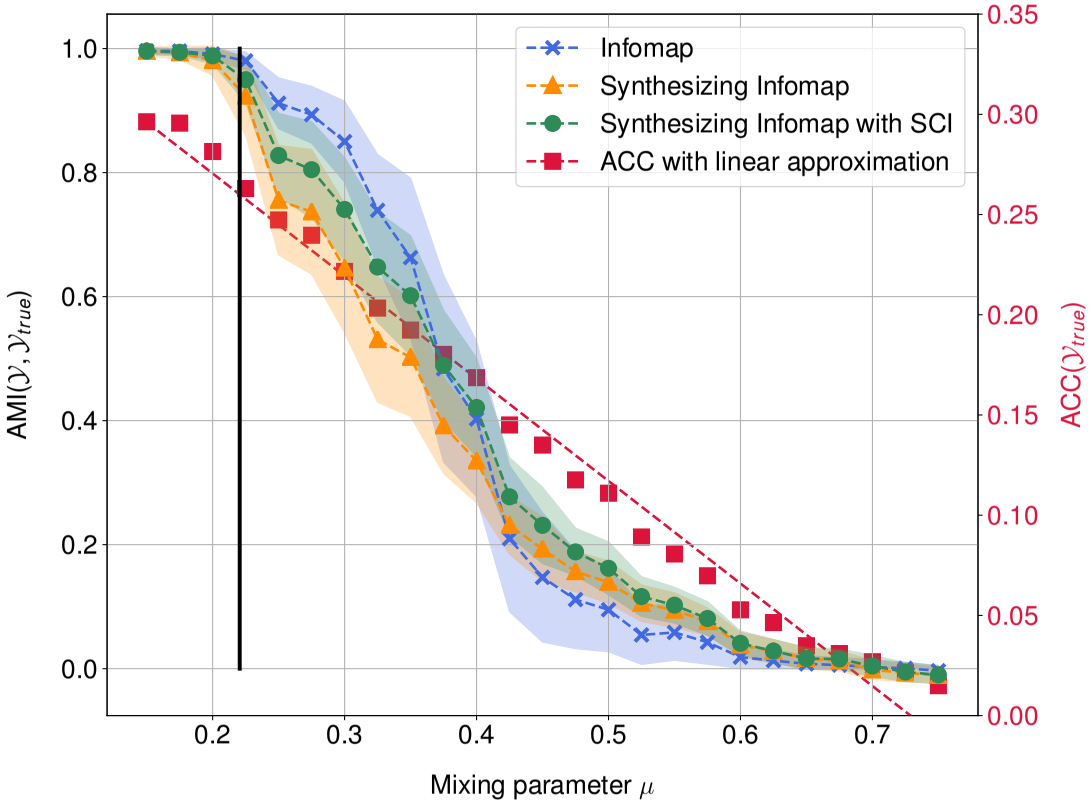


Figure 5.12.: Comparison of Infomap and synthesizing Infomap variants. The plot shows mean AMI and ACC scores achieved on 100 different LFR benchmark graphs with $N = 300$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.22$ corresponds to an ACC value of 0.26 in the linear approximation.

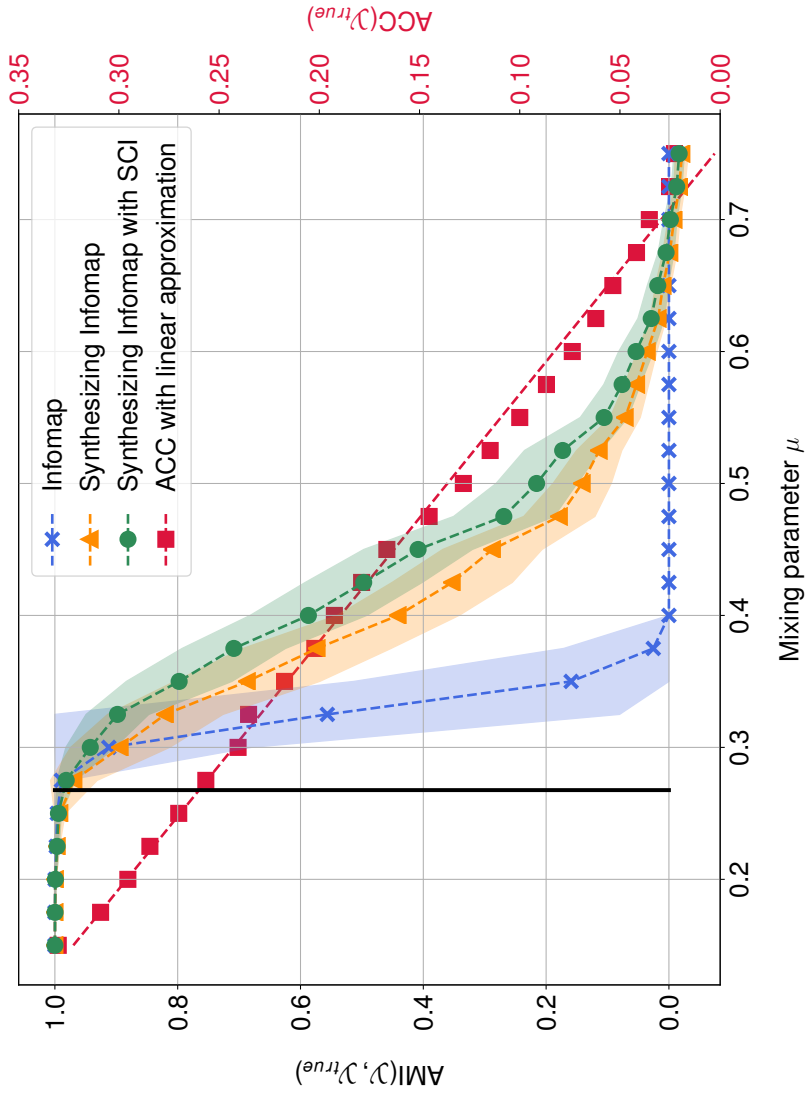


Figure 5.13: Comparison of Infomap and synthesizing Infomap variants. The plot shows mean AMI and ACC scores achieved on a minimum of 99 different LFR benchmark graphs with $N = 600$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.27$ corresponds to an ACC value of 0.26 in the linear approximation.

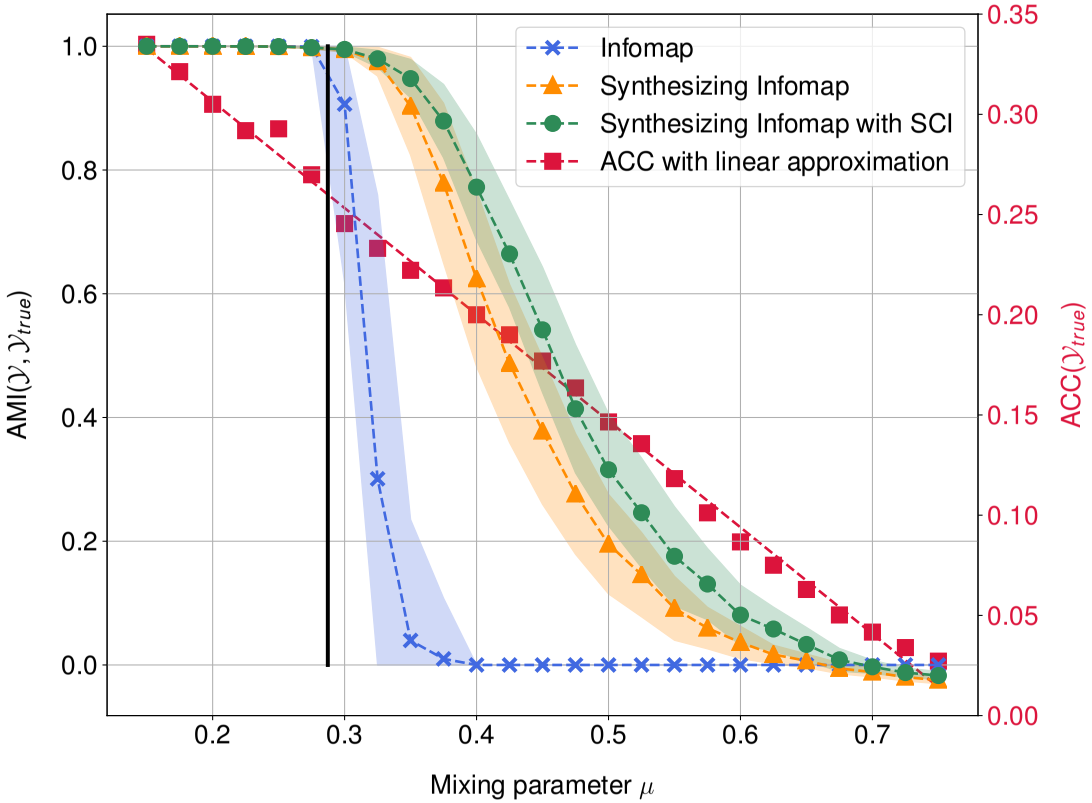


Figure 5.14.: Comparison of Infomap and synthesizing Infomap variants. The plot shows mean AMI and ACC scores achieved on a minimum of 95 different LFR benchmark graphs with $N = 1200$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.29$ corresponds to an ACC value of 0.26 in the linear approximation.

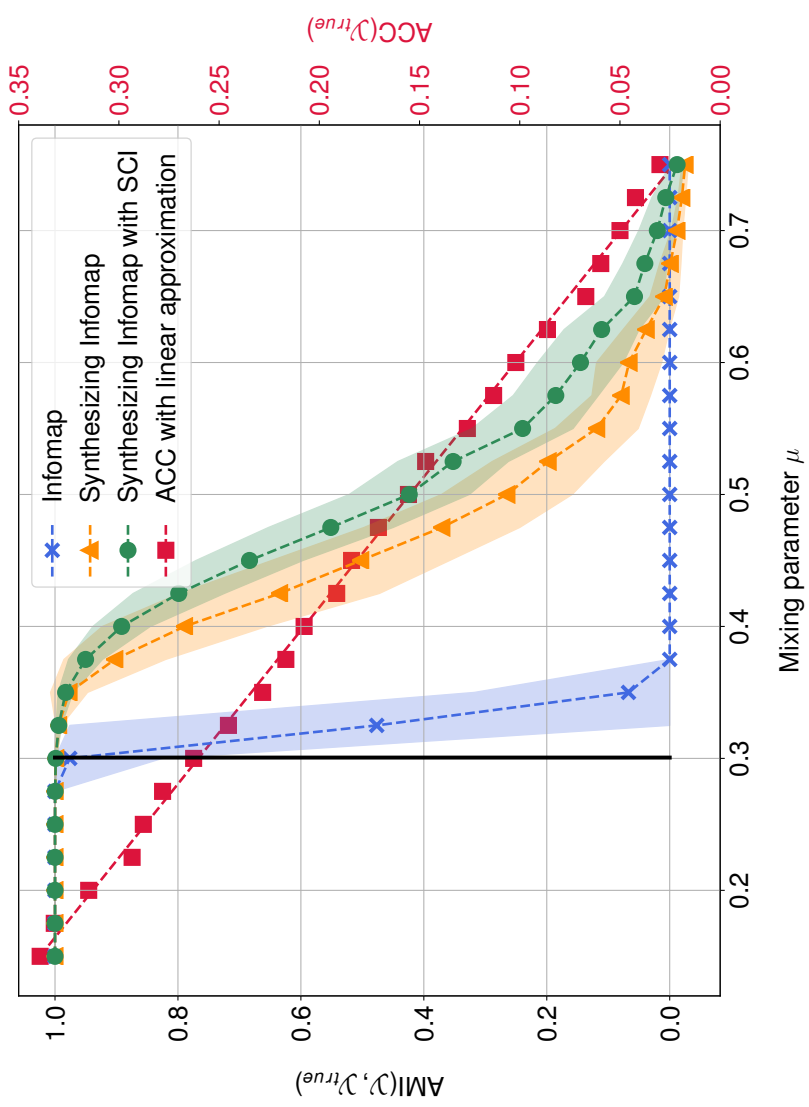


Figure 5.15.: Comparison of Infomap and synthesizing Infomap variants. The plot shows mean AMI and ACC scores achieved on a minimum of 77 different LFR benchmark graphs with $N = 2400$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.30$ corresponds to an ACC value of 0.26 in the linear approximation.

