ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Multi-Order Flow Compression in Time Series Data on Complex Networks

Master Thesis

Zhichao Han

Saturday 1st September, 2018

Advisors: Prof. Ingo Scholtes, Prof. Frank Schweitzer, Prof. Andreas Krause
Chair of Systems Design AND Learning & Adaptive Systems Group, ETH Zürich

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

MULTI-ORDER FLOW COMPRESSION IN TIME SERIES DATA ON COMPLEX NETWORKS

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
HAN	ZHICHAO

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, September 1, 2018	韩智超.

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Abstract

Detecting the community structure in graphs is an important task in network analysis. Most previous community detection algorithms are designed for static networks. Detecting the communities in networks based on time series data is still a fundamental challenge. In this thesis, we generalize the stardard Infomap method to the multi-order Markov model for time series data and we call it `Multiorder Infomap`. `Multi-order Infomap` uses the same idea of the stardard Infomap that the walker tends to be "trapped" within each cluster and it keeps the framework of the stardard Infomap. However, `Multi-order Infomap` adapts the map equation to consider different-order transitions of the sequence. In this way, the `Multi-order Infomap` can directly detect the communities based on the sequence data without constructing the underline graph explicitly. And because the `Multi-order Infomap` relies on the multi-order Markov model to capture the transition property of the sequences, the order of `Multi-order Infomap` could be determined by the order detection method implementedx in the multi-order Markov model. The experiments show that the multi-order Infomap can find meaningful community structure existed in the sequence. Besides, we provide a deep discussion about the description length which is treated as the measurement for model selectoin in previous studies.

Contents

Chapter 1

Introduction

Detecting community structure in graphs is an important task. It has wide applications in various areas, e.g. studying social groups, discovering the functional modules in biology, designing recommender systems, analyzing the modularity in software architectures and graph-based image segmentation [10] [9]. While the community detection problem has been studied extensively, most previous studies are designed for static network. Particularly, the majority of existing studies assumes that paths in the network are transitive, i.e. the existence of two links $a \to b$ and $b \to c$ implies that a is transitively connected to c via b. However, this assumption may not hold in real-time series data / sequences. For the above example, a transitive path from a via b to c only exists if the link $a \to b$ occurs before $b \to c$, while it is absent if the ordering of the two links is reversed. So it is still a fundamental challenge that how can we detect communities *directly* from time series data that not only capture the topology of links but also the ordering of these transitions.

Recent studies have shown that the correlations in the ordering of transitions in time series data can be captured by the high-order Markov model. The idea is to use high order nodes to model the states transitions. Each high order node with order k corresponds to a subsequence $(v_1, v_2, \ldots, v_k)$ with length k. The state node $(v_1, v_2, \ldots, v_k)$ means the sequence is now on node v_k and the person comes to v_k from nodes $(v_1, v_2, \ldots, v_{k-1})$. In this way, the sequence can be seen as the transitions between high order nodes and we can preserve statistical dependencies between the next node on a path and the k nodes previously traversed. Moreover, the recent work [11] combines high-order Markov model with different orders to different layers together and uses high-order state nodes with different orders to model the sequences. And this multi-order Markov model [11] also provides a model selection method that allows to infer the optimal order for a given time series data set. The details are introduced in section 3.1.

In this thesis, we generalize the Infomap algorithm to the multi-order Markov model. Infomap is a classical method to detect the community structure of graph. It relies on the fact that a walker in a graph tends to be trapped within each cluster. For example, Figure 1.1 plots the heatmap of traffic in a city. We can see the traffic is more likely to stay within some districts. With this assumption, Infomap maps the task of finding "natural" clusters in a graph to the problem of finding an optimal "compression" of the flows. In the Infomap model, we partition first-order or high-order nodes to different communities. The communities are assigned with a unique id and the nodes in the same community are also assigned with different ids [1]. The sequence that could be seen as the transitions between nodes can also be encoded by this coding scheme. Specifically, if the sequence enters into a new community, we use the id of the new community to indicate that this sequence enters to this community and samely if the sequence leaves a community, we use the id of this community to indicate that this sequence leaves this community. Besides, we use the id of codes to indicate which nodes the sequence is on. For different partitions of the nodes, the probability flow has different description length. The partition which minimizes the coding length is the natural cluster structure in the graph. The details of Infomap are reviewed in section 3.2. Because the Infomap relies on the Markov model to capture the transitions in the sequences. And the multi-order Markov model has shown its power to capture the properties of transitons of the sequences better. So it is a natural idea to combine Infomap and the multi-order Markov model together. We can use the multi-order Markov model to capture the topological and ordering properties of transitions, e.g. the transition probabilities. Then we adapt the Infomap to consider these different-order transitions of the sequence based on the multi-order Markov model.

The main contributions of this thesis are summarized as follows:

- We provide a deep discussion about the description length used as measurement for the quality of the found communities. Infomap maps the task of finding "natural" clusters in a graph to the problem of finding an optimal "compression" of the flows. The flows have different coding lengths with different partitions of the nodes. And the partition corresponding to the minimum description length is the best partition [7]. This claim is true if we use the coding length to measure the quality the different partitions with the *fixed* order Infomap. However, we argue that the coding length cannot be used to measure the quality of the communities found by Infomap with different orders. The details are discussed in section 3.3.
- We generalize the Infomap to the multi-order Markov model for the time series / sequences (in chapter 4). The developed multi-order In-

[1] Note that the nodes in different communities could have same id

Figure 1.1: The headmap of traffic of a city. The red regions represent high volume of traffic, whereas the blue regions indicate low volume of traffic. The image is from [1]. The image shows the traffic are likely to "trap" in several districts which correspond to different communities.

fomap operates on the sequences and does not require contructing the underline graph explicitly. The order of the multi-order Infomap could be determined by the order detection method implemented in the `PATHPY`[11]. The experiments show that the multi-order Infomap could find meaningful communities.

The rest parts are orginized as following. Firstly we review the related works in chapter 2. In chapter 3, we briefly review the Markov model and the Infomap model which are the preliminaries for our model. Besides, we discuss the description length for Infomap with different orders and argue that the description length cannot be used for model selection in chapter 3. Then in chapter 4, we formally define the problem to study in this thesis, discuss how to generalize the Infomap on the Multi-order graphical model (multi-order Infomap) for pathway data, and use experiments to show that the multi-order Infomap developed in this thesis can find meaningful communities. Finally we summerize the thesis and list the potential improvements in the last chapter 5.

Chapter 2

Related Works

There exist different kinds of community detection algorithms. Here we briefly review the flow-based approaches that are closely related with our method. We refer [4] and [9] for further overview.

2.1 Walktrap: distance based on random walk

The idea of **Walktrap** is to define a distance for the vertices. And the vertices in the same community are expected to have small distance and the vertices in different communities are expected to have large distance. As the result, clustering algorithm could be used to cluster these vertices. **Walktrap**[6] use the random walk to define the distance measurement. The reasonale is based on the observation that a random walker is more likely to stop in the same community with a proper length since nodes in the same community have more interaction compared with nodes in different communities. Based on the different visiting probability of nodes in same community and different community, **Walktrap** defines a distance measurement such that the the nodes in same community have small distance.

2.2 Conductance: the probability of jumping to other community

The assumption is that if the network exhibits community structure, a random walker is more likely to stay in the same community rather than going to another community. And the conductance is a measurement which describes the probability of a random walker going to another module considering the size of module. Suppose we divide the network to two parts, the

corresponding conductance is

$$\varphi(S) = \frac{cut(S, \overline{S})}{min(vol(S), vol(\overline{S}))} \tag{2.1}$$

where S is a set of vertices and $\overline{S}$ is the set of remaining vertices.

If we divide the network into several modules according to its community structure, the conductance corresponding to such division should be small. So the community detection problem can be solved by finding the division which minimize the corresponding conductance.

2.3 Infomap: reveal community structure by optimizing map equation

Infomap [7] [3] is also based on the assumption that a random walker is likely to spend long periods of time within a community instead of frequently going to another community. So if we divide the network into different areas and give every area an identification. We also give each node in each area an id. Then we can use this two-level coding (community id + vertex id) to describe the random walk. Note that the identifications of different communities should be different and the identifications of vertices within the same community should be different. But the identifications of vertices in different communities could be same. This coding scheme is very similar to a real map. For example, we could describe the location of a person be specifying the city name and the street name within the city. Streets in different cities could have the same name. In **Infomap**, the minimum length codes are used to name the communities and vertices. And the coding length of a random walk should be small if the network is divided according to its community structure. So we search the division of the network which minimizes the map equation:

$$\underset{\mathbf{M}}{\operatorname{argmin}}\ L(\mathbf{M}) = qH(Q) + \sum_{i=1}^{m} p^i H(P^i) \tag{2.2}$$

where $\mathbf{M}$ is a partition, q is the probability of this random walk switching clusters at each step, $H(Q)$ is the entropy of module names, p^i is the probability of within-module movements that occur in module i and $H(P^i)$ is the entropy of names of nodes in module i. The first term corresponds to the coding length of the movement between modules and second term corresponds to the coding length of the movement within a module. And $L(M)$ is average coding length to code each step of the random walk.

