Random Walk Approaches to Clustering Directed Networks

Dissertation
zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von
Stefan Rüdrich

Berlin, 2019

Erstgutachter: Prof. Dr. Christof Schütte
Fachbereich Mathematik und Informatik
Freie Universität Berlin

Zweitgutachterin: Prof. Dr. Kathrin Padberg-Gehle
Institut für Mathematik und ihre Didaktik
Leuphana Universität Lüneburg

Disputation am 28. Februar 2020

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe. Ich erkläre weiterhin, dass ich die vorliegende Arbeit oder deren Inhalt nicht in einem früheren Promotionsverfahren eingereicht habe.

_____

Stefan Rüdrich
Berlin, September 26, 2019

Acknowledgements

I would like to acknowledge my supervisors Christof Schütte and Sebastian Hainzl for their patient support, my mentor Heike Siebert for her counsel and encouragement, and my dear colleagues at the Biocomputing group for their help and enjoyable company.

A special compliment goes to the members of the Systems Pharmacology and Disease Control group, that shared many wonderful years and countless cups of coffee with me. I am most grateful to Nada Cvetković and Maureen Smith for proof-reading and giving comments on earlier drafts of this work.

My research was funded by fellowships from Helmholtz Association's graduate program 'GeoSim' and Berlin Mathematical School.

Contents

1 | Introduction

Some networks obtained from real world problems such as social networks, biological cells or earthquake recurrence networks, may consist of many thousands of vertices and exhibit a pronounced modular structure. To facilitate further analysis or derive a simple model of the whole system, it is often useful to identify subunits of these networks that are nearly disconnected from the remaining graph. These subunits are called 'clusters' and may correspond to a group of people sharing similar interests, a cell compartment responsible for a special type of metabolic reaction or an active fault along which seismic events tend to line up.

A well-known method to find clusters in undirected networks is based on random processes whose state space is the network's set of vertices, moving in each time step to a randomly chosen neighboring vertex. A close correspondence between network substructures and properties of such a random walk process has been exploited in numerical algorithms, which aim at decomposing the random walk's state space into disjoint 'metastable' sets, describing nearly stable conditions of the system which are rarely changed [8].

Identification of these metastable sets can be realized numerically with spectral methods that rely on computing eigenvalues and eigenvectors of the transition matrix that defines the behavior of a random walk. However, computing these eigenvectors can be arbitrarily hard for some cases, e.g. when there are bottlenecks in the network and the transition matrix is ill-conditioned [58]. In addition, it is not yet fully understood how to work with complex-valued eigenvectors that arise from random walks on directed networks.

To address this issue, other methods of numerical analysis based on random walks have been proposed, also focused on metastability, but exploiting it differently. It has been known for a long time that metastable sets of a random walk on a given graph may be characterized as sets which, like traps, are difficult to leave once entered [25]. To put this in mathematically rigorous terms, it is useful to quantify how long it takes a random walk to leave or enter a specified set. The average time a random walk takes to reach some fixed target set may be regarded as a non-negative function on the vertex set of the graph. The value of this function at a given vertex describes the 'mean hitting time' of the target set, conditioned on the walker starting at the vertex.

This thesis presents a new network clustering approach exploiting the fact that mean hitting times of arbitrary target sets can be computed quickly by solving a system of linear equations [66]. This approach is built upon work by Schütte [69], Đurđevac [23] and Sarich [63] and it employs hitting times to extend random walk based clustering to directed networks without resorting to any form of symmetrization. Based on this approach, we developed and implemented a novel network clustering algorithm. To this end, we studied

in detail the choice and number of target sets, the identification of clusters from their mean hitting times, requirements of this method regarding the network structure and its computational aspects. In particular, we focused on the computational effort and the accuracy of results.

The computational effort of this algorithm is dominated by solving linear systems to compute hitting times. For 'sparse' networks, i.e. for networks where the number of neighbors any vertex may have is small relative to the total number of vertices, our algorithm can exploit sparsity in order to achieve $\mathcal{O}(n)$ runtime, provided that the associated sparse linear systems are solved using efficient numerical solvers such as GMRES [62, 61]. Existing random walk methods dependent on eigenvector approximations require matrix multiplications whose complexity is estimated to be $\mathcal{O}(n^\omega)$ with an exponent ω between 2 and 2.376 [18].