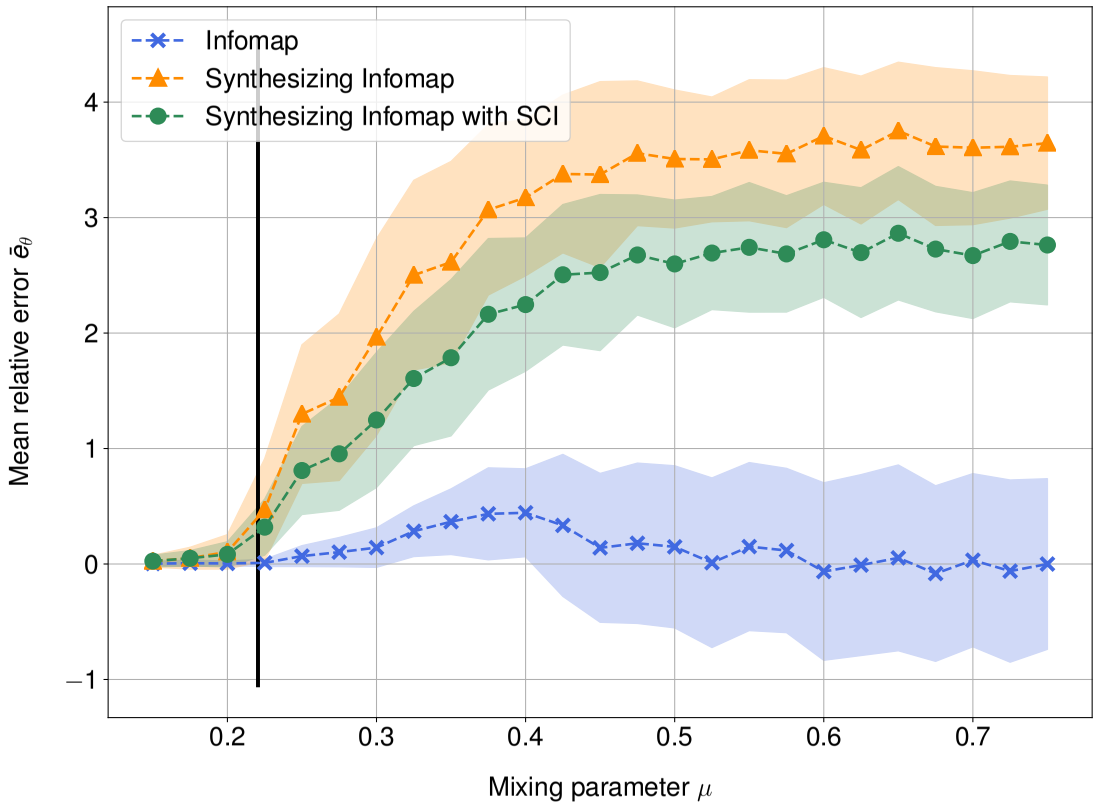


Figure 5.16.: Comparison of Infomap and synthesizing Infomap variants. The plot shows MRENDC scores achieved on 100 different LFR benchmark graphs with $N = 300$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.22$ corresponds to an ACC value of 0.26 in the linear approximation.

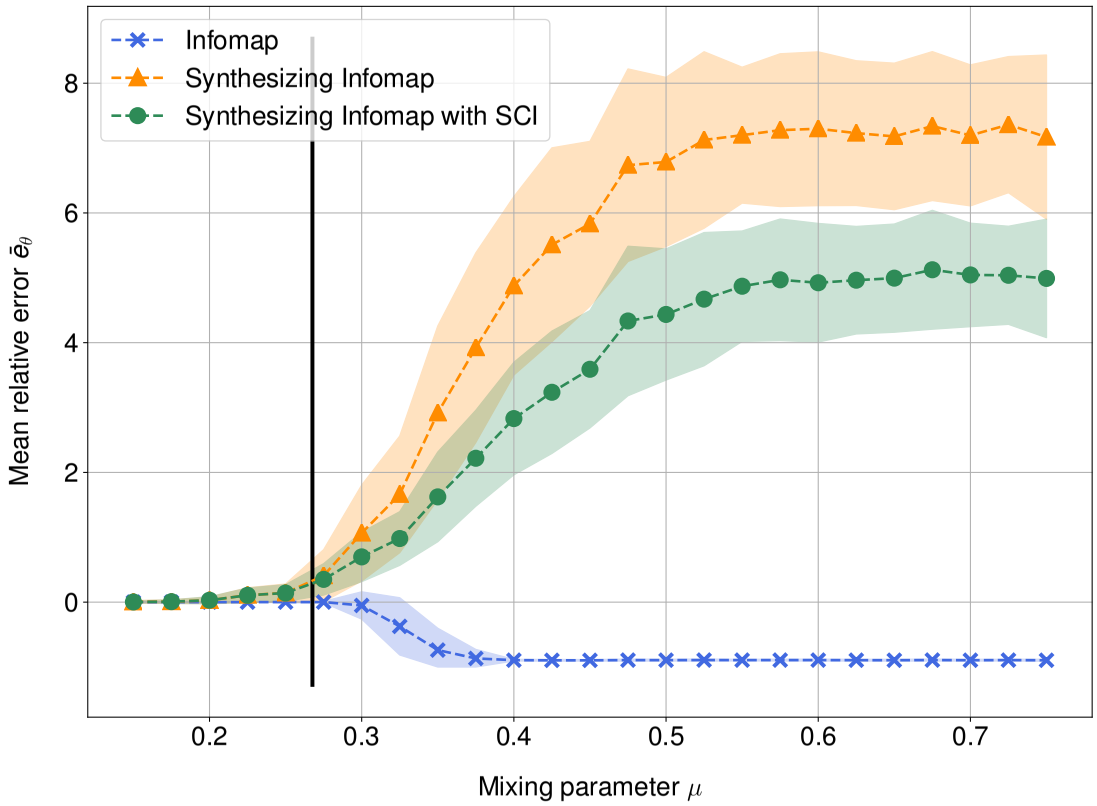


Figure 5.17.: Comparison of Infomap and synthesizing Infomap variants. The plot shows MRENDC scores achieved on a minimum of 99 different LFR benchmark graphs with $N = 600$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.27$ corresponds to an ACC value of 0.26 in the linear approximation.

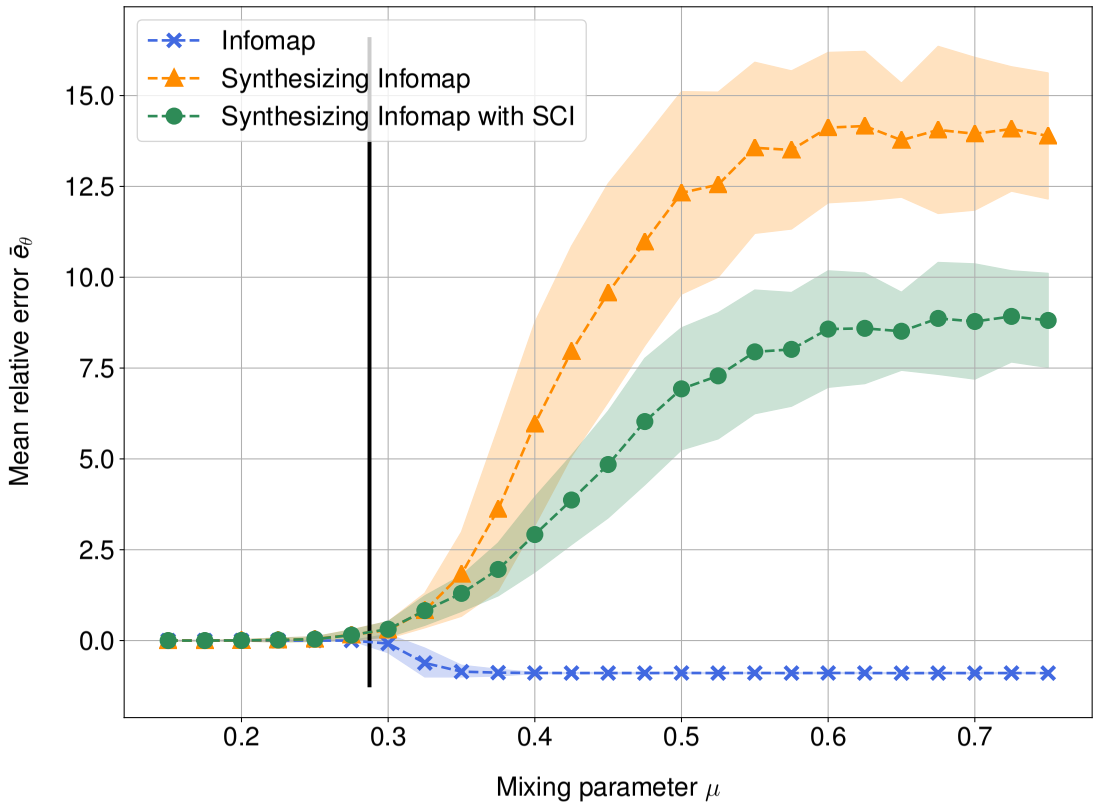


Figure 5.18.: Comparison of Infomap and synthesizing Infomap variants. The plot shows MRENDC scores achieved on a minimum of 95 different LFR benchmark graphs with $N = 1200$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.29$ corresponds to an ACC value of 0.26 in the linear approximation.

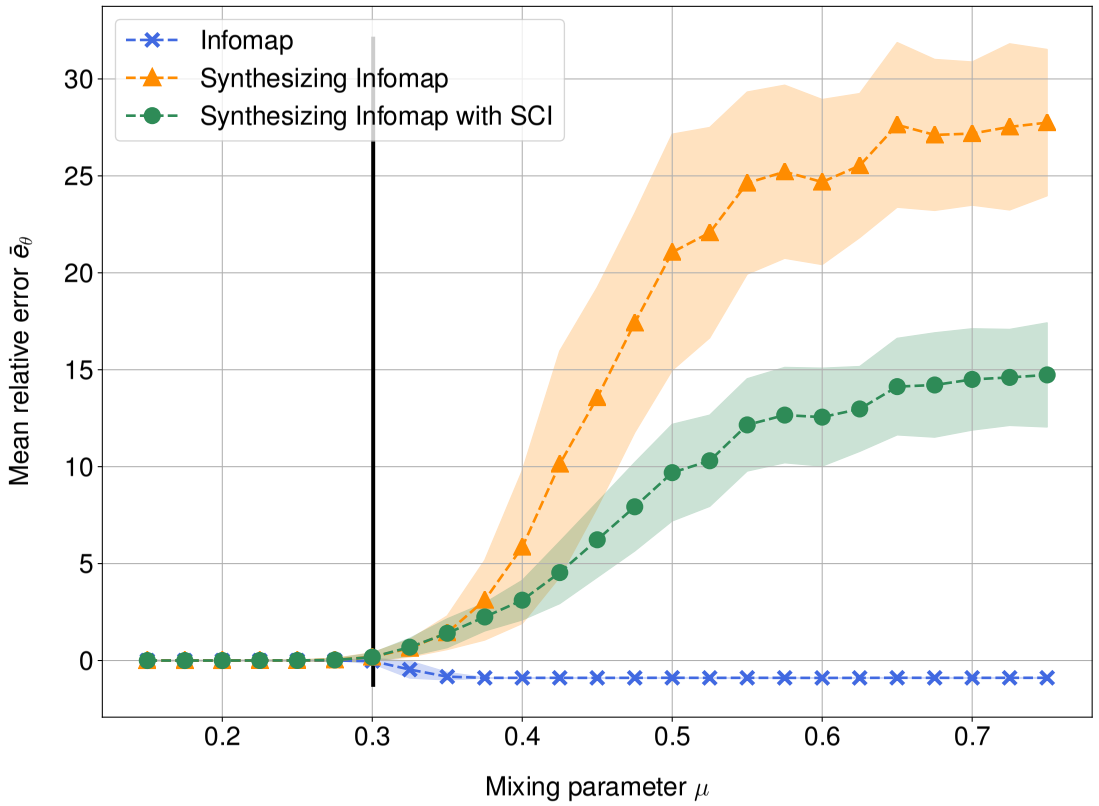


Figure 5.19.: Comparison of Infomap and synthesizing Infomap variants. The plot shows MRENDC scores achieved on a minimum of 77 different LFR benchmark graphs with $N = 2400$ nodes as a function of the mixing parameter μ. We depict variation in the results as a margin of one standard deviation around the mean scores. The black line at $\mu = 0.30$ corresponds to an ACC value of 0.26 in the linear approximation.

5.2.3. AMI as a Function of the Network Size

The results for AMI performances as a function of the network size are shown in Figures 5.20, 5.21 and 5.22. Infomap correctly identifies communities for sufficiently small values of μ and larger network sizes (see Figure 5.20) and it fails to do so for increasing values of μ and larger network sizes as in Figure 5.22. Here, a behavioral change occurs within a critical region of the mixing parameter.