2.4 MCL: Graph Clustering by Flow Simulation

MCL [12] [8] simulates the flow on network and finds the areas having dense flow as communities. Suppose A is the (weighted) adjacency matrix and M is column-stochastic transition matrix with $M(i,j) = \frac{A(i,j)}{\sum_k A(k,j)}$. **MCL** repeats these three operations on M until M converges:

- **Expand** $Expand(M) := M * M$. During the expanding process, flows are spreaded on the network and the flows to vertices reachable by multiple paths are enhanced. Thus within-cluster flows are enhanced since there are more connection within the community.
- **Inflate** $Inflate(M) := \frac{M(i,j)^r}{\sum_{k=1}^{n} M(k,j)^k}$. The inflation step further enhances within-cluster flows and it brings non-linearity into the process. The non-linearity is necessary since it prevents all the columns of M becoming equal to the principal eigenvector of the canonical transition matrix [8].
- **Prune** Remove those entries which have very small values and rescale the retained entries to have the column sum to 1. This step reduces the number of non-zero entries in the matrix and saves memory.

The MCL process can be understood as expanding and contracting the flow in the graph alternately [8]. When M converges, we can interpret M as communities.

2.5 Modelling sequence on community structures

The idea of this approach is to use a parameterized model to model the generalization of a sequence and the community structure is encoded as the parameters. Communities can be found by learning the parameters from data. The simple model used in [5] is

$$p(x|\mathbf{x}) = \theta_x \lambda_{b_x b_{\mathbf{x}}} \tag{2.3}$$

where $p(x|\mathbf{x})$ is the transition probability of visiting token x given the memory $\mathbf{x}$; θ_x is the probability at which token x is selected among those that belong to the same group; $\lambda_{b_x b_{\mathbf{x}}}$ is the transition probability from memory group $\mathbf{x}$ to token x. This model can be learnt by the maximum likelihood method. And community structure is found by finding the most appropriate way for dividing the blocks.

Note that this method can also be applied on dynamic network where nodes do not change but the edges are changed with time.

Chapter 3

Preliminaries

In this thesis, we study the discrete time series [1]. We represent time series as sequences and do not care about the specific time stamp of each point. Formally, the time series is represented as a multi-set $T = \{t_1, t_2, \ldots, t_N\}$ with N independent observations of sequences. Each sequence $t_i = (v_{i_0}, v_{i_1}, \ldots, v_{i_l})$ contains several ordered vertices and each vertex could be seen as a state. And the sequence could be seen as the transitions on these states. For example, if the time series data is the trajectories of taxis, each vertex is the geometric position of the taxi and each sequence shows the movements of a taxi from one place to another place and so on. Other examples include the click paths of users in the Web, the chains of molecular interactions in a cell or itineraries of passengers in a transportation network. In the following parts of the thesis, we assume that we are given the time series data T and we use *time series* and *sequences* samely to refer to the time series data.

For the given time series, the important task is to describe the transition property of these sequences. Viewing the vertices as states which is the realization of a random variable, we can use (discrete) Markov model to describe the transition property and the generation probability of these sequences. The assumption of Markov model is the future state only depends on the current node or several nodes before the future node, regardless of the histories that occurs before. Furthermore, we will see that the Markov model is also the foundation of the Infomap commuinity detection algorithm we will discuss in this thesis. So before introducing how to detect communities from time series, we firstly review the Markov model which is used by the Infomap to capture the transition properties of the sequences.

[1] Time series: https://en.wikipedia.org/wiki/Time_series

3.1 Model Flows: Markov Models

3.1.1 First-Order Markov Model.

The stardard (first-order) Markov model assumes that the future state only depends on the current state, regardless of the previous states, such that

$$P(v_i|v_1 \to \ldots \to v_{i-1}) = P(v_i|v_{i-1}) \tag{3.1}$$

where $P(v_i|v_{i-1})$ is the transition probability from node v_{i-1} to node v_i. $P(v_i|v_{i-1})$ could be computed from the given time series data T by maximizing the likelihood (e.q. 3.2),

$$P(v_i|v_{i-1}) = \frac{|(v_{i-1} \to v_i) \in T|}{\sum_{u \in V} |(v_{i-1} \to u) \in T|} \tag{3.2}$$

where V is the set that contains all the vertices appeared in the time series T and $|(v_{i-1} \to v_i) \in T|$ is the number of the transition $(v_{i-1} \to v_i)$ appeared in T.

And using the chain rule, the generation probability of sequence $t_i = (v_1 \to v_2 \to \ldots \to v_l)$ is

$$\begin{aligned}
&P(v_1 \to v_2 \to \ldots \to v_l) \\
&= P(v_1 \to v_2 \to \ldots \to v_{l-1}) * P(v_l|v_1 \to \ldots \to v_{l-1}) \\
&= P(v_1 \to v_2 \to \ldots \to v_{l-1}) * P(v_l|v_{l-1}) \\
&= P(v_1 \to v_2 \to \ldots \to v_{v-2}) * P(v_{l-1}|v_1 \to v_2 \to \ldots \to v_{v-2}) * P(v_l|v_{l-1}) \\
&= P(v_1 \to v_2 \to \ldots \to v_{v-2}) * P(v_{l-1}|v_{v-2}) * P(v_l|v_{l-1}) \\
&\ldots \\
&= P(v_1) * P(v_2|v_1) * \ldots * P(v_{l-1}|v_{v-2}) * P(v_l|v_{l-1})
\end{aligned} \tag{3.3}$$

Because the sequences $\{t_1, t_2, \ldots, t_N\}$ in T are independent observations, the generation probability of the time series set T is the multiplication of the generation probability of all sequences t_i (eq. 3.4)

$$P(T) = \prod_{j=1}^{N} P(p_j) \tag{3.4}$$

where $P(p_j)$ is the generation probability of sequence p_j which is computed by equation 3.3.

3.1.2 High-Order Markov Model

Sometimes, only considering the current state is not enough to capture the transition property of sequences. For example, we can construct two time series sets that exhibit totally different transition properties. But the first-order Markov model is not enough to capture their different transition properties. These two time series sets are constructed as following. The first time series set T_1 contains two independent sequences on five nodes: $T_1 = \{t_1 = (`A' \rightarrow `C' \rightarrow `B' \rightarrow `A'), t_2 = (`E' \rightarrow `C' \rightarrow `D' \rightarrow `E')\}$. The second sequence set T_2 contains one sequence on the same node set: $T_2 = \{t = (`A' \rightarrow `C' \rightarrow `D' \rightarrow `E' \rightarrow `C' \rightarrow `B' \rightarrow' A')\}$. If we describe these two time series sets using the first-order Markov model, T_1 and T_2 both have transitions $(`A' \rightarrow `C')$, $(`C' \rightarrow `B')$, $(`B' \rightarrow `A')$, $(`E' \rightarrow `C')$, $(`C' \rightarrow `D')$ and $(`D' \rightarrow `E')$. And the transition probability are totally same in these two data sets (e.g. the transition $(`A' \rightarrow `C')$ has same probability to happen in these two time series sets). However, these two data sets T_1 and T_2 have different transition property at all. In the first data set T_1, there are two sequences that travels different areas of the graph (t_1 in T_1 travels nodes $`A', `C', `B'$ and $t_2$ in $T_1$ travels nodes $`E', `C', `D'$). However, in the second data set T_2, there is only one sequence that travels through all nodes of the graph. So for this example, the first-order Markov model cannot capture the different transition properties of these two time series sets. The key here is only considering the current node is not enough to determine the next state for these two data sets. To determine the next state, we not only have to consider the current state, but also the previous states before current state. For example, if we consider the current state and the previous one state, we can distinguish the different transition properties of these two time series sets. For example, let us look at the first time series set T_1 and suppose the person is now on node $`C'$. Then the person will go to node $`B'$ if this walker comes to $`C'$ via $`A'$ or this person will go to $`D'$ if this person comes to $`C'$ via $`E'$. However, if we look at the second data set, the person will go to $`D'$ if this walker comes to $`C'$ via $`A'$ or this walker will go to $`B'$ if this walker comes to $`C'$ via $`E'$. We use this example to show the demand of high-order Markov model.

The key point of high-order Markov model is that the next state not only depends on the current state, but also depends on the the previous several states. In a order k Markov model, the next state v_{i+1} is determined by the current state v_i and previous $k-1$ states (eq 3.5).

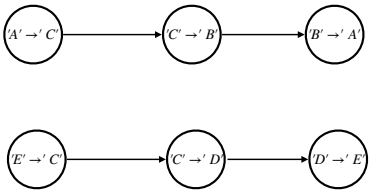
$$P(v_{i+1}|v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_i) = P(v_{i+1}|v_{i-k+1} \rightarrow \ldots \rightarrow v_i) \tag{3.5}$$

And the transition probability $P(v_{i+1}|v_{i-k+1} \rightarrow \ldots \rightarrow v_i)$ is

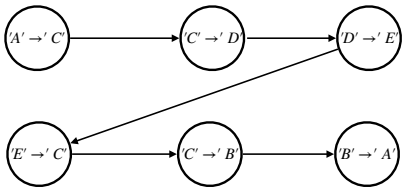
$$P(v_{i+1}|v_{i-k+1} \rightarrow \ldots \rightarrow v_i) = \frac{|(v_{i-k+1} \rightarrow \ldots \rightarrow v_{i-1} \rightarrow v_i) \in T|}{\sum_{u \in V} |(v_{i-k+1} \rightarrow \ldots \rightarrow v_{i-1} \rightarrow u) \in T|} \tag{3.6}$$

The generation probability of an abserved path $p = (v_1, \ldots, v_l)$ could be derived using the chain rule samely. And the likelihood to generate the data set is the production of probability to generate each sequence in the data set (eq 3.4).