For the purpose of assessing runtime scaling, we have generated a family of sparse, directed, random networks spanning several orders of magnitude in vertex count. A series of tests verified that runtime of the presented algorithm scales linearly with the number of vertices in these network, in accordance with our expectations. Therefore, our approach is well suited for very large networks that are sufficiently sparse and shows a computational advantage over existing methods requiring eigenvector approximations.

After introducing definitions and concepts from graph theory and probability theory in Chapter 2, the basics of clustering for undirected and directed graphs through spectral and non-spectral methods will be covered in Chapters 3 and 4, respectively. A selection of existing approaches to clustering graphs are described briefly and popular ways of measuring the similarity of clusterings for the purpose of comparing and evaluating competing approaches to each other or to ground truth data are presented. Chapter 5 introduces and analyzes our clustering method based on mean hitting times, including computational aspects and limitations following from the hitting time approach. The final discussion in Chapter 6 summarizes the presented results and sketches early approaches to overcome existing restrictions on the structure of networks, to further extend the class of eligible networks and generalize the method to even more use cases.

2 | Theoretical background

Graph Theory has a century-old tradition, reaching back at least to the year 1736, when Leonhard Euler created the foundations of graph theory to tackle the problem of finding a path around the city of Königsberg without crossing any of its bridges twice. The story is worth telling as it exemplifies how graphs have been invented and why the model is so universally applicable.

The city was located at the Pregel river and included two islands in addition to districts on either side of the river. The tour he was asked to find was meant to cover sights in each of the four parts of the city and all seven bridges connecting them exactly once. Euler soon noted that the paths taken through each of the four parts of the city were irrelevant to the problem.

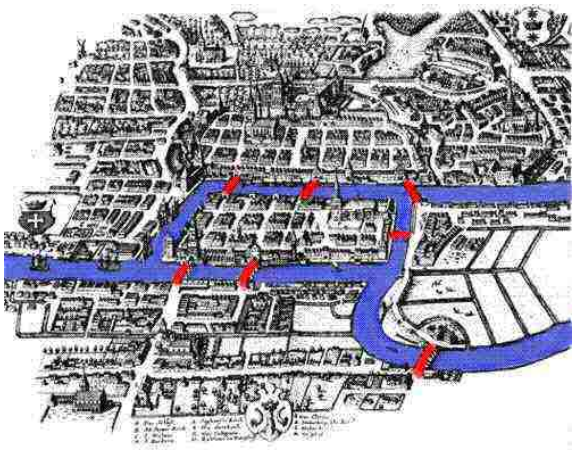


Figure 2.1: A map of the old city core of Königsberg, showing the seven bridges across the Pregel river. Image reprinted from http://www-history.mcs.st-andrews.ac.uk/Miscellaneous/Konigsberg.html, visited on 1st of May 2010.

To phrase the problem with mathematical rigor, he derived a model on a high level of abstraction, where each land mass was represented by a vertex, each bridge by an edge. Being an ambitious researcher, Euler was not content to find just one admissible solution or disprove its existence. He aimed at identifying conditions necessary or sufficient to solve any such problem, for any kind of network topology. He succeeded in proving that the existence of a solution depends on whether the network vertices have even degrees, except for the first and final vertex of a potential path.

Given the simplicity of this model and the universality of his findings, Euler's results could easily be applied to any city of arbitrary size or even to systems unrelated to city maps.

2.1 Graph theory basics

All over the different fields of science, graphs are used to model interactions, dependencies or proximity between the elements of a system. This model is so popular and widely used because of its simplicity: All interior properties of the constituents are neglected, their relation to each other is considered to differentiate network vertices. Having such a simple and universal model allows to employ graph theoretic results and methods to derive insights about systems as diverse as electric circuits, metabolic networks, earthquake recurrence maps or social networks. Identifying bottlenecks, shortest paths, clusters or hubs reveals key information about the structure of a system, independent of its origin.

We aim to present and analyze recently developed methods of network analysis and thus start off by giving formal definitions for graphs and some basic terminology. Let us start by introducing the notion of a graph.

Definition 1. A pair $G = (V, E)$, consisting of **vertex set** V and **edge set** E, a set of pairs of vertices $e = \{u, v\} \in V \times V$, is called a **(simple) graph**. The elements of V are also called **nodes**. We will focus on finite directed graphs, whose vertex set is finite and edges are oriented, turning E into a set of ordered pairs of vertices. Optionally, an edge weighting function $w : \ E \to \mathbb{R}$ may be given, assigning a value, cost or length to edges, often describing the distance, capacity or flow intensity between two vertices. If none is provided, we consider the edges to have uniform weights $w \equiv 1$.