Regarding synthesizing Infomap variants, one can observe that regardless of the mixing parameter the AMI performance rises with increasing network size. Although counterintuitive at first, the behavior can be explained by the average community clustering. As ACC increases with the network size it resembles the AMI behavior very well. Again, synthesizing Infomap consistently performs better that its standard variant.

5.2.4. Visual Comparison

In the following we display multiple partitions of a sample LFR benchmark graph with $N = 600$ nodes and a mixing parameter of $\mu = 0.35$. Figure 5.23 shows the nine ground truth modules as given by the benchmark generation. The resulting partition for Infomap is visualized in Figure 5.24. As can be seen, Infomap yields a single community including all nodes. Partitions for synthesizing Infomap with and without SCI are given in Figure 5.25 and Figure 5.26 respectively, where we aggregate nodes that are not members of the nine largest detected modules into a common residual module. For this sample graph, synthesizing Infomap reasonably uncovers seven out of the nine ground truth communities whereas the variant with SCI detects eight out of nine.

Interestingly, smaller communities are uncovered with greater accuracy than larger communities, where at first this behavior seems counter-intuitive. Nevertheless, we explain the phenomenon in the following way. Consider an undirected graph $\mathcal{G}$ with only two modules $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2\}$ where $|\mathcal{Y}_1| = 10$,

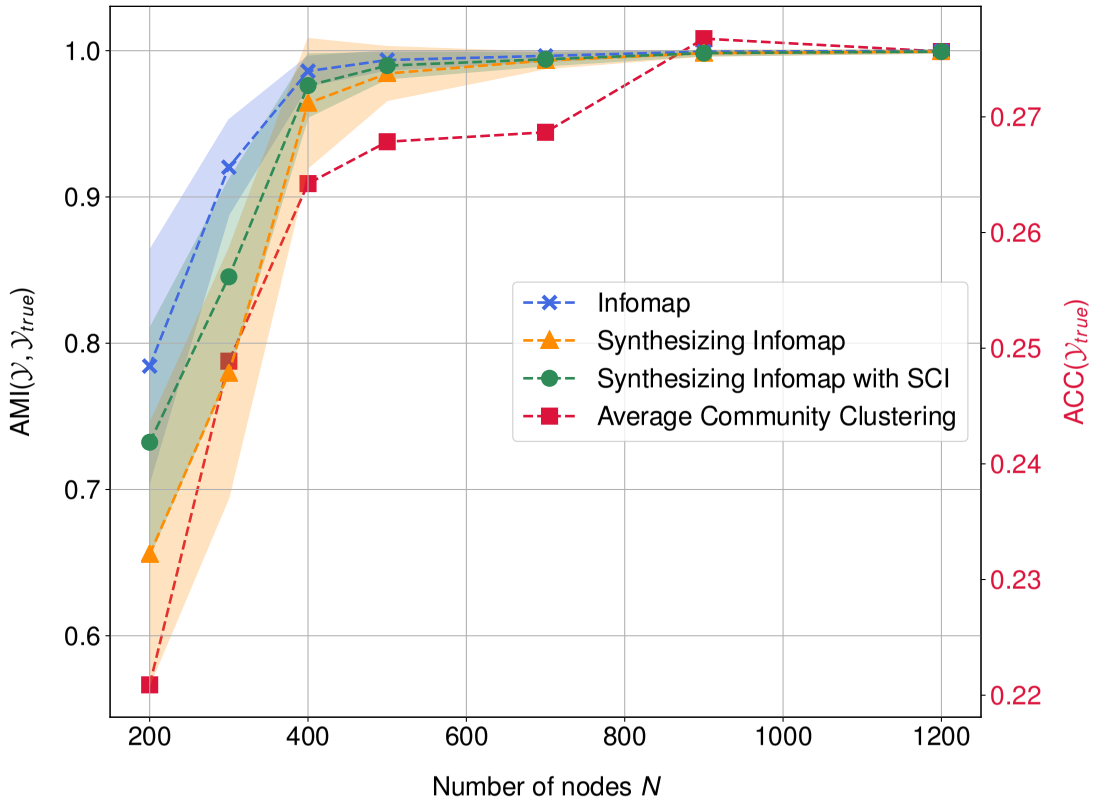


Figure 5.20.: Comparison of Infomap and synthesizing Infomap variants. The plot shows AMI and ACC scores achieved on a minimum of 99 different LFR benchmark graphs with a fixed mixing parameter $\mu = 0.25$ nodes as a function of the network size N. We depict variation in the results as a margin of one standard deviation around the mean scores.

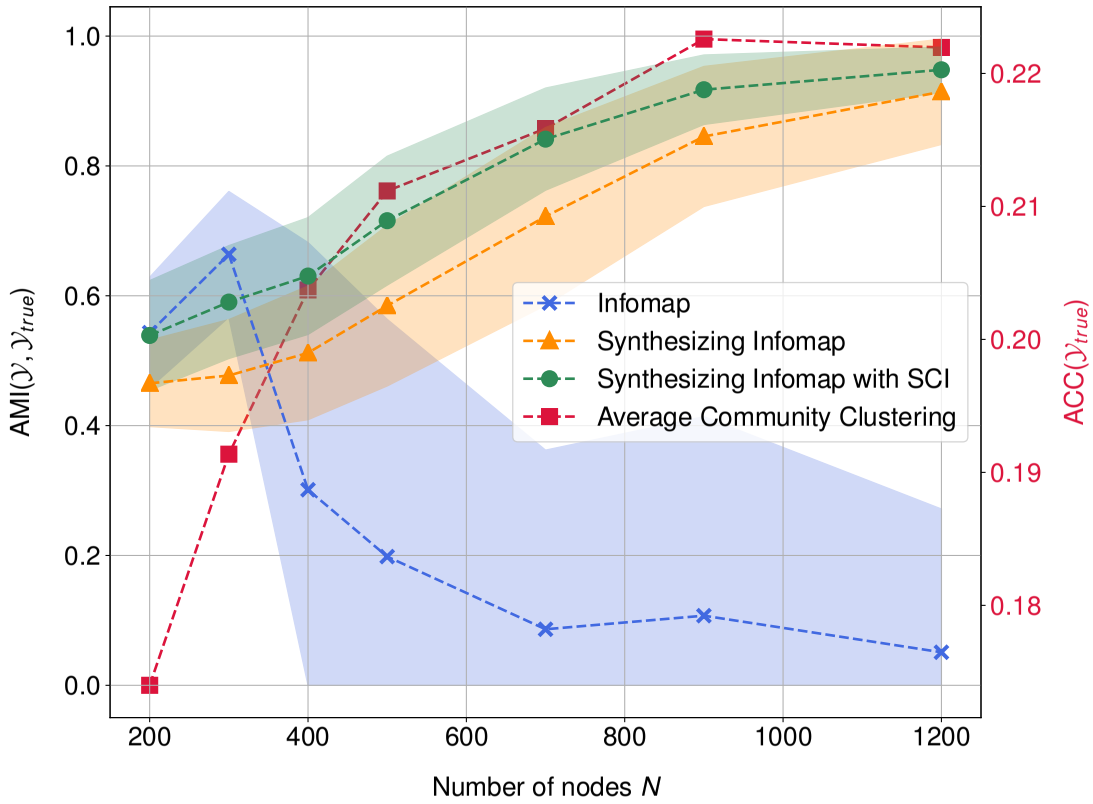


Figure 5.21.: Comparison of Infomap and synthesizing Infomap variants. The plot shows AMI and ACC scores achieved on a minimum of 96 different LFR benchmark graphs with a fixed mixing parameter $\mu = 0.35$ nodes as a function of the network size N. We depict variation in the results as a margin of one standard deviation around the mean scores.

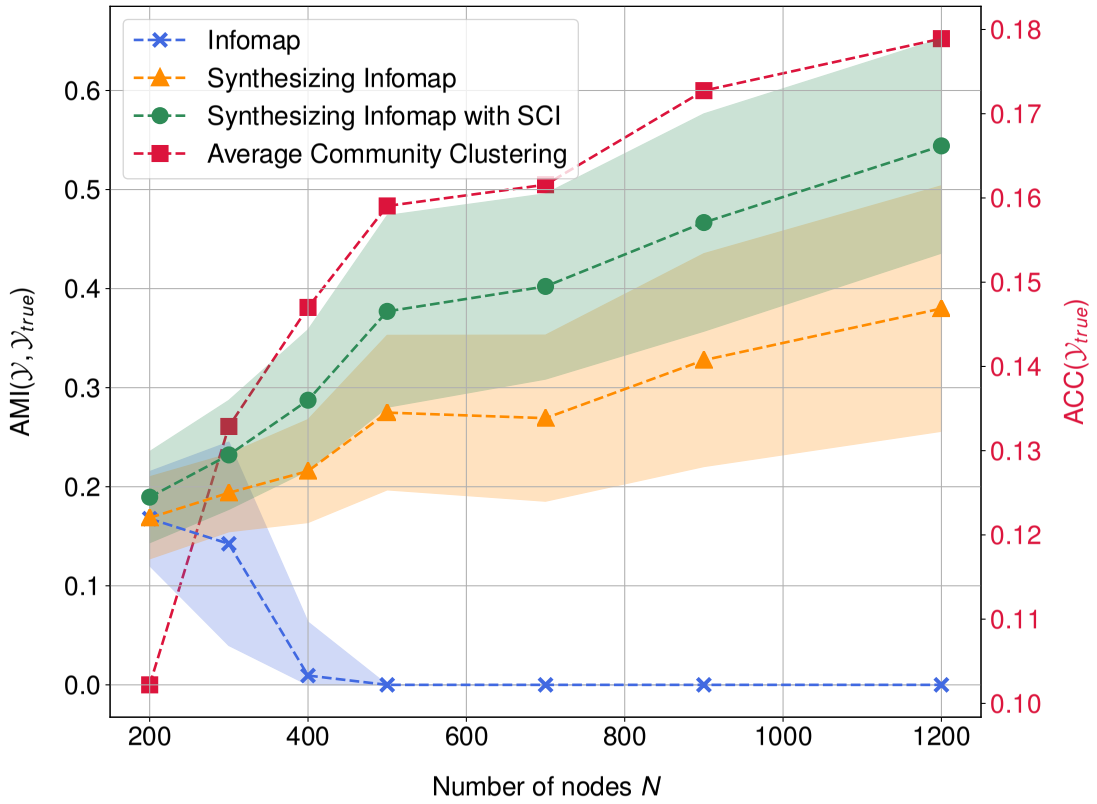


Figure 5.22.: Comparison of Infomap and synthesizing Infomap variants. The plot shows AMI and ACC scores achieved on a minimum of 99 different LFR benchmark graphs with a fixed mixing parameter $\mu = 0.45$ nodes as a function of the network size N. We depict variation in the results as a margin of one standard deviation around the mean scores.

$|\mathcal{Y}_2| = 20$ and $\mathcal{Y}_1$ is a maximal clique. Further, assume a fixed mixing parameter $\mu = 0.1$ *for all nodes*, corresponding to a strong community definition according to [4]. The internal degree of any node $\alpha \in \mathcal{Y}_1$ is $k^{int} = |\mathcal{Y}_1| - 1 = 9$. Hence, the total number of outgoing links for $\mathcal{Y}_1$ can be computed from Equations (2.11)–(2.13) as

$$k^{ext}_{\mathcal{Y}_1} = \sum_{\alpha \in \mathcal{Y}_1} k^{ext}_{\alpha} = \frac{\mu}{1-\mu} \sum_{\alpha \in \mathcal{Y}_1} k^{int}_{\alpha} = 10. \tag{5.2}$$

Since there are only two communities in our example, it holds that

$$k^{ext}_{\mathcal{Y}_2} = k^{ext}_{\mathcal{Y}_1} \tag{5.3}$$

and therefore

$$k^{int}_{\mathcal{Y}_2} = \frac{1-\mu}{\mu} k^{ext}_{\mathcal{Y}_2} = \frac{1-\mu}{\mu} k^{ext}_{\mathcal{Y}_1} = k^{int}_{\mathcal{Y}_1} = 90, \tag{5.4}$$

i.e. the total internal degree of the smaller community $\mathcal{Y}_1$ limits the total internal degree of the larger community $\mathcal{Y}_2$. Whereas in our example $\mathcal{Y}_1$ is already fully connected internally, $\mathcal{Y}_2$ only has 45 out of 190 possible internal edges and is thus less well connected internally. For the community clustering coefficients it follows that $\bar{c}_{\mathcal{G}}(\mathcal{Y}_2) < \bar{c}_{\mathcal{G}}(\mathcal{Y}_1) = 1$ as long as $\mathcal{Y}_2$ is larger than $\mathcal{Y}_1$. We note that the global mixing parameter cannot be achieved by all nodes in $\mathcal{Y}_2$ but only as a community average.