Finally let us explain the relation between higher-order Markov model and graph abstraction of the time series data. For the stardard Markov model ($k = 1$), the corresponding graphical representation is a graph whose nodes are the same vertices appeared in the sequences and the weight of each (directed) edge is the transition probability between these two vertices (computed by equation 3.1). For the high-order Markov model with order k ($k > 1$), its graphical representation is the high-dimensional *De Bruijn graphs* [2]. The nodes in the *De Bruijn graphs* are $k - th$ order vertices which corresponds to a subpath with length $k - 1$ and he transition $P(v_{i+1}|v_{i-k+1} \rightarrow \ldots \rightarrow v_i)$ is representated as an edge between two $k - th$ order vertices $(v_{i-k+1} \rightarrow \ldots \rightarrow v_i)$ and $(v_{i-k+2} \rightarrow \ldots \rightarrow v_{i-1} \rightarrow v_i)$. And the weight of this edge is the transition probability between this two high-order nodes. The $k - th$ order node $(v_{i-k+1} \rightarrow \ldots \rightarrow v_i)$ represents the person is now on vertex v_i and the person comes to v_i from $(v_{i-k+1} \rightarrow \ldots \rightarrow v_{i-1})$. For the previous time series sets T_1 and T_2, their first-order graphical representation are the same. However, their second graphical representations are different (shown in figure 3.1). Using the $2 - th$ order Markov, we can distinguish the different transition properties of T_1 and T_2.



(a) The 2-order *De Bruijn graph* (corresponding to the second-order Markov model) of T_1.



(b) The 2-order *De Bruijn graph* (corresponding to the second-order Markov model) of T_2.

Figure 3.1: The 2-order *De Bruijn graph* of T_1 and T_2.

3.1.3 Multi-Order Markov Model.

While high-order Markov model has shown its power to capture the high-order transition correlations existed in the time series, it ignores the first several transitions of each sequence [11]. This could be problematic if the sequences are short as high-order Markov model discards the infomation of the first several transitions of each sequence. Let us take an extreme case as example. Suppose we have a short sequence $t = (v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5)$

and we use the high-order Markov model with $k = 3$ to describe this sequence. Then this sequence only contains two transitions $(v_1 \rightarrow v_2 \rightarrow v_3) \rightarrow (v_2 \rightarrow v_3 \rightarrow v_4)$ and $(v_2 \rightarrow v_3 \rightarrow v_4) \rightarrow (v_3 \rightarrow v_4 \rightarrow v_5)$. The transitions within the first 3 vertices $(v_1 \rightarrow v_2 \rightarrow v_3)$ are ignored. In fact, the real situation to generate this sequence is: the walker first goes to the first node v_1, and then the walker goes to the second node v_2 based on its previous node v_1, then the walker goes further to v_3 based on previous two states v_1 and v_2 and so on. To overcome this problem, Ingo Scholtes develops the multi-order Markov model [11]. Specifically, the multi-order model whose maximal order is K uses multiple k-th order modes ($k = 0, 1, \ldots, K$) to capture the first K transitions and then use $K - th$ order Markov model to catpure the remaining transitions. Suppose the transition probability of $k - th$ order Markov mode is $P^{(k)}$ (equation 3.5), then the probability $\bar{P}^{(K)}$ to generate a path $(v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_l)$ with multi-order Markov model with maximal order K is:

$$\begin{aligned}\bar{P}^{(K)}(v_0, v_1, \ldots, v_l) = &\prod_{k=0}^{K} P^{(k)}(v_k | v_0 \rightarrow \ldots \rightarrow v_{k-1}) \\ &\prod_{i=K+1}^{l} P^{(k)}(v_i | v_{i-K} \rightarrow \ldots \rightarrow v_{i-1})\end{aligned} \tag{3.7}$$

where $P^{(k)}$ is computed with equation 3.5 and 3.6. And the likelihood to generate the given time series T using multi-order Markov model with maximal order K is:

$$\bar{P}(T) = \prod_{p_j \in S} \bar{P}^{(K)}(p_j) \tag{3.8}$$

where $\bar{P}^{(K)}$ is from equation 3.7.

3.2 Compress Flows: Infomap

The idea of Infomap algorithm is to cluster nodes to different clusters and assigns each cluster a unique id. Furthermore, the nodes within every cluster are given different ids (Note that the nodes in different communities could have same id). Using this two-level coding scheme [7] [2], each node is encoded by its community code and the code inside the community. Because the time series data is ordered sequences of vertices, the time series could

[2]Later, Rosvall et. al. generalized it to multi-level cases [3]. The idea of multi-level coding scheme is similar to the two-level coding scheme and we consider the two-level case in this thesis.

also be encoded by this two-level coding scheme. Specifically, when the walker enters into a new community, Infomap uses the id of the new community to indicate that this sequence enters to this community and samely if the random walker leaves a community, Infomap uses a code of this community to indicate that this sequence leaves this community. Besides, everytime the walker reaches a new vertex, the (within cluster) id of this vertex is used to indicate the position of the walker within each community. For different parititions of the nodes, the sequences will have different description lengths. Rosvall et. al find that finding community structure of the graph corresponds to minimizing the description length [7]. So the community detection problem is changed to find the best clustering which minimizes the description length of the sequences.

Infomap uses the map equation (equation 3.9) to compute description length. In order to compute the description length, we need to know the probability to enter and exit each community and the probability to visit each node. Infomap assumes that the sequences follow the first-order or high-order Markov assumption. So the probability to enter and exit each community and the probability to visit each node can be computed by the Markov model. In the following, we will review the stardard (first-order) Infomap [7] which relies on the first-order Markov model to compute the transition probability between state nodes and the high-order Infomap [3] which relies on the high-order Markov model to compute the transition probability between state nodes.

3.2.1 First-Order Infomap

First-order Infomap assumes that the sequences follow the first-order Markov assumption such that the next state only depends on the current state, regardless of the previous states (see the discussion in section 3.1.1). Suppose the vertex set V of a graph contains n vertices $v_1, v_2, \ldots, v_n$. We can partition these n vertices $v_1, v_2, \ldots, v_n$ into m non-overlapped communities $M_1, M_2, \ldots, M_m$ (community M_i containing the nodes assigned to this module is a subset of V). Stardard Infomap shows that finding the community structure of the graph is equal to search the best $\mathbf{M}$ which minimizes the average description length per step. The description length is computed by the following map equation (3.9) [3].

$$L(\mathbf{M}) = q_{\curvearrowleft} H(Q) + \sum_{i=1}^{m} p_i^{\circlearrowright} H(P^i) \tag{3.9}$$

[3] At the earliest paper [7], the authers do not disginguish the enter probability $q_{\curvearrowleft}$ and exit probability $q_{\curvearrowright}$. And the author use $q_{\curvearrowleft}$ and $q_{\curvearrowright}$ samely to represent the switch probability of modules. But in their later papers [3], the authors use $q_{\curvearrowleft}$ to compute the entropy between modules and $q_{\curvearrowright}$ inside the entropy of movements within modules. We disginguish $q_{\curvearrowleft}$ and $q_{\curvearrowright}$ in this thesis

The map equation (equation 3.9) contains two terms: the first term is the entropy of the movements between different communities and the second item is the entropy of movements within modules. Here $q_{\curvearrowleft}$ is the probability that the random walker enters a new module per step. The probability of entering a new module $q_{\curvearrowleft}$ is computed as $q_{\curvearrowleft} = \sum_{i=1}^{m} q_{i\curvearrowleft}$, where $q_{i\curvearrowleft}$ is the probability of entering module M_i ($i \in [1, m]$). And $q_{i\curvearrowleft}$ is computed as

$$q_{i\curvearrowleft} = \sum_{v_j \notin M_i} \sum_{v_k \in M_i} \pi_{v_j} p_{v_j v_k} \tag{3.10}$$

π_{v_j} is the ergodic visit probability of node v_j and $p_{v_j v_k}$ is the probability going to node v_k from node v_j. $p_{v_j v_k}$ is computed by the first-order Markov model (equation 3.2) and the ergodic visit probability of every node is gotten by the power method based on the transition probability between any to state nodes. $H(Q)$ is the entropy between different modules and $H(Q)$ is computed by equation 3.11

$$H(Q) = \sum_{i=1}^{m} \frac{q_{i\curvearrowleft}}{\sum_{j=1}^{m} q_{j\curvearrowleft}} log(\frac{q_{i\curvearrowleft}}{\sum_{j=1}^{M} q_{j\curvearrowleft}}) \tag{3.11}$$

$p_i^{\circlearrowleft}$ is the fraction of within-module movements that occurs in module M_i, plus the probability of exiting module M_i such that

$$p_i^{\circlearrowleft} = \sum_{v_j \in M_i} \pi_{v_j} + q_{i\curvearrowright} \tag{3.12}$$

where $q_{i\curvearrowright} = \sum_{v_j \in M_i} \sum_{v_k \notin M_i} \pi_{v_j} p_{v_j v_k}$ is the probability fo exiting module Mi. π_{v_j} is still the ergodic node visit probability of node v_j and $p_{v_j v_k}$ is the probability going to node v_k from node v_j. The reason to add the probability $q_{i\curvearrowright}$ of exiting module M_i into the fraction of within-module movements $p_i^{\circlearrowleft}$ is when the random walker exits a module, we assign an exit code to indicate that the walker is leaving this module. It is equivalent to add a special node to this module which indicates the walker will leave this module in the next step. And the entropy of within movements $H(P^i)$ is

$$\begin{aligned} H(P^i) = & \frac{q_{i\curvearrowright}}{q_{i\curvearrowright} + \sum_{v_k \in i} \pi_{v_k}} log(\frac{q_{i\curvearrowright}}{q_{i\curvearrowright} + \sum_{v_k \in i} \pi_{v_k}}) + \\ & \sum_{v_j \in i} \frac{\pi_{v_j}}{q_{i\curvearrowright} + \sum_{v_k \in i} \pi_{v_k}} log(\frac{\pi_{v_j}}{q_{i\curvearrowright} + \sum_{v_k \in i} \pi_{v_k}}) \end{aligned} \tag{3.13}$$

3.2.2 High-Order Infomap

Infomap uses the map equation (equation 3.9) to compute the description length for the flows. In the map equation, we need the enter/exit probability of each module and the ergodic visit of each node. The first-order

Infomap assumes the flows follow the first-order Markov assumption. The enter/exit probability of each module and the ergodic visit of each node are computed based on first-order Markov model. As discussed in the section 3.1.2, sometimes high-order Markov Model could better capture the transition property of the time series. So Rosvall et. al. considered to generalize the Infomap to high-order Markov model [3]. The details are following.

For the high-order Infomap [3], the framework remains the same. We want to find the partition $\mathbf{M}$ which minimizes the map equation (3.9). The difference is that high-order Infomap operates on the high-order state nodes [4]. We use $\{\alpha, \beta, \gamma, \ldots\}$ to represent the high-order state nodes to differentiate them with the original vertex set $V = (v_1, v_2, \ldots)$. High-order Infomap partitions the state nodes $\{\alpha, \beta, \gamma, \ldots\}$ to m non-overlapped communities and find the best partition $\mathbf{M}$ which minimizes the map equation (equation 3.9). Accordingly, the enter/exit probability of each module and the ergodic visit probability of each state nodes, that are used in the map equation, are computed based on the corresponding high-order Markov model.

In the k-order Markov model, the state node $\alpha = (v_{i-k+1} \to \ldots \to v_i)$ corresponding to a subpath in the time series represents the walker is now on node v_i and the walker comes to node v_i from $(v_{i-k+1} \to \ldots \to v_{i-1})$. In order to distinguish the state nodes $\{\alpha, \beta, \gamma, \ldots\}$ and the original nodes $\{v_1, v_2, \ldots\}$ in sequences, we call the original nodes as physical node. We say state node α belongs to physical node v_i ($\alpha \in v_i$) iff the last node in the state node $\alpha = (v_{i-k+1} \to \ldots \to v_i)$ is v_i. And each physical node v_i could have more than one state node and these different state nodes of the same physical node v_i represent the walker could reach physical node v_i via different paths. In the high-order Markov model, we can compute the ergodic visit probability of each state node $\alpha = (v_{i-k+1} \to \ldots \to v_i)$ as we have the transition probabilities between these state nodes. And the ergodic visit probability of each physical node π_{v_i} is the sum of its state nodes $\pi_{v_i} = \sum_{\alpha \in v_i} \pi_\alpha$.

Now let us see the details how to compute the probabilities in map equation using state nodes. For the first term in eq (3.9), we can directly compute the probability of entering a new module $q_{\curvearrowleft}$ by summing the probability of entering each module $q_{\curvearrowleft} = \sum_{i=1}^{M} q_{i\curvearrowleft}$, where the probability of entering module i is

$$q_{i\curvearrowleft} = \sum_{\alpha \notin M_i} \sum_{\beta \in M_i} \pi_\alpha p_{\alpha\beta} \tag{3.14}$$

π_α is the ergodic visit probability of state node α. $p_{\alpha\beta}$ is the transition probability from state node α to state node β. And the entropy between modules $H(Q)$ is same as eq (3.11).

[4] The high-order state nodes are introduced in section 3.1.2 (The nodes in the high-dimensional *De Bruijn graphs*).

For the second term, we need to pay attention that the state nodes of the same physical node could be partitioned into different communities. In this case, this physical node is clustered into different communities and we need to compute the ergodic visit probability of this physical node in different communities. For example, the probability of visiting physical node v_i in module M_m is the sum of all state nodes belonging to physical node v_i that are in module M_m, such that

$$\pi_{v_i \in M_m} = \sum_{\alpha \in v_i \cap \alpha \in M_m} \pi_\alpha \tag{3.15}$$

And the the fraction of within-module movements $p_i^{\circlearrowleft}$ that occurs in module M_i is the sum of probability of visiting each physical in this module plus the probability to exit module M_i

$$p_i^{\circlearrowleft} = \sum_{v_k \in M_i} \pi_{v_k \in M_i} + q_{i\curvearrowright} \tag{3.16}$$

$\pi_{v_k \in M_i}$ is computed by eq (3.15). $q_{i\curvearrowright}$ is probability of exiting module M_i and it is computed as

$$q_{i\curvearrowright} = \sum_{\alpha \in M_i} \sum_{\beta \notin M_i} \pi_\alpha p_{\alpha\beta} \tag{3.17}$$

And the entropy of within movements $H(P^i)$ is

$$\begin{aligned} H(P^i) = & \frac{q_{i\curvearrowright}}{\sum_{v_k \in M_i} \pi_{v_k \in M_i} + q_{i\curvearrowright}} log(\frac{q_{i\curvearrowright}}{\sum_{v_k \in M_i} \pi_{v_k \in M_i} + q_{i\curvearrowright} + \sum_{\beta \in M_i} \pi_\beta}) + \\ & \sum_{v_k \in M_i} \frac{\pi_{v_k \in M_i}}{q_{i\curvearrowright} + \sum_{v_j \in M_i} \pi_{v_j \in M_i}} log(\frac{\pi_{v_k \in M_i}}{q_{i\curvearrowright} + \sum_{v_j \in M_i} \pi_{v_j \in M_i}}) \end{aligned} \tag{3.18}$$

3.2.3 Comparison of Infomap with different orders

In summary, the framework of first-order Infomap and the framework of high-order Infomap are same. First-order Infomap and high-order Infomap both compute the probability of entering each module, the probability of exiting each module and the ergodic visit probability of each node in different communities. Based on these probabilities, they compute the entropy of movements between different modules and the entropy within each community. However, first-order Infomap relies on first-order Markov model to compute the visit probability of each node and the transition probability between different nodes. High-order Infomap operates on the high-dimensional state nodes. It is built on high-order Markov model to compute the visit probability of each state node and the transition probability between different state nodes. The difference between first-order Infomap and high-order Infomap is summarized as following:

- When computing the probability of entering each module and the probability of exiting each module, the first-order Infomap operates on the physical nodes while high-order Infomap operates on high-order state nodes.

- The first-order Infomap returns non-overlapped communities while the communities returned by high-order Infomap could overlap. The reason is that the high-order Infomap operates on the high high-dimensional state nodes. While the communities of state nodes in high-order Infomap are non-overlapped, state nodes of same physical node could be clustered into different communities. So finally the physical nodes could appear in these different communities.

- First-order Infomap and high-order Infomap both need to compute the probability to visit the nodes in each community (in the second term in eq. (3.9)). When computing the probability of visiting the node v_k in module M_i, the high-order Infomap sums the probability to visit all state nodes in module M_i that belongs to node v_k.

3.3 Discussion about the Description Length

In the original proposal of this thesis, the second task is to integrate the high-order Infomap with the multi-order Markov framework. We know if we fix the order of the Infomap, we can use this Infomap to find a optimal partition which minimizes the description length (equation 3.9). We are interested to know whether the Infomap with different orders has different minimal description lengths. Furthermore we know that the order of the path could be detected by the model selection method implemented `PATHPY`. If the Infomap with different orders has different minimal description lengths, we are particularly interested in the question that whether the optimal order K_{opt} detected by the model selection method implemented in `PATHPY` corresponds to the order of Infomap which gives us the minimum description length.

For this problem, we once thougth that the Infomap with K_{opt} should gives us the minimum description length. However I find the realilty is not as we expected when I conduct the experiments proposed in the proposal. Let us use an example for detailed explanation.

We construct the following $2-th$ order pathway data set as:$T_1 = \{t_1 = (a1 \to 1 \to a2 \to a1 \to 1 \to a2), t_2 = (b1 \to 1 \to b2 \to b1 \to 1 \to b2), t_3 = (a1 \to 1 \to b2 \to b1 \to 1 \to a2 \to a1 \to 1 \to b2 \to b1)\}$. This data set contains three paths such that the first path t_1 travels on nodes $\{a1, a2, 1\}$, the second path t_2 travels on nodes $\{b1, b2, 1\}$ and the third path t_3 travels on all nodes $\{a1, a2, 1, b1, b2\}$. We set the frequency of the first path t_1 and the second path t_2 to 10 and the frequency of the third path to 1. In fact this data set corresponds to figure 4.2a.