Transferring these observations to the LFR benchmark graphs, in order to achieve a global mixing parameter, larger communities have to "adapt" their internal degrees to match the internal degrees of the smaller communities in the network. Therefore, smaller communities will be more strongly connected internally than larger communities and thus *be easier to detect*. Hence we argue that, although the mixing parameter is decisive of the *existence* of a community, its *discoverability* rather depends on the actual connectedness within the community, which can be measured using the community clustering coefficient.

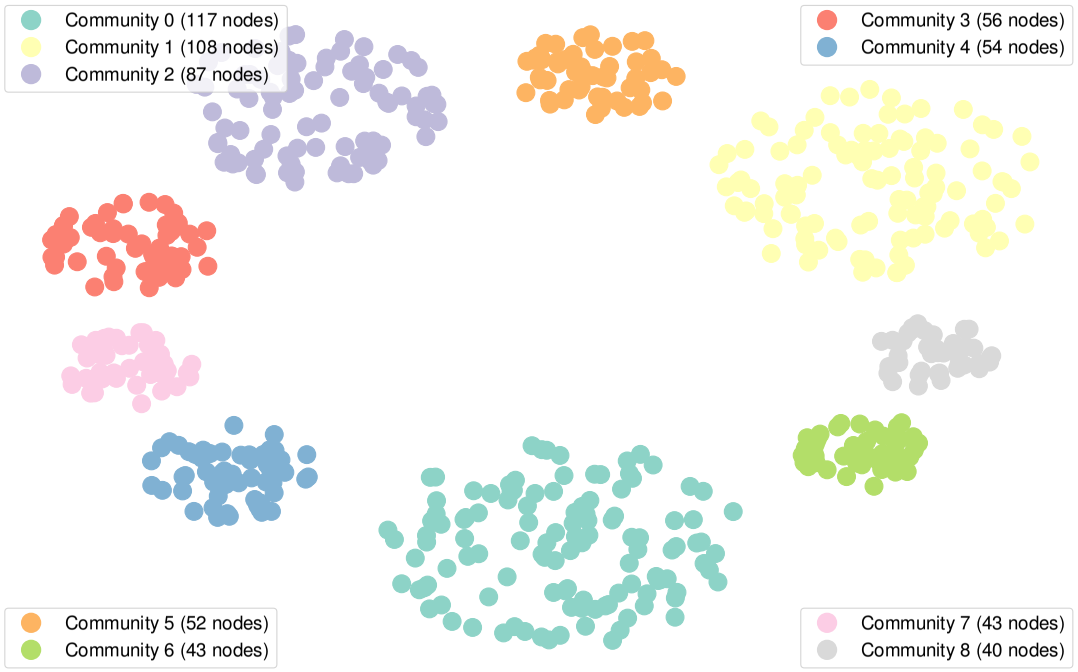


Figure 5.23.: Sample LFR benchmark graph with ground truth communities. The graph has $N = 600$ nodes and a mixing parameter of $\mu = 0.35$. Nodes with the same color belong to the same community. We do not display any edges to avoid visual overloading.

Figure 5.24.: Sample LFR benchmark graph with communities as detected by Infomap. The graph has $N = 600$ nodes and a mixing parameter of $\mu = 0.35$. Nodes with the same color belong to the same community. We do not display any edges to avoid visual overloading.

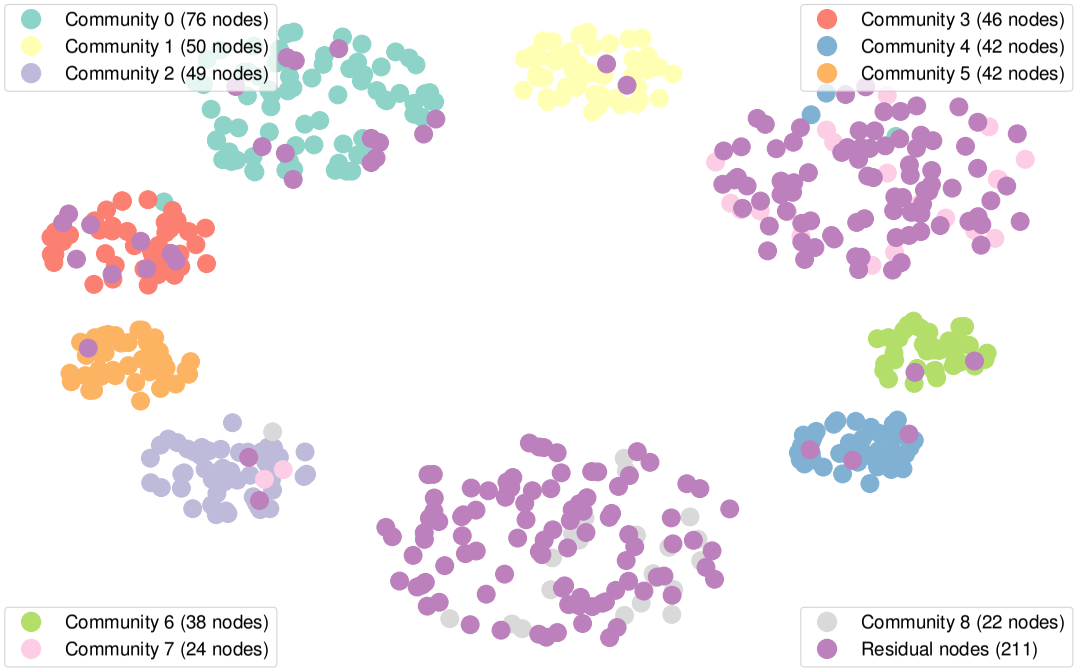


Figure 5.25.: Sample LFR benchmark graph with communities as detected by synthesizing Infomap. The graph has $N = 600$ nodes and a mixing parameter of $\mu = 0.35$. For the eight largest communities, nodes with the same color belong to the same community. The residual nodes (not in the eight largest modules) share a common coloring. We do not display any edges to avoid visual overloading.

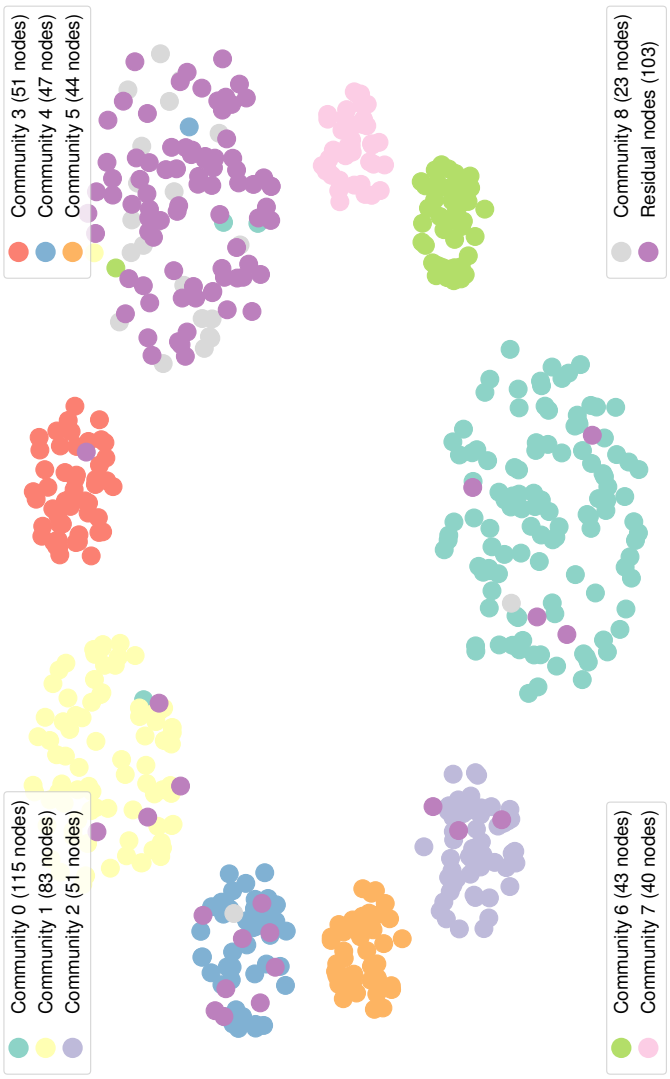


Figure 5.26.: Sample LFR benchmark graph with communities as detected by synthesizing Infomap with SCI. The graph has $N = 600$ nodes and a mixing parameter of $\mu = 0.35$. For the eight largest communities, nodes with the same color belong to the same community. The residual nodes (not in the eight largest modules) share a common coloring. We do not display any edges to avoid visual overloading.

	Original network		Reduced network		
Network	# nodes	# edges	# nodes	# edges	$\overline{N_c}$
com-Amazon	334,863	925,872	16,716	48,739	13
com-DBLP	317,080	1,049,866	93,432	335,520	22
com-LiveJournal	3,997,962	34,681,189	84,438	1,521,988	28
com-Youtube	1,134,890	2,987,624	39,841	224,235	15

Table 5.1.: Properties of the examined real-world networks. We report network sizes for the original and the reduced networks. Additionally, we provide the average community size $\overline{N_c}$ of the reduced networks.

5.3. Real-World Networks

We compare Infomap and synthesizing Infomap on a selection of real-world networks taken from the SNAP network dataset collection [31]. Specifically, we limit our experiments to networks with given ground-truth communities, originally provided by the work in [42]. Since the ground truth communities are overlapping, we compare the methods using ONMI, Omega index and RENDC. We compute these metrics with respect to the top 5000 communities (see [42] for ranking details) on reduced networks. We reduce a given network by removing all nodes from the network that are not included in any of the top 5000 communities. Network properties are given in Table 5.1. The results of our experiments are summarized in Table 5.2.

As can be seen, the results of Infomap and synthesizing Infomap are close to each other on all tested networks. In fact, synthesizing Infomap scores equally or slightly better regarding the Omega Index, and performs worse with respect to ONMI only on the *com-Youtube* network. RENDC values stay below 100% in all test cases. These results indicate that synthesizing Infomap is a suitable method to uncover communities on real-world networks.

	Infomap (reduced network)			Synthesizing Infomap (reduced network)		
Network	RENDC	ONMI	OI	RENDC	ONMI	OI
com-Amazon	−0.72	0.89	0.35	−0.72	0.90	0.37
com-DBLP	0.21	0.47	0.01	0.19	0.47	0.01
com-LiveJournal	−0.32	0.88	0.60	−0.35	0.88	0.66
com-Youtube	−0.50	0.52	0.19	−0.39	0.43	0.21

Table 5.2.: Comparison of Infomap and synthesizing Infomap results achieved on reduced real-world networks. We report RENDC, overlapping mutual information (ONMI) and Omega index (OI) values with respect to the top 5000 communities.

6. Conclusion and Future Work

In this final chapter we summarize the main findings of this thesis in Section 6.1 and give an outlook on potential future extensions and research questions in Section 6.2.

6.1. Summary

This thesis presents synthesizing Infomap, which is a novel, Kullback-Leibler divergence-based method for community detection. Our work comprises the following main points.

We integrated the synthesizing Infomap objective into the existing Infomap software package, considering issues regarding numerical stability and making the extended functionality available via the command line interface. The modified framework is publicly accessible in our repository at `https://github.com/chritoth/infomap`. Hence, we provide a ready-to-use implementation of the synthesizing Infomap objective for the further evaluation of this approach and any potential application as a community detection method to real-world networks.

To this end, we extensively evaluated our implementation in experiments on artificially generated LFR benchmark graphs. Our results show that our method performs on par with Infomap on networks with strong community structure and outperform Infomap as the community structure of the networks weakens. Furthermore, we observe that the performance of our method is not dictated by the network size as such, but rather on the

internal "connectedness" of the communities in the network. We argue that the mixing parameter serves as a measure for the existence of communities in a network, whereas our defined average community clustering governs their detectability. On a selection of real-world networks we perform on par with Infomap, proving that our method is applicable to real-world tasks.

Additionally, we provide fundamental bounds of the synthesizing Infomap objective and we explore basic behavior of the optimization procedure by looking a set of prototypical toy graphs. Our findings indicate that the synthesizing Infomap objective is characterized by local minima in the form of maximal independent sets, depending on the network structure. However, as the search algorithm does not assemble independent sets by design, this does not impact our application to community detection as discussed in this thesis. To mitigate initialization issues connected to independent sets, an alternative initialization strategy based on spectral clustering was proposed.

Lastly, we provide a complete derivation of synthesizing Infomap and position our method within the related scientific literature.