Using the order detection method implemented in the `PATHPY`, the optimal order of this pathway data set is 2. And we test the Infomap with $order = 1, 2, 3, 4$ on this data set. The optimal partition together with the minimum description length corresponding to each order is summarized in table 3.1. We can see that the description length corresponding to 4-order Infomap is smaller than the description length corresponding to 2-order Infomap. So the answer to the our question is that the optimal order detected by the `PATHPY` may not be the order for Infomap which has smallest description length. The reason is that the high-order Infomap oper-

ates on the state node space, [5] and the Infomap with different orders has different state node space. Even if the optimal partitions of Infomap with $order = 2, 3, 4$ are same (as shown in hte third column in table 3.1), their description lengths could be different. And the Infomap with K_{opt} may not give us the minimal description length.

Infomap for pathway with different orders		
Infomap order	minimum score	corresponding partition
1	2.789	{ {'a1', 'b1', 'a2', 'b2', '1'} }
2	2.511	{ {'b1', 'b2', '1'}, {'a1', 'a2', '1'} }
3	2.528	{ {'a2', '1', 'a1'}, {'1', 'b2', 'b1'} }
4	2.496	{ {'b2', 'b1', '1'}, {'a1', 'a2', '1'} }

Table 3.1: Infomap with different orders for 2-order pathway data set. We can find that order (order = 4) of Infomap which corresponding to the minimum description length is not same as the K_{opt} detected by the `PATHPY`.

In the original paper of Infomap [7], the authors state that finding the community structure of the graph corresponds to minimizing the description length. So the description length is used to measure the quality of different partitions. **We argue we need to add a contraint to this statement that the description length can only act the measurement for different partitions corresponding to the Infomap with the fixed order**. If we change the order of Infomap, we cannot use the description length as the measurement to determine which order gives us the best partition of the nodes. The reason is explained before that the description length is computed based on the state nodes. If we change the order for Infomap, we will have different state nodes and we cannot compare the description length corresponding to different state nodes. Just like the previous example, the best partitions corresponding to Infomap with $order = 2, 3, 4$ are the same. However the description lengths are different. So we cannot use the description length to determine the quality of partitions of Infomap with different orders.

Next we use another counterexample for further verification. The idea is to construct a path data set without community structure. Suppose the time series are totally random and do not exhibit community structures. So if we apply Infomap with different orders, the partitions found by these models should be equally 'bad'. And it is meaningless to compare the communities found by Infomap with different orders for this data set. As shown in figure 4.3, if we increase the order of the Infomap, there is a trend that the coding lenght decreases. However we cannot say the partition of Infomap with higher order is better because the sequences do not have community structure at all.

[5] The ordinary Infomap could also be seen as the special case of high-order Infomap with 1-order.

High-order Infomap for pathway without Community Structure		
	Infomap order	best score
pathway order 1	1	5.371
	2	4.389
	3	3.933
	4	3.535
	5	3.443
	Infomap order	best score
pathway order 2	1	4.652
	2	3.767
	3	3.516
	4	3.224
	5	3.182

Table 3.2: High-order Infomap on sequences without community structure. The path data set is generated in the following way: 1) generate the random graph without community structure; 2) generate the sequences on the random graph with specified order. The path data set contains 10 independent sequences and the maximum length is 8. And the code is from Ingo [11]. The best score is the minimum description length in 20 random experiments. The maxmum number of optimization iteration in each turn is 50000. The results support our argument that the description length cannot be used to determine which order yeilds the best partition of the graph. And we can further observe that the description length tends to decrease when we increase the order of Infomap.

In a conclusion, the description length cannot be used to measure the quality of partitions of Infomap with different orders. The reason is that the map equation is computed based on the state nodes and changing the order for Infomap will change the state nodes. And even the partitions of Infomap with different orders are same, the description length corresponding to these same partitions could be different. So we need to directedly examine the partitions found by the Infomap. But for the second task in initial proposal, we still find something that if the order of Infomap is smaller than the order detected by `PATHPY`, the Infomap may not find the true community structure. The reason is that if the order of Infomap is smaller than the true order, the transitions of the state nodes in Infomap cannot capture the properties of the pathway. So we can integrate the `PATHPY` and high-order Infomap in a way such that we can consider to first use the order detection method in `PATHPY` to detect the order for Infomap and then apply the Infomap with the detected optimal order.

Chapter 4

Multi-order Compression

Problem Description Our aim is to find "nature" communities from the time series data. The formal problem description is following. Suppose we know N independent observations of time series represented as a multi-set $T = \{t_1, t_2, \ldots, t_N\}$ where each sequence $t_i = (v_{i_0}, v_{i_1}, \ldots, v_{i_l})$ is a series of transitions between several vertices. The vertex set $V = \{v_1, \ldots, v_n\}$ contains the vertices appeared on all sequences. The task is to find a partition $\mathbf{M}$ of the nodes $V = \{v_1, \ldots, v_n\}$ to different communities $M_1, M_2, \ldots, M_m$ (each community M_i is a subset of V) such that most transitions of the sequences happen within the communities and these sequences are less likely to go across different communities. Note that the found communities could overlap such that a node could belong to several different communities.

4.1 Motivation

In section 3.2, we introduced the basic idea and framework of the stardard (first-order) and high-order Infomap. On the one hand, Infomap relies on the corresponding Markov model to capture the transition property of the sequences. For example, the k-th order Markov model uses k-th order nodes to capture the transition of the sequences. The k-th order node $\alpha = (v_1, v_2, \ldots, v_k)$ means the walker is now on node v_k and this walker reaches v_k from $(v_1, v_2, \ldots, v_{k-1})$. The corresponding k-order Infomap considers the transitions between the k-th order state nodes. For example, suppose the sequence is $t = (v_1 \to v_2 \to v_3 \to v_4 \to v_5)$. The transitions in the high-order Infomap with $order = 3$ are $(v_1 \to v_2 \to v_3) \to (v_2 \to v_3 \to v_4) \to (v_3 \to v_4 \to v_5)$. On the other hand, as shown in 3.1, the stardard markov model cannot capture the high-order correlation existed in the sequences. The high-order Infomap ignores the first several movements and could be problematic for the short sequences. Considering these two points, we want to generalize the Infomap to the multi-order Markov model which could capture the transition property of the sequences better.

Besides, the stardard/high-order Infomap is designed to detect comminities on the static network. It *imagines* a probability flow on the graph and the transition probabilty of the flow is based on the weights of out-edges of current nodes. It operates on the network and only uses the imagined flow as a proxy. However,

sometimes we are given the real time series sequences instead of the graph. So we want to detect the communities directly from the sequences, without explicitly constructing the underline graph. In the following, we will first describe the multi-order Infomap model. Then, we will use examples and experiments to show how it works on the time series data.

4.2 Multi-order Infomap: Model Formulation

State Nodes in Multi-order Infomap. In order to build the Infomap on multi-order Markov model, we take into account all layers of the multi-order Markov model [11] at once. And this is the main difference with the existed high-order Infomap. Specifically, for the multi-order Infomap with $order = k$, we use the low order nodes in layer $(1, \ldots, k-1)$ of the multi-order Markov model to describe the first k transitions of each sequence. We use the nodes in k-th layer of the multi-order Markov model to describe the remaining transitions. And the state node has same meaning as before such that the state node $\alpha = (v_1, v_2, \ldots, v_{k-1}, v_k)$ means that the walker is now on physical node v_k and this walker comes to v_k from physical nodes $v_1, v_2, \ldots, v_{k-1}$. Consider the previous path $t = (v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_5)$. The multi-order Infomap with $order = 3$ views this path as state transitions $(v_0) \rightarrow (v_0, v_1) \rightarrow (v_0, v_1, v_2) \rightarrow (v_1, v_2, v_3) \rightarrow (v_2, v_3, v_4)$. Besides, in order to distinguish the high order nodes with the original nodes, we denote the high order nodes as state nodes and denote the original nodes as physical nodes. This notation the same as the notation in [3]. And we denote the state node set of the multi-order Infomap with $order = k$ with $S^{(k)}$. So the state node set $S^{(3)}$ in the previous case is $S^{(k)} = \{(v_0), (v_0, v_1), (v_0, v_1, v_2), (v_1, v_2, v_3), (v_2, v_3, v_4)\}$.

As for every state node α in $S^{(k)}$, its ergodic visit probability π_α is computed by the multi-order Markov model. Recall that the state node $\alpha = (v_1, v_2, \ldots, v_{k-1}, v_k)$ means that the walker is now on physical node v_k and this walker comes to v_k from physical nodes $v_1, v_2, \ldots, v_{k-1}$. The probability to visit physical node v_i is the sum of probability to visit all state nodes whose last physical node is v_i.

$$\pi_{v_i} = \sum_{\alpha \sqsubseteq v_i} \pi_\alpha \tag{4.1}$$

where $\alpha \sqsubseteq v_i$ means the last physical node of state node α is v_i.

The Map Equation. Infomap [7, 3] uses the fact that the random walker in a graph tends to be trapped within each community. It partitions nodes to different modules and assign a unique code to each module. The nodes within each module are also assigned with different codes to indicate the state of the walker within this module [1]. And a special codeword, the exit code, is chosen as part of the within-cluster coding to indicate the walk will leave this module in the next movement. Infomap uses this two-level coding scheme - the module code and node id - to encode the random path. If the movement happens within a module, the code of the new node is used to indicate the new state of the walker within this module. If the movement is between two modules, we first use the exit code to indicate that the walker is leaving the module, and use the code of new module and the code of the node in

[1] Note that the nodes in different modules could have same code.

the new module to indicate the new state of the walker. This coding is illustrated by figure 4.1.

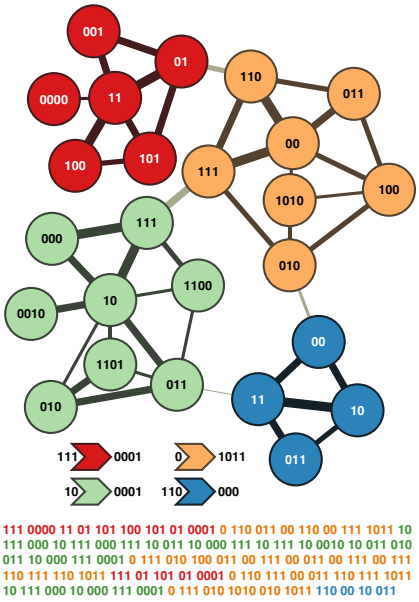


Figure 4.1: The illustration of coding the sequence with the two-level coding scheme. This figure is from [7].

Under this coding scheme, the average description length per step for the sequence is computed using the map equation 4.2 based on the ergodic visit probability of each node and the transition probability bewteen different communities. And detecting the "nature" communities is turned to a problem to find the partition which minimizes the description length [7].

When generalizing the Infomap to the multi-order Markov model, we keep the framework unchanged. We still use the two-level coding scheme and search the partition of the state nodes which minimizes the average description length for the given sequence. However, we need to build the map equation based on the state nodes in all layers of the corresponding multi-order Markov model.

$$L(\mathbf{M}) = q_{\curvearrowleft} H(Q) + \sum_{i=1}^{M} p_i^{\circlearrowleft} H(P^i) \tag{4.2}$$

Multi-order Infomap operates on the state nodes. It partitions the state nodes $S^{(k)}$ of the sequences T in all layers of the k-order multi-order Markov model to m different (non-overlapped) communities. It is equivalent to apply the stardard (first-order) Infomap on the state nodes $S^{(k)}$. However, the aim is to find the communities of the original (physical) nodes instead of the state nodes. So we need to map the partition of the state nodes to the partition of the physical nodes. We know that each physical node v_i could have more than one state nodes whose last physical node is

this physical nodes. If the state nodes of the same physical nodes are partitioned to different communities, it means that this physical node belongs to these different communities. So the final communities of the physical nodes returned by the multi-order Infomap could overlap.

In the multi-order Infomap, if we partition different state nodes of physical node v_i to one module M_m, the probability to visit physical node v_i is the sum of probability to visit its all state nodes in this module.

$$\pi_{v_i} = \sum_{\substack{\alpha \sqsubseteq v_i \\ \alpha \in M_m}} \pi_\alpha \tag{4.3}$$

where π_α is the ergodic visit probability of state node α that is computed from the Multi-order Markov model. Specifically, we learn the transition probability $P_{\alpha\beta}$ between state nodes α and β in the multi-order Markov model by

$$P_{\alpha\beta} = \frac{w_{\alpha\beta}}{\sum_{\gamma \in \alpha.successor} w_{\alpha\gamma}} \tag{4.4}$$

where $w_{\alpha\beta}$ is the number of such transition from state α to state β appeared in all sequences. And we can use the power method to compute the ergodic visit probability $\pi_\alpha = \pi_{\beta_j} P_{\beta_j \alpha}$. Similar to PageRank algorithm, a damping factor is introduced to guarantee a unique steady state distribution for the directed network. The damping factor is set to 0.85.

After having the ergodic visit probability of state nodes and physical nodes, we can compute the description length for the given partition **M**. In the map equation (equation 4.2), the probability $q_{\curvearrowleft}$ of entering a module at any step is the sum of the probability to enter each module $q_{\curvearrowleft} = \sum_i q_{i\curvearrowleft}$. And $q_{i\curvearrowleft}$ is computed by plugging the transition probability (equation 4.3) into equation 3.14. After having $q_{i\curvearrowleft}$, the entropy of movements between modules $H(Q)$ can be computed via equation (3.11). For the second term in the map equation (equation 4.2), the within-module movement $p_i^{\circlearrowleft}$ is computed by plugging the ergodic visit probability and the transition probability (equation 4.4) in equation 3.16 and equation 3.17. And the entropy of within movements $H(P^i)$ is computed via equation 3.18. The algorithm of Multi-order Infomap is summarized in Algorithm 1.

Additional assumptions Furthermore, we explicitly make two assumptions with respect to the first state node and the last state node in each sequence.

- When the walker reaches the first state node of a sequence, the walker enters the corresponding community of the first state node.
- When the walker enters the last state node of a sequence, the walker then exits the corresponding community of the last state node.

The rationale for this two assumptions is that the sequences are independent. If a walker appears on the first node of the path, we think the walker arrives this node from somewhere else, so the walker *enters* the corresponding cluster. And if the walker appears on the last node of the path, we think the walker will go to somewhere else, so the walker *exits* the corresponding module of the last node. Without these two assumption, we may encounter the computation problem that

zero appears in the logarithm operation. The reason is some modules may not have in-edge from other modules or may not have out-edge to other modules. So the entering probability or the exit probability for these modules are zero, which is problematic in computing the map equation (Equation 4.2).

Optimization In the Infomap, our objective is to find the best paritition of the nodes which minimizes the description length (equation 4.2). However, it is unrealistic to try every possible partition because of the huge searching space (NP hard). So we need to use approximation method to find the partition as good as possible. The optimization process operates in the following way. Each state node is assigned to a singles cluster at beginning. In each iteration of the optimization, we merge some nodes together and get a new partition of the state nodes. We accept the new partition if its corresponding description length is smaller or we can use the simulated annealing algorithm to determine whether to accept the new partition or not. We repeat this process until we cannot find a better partition of the state nodes. We have two strategies to merge the nodes in each iteration. And we will see these two strategies could produce different results.

The first strategy operates on the module level. It randomly chooses a module and merges all of the nodes on this module to another module. The advantage of this approach is it converges fast as we reduce one cluster at each iteration. Furthermore, we can use advanved data to avoid to re-compute the visit probabilities of all communities (we only need to change the visit probabilities of these two modules and the related modules). For example, for the flight data set, there are 286,810 independent sequences and 1,774 state nodes in the multi-order Infomap with $order = 2$. And the approach can find the partition in several minutes. However, this approach is easy to get caught into suboptimal solution. The reason is that each time we merge two communities and the merged nodes will always stay in the same communities. It is possible that we "wrongly" merge some nodes together in the early steps and get into the suboptimal results. For example, suppose there are four different communities C_1, C_2, C_3 and C_4 at begining and we know that if we merge C_1 with C_2 to a new community, C_3 and C_4 to a another community, the description length is minimum. However, it could happen that we first merge C_2 and C_3 because we randomly choose the communities to be merge and this merge gives us shorter coding description. Now we have C_1, $\{C_2, C_3\}$ and C_4 three communities. And we cannot further get shorter description length by merging these three communities. So we stick in the local optimum. In the experiment 4.1, the 2-order (multi-order) Infomap clusters all nodes into one community for the 2-order path. And this is a local minimum while the global minimum is found by the next approach which would be discussed in the following.

The second optimzation strategy operates on the node level. It randomly chooses a node at each iteration and merges this node to a different module. The advantage of this method is it can reduce the probability to stick in the local minimum if iteration is long enough. The reason is the search space of this approach is larger than the search space of first approach. Let us take the previous example. At begining we have C_1, C_2, C_3 and C_4 four communities and we know that if we merge the nodes in C_1 with C_2 to a new community and the nodes in C_3 and C_4 to a another community, the corresponding description length is minimum. In the early steps, even if we "wrongly" merge the nodes in C_2 and the nodes in C_3 together, we still have the chance to later choose the nodes in C_1 and the nodes in $\{C_2, C_3\}$ in later

iteration. So we still have the chance to reach the global minimum instead of the local minimum. However, the search space is very huge and it can only handle the small data sets. Besides, it is much more complicated to design the advanced algorithm to speed up the optimization process like the first approach. So we need to re-compute all the probabilities need in the map equation (equation (4.2)) and it takes much time.

Algorithm 1 Multi-order Infomap

1: Input: the sequences $T = \{t_1, t_2, \ldots, t_N\}$
2: Output: the partition of nodes $V = \{v_1, v_2, \ldots, v_n\}$ to m (overlapped) communities $M_1, M_2, \ldots, M_m$.

3: Extract the state nodes $S = \{\alpha, \beta, \gamma, \ldots\}$ in multi-order Markov model from sequences T.
4: Compute the ergodic visit probability of the state nodes and physical nodes, and the transition probability between state nodes.
5: Initialize the initial paritition M of the state nodes, e.g. each state node is assigned to a single community.
6: Initialize the maximum iteration number, e.g. $Iterations \leftarrow 10000$

7: Compute the initial description length $L \leftarrow CodingLength(M, T, S)$
8: **for** $loop \leftarrow [0, Iterations)$ **do**
9: Try the new assignment of the nodes and get the new partition M^{new}.
10: Compute the description length $L^{new} \leftarrow CodingLength(M^{new}, T, S)$
11: Accept the new partition and update $M \leftarrow M^{new}$ and $L^{new} \leftarrow L$, if $L^{new} < L$ or based on the criterion of simulated annealing algorithm.