6.2. Future Work

The promising performance of synthesizing Infomap in our experiments calls for further evaluation, especially regarding the applicability on real-world networks. However, as many real-world networks exhibit highly overlapping or hierarchical community structure, an generalized formulation of synthesizing Infomap to serve such network properties is a highly relevant future extension of the method. Additionally, exploring a synthetic random walk model with higher-order Markov dynamics, similar to [14], may yield an interesting functional expansion. Furthermore, we have yet to further study and exploit local optimality of synthesizing Infomap regarding maximal independent sets as an application to multipartite graphs. The design of an alternative search algorithm for such a scenario could pose an interesting field of work.

In order to evaluate community detection methods on artificially generated benchmark graphs the LFR benchmark is widely used within the scientific literature. In our experiments, we observed that the number of nodes in the network as a primary parameter to the generation routine limits the ability to generate benchmark networks with specific community properties. In more detail, fundamental network parameters as the number of communities and their sizes cannot be fixed but are a result of network size and distribution parameters. We argue that when testing community detection algorithms what we want to specify in a benchmark graph is exactly the fundamental parametrization of communities in the network, as this would improve the variance of simulation results and hence enable a more objective and efficient comparison of individual community detection methods. In our opinion, the design of a benchmark providing such features could would be a worthwhile endeavor to the scientific community.

Appendix

Appendix A.

Implementation Details

In this appendix we document different aspects of our synthesizing Infomap implementation. Since we integrated the synthesizing Infomap objective into the existing Infomap framework [8], we first give a compact overview of the Infomap software package and its functionality in Section A.1. Secondly, we summarize the integration process of our synthesizing Infomap in to the existing Infomap framework in Section A.2. Lastly, we explain implementation strategies regarding the numerical stability of our objective function in Section A.3. The modified framework used in the course of this work can be found at `https://github.com/chritoth/infomap`.

A.1. The Infomap Software Package in a Nutshell

Infomap is a well-known software package for community detection, able to analyze a broad range of different network types (directed vs. undirected, weighted vs. unweighted) and community structures (two-level vs. multi-level hierarchies, overlapping vs. non-overlapping communities). Infomap is implemented in the C++ programming language and its functionality is accessible via a command line interface (CLI). The CLI provides numerous options, e.g. for the mode of operation and optimization hyperparameter tweaking. In our experiments we setup Infomap for undirected

(`--undirected`) networks with a two-level hierarchy (`--two-level`). We do not specify any optimization hyper-parameters using the CLI.

As discussed in Section 2.3, Infomap tries to extract the community structure of a given network by finding a suitable partition that minimizes the map equation (see Equation (2.16)). Since finding the optimal partition is a hard problem, Infomap employs a sophisticated search algorithm to find good solutions. In the following we will briefly describe the two-level search procedure.

The Infomap search algorithm can be described by a core algorithm and two tuning mechanisms as follows.

- *Core Algorithm:* In the core algorithm neighboring modules are repeatedly joined to form a reasonable partition, serving as a starting point for the tuning mechanisms. Initially, each node serves as its own module. In a first iteration each node repeatedly joins with a neighboring modules such that the resulting decrease of the map equation is maximal. In case no improvement is possible by moving a specific node, it stays in its module. This process is run until no further improvements are achieved. The resulting modules serve as nodes in the next iteration and the merging process as described above is repeated. This hierarchical clustering process is repeated until no further decrease of the map equation is achieved.
- *Coarse Tuning:* Assume a partition $\mathcal{Y}$ provided by the core algorithm. In the coarse tuning procedure each module $\mathcal{Y}_i \in \mathcal{Y}$ within this partition is considered an independent network. Applying the core algorithm to the modules yields a partition $\mathcal{Z}$ into sub-modules $\mathcal{Z}_i$ for each module $\mathcal{Y}_i$. Then the core algorithm is re-applied to the original network with partition $\mathcal{Y}$ whereas now the sub-modules $\mathcal{Z}_i$ are movable between the modules $\mathcal{Y}_i$.
- *Fine Tuning:* Assume a partition $\mathcal{Y}$ provided by the core algorithm. Then, in the fine tuning procedure, each node $\alpha \in \mathcal{X}$ of the original network is considered to be a separate module. The core algorithm is reapplied to the original network with partition $\mathcal{Y}$ whereas now single nodes are movable between the modules $\mathcal{Y}_i \in \mathcal{Y}$.

Coarse and fine tuning procedures are repeated sequentially until no further improvements are achieved, resulting in the (approximately) optimal partition with respect to the map equation.

A.2. Integration of the Synthesizing Infomap Objective

We integrated the synthesizing Infomap objective $\mathcal{J}(\mathcal{Y})$ (see Equation (3.7)) into the existing Infomap software package, motivated by two main reasons. First, Infomap has already implemented and tested input/output mechanisms, a CLI, and a practical search algorithm. By reusing these features the lengthy process of implementing a whole software package from scratch could be circumvented in favor of experiments. Secondly, the common optimization algorithm enables better comparability of the Infomap and synthesizing Infomap objectives. The synthesizing Infomap objective is available via the CLI option `--altmap`.

Infomap provides the possibility of passing an initial clustering for a given network via the CLI option `--cluster-data <cluster-file>`. We use this option in our experiments to pass a pre-computed spectral clustering to Infomap, as we did not implement a spectral clustering initialization into the Infomap framework itself.

As Infomap minimizes the map equation and synthesizing Infomap poses a maximization problem, we minimize the negative synthesizing Infomap objective $-\mathcal{J}(\mathcal{Y})$ in the implementation. Since original Infomap already computes the marginal and joint probabilities of the process on the network modules as described in Section 2.1.3, it is convenient to reuse these quantities. Indeed, we can express the synthesizing Infomap objective solely in terms of marginal and joint probabilities by rewriting Equation (3.7) as

Filename	Modifications
Config.h	add `--altmap` option
Infomap.cpp	add `--altmap` option
InfomapBase.cpp	correct compression rate for negative cost
InfomapGreedyCommon.h	implement synthesizing Infomap objective
InfomapGreedySpecialized.h	implement synthesizing Infomap objective

Table A.1.: Files modified in the integration process including the respective changes.

follows.

$$\mathcal{J}(\mathcal{Y}) = \sum_{i \in \mathcal{Y}} p_i D(p_{i \to i} \| p_i) \tag{A.1a}$$

$$= \sum_{i \in \mathcal{Y}} p_i p_{i \to i} \log \frac{p_{i \to i}}{p_i} + p_i p_{i \not\to i} \log \frac{p_{i \not\to i}}{1 - p_i} \tag{A.1b}$$

$$= \sum_{i \in \mathcal{Y}} p_{i,i} \log \frac{p_{i,i}}{p_i^2} + p_{i,\neg i} \log \frac{p_{i,\neg i}}{p_i p_{\neg i}} \tag{A.1c}$$

$$= \sum_{i \in \mathcal{Y}} p_{i,i} \log p_{i,i} - 2 \cdot p_{i,i} \log p_i + p_{i,\neg i} \log p_{i,\neg i} - p_{i,\neg i} \log p_i p_{\neg i} \tag{A.1d}$$

Modifications to any files in the original Infomap framework are listed in Table A.1.

A.3. Implementation Strategies regarding Numerical Stability

From the implemented form of the synthesizing Infomap objective in Equation (A.1d) it is apparent that numerical issues may arise for any of the logarithm arguments vanishing. Looking at an arbitrary module i we consider the three possible cases for the marginal p_i as follows.

1. $\mathbf{p_i = 1}$: For the marginal probability to be $p_i = 1$ a random walker must spend all the time moving withing module i. Hence, also the joint probability $p_{i,i} = 1$. It is easy to see, that for $p_{i,i} = p_i p_{i\to i} = 1$ also $p_i = 1$. Therefore, $p_i = 1 \iff p_{i,i} = 1$. Moreover, $p_i = 1 \wedge p_{i,i} = 1 \implies p_{i,\neg i} = 0$. A single module contribution to Equation A.1d therefore is evaluated as

$$\begin{aligned}\mathcal{J}(\mathcal{Y}_i) &= p_{i,i}\log p_{i,i} & -2\cdot p_{i,i}\log p_i & & +p_{i,\neg i}\log p_{i,\neg i} & & -p_{i,\neg i}\log p_i p_{\neg i}\\ &= 1\log 1 & -2\cdot 1\log 1 & & +0\log 0 & & -0\log(1\cdot 0)\\ &= 0\end{aligned}$$

   where we set $0\log 0 = \lim_{x\to 0} x\log x = 0$.
2. $\mathbf{p_i = 0}$: For $p_i = 0$ it trivially follows that $p_{i,i} = p_{i,\neg i} = 0$, i.e. a random walker never visits module i. Thus, the contribution of module i is

$$\begin{aligned}\mathcal{J}(\mathcal{Y}_i) &= p_{i,i}\log p_{i,i} & -2\cdot p_{i,i}\log p_i & & +p_{i,\neg i}\log p_{i,\neg i} & & -p_{i,\neg i}\log p_i p_{\neg i}\\ &= 0\log 0 & -2\cdot 0\log 0 & & +0\log 0 & & -0\log(0\cdot 1)\\ &= 0.\end{aligned}$$

3. $\mathbf{0 < p_i < 1}$: For $0 < p_i < 1$ the only critical terms ins the objective are $p_{i,i}\log p_{i,i}$ and $p_{i,\neg i}\log p_{i,\neg i}$. However, as $p_{i,i} \to 0$ (module i is an independent set) also $p_{i,i}\log p_{i,i} \to 0$ and similarly as $p_{i,\neg i} \to 0$ (module i is absorbing; not feasible for $p_i < 1$) $p_{i,\neg i}\log p_{i,\neg i} \to 0$

The considerations above have been incorporated into the implementation.

Appendix B.

Connections of Synthesizing Infomap to Previous Work

In this section we give several connections our proposed synthesizing Infomap objective to related scientific areas. In Section B.1 we present a further relaxation of Problem 1 and list its connection to Markov aggregation and random walk-based clustering. In Section B.2 locate synthesizing Infomap within the scientific literature around Markov aggregation. Finally, in Section B.3 we relate our approach to stochastic block modeling.

B.1. A Relaxation and its Connection to Related Work

We present the following relaxation for the sake of completeness, although it does not affect our experimental sections. However, this relaxation allows to bring our work in context with other existing information-theoretic approaches to community detection and to the aggregation of Markov chains.

We relax the optimization problem (3.2) by admitting not only one multinomial distributions over modules $[u_i]_{i\in\mathcal{Y}}$, but one distribution $[v^i_j]_{j\in\mathcal{Y}\setminus\{i\}}$

for each module $\mathcal{Y}_i$. Here, v_j^i denotes the probability that, if the random walker leaves module $\mathcal{Y}_i$, the next visited module will be $\mathcal{Y}_j$. The transition probability matrix $Q^{\mathcal{Y}} = [q_{\alpha\to\beta}]_{\alpha,\beta\in\mathcal{X}}$ characterizing the relaxed random walker is hence given by

$$q_{\alpha\to\beta} = \begin{cases} r_\beta^{m(\beta)} \quad (1 - s_{m(\alpha)}) & \text{for} \quad m(\alpha) = m(\beta), \\ r_\beta^{m(\beta)} \quad s_{m(\alpha)} \quad v_{m(\beta)}^{m(\alpha)} & \text{for} \quad m(\alpha) \neq m(\beta). \end{cases} \tag{B.1}$$

We can now state the relaxed problem accordingly.

Problem 3. Let $\mathcal{G} = (\mathcal{X}, E, W)$ be a graph and let $P \sim W$ be a transition probability matrix derived from the weight matrix W. Moreover, let $Q^{\mathcal{Y}}$ denote a transition probability matrix dependent on some partition $\mathcal{Y}$ as described in (3.1). We then want to find an optimal distribution $\mathcal{Y}^*$, subject to

$$\mathcal{Y}^* \in \arg\min_{\mathcal{Y}} \left[\min_{\{[r_\alpha^i]_{\alpha\in i},\, s_i,\, [v_j^i]_{j\in\mathcal{Y}\setminus\{i\}}\}_{i\in\mathcal{Y}}} \overline{D}(P\|Q^{\mathcal{Y}}) \right]. \tag{B.2}$$

In Appendix C.3 we prove that the minimization with respect to the parameters of $Q^{\mathcal{Y}}$ yields the following proposition.

Proposition 1. *Let $\{X_t\}$ be a stationary Markov chain with transition probability matrix P derived from the weight matrix W of the graph $\mathcal{G}$ and let $Q^{\mathcal{Y}}$ be a transition probability matrix of the same size parameterized as in (3.1). Additionally, let $Y_t = m(X_t)$. Then, for every partition $\mathcal{Y}$, it holds that*

$$\min_{\{[r_\alpha^i]_{\alpha\in i},\, s_i,\, [v_j^i]_{j\in\mathcal{Y}\setminus\{i\}}\}_{i\in\mathcal{Y}}} \overline{D}(P\|Q^{\mathcal{Y}}) = I(X_t; X_{t-1}) - I(Y_t; Y_{t-1}). \tag{B.3}$$

The alternative objective for the relaxation according to Problem 3 therefore reads

$$\widetilde{\mathcal{J}}(\mathcal{Y}) := I(Y_t; Y_{t-1}). \tag{B.4}$$

By the data processing inequality, $I(Y_t; Y_{t-1})$ increases under refinements of the partition. Indeed, the trivial partition in which every state is a module maximizes $I(Y_t; Y_{t-1})$ to $I(X_t; X_{t-1})$. Hence, this objective function cannot be used to determine the number of communities (other than by using an "elbow rule" as in [43, Fig. 8]). This objective function has furthermore appeared in Markov aggregation (see Section B.2) and random walk-based clustering [44].

B.2. Relationship with Cost Functions for Markov Aggregation

Markov aggregation is concerned with replacing a Markov chain $\{X_t\}$ on a large state space $\mathcal{X}$ by another Markov chain $\{Y_t\}$ on a significantly smaller state space $\mathcal{Y}$. Several authors have proposed using the Kullback-Leibler divergence rate as a cost function for Markov aggregation [43, 45, 46, 47]. Specifically, for the task of comparing two Markov chains with different state spaces, the authors of [43] proposed "lifting" the aggregated Markov chain $\{Y_t\}$ to the original state space $\mathcal{X}$. Let $P = [p_{\alpha\to\beta}]_{\alpha,\beta\in\mathcal{X}}$ and $Q = [q_{i\to j}]_{i,j\in\mathcal{Y}}$ denote the transition probability matrices of the Markov chains $\{X_t\}$ and $\{Y_t\}$ respectively. Then the authors defined a Markov chain $\{X_t^\bullet\}$ on $\mathcal{X}$ with transition probability matrix $P^\bullet = [p^\bullet_{\alpha\to\beta}]_{\alpha,\beta\in\mathcal{X}}$ defined by

$$p^\bullet_{\alpha\to\beta} = \frac{p_\alpha}{\sum_{\alpha\in\mathcal{Y}_{m(\alpha)}} p_{\alpha^\bullet}} q_{m(\alpha)\to m(\beta)} \tag{B.5}$$

and thus formulated the Markov aggregation problem as one of finding an aggregation function $m\colon \mathcal{X} \to \mathcal{Y}$ that minimizes $\overline{D}(P\|P^\bullet)$. If the transition probability matrix Q is equivalent to the one-step conditional probabilities

of the process $\{m(X_t)\}$, then one can show that minimizing $\overline{D}(P\|P^\bullet)$ is equivalent to maximizing $I(Y_t;Y_{t-1})$. In other words, our relaxed utility function for community detection from Proposition 1 is equivalent to the cost function for Markov aggregation proposed by [43].

The authors of [43] have argued that maximizing $I(Y_t;Y_{t-1})$ aims at finding modules such that the process $\{Y_t\}$ on the modules is "predictable". This has been made precise in [48, Sec. IV.B] by connecting this objective function with the error probability of estimating the next module to which the random walker moves. Indeed, [43] reports success mainly for Markov chains that are "nearly completely decomposable", i.e. the transition graph of such Markov chains displays a clear community structure. It has further been shown that a relaxation of maximizing $I(Y_t;Y_{t-1})$ for only two modules is connected to the spectral theory of Markov chains [43, Sec. IV.B]. The authors further showed that the transition probability matrix Q minimizing the problem in Proposition 1 is such that $q_\beta = p_\beta$, i.e. the processes $\{X_t^\bullet\}$ and $\{X_t\}$ have the same invariant distribution [43, Th. 2]. Finally, the authors of [47] give yet another justification for maximizing $I(Y_t;Y_{t-1})$. Maximizing this objective strikes a perfect balance between two operational goals for Markov aggregation [47, Lemma 4]. On the one hand, the process $\{Y_t\}$ on the modules should be close to a Markov chain (but not necessarily for every initial distribution) and, on the other hand, it should retain most of the temporal dependence structure of the original Markov chain $\{X_t\}$.

In contrast to this, the authors of [45] have proposed a cost function based on a sufficient condition for strong lumpability, i.e. the requirement that $\{Y_t\} = \{m(X_t)\}$ is a Markov chain for every initial distribution. Similarly, their cost function can be written as the Kullback-Leibler divergence rate between the original Markov chain, and a (different) lifting of the aggregated Markov chain with transition probability matrix $Q^\bullet$. The resulting cost function

$$H(Y_t|Y_{t-1}) - H(Y_t|X_{t-1}) \tag{B.6}$$

was shown to be a lower bound of our relaxed objective (B.4). Hence, the lifting proposed by [45] is expected to yield a closer approximation of the original transition probability matrix P (or equivalently, of the original

process $\{X_t\}$). In contrast to this, Corollary 2 shows that synthesizing Infomap gives an upper bound on $I(X_t; X_{t-1}) - I(Y_t; Y_{t-1})$. Therefore, Q in (3.1) is assumed to be a worse fit to P than both Q in (B.1) and P' as proposed by the lifting in [45].

The link between Markov chains and community detection is prominent and well-known. The preceding discussion displays an even stronger link between our proposed cost function for community detection and the literature on Markov aggregation with information-theoretic cost functions. In both [43, 45] and our Problem 2, the Kullback-Leibler divergence rate between the transition probability matrix P of the original Markov chain (or random walk on the considered graph) and an approximation is used as a cost function. Furthermore, the resulting cost functions can be shown to form an ordered set. The main difference between our Problem 2 on the one hand and [43, 45] on the other is that the transition probability matrix Q used in the Kullback-Leibler divergence rate is *not* obtained via lifting the one-step conditional probabilities of the process $\{m(X_t)\}$.

B.3. Relationship to Stochastic Block-Modeling

The definition of Q in (3.1) and of the optimization problem (3.2) are reminiscent of stochastic block-modeling under Kullback-Leibler divergence. The main difference is that in stochastic block-modeling, one tries to infer model parameters such that the likelihood of a given graph is maximized *without* first transforming this graph into a probabilistic model via for example a random walker. In other words, block-modeling infers the parameters of a random graph model by maximizing the likelihood for a given graph. Here, in contrast, we infer the parameters of a restricted random walk model that resembles the probabilistic behavior of a random walker on the given graph.

Appendix C.

Proofs

In this appendix we provide proofs which we withheld in the main part of this thesis for the sake of readability. In Section C.1 we prove the feasibility of the relaxation of Problem 1 into Problem 2. We show the local optimality of maximal independent sets as a coarsening of independent subsets in Section C.2. Finally, we prove Proposition 1 in Section C.3.

C.1. Relaxation of Problem 1

Proposition 2. *Let $\{X_t\}$ be a stationary Markov chain with transition probability matrix P derived from the weight matrix W of the graph $\mathcal{G}$ and let $Q^{\mathcal{Y}}$ be a transition probability matrix of the same size parameterized as in (3.1). Then, for every partition $\mathcal{Y}$, it holds that*

$$\min_{\{[r_\alpha^i]_{\alpha\in i},\, s_i,\, u_i\}_{i\in\mathcal{Y}}} \overline{D}(P\|Q^{\mathcal{Y}}) \leq I(X_t; X_{t-1}) - \sum_{i\in\mathcal{Y}} p_i D(p_{i\to i}\|p_i) \qquad \text{(C.1)}$$

where p_i and $p_{i\to i}$ are defined as in (2.5).

Proof 1. Let $p_{\alpha\to i} := \sum_{\beta\in i} p_{\alpha\to\beta}$ and $p_{\alpha\not\to i} := \sum_{j\neq i} p_{\alpha\to j} = 1 - p_{\alpha\to i}$. Then from (3.1) follows that

$$\begin{aligned}
\overline{D}(P\|Q^{\mathcal{Y}}) &= \sum_{\alpha,\beta} p_\alpha p_{\alpha\to\beta} \log \frac{p_{\alpha\to\beta}}{q_{\alpha\to\beta}} \\
&= \sum_{\alpha,\beta} p_\alpha p_{\alpha\to\beta} \left[\mathbb{I}_{m(\alpha)}(\beta) \log \frac{p_{\alpha\to\beta}}{r_\beta^{m(\beta)}(1-s_{m(\alpha)})} \right. \\
&\qquad \left. + (1-\mathbb{I}_{m(\alpha)}(\beta)) \log \frac{p_{\alpha\to\beta}}{r_\beta^{m(\beta)} s_{m(\alpha)} \frac{u_{m(\beta)}}{1-u_{m(\alpha)}}} \right] \\
&= \sum_j \sum_\alpha \sum_{\beta\in j} p_\alpha p_{\alpha\to\beta} \log \frac{\frac{p_{\alpha\to\beta}}{p_{\alpha\to j}}}{r_\beta^j} + \sum_i \sum_{\alpha\in i} p_\alpha p_{\alpha\to i} \log \frac{p_{\alpha\to i}}{1-s_i} \\
&\qquad + \sum_i \sum_{j\neq i} \sum_{\alpha\in i} p_\alpha p_{\alpha\to j} \log \frac{p_{\alpha\to j}}{s_i \frac{u_j}{1-u_i}} \\
&= \sum_j \sum_\alpha \sum_{\beta\in j} p_\alpha p_{\alpha\to\beta} \log \frac{\frac{p_{\alpha\to\beta}}{p_{\alpha\to j}}}{r_\beta^j} + \sum_i \sum_{j\neq i} \sum_{\alpha\in i} p_\alpha p_{\alpha\to j} \log \frac{\frac{p_{\alpha\to j}}{p_{\alpha\not\to i}}}{\frac{u_j}{1-u_i}} \\
&\quad + \sum_i \sum_{\alpha\in i} p_\alpha p_{\alpha\to i} \log \frac{p_{\alpha\to i}}{1-s_i} + \sum_i \sum_{\alpha\in i} p_\alpha p_{\alpha\not\to i} \log \frac{p_{\alpha\not\to i}}{s_i} \\
&= \sum_j \sum_\alpha \sum_{\beta\in j} p_\alpha p_{\alpha\to\beta} \log \frac{\frac{p_{\alpha\to\beta}}{p_{\alpha\to j}}}{r_\beta^j} + \sum_i \sum_{j\neq i} \sum_{\alpha\in i} p_\alpha p_{\alpha\to j} \log \frac{\frac{p_{\alpha\to j}}{p_{\alpha\not\to i}}}{\frac{u_j}{1-u_i}} \\
&\quad + \sum_i \sum_{\alpha\in i} \left[p_\alpha p_{\alpha\to i} \log \frac{p_\alpha p_{\alpha\to i}}{p_\alpha(1-s_i)} + p_\alpha p_{\alpha\not\to i} \log \frac{p_\alpha p_{\alpha\not\to i}}{p_\alpha s_i} \right]
\end{aligned}$$

$$= \sum_j \sum_\alpha \sum_{\beta \in j} p_\alpha p_{\alpha\to\beta} \log \frac{\frac{p_{\alpha\to\beta}}{p_{\alpha\to j}}}{r^j_\beta} + \sum_i \sum_{j\neq i} \sum_{\alpha\in i} p_\alpha p_{\alpha\to j} \log \frac{\frac{p_{\alpha\to j}}{p_{\alpha\not\to i}}}{\frac{r_j}{1-r_i}} + \sum_i D(p_\alpha p_{\alpha\to i} \| p_\alpha(1-s_i)). \tag{C.2}$$