12: Map the best partition of the state nodes to the partition of the physical nodes.
13: **return** The partition of the physical nodes $M_1, M_2, \ldots, M_m$.

Algorithm 2 Function CodingLength(Partition M, Time Series T, State Nodes S)

1: **function** CodingLength(Partition M, Time Series T, State Nodes S)
2: Compute the transition probability between state nodes S, according to equation 4.4.
3: Compute the ergodic visit probability of each state nodes and physical nodes, according to equation 4.3.
4: Compute the enter probability $q_{\curvearrowright}$ and the exit probability $q_{\curvearrowright}$ of each module.
5: Compute the description length L according to 4.2.
6: **return** L

To summarize, the generalization of Infomap to multi-order Markov model keeps the original framework of the Infomap. The difference between the existed high-order Infomap is multi-order Infomap uses low-order state nodes to describe the first several transitions of the sequences and uses high-order state nodes for the remaining transitions. And the ergodic visit probability and transition probability needed in the map equation are computed using the Multi-order Markov model. This brings two advantages for the multi-order Infomap:

- The multi-order Infomap combines the low-order state nodes and high-order state nodes together to describe transitions in each sequence. In this way, we can reduce the infomation loss and capture the movements of the states better, especially for the short sequences. The experiments shows the multi-order Infomap finds meaningful communities.

- The multi-order Infomap is built on the multi-order Markov model to capture the property of the transitions in the sequences. So the optimal order of the Multi-order Markov model could be determined using the order detection method implemented in the Multi-order Markov model that tells us which order could capture the transition property of the sequences best.

4.3 Experiment

4.3.1 Description length vs. order

First of all, samely with the high-order infomap, we argue that the description length cannot be used to measure the quality of communities found by multi-order Infomap with different orders. Here we still generate a random graph and then sample random sequences on the graph. Table 4.1 shows the description length corresponding to the multi-order Infomap with different orders. Similar to the trend of high-order-infomap (section 3.3), the description length tends to decrease if we increase the order. But we cannot say the higher order yields better results because the sequences do not have community structure at all.

Multi-order Infomap for pathway without Community Structure		
	Infomap order	best score
pathway order 1	1	4.307
	2	3.878
	3	3.722
	4	3.632
	5	3.182
	Infomap order	best score
pathway order 2	1	4.226
	2	3.509
	3	3.359
	4	3.254
	5	3.212

Table 4.1: Multi-order Infomap on sequences without community structure. The path data set is generated in the following way: 1) generate the random graph without community structure; 2) generate the sequences on the random graph with specified order. The path data set contains 10 independent sequences and the maximum length is 8. And the code is from Ingo [11]. The best score is the minimum description length in 20 random experiments. The maxmum number of optimization iteration in each turn is 50000. The results support our claim that the description length cannot be used to determine which order yields the best result.

4.3.2 Optimization Strategy Matters

Here we use a toy example to compare the two different optimization strategies discussed above. Firstly, let us show that the first optimization strategy operating on the module level is more easy to get caught in the local minimum than the second strategy which operates on the state nodes. We construct the following three data sets with different orders.

- $T_1 = \{t_1 = (a1 \to 1 \to a2 \to a1 \to 1 \to a2), t_2 = (b1 \to 1 \to b2 \to b1 \to 1 \to b2), t_3 = (a1 \to 1 \to b2 \to b1 \to 1 \to a2 \to a1 \to 1 \to b2 \to b1)\}$. The frequency of t_1 and t_2 is 10 and the frequency of t_3 is 1. Its order is 2 which is confirmed by the order detection method implemented in the `PATHPY`. (corresponding to 4.2a.)

- $T_2 = \{t_1 = (a1 \to 2 \to 1 \to a2 \to a1 \to 2 \to 1 \to a2), t_2 = (b1 \to 2 \to 1 \to b2 \to b1 \to 2 \to 1 \to b2), t_3 = (a1 \to 2 \to 1 \to b2 \to b1 \to 2 \to 1 \to a2 \to a1 \to 2 \to 1 \to b2 \to b1)\}$. The frequency of t_1 and t_2 is 10 and the frequency of t_3 is 1. Its order is 3 which is confirmed by the order detection method implemented in the `PATHPY`. (corresponding to 4.2b.)

- $T_3 = \{t_1 = (a1 \to 3 \to 2 \to 1 \to a2 \to a1 \to 3 \to 2 \to 1 \to a2), t_2 = (b1 \to 3 \to 2 \to 1 \to b2 \to b1 \to 3 \to 2 \to 1 \to b2), t_3 = (a1 \to 3 \to 2 \to 1 \to b2 \to b1 \to 3 \to 2 \to 1 \to a2 \to a1 \to 3 \to 2 \to 1 \to b2 \to b1)\}$. The frequency of t_1 and t_2 is 10 and the frequency of t_3 is 1. Its order is 4 which is confirmed by the order detection method implemented in the `PATHPY`. (corresponding to 4.2c.)

These three data sets all contains three paths. We can divide the nodes to two parts such that each part is traveled by 5 independent sequences. And the two parts overlap at some nodes. And for each data set, there is an another sequence that travels all the nodes which acts as the noise sequence. The best partition should partition the nodes to two overlapped communities such that most of the sequences are trapped within these two communities. We apply the multi-order Infomap with these two different optimization strategies on these three constructed data sets. The order of the multi-order Infomap is the same as the order of the sequences. For each data set, we run multi-order Infomap for 10 times and return the best result (the partition with the minimum description length) within these 10 independent experiment. The results for these three small data sets are summarized in table 4.2.

path	order	First Strategy		Second Strategy	
		partition	score	partition	score
T_1	2	$\{\{'b2','a1','b1','a2','1'\}\}$	2.896	$\{\{'b2','b1','1'\},\{'a2','a1','1'\}\}$	2.355
T_2	3	$\{\{'b2','a1','b1','a2','1','2'\}\}$	3.004	$\{\{'b2','b1','1','2'\},\{'a2','a1','1','2'\}\}$	2.614
T_3	4	$\{\{'b2','a1','b1','a2','1','2','3'\}\}$	3.142	$\{\{'b2','b1','1','2','3'\},\{'a2','a1','1','2','3'\}\}$	2.838

Table 4.2: Comparsion between the first optimzation strategy and the second optimization strategy. The data sets T_1, T_2 and T_3 are introduced before and they have order $2, 3$ and 4 respectively. The second column refers the order of the multi-order Infomap applied on this data set. We run every experiment for 10 times and report the best result. For each sequence set, the best partition detected by the multi-order Infomap using the first optimization strategy (colunm 3 and 4) has larger description length than the best partition returned using the second optimization strategy (column 5 and 6). This means the first optization strategy is more easy to get stuck in the suboptimal case.

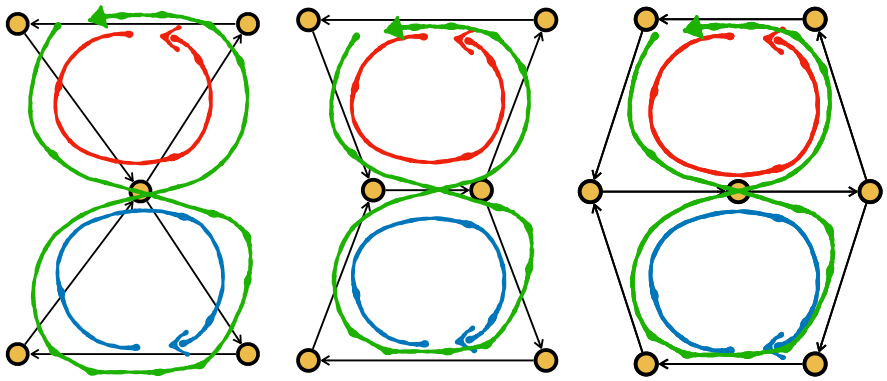
The multi-order Infomap with the second optimzation strategy returns two overlapped communities for these three synthetic sequence sets which is the same as our expectation. However, the multi-order Infomap with the first optimzation strategy partitions the nodes to a single community. Furthermore, we can compare the description length of the partition. The partition returned by the first optimzation strategy has larger description length which confirms that the partition found by the multi-order Infomap with first optimzation strategy is suboptimal in these three data sets. This example shows that the Infomap with first optimization strategy operating on the module level is more easy to get caught in the local minimum, regardless of the order of the sequences. The reason is discussed in 4.2.

4.3.3 Real Datasets

We apply the multi-order Infomap on a real pathway data set [11] [2]. This data set contains passenger itineraries along flight routes between US airports in 2001. The number of itineraries is 286,810 on 175 different airports.

As said before, we can estimate the order of this flight set using the method developed in the multi-order Markov model. And the estimated order is two. So we

[2]The results are saved in *real_data/flight_result.txt* file



(a) Order 2 pathway (b) Order 3 pathway (c) Order 4 pathway

Figure 4.2: Pathways with different orders. The red and blue lines correspond to the paths in two communities. The green line is path through two communities.

apply the multi-order Infomap with $order = 2$ for this dataset. However, we do not have the ground truth for this data set. Thus we cannot quantitively measure the quality of the found communities. Furthermore, it is very hard to visualize the overlapped communities for this real data set. Nevertheless, we can still find something meaningful from the found communities.