We can now independently minimize the first summation term w.r.t $\{[r^j_\beta]_{\beta\in j}\}_{j\in\mathcal{Y}}$, the second summation term w.r.t $[u_i]_{i\in\mathcal{Y}}$, and the last summation term w.r.t $[s_i]_{i\in\mathcal{Y}}$. Considering the latter, one can show [15, Lemma 10.8.1] that the Kullback-Leibler divergence $D(p_\alpha p_{\alpha\to i}\|p_\alpha(1-s_i))$ is minimized for

$$1 - s_i = \sum_{\alpha\in i} p_\alpha p_{\alpha\to i} = p_{i\to i} \tag{C.3}$$

and hence

$$s_i = 1 - p_{i\to i} = p_{i\not\to i}. \tag{C.4}$$

The minimizer regarding the distribution over nodes can be found along similar lines. For the first summation term in (C.2) we observe that

$$\sum_j \sum_\alpha \sum_{\beta\in j} p_\alpha p_{\alpha\to\beta} \log \frac{\frac{p_{\alpha\to\beta}}{p_{\alpha\to j}}}{r^j_\beta} = \sum_j \sum_\alpha p_\alpha p_{\alpha\to j} \sum_{\beta\in j} \frac{p_{\alpha\to\beta}}{p_{\alpha\to j}} \log \frac{\frac{p_{\alpha\to\beta}}{p_{\alpha\to j}}}{r^j_\beta}$$

$$= \sum_j p_j \sum_\alpha \frac{p_\alpha p_{\alpha\to j}}{p_j} \sum_{\beta\in j} \frac{p_{\alpha\to\beta}}{p_{\alpha\to j}} \log \frac{\frac{p_{\alpha\to\beta}}{p_{\alpha\to j}}}{r^j_\beta}$$

$$= \sum_j p_j \sum_\alpha \frac{p_\alpha p_{\alpha\to j}}{p_j} \sum_{\beta\in j} \frac{p_{\alpha\to\beta}}{p_{\alpha\to j}} \log \frac{\frac{p_\alpha p_{\alpha\to\beta}}{p_j}}{\frac{p_\alpha p_{\alpha\to j}}{p_j} r^j_\beta}$$

$$= \sum_j p_j \sum_\alpha \sum_{\beta\in j} \frac{p_\alpha p_{\alpha\to j}}{p_j} \frac{p_{\alpha\to\beta}}{p_{\alpha\to j}} \log \frac{\frac{p_\alpha p_{\alpha\to\beta}}{p_j}}{\frac{p_\alpha p_{\alpha\to j}}{p_j} r^j_\beta}$$

$$= \sum_j p_j D\left(\frac{p_\alpha p_{\alpha\to j}}{p_j} \cdot \frac{p_{\alpha\to\beta}}{p_{\alpha\to j}} \middle\| \frac{p_\alpha p_{\alpha\to j}}{p_j} \cdot r^j_\beta\right)$$

and by again employing [15, Lemma 10.8.1], for a fixed module j this quantity is minimized for

$$r^j_\beta = \sum_\alpha \frac{p_\alpha p_{\alpha\to\beta}}{p_j} = \frac{p_\beta}{p_j} = \mathbb{P}(X_t = \beta | Y_t = j). \tag{C.5}$$

Applying these minimizers yields

$$\min_{\{[r^i_\alpha]_{\alpha\in i},\, s_i,\, u_i\}_{i\in\mathcal{Y}}} \overline{D}(P\|Q^{\mathcal{Y}}) = \min_{[u_i]_{i\in\mathcal{Y}}} I(X_t; X_{t-1}) - H(Y_t) + H(S_t|Y_t) - \sum_i \sum_{j\neq i} p_i p_{i\to j} \log \frac{u_j}{1-u_i} \tag{C.6}$$

where $S_t = 1$ is a binary RV reflecting whether we stay in or leave a module at time t. Since for the distribution over modules $[u_i]_{i\in\mathcal{Y}}$ there exists no closed-form solution to the best of our knowledge, we choose $[u_i]_{i\in\mathcal{Y}} = [p_i]_{i\in\mathcal{Y}}$ as a sub-optimal solution. In other words, we utilize the stationary distribution of $\{Y_t\}$. Inserting this choice finally yields

$$\min_{\{[r^i_\alpha]_{\alpha\in i},\, s_i,\, u_i\}_{i\in\mathcal{Y}}} \overline{D}(P\|Q^{\mathcal{Y}}) \leq I(X_t; X_{t-1}) - \sum_i p_i D(p_{i\to i}\|p_i). \tag{C.7}$$

□

Corollary 1. *From the non-negativity of the Kullback-Leibler divergence, Kullback-Leibler divergence rate and the mutual information it follows trivially, that*

$$0 \leq \sum_{i \in \mathcal{Y}} p_i D(p_{i \to i} \| p_i) \leq I(X_t; X_{t-1}). \tag{C.8}$$

C.2. Behavior of Coarsenings of Independent Sets

Proposition 3. *Let $\mathcal{Y} = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_M\}$ be partition into (not exclusively maximal) independent sets of a graph $\mathcal{G} = (\mathcal{X}, E, W)$. Consider further a coarsening $\mathcal{Y}^\bullet = \{\mathcal{Y}_1^\bullet, \ldots, \mathcal{Y}_M^\bullet\}$ of $\mathcal{Y}$ such that each module in $\mathcal{Y}^\bullet$ is a maximal independent set. Then, for every partition $\mathcal{Y}$ it holds that*

$$\mathcal{J}(\mathcal{Y}^\bullet) > \mathcal{J}(\mathcal{Y}). \tag{C.9}$$

Proof 2. Let

$$f(\mathcal{Y}_i) := p_i D(p_{i \to i} \| p_i) \tag{C.10}$$

denote the contribution to the synthesizing Infomap objective $\mathcal{J}(\mathcal{Y}) = \sum_{\mathcal{Y}_i \in \mathcal{Y}} f(\mathcal{Y}_i)$ (cf. Equation (3.7)) for a given module $\mathcal{Y}_i$. If $\mathcal{Y}_i$ is an independent set it holds that $p_{i \to i} = 0$ and therefore $f(\mathcal{Y}_i)$ reduces to

$$f_{ind}(\mathcal{Y}_i) = -p_i \log(1 - p_i). \tag{C.11}$$

It is clear to see, that

$$f_{ind}(\mathcal{Y}_i) > f_{ind}(\mathcal{Y}_j) \quad \text{if} \quad p_i > p_j \tag{C.12}$$

for the marginal probabilities $0 < p_i, p_j < 1$. For a union of two independent sets $\mathcal{Y}_i$ and $\mathcal{Y}_j$ into a larger independent set the objective reads

$$f_{ind}(\mathcal{Y}_i \cup \mathcal{Y}_j) = -(p_i + p_j)\log[1 - (p_i + p_j)] \tag{C.13}$$
$$= -p_i \log[1 - (p_i + p_j)] - p_j \log[1 - (p_i + p_j)]. \tag{C.14}$$

Now, since

$$-p_i \log[1 - (p_i + p_j)] > -p_i \log(1 - p_i) = f_{ind}(\mathcal{Y}_i) \tag{C.15}$$

and

$$-p_j \log[1 - (p_i + p_j)] > -p_j \log(1 - p_j) = f_{ind}(\mathcal{Y}_j) \tag{C.16}$$

it further holds that

$$f_{ind}(\mathcal{Y}_i \cup \mathcal{Y}_j) > f_{ind}(\mathcal{Y}_i) + f_{ind}(\mathcal{Y}_j). \tag{C.17}$$

It follows for a maximal independent set

$$\mathcal{Y}_i^\bullet = \bigcup_{\mathcal{Y}_i \in \mathcal{Y}_i^\bullet} \mathcal{Y}_i \tag{C.18}$$

that

$$f_{ind}(\mathcal{Y}_i^\bullet) > \sum_{\mathcal{Y}_i \in \mathcal{Y}_i^\bullet} f_{ind}(\mathcal{Y}_i) \tag{C.19}$$

where $\{\mathcal{Y}_i\}_{\mathcal{Y}_i \in \mathcal{Y}_i^\bullet}$ are independent subsets of $\mathcal{Y}_i^\bullet$. Hence,

$$\mathcal{J}(\mathcal{Y}^\bullet) = \sum_{\mathcal{Y}_i^\bullet \in \mathcal{Y}^\bullet} f_{ind}(\mathcal{Y}_i^\bullet) > \sum_{\mathcal{Y}_i \in \mathcal{Y}} f_{ind}(\mathcal{Y}_i) = \mathcal{J}(\mathcal{Y}) \tag{C.20}$$

if $\mathcal{Y}^\bullet$ is a coarsening of $\mathcal{Y}$. □

C.3. Proof of Proposition 1

Proof 3. Reusing intermediate results from Proof 1 we can minimize w.r.t $\{[v_j^i]_{j\in\mathcal{Y}\setminus\{i\}}\}_{i\in\mathcal{Y}}$ by looking at the second summation term of Equation (C.2).

$$\begin{aligned}
\sum_i \sum_{j\neq i} \sum_{\alpha\in i} p_\alpha p_{\alpha\to j} \log \frac{\frac{p_{\alpha\to j}}{p_{\alpha\not\to i}}}{v_j^i} &= \sum_i \sum_{\alpha\in i} p_\alpha p_{\alpha\not\to i} \sum_{j\neq i} \frac{p_{\alpha\to j}}{p_{\alpha\not\to i}} \log \frac{\frac{p_{\alpha\to j}}{p_{\alpha\not\to i}}}{v_j^i} \\
&= \sum_i p_i p_{i\not\to i} \sum_{\alpha\in i} \frac{p_\alpha p_{\alpha\not\to i}}{p_i p_{i\not\to i}} \sum_{j\neq i} \frac{p_{\alpha\to j}}{p_{\alpha\not\to i}} \log \frac{\frac{p_{\alpha\to j}}{p_{\alpha\not\to i}}}{v_j^i} \\
&= \sum_i p_i p_{i\not\to i} \sum_{\alpha\in i} \frac{p_\alpha p_{\alpha\not\to i}}{p_i p_{i\not\to i}} \sum_{j\neq i} \frac{p_{\alpha\to j}}{p_{\alpha\not\to i}} \log \frac{\frac{p_\alpha p_{\alpha\to j}}{p_i p_{i\not\to i}}}{\frac{p_\alpha p_{\alpha\not\to i}}{p_i p_{i\not\to i}} v_j^i} \\
&= \sum_i p_i p_{i\not\to i} D\Big(\frac{p_\alpha p_{\alpha\not\to i}}{p_i p_{i\not\to i}} \cdot \frac{p_{\alpha\to j}}{p_{\alpha\not\to i}} \Big\| \frac{p_\alpha p_{\alpha\not\to i}}{p_i p_{i\not\to i}} \cdot v_j^i\Big).
\end{aligned}$$

Again using [15, Lemma 10.8.1], the Kullback-Leibler divergence attains its minimum for

$$\begin{aligned}
v_j^i &= \sum_{\alpha\in i} \frac{p_\alpha p_{\alpha\not\to i}}{p_i p_{i\not\to i}} \frac{p_{\alpha\to j}}{p_{\alpha\not\to i}} = \frac{p_{i\to j}}{p_{i\not\to i}} && \text{(C.21)} \\
&= \mathbb{P}(Y_t = j | Y_{t-1} = i, Y_t \neq i). && \text{(C.22)}
\end{aligned}$$

for a fixed module i. Recapitulating Equation (C.6) we have

$$\begin{aligned}
\min_{\{[r_\alpha^i]_{\alpha\in i},\, s_i,\, [v_j^i]_{j\in\mathcal{Y}\setminus\{i\}}\}_{i\in\mathcal{Y}}} \overline{D}(P\|Q^{\mathcal{Y}}) = \min_{\{[v_j^i]_{j\in\mathcal{Y}\setminus\{i\}}\}_{i\in\mathcal{Y}}} & I(X_t; X_{t-1}) \\
-H(Y_t) + H(S_t|Y_t) - \sum_i \sum_{j\neq i} & p_i p_{i\to j} \log v_j^i && \text{(C.23)}
\end{aligned}$$

and inserting the result from Equation (C.21) finally yields

$$\min_{\{[r_\alpha^i]_{\alpha\in i},\, s_i,\, [v_j^i]_{j\in\mathcal{Y}\setminus\{i\}}\}_{i\in\mathcal{Y}}} \overline{D}(P\|Q^{\mathcal{Y}}) = I(X_t;X_{t-1}) - I(Y_t;Y_{t-1}). \tag{C.24}$$

□

Corollary 2. *Since the optimization problem* (B.2) *has a larger feasible set than* (3.2) *it holds that*

$$I(X_t;X_{t-1}) - I(Y_t;Y_{t-1}) \leq I(X_t;X_{t-1}) - \sum_{i\in\mathcal{Y}} p_i D(p_{i\to i}\|p_i). \tag{C.25}$$

This implies

$$\mathcal{J}(\mathcal{Y}) \leq I(Y_t;Y_{t-1}), \tag{C.26}$$

which presents a different upper bound of the synthesizing Infomap objective (3.7).

Bibliography

[1] Santo Fortunato. "Community detection in graphs." In: *Physics reports* 486.3-5 (2010), pp. 75–174 (cit. on p. 1).

[2] Santo Fortunato and Darko Hric. "Community detection in networks: A user guide." In: *Physics Reports* 659 (2016). Community detection in networks: A user guide, pp. 1–44. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2016.09.002. URL: http://www.sciencedirect.com/science/article/pii/S0370157316302964 (cit. on pp. 1, 9, 26).

[3] M. Girvan and M. E. J. Newman. "Community structure in social and biological networks." In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 7821–7826. ISSN: 0027-8424. DOI: 10.1073/pnas.122653799. eprint: https://www.pnas.org/content/99/12/7821.full.pdf. URL: https://www.pnas.org/content/99/12/7821 (cit. on pp. 1, 26, 27).

[4] Filippo Radicchi et al. "Defining and identifying communities in networks." In: *Proceedings of the National Academy of Sciences* 101.9 (2004), pp. 2658–2663. ISSN: 0027-8424. DOI: 10.1073/pnas.0400054101. eprint: https://www.pnas.org/content/101/9/2658.full.pdf. URL: https://www.pnas.org/content/101/9/2658 (cit. on pp. 1, 10, 64).

[5] Zhao Yang, René Algesheimer, and Claudio Tessone. "A Comparative Analysis of Community Detection Algorithms on Artificial Networks." In: *Scientific Reports* 6 (Aug. 2016). DOI: 10.1038/srep30750 (cit. on pp. 1, 2, 26, 28, 50).

[6] Martin Rosvall and Carl T. Bergstrom. "Maps of random walks on complex networks reveal community structure." In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123. ISSN: 0027-8424. DOI: 10.1073/pnas.0706851105. eprint: https://www.pnas.org/content/105/4/1118.full.pdf. URL: https://www.pnas.org/content/105/4/1118 (cit. on p. 1).

[7] M. Rosvall, D. Axelsson, and C. T. Bergstrom. "The map equation." In: *The European Physical Journal Special Topics* 178.1 (Nov. 2009), pp. 13–23. ISSN: 1951-6401. DOI: 10.1140/epjst/e2010-01179-1. URL: http://dx.doi.org/10.1140/epjst/e2010-01179-1 (cit. on pp. 1, 11).

[8] Daniel Edler, Anton Eriksson, and Martin Rosvall. *The MapEquation software package*. URL: http://www.mapequation.org (cit. on pp. 1, 25, 77).

[9] Martin Rosvall and Carl T. Bergstrom. "Multilevel Compression of Random Walks on Networks Reveals Hierarchical Organization in Large Integrated Systems." In: *PLOS ONE* 6.4 (Apr. 2011), pp. 1–10. DOI: 10.1371/journal.pone.0018209. URL: https://doi.org/10.1371/journal.pone.0018209 (cit. on p. 1).

[10] Alcides Viamontes Esquivel and Martin Rosvall. "Compression of Flow Can Reveal Overlapping-Module Organization in Networks." In: *Phys. Rev. X* 1 (2 Dec. 2011), p. 021025. DOI: 10.1103/PhysRevX.1.021025. URL: https://link.aps.org/doi/10.1103/PhysRevX.1.021025 (cit. on p. 1).

[11] Tatsuro Kawamoto and Martin Rosvall. "Estimating the resolution limit of the map equation in community detection." In: *Phys. Rev. E* 91 (1 Jan. 2015), p. 012809. DOI: 10.1103/PhysRevE.91.012809. URL: https://link.aps.org/doi/10.1103/PhysRevE.91.012809 (cit. on p. 1).

[12] M. E. J. Newman and M. Girvan. "Finding and evaluating community structure in networks." In: *Phys. Rev. E* 69 (2 Feb. 2004), p. 026113. DOI: 10.1103/PhysRevE.69.026113. URL: https://link.aps.org/doi/10.1103/PhysRevE.69.026113 (cit. on pp. 1, 32).

[13] Michael T. Schaub, Renaud Lambiotte, and Mauricio Barahona. "Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation." In: *Phys. Rev. E* 86 (2 Aug. 2012), p. 026112. DOI: 10.1103/PhysRevE.86.026112. URL: https://link.aps.org/doi/10.1103/PhysRevE.86.026112 (cit. on pp. 1, 2).

[14] Masoumeh Kheirkhahzadeh, Andrea Lancichinetti, and Martin Rosvall. "Efficient community detection of network flows for varying Markov times and bipartite networks." In: *Phys. Rev. E* 93 (3 Mar. 2016), p. 032309. DOI: 10.1103/PhysRevE.93.032309. URL: https://link.aps.org/doi/10.1103/PhysRevE.93.032309 (cit. on pp. 2, 72).

[15] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. 2nd ed. USA: Wiley-Interscience, 2006. ISBN: 0471241954 (cit. on pp. 8, 91, 92, 95).

[16] Ziad Rached, Fady Alajaji, and L. Lorne Campbell. "The Kullback-Leibler divergence rate between Markov sources." In: *IEEE Transactions on Information Theory* 50.5 (May 2004), pp. 917–921 (cit. on p. 8).

[17] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. "Benchmark graphs for testing community detection algorithms." In: *Phys. Rev. E* 78 (4 Oct. 2008), p. 046110. DOI: 10.1103/PhysRevE.78.046110. URL: https://link.aps.org/doi/10.1103/PhysRevE.78.046110 (cit. on pp. 9, 26–28).

[18] Andrea Lancichinetti and Santo Fortunato. "Community Detection Algorithms: A Comparative Analysis." In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 80 (Nov. 2009), p. 056117. DOI: 10.1103/PhysRevE.80.056117 (cit. on pp. 9, 10).

[19] Günce Keziban Orman and Vincent Labatut. "A Comparison of Community Detection Algorithms on Artificial Networks." In: *Discovery Science*. Ed. by João Gama et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 242–256. ISBN: 978-3-642-04747-3 (cit. on pp. 9, 26, 28).

[20] C. E. Shannon. "A Mathematical Theory of Communication." In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x. eprint: https://onlinelibrary.

wiley.com/doi/pdf/10.1002/j.1538-7305.1948.tb01338.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x (cit. on p. 11).

[21] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX." In: (2008). Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman, pp. 11–15. URL: https://networkx.github.io (cit. on p. 26).

[22] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 26).

[23] Alexander J. Gates and Yong-Yeol Ahn. "CluSim: a Python package for the comparison of clusterings and dendrograms." In: *bioRxiv* (2018). DOI: 10.1101/410084. URL: https://www.biorxiv.org/content/early/2018/09/06/410084 (cit. on p. 26).

[24] J. D. Hunter. "Matplotlib: A 2D graphics environment." In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cit. on p. 26).

[25] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation." In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30. DOI: 10.1109/MCSE.2011.37. URL: https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37 (cit. on p. 26).

[26] Wes McKinney. "pandas: a foundational Python library for data analysis and statistics." In: *Python for High Performance and Scientific Computing* 14 (2011) (cit. on p. 26).

[27] Ulrike Luxburg. "A Tutorial on Spectral Clustering." In: *Statistics and Computing* 17.4 (Dec. 2007), pp. 395–416. ISSN: 0960-3174. DOI: 10.1007/s11222-007-9033-z. URL: http://dx.doi.org/10.1007/s11222-007-9033-z (cit. on p. 26).

[28] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an Algorithm." In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. NIPS'01. Vancouver, British Columbia, Canada: MIT Press, 2001,

pp. 849–856. URL: http://dl.acm.org/citation.cfm?id=2980539.2980649 (cit. on p. 26).

[29] Jianbo Shi and J. Malik. "Normalized cuts and image segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (Aug. 2000), pp. 888–905. DOI: 10.1109/34.868688 (cit. on p. 26).

[30] Wayne W. Zachary. "An Information Flow Model for Conflict and Fission in Small Groups." In: *Journal of Anthropological Research* 33.4 (1977), pp. 452–473. DOI: 10.1086/jar.33.4.3629752. URL: https://doi.org/10.1086/jar.33.4.3629752 (cit. on p. 27).

[31] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. http://snap.stanford.edu/data. June 2014 (cit. on pp. 27, 28, 69).

[32] Jérôme Kunegis. "KONECT: The Koblenz Network Collection." In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW '13 Companion. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 1343–1350. ISBN: 9781450320382. DOI: 10.1145/2487788.2488173. URL: https://doi.org/10.1145/2487788.2488173 (cit. on p. 27).

[33] Ryan A. Rossi and Nesreen K. Ahmed. "The Network Data Repository with Interactive Graph Analytics and Visualization." In: *AAAI*. 2015. URL: http://networkrepository.com (cit. on p. 27).

[34] Leto Peel, Daniel B. Larremore, and Aaron Clauset. "The ground truth about metadata and community detection in networks." In: *Science Advances* 3.5 (2017). DOI: 10.1126/sciadv.1602548. URL: https://advances.sciencemag.org/content/3/5/e1602548 (cit. on p. 27).

[35] Andrea Lancichinetti and Santo Fortunato. "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities." In: *Phys. Rev. E* 80 (1 July 2009), p. 016118. DOI: 10.1103/PhysRevE.80.016118. URL: https://link.aps.org/doi/10.1103/PhysRevE.80.016118 (cit. on p. 28).

[36] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance." In: *J. Mach. Learn. Res.* 11

(Dec. 2010), pp. 2837–2854. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1756006.1953024 (cit. on p. 30).

[37] Leon Danon et al. "Comparing community structure identification." In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.09 (Sept. 2005), P09008–P09008. DOI: 10.1088/1742-5468/2005/09/p09008. URL: https://doi.org/10.1088%2F1742-5468%2F2005%2F09%2Fp09008 (cit. on p. 30).

[38] S. Romano et al. "Standardized mutual information for clustering comparisons: One step further in adjustment for chance." In: *31st International Conference on Machine Learning, ICML 2014* 4 (Jan. 2014), pp. 2873–2882 (cit. on p. 30).

[39] D. Watts and S. Strogatz. "Collective dynamics of 'small-world' networks." In: *Nature* 393 (1998), pp. 440–442. DOI: https://doi.org/10.1038/30918 (cit. on p. 31).

[40] Linda M. Collins and Clyde W. Dent. "Omega: A General Formulation of the Rand Index of Cluster Recovery Suitable for Non-disjoint Solutions." In: *Multivariate Behavioral Research* 23.2 (1988), pp. 231–242. DOI: 10.1207/s15327906mbr2302_6. URL: https://doi.org/10.1207/s15327906mbr2302_6 (cit. on pp. 32, 33).

[41] Andrea Lancichinetti, Santo Fortunato, and János Kertész. "Detecting the overlapping and hierarchical community structure in complex networks." In: *New Journal of Physics* 11.3 (Mar. 2009), p. 033015. DOI: 10.1088/1367-2630/11/3/033015. URL: https://doi.org/10.1088%2F1367-2630%2F11%2F3%2F033015 (cit. on p. 33).

[42] Jaewon Yang and Jure Leskovec. "Defining and evaluating network communities based on ground-truth." In: *Knowledge and Information Systems* 42.1 (Jan. 2015), pp. 181–213. ISSN: 0219-3116. DOI: 10.1007/s10115-013-0693-z. URL: https://doi.org/10.1007/s10115-013-0693-z (cit. on p. 69).

[43] K. Deng, P. G. Mehta, and S. P. Meyn. "Optimal Kullback-Leibler Aggregation via Spectral Theory of Markov Chains." In: *IEEE Transactions on Automatic Control* 56.12 (Dec. 2011), pp. 2793–2808. ISSN: 2334-3303. DOI: 10.1109/TAC.2011.2141350 (cit. on pp. 85–87).

[44] Amir Alush, Avishay Friedman, and Jacob Goldberger. "Pairwise clustering based on the mutual-information criterion." In: *Neurocomputing* 182 (2016), pp. 284–293. ISSN: 0925-2312 (cit. on p. 85).

[45] B. C. Geiger et al. "Optimal Kullback-Leibler Aggregation via Information Bottleneck." In: *IEEE Transactions on Automatic Control* 60.4 (Apr. 2015). open-access: arXiv:1304.6603 [cs.SY], pp. 1010–1022. ISSN: 2334-3303. DOI: 10.1109/TAC.2014.2364971 (cit. on pp. 85–87).

[46] M. Vidyasagar. "Reduced-Order Modeling of Markov and Hidden Markov Processes via Aggregation." In: *49th IEEE Conference on Decision and Control (CDC)*. Dec. 2010, pp. 1810–1815. DOI: 10.1109/CDC.2010.5717206 (cit. on p. 85).

[47] R. A. Amjad, C. Blöechl, and B. C. Geiger. "A Generalized Framework for Kullback-Leibler Markov Aggregation." In: *IEEE Transactions on Automatic Control* (2019), pp. 1–1. ISSN: 2334-3303. DOI: 10.1109/TAC.2019.2945891 (cit. on pp. 85, 86).

[48] B. C. Geiger and Y. Wu. "Higher-Order Optimal Kullback-Leibler Aggregation of Markov Chains." In: *SCC 2017; 11th International ITG Conference on Systems, Communications and Coding*. open-access: arXiv:1608.04637 [cs.IT]. Feb. 2017, pp. 1–6 (cit. on p. 86).