We count the number of transitions within the same community and across different communities. For the multi-order Infomap with $order = 2$, the number of transitions within the same community is 224264 and the number of transitions across different communities is 198. In comparison, for the ordinary Infomap (1-order), the number of transitions happened within the same community is 222610 and the number of transitions across different communities is 1852. Considering the objective of Infomap is to find a partition of the nodes such that less transitions happen across different communities, we can say the communities returned by 2-order Multi-order Infomap is better.

	inter-cluster transitions	within-cluster transitions	# communities
Stardard Infomap	1852	222610	167
Multi-order Infomap	198	224264	14

Table 4.3: High-order Infomap with $order = 2$ and Stardard Infomap on the flight data.

If we compare the communities found by the ordinary Infomap and the multi-order Infomap, we can find these two Infomap both return a big community and several small communities. For the big community, the ordinary Infomap and the multi-order Infomap almost return the same result. However, most of the small communities returned by the ordinary Infomap only contain a single airport. Differently, the multi-order Infomap gives us some small communities which contain several

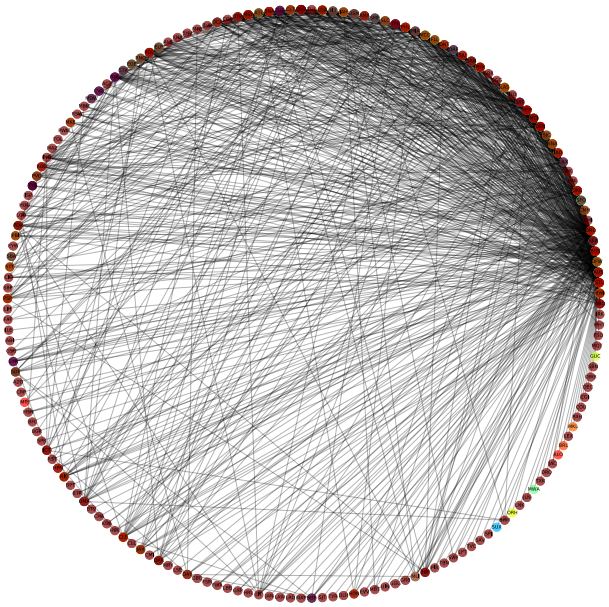


Figure 4.3: The visualization of the flight pathway. The edges between two airports represent these two airports appear in the same itinerary. And each color corresponds to a community. Note that some node could belong to several different communities. In this case, the color of overlapped node is the mixed color of these different communities.

airports. And if we examine the nodes in the same small communities, we can find that these airports are more frequently visited within the same itinerary. For example, in the communities returned by the multi-order Infomap, there is a small community containing two airports: GUC and DFW. DFW is a frequently-visited airport and it appears in many different communities. But GUC only appears in this community with DFW. So I checked the paths containing GUC. I found in all paths containing GUC, the previous node before GUC and the next node after GUC are both DFW. This means that if the traveler goes to GUC, the traveler mush first arrive DFW. If the traveler goes to another airport from GUC, the next step mush be DFW. I think this could be a example to show that the communities returned by 2-order multi-order Infomap are meaningful because the community captures the correlation between GUC and DFW.

In a conclusion, while it is hard to quantitively measure the communities found by multi-order Infomap because of the lack of ground-truth, the communities found by multi-order Infomap are more meaningful than the communities returned by the ordinary Infomap. Particularly the found communities by multi-order Infomap can tell us which airports are more frequently visisted within the same itinerary.

Chapter 5

Conclusion and Discussion

In this thesis, there are mainly two contributions: (1) the first one is that we find the description length cannot be used to measure the quality of communities found by different models. The reason is the map equation is computed based on the state. We used two experiments in section 3.2.3 to support our argument. (2) the second contribution is to generalize the Infomap to the multi-order Markov model. Particularly, the developed multi-order Infomap directly operates on the given sequences and do not explicitly contructing the underline graph. Different from the previous works of Infomap, the multi-order Infomap takes into account all layers of a multi-order model at once. To achieve this goal, we keep the framework of the original Infomap and adapt the map equation in Infomap to the multi-order Markove model. Particularly, we compute the entering probabilities, exiting probabilities and the ergodic node probabilities taking account to all layers of a multi-order model at once. For the multi-order Infomap, we test it on a small synthetic data set with ground-truth. And it correctly find the best partition. And we also apply it on a real data set and qualitively analyze the communities found by the multi-order Infomap. The reason we cannot do quantive measurement is we do not know the ground-truth of the real data set and it is very hard to measure the overlapped communities found by Infomap.

5.1 Potential Improvements

The optimization strategy.

When conducting the experiments, I find the optimation of searching best partition of the nodes influences the result much. In these different variaions of Infomap, the nodes are initialized to different communities at begining. In each iteration, we randomly change the module of one node or several nodes and get a new partition of the communities. And the way to change the module of nodes influences the results much.

In the multi-order infomap, we tried two different strategies to change the modules of the nodes. The first one operates on the community level such that in each iteration we randomly choose two different communities and merge the nodes in these two communities into a new community. And we accept the new partition

if the description length corresponding to the new partition is shorter than the old partition or we could use the simulated annealing approach. So with this strategy, we reduce one community at each iteration. The advantage of this approach is that the it can converge to the stable state fast because every time we reduce one community. Furthermore, we can use advanved data to avoid to re-compute the visit probabilities of all communities (we only need to change the visit probabilities of these two modules and the related modules). However, this approach may sacrifice the quality. The reason is that each time we merge two communities and the merged nodes will always stay in the same communities. This means we are easy to go into local minimum because we could "wrongly" merge some nodes together in the early steps.

The second optimzation stratage operates on the node level. In this approach, we randomly merge two nodes in different communities in to a new community. Samely we accept the new partition if the description length corresponding to the new partition is shorter than the old partition or we could use the simulated annealing approach. The advantage of this method is it can reduce the probability to stick in the local minimum if iteration is long enough because the search space of this approach is larger than the search space of first approach. In fact, if the iteration is long enough, this method searches all the possible partitions in fact. However, because of the hugh search space, this strategy can only handle the small data sets. Furthermore, it is much complicated to design the advanced algorithm to speed up the optimization process like the first approach.

The implementation of optimization.

In the real implementation of the multi-order Infomap, we use the advanced data structure to speed up the optimizaiton process for the first strategy. The rationale is that in each iteration turn, we do not need to compute the entering/exiting probability for each module and the visiting probability of physical nodes in every module. However, designing the advanced methods for the second strategy is much more complecated. However, in each optimization turn of the second strategy, we also only change the nodes in two communities. So we only need to change the switch probabilities of these two modules and the modules that has link with these two communities. Thus I think we can still improve the speed of optimization. And the speed improvement of the second strategy is meaningful because the second strategy is less to stick into the local minimum. If we can reduce the complexity of this method, we can apply it to larger data set.

The experiments for multi-order Infomap.

In section 4.3.2 and 4.3.3, we use a very small manually contructed data set and a real data set (the flight data) to demonstrate the effectiveness of the multi-order Infomap developed in this thesis. However, it is better if we have additional experiments on the larger synthetic data set. The difficult here is how to construct the synthetic time series with the (overlapped) community structure. Besides, the synthetic time series should have high order correlation. If we can construct this data set and demonstrate the advantage of multi-order Infomap on the larger synthetic data set, the effectiveness of developed multi-order Infomap is more convincing.

5.2 About the initial proposal of this thesis

In the initial proposal, there are two tasks related to the description length. When doing the experiments in task two, I find the real case is not as we expected at beginning. We show that we cannot use the description length to determine which order yields the best partition (in section 3.3). For the task four, we still cannot use the description length as the measurement to analyze the advantage of the multi-order Infomap developed in this thesis. Instead we qualitively analyze the communities found by multi-order Infomap because we do not know the ground-truth of the real data set and cannot use description length or other measurement to quantively show the quality of the found communities.

Bibliography

[1] Wei Chen, Fangzhou Guo, and Fei-Yue Wang. A survey of traffic data visualization. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2970–2984, 2015.

[2] Nicolaas Govert De Bruijn. A combinatorial problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, 49(49):758–764, 1946.

[3] Daniel Edler, Ludvig Bohlin, et al. Mapping higher-order network flows in memory and multilayer networks with infomap. *Algorithms*, 10(4):112, 2017.

[4] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[5] Tiago P Peixoto and Martin Rosvall. Modelling sequences and temporal networks with dynamic community structures. *Nature communications*, 8(1):582, 2017.

[6] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005.

[7] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[8] Venu Satuluri and Srinivasan Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 737–746. ACM, 2009.

[9] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[10] Michael T Schaub, Jean-Charles Delvenne, Martin Rosvall, and Renaud Lambiotte. The many facets of community detection in complex networks. *Applied Network Science*, 2(1):4, 2017.

[11] Ingo Scholtes. When is a network a network?: Multi-order graphical model selection in pathways and temporal networks. In *Proceedings of the 23rd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1037–1046. ACM, 2017.

[12] Stijn Marinus Van Dongen. *Graph clustering by flow simulation*. PhD thesis, 2001.