When another graph $G' = (V', E')$ is given, contained within G in the sense that $V' \subset V$ and $E' = \{(u, v) \in E : \ u \in V', v \in V'\}$, G' is called a **subgraph** of G, 'induced' by the set V'.

The term 'network' will be used synonymously for directed graphs throughout this document. The following definitions are related to graphs, so let $G = (V, E)$ be a finite directed graph with edge weighting w for the remainder of this section.

Definition 2. For an arbitrary vertex v, any vertex u that shares an edge with v, that is $(u, v) \in E$ or $(v, u) \in E$, is called a **neighbor** of v. The **degree** $d(v)$ of v is defined as the number of its neighbors.

In directed graphs, we may differentiate in-degree and out-degree of a vertex v. The **out-degree** $d_{out}(v)$ is given by the number of out-going edges oriented away from v, the **in-degree** $d_{in}(v)$ corresponds to the number of edges oriented towards v:

$$d_{out}(v) := |\{u \in V : \ (v, u) \in E\}|, \quad d_{in}(v) := |\{u \in V : \ (u, v) \in E\}|.$$

For many purposes, it is convenient to encode all information characterizing a graph, including existence, direction and weights of edges in matrix form. Linear algebra tools may then be used to analyze networks and link certain matrix properties to the structure of a graph, which we will exploit later.

Definition 3. A graph on vertex set $V = v_1, \ldots, v_n$ with $n = |V|$ vertices may be represented by an **adjacency matrix** $A \in \mathbb{R}^{n \times n}$ whose entries $A = (a_{ij})_{i,j=1}^{n}$ define which

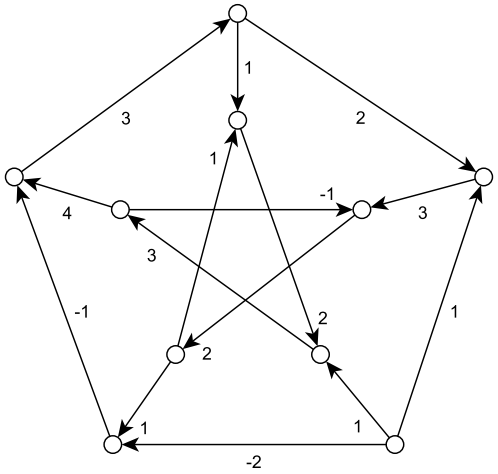


Figure 2.2: Representation of the Petersen graph, named after Danish mathematician Julius Petersen (1839–1910), with exemplary orientation of edges and integer-valued edge weights. Each vertex has exactly three neighbors, so the graph has a uniform vertex degree of three.

vertices are linked by an edge and their respective weights:

$$a_{ij} = \begin{cases} w((v_i, v_j)), & (v_i, v_j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{2.1.1}$$

Preparing the introduction of connectivity and components of a graph, we need a notion of which vertices u and v are indirectly connected to each other through a sequence of edges that form a loop-less connection from u to v.

Definition 4. For any two vertices u and v, a **path** from u to v is a finite and non-empty sequence of vertices $u = v_1, \ldots, v_n = v$, where all vertices except u and v are disjoint and each succeeding pair of vertices in the sequence is connected by an edge. When a path has n elements, its **length** is defined as $n - 1$ and if u equals v, the path is called a **cycle**.

In directed graphs, we are interested in directed paths and directed cycles. To qualify as **directed path**, the orientation of edges connecting two succeeding vertices in the sequence must 'follow' the order of the sequence, that is:

$$(v_i, v_{i+1}) \in E \quad \forall i = 1, \ldots, n-1.$$

When two vertices u and v are connected by a path, we say that the two vertices 'communicate'. Communication is a transient, symmetric and reflexive relation by definition. So it defines equivalence classes on vertex set V. They are used to define a disjoint partition of a graph into connected subgraphs.

Definition 5. For any non-empty graph $G = (V, E)$, the vertex set V contains equivalence classes of vertices with respect to connectivity. The subgraphs induced by these classes are called the **components** of G and those graphs which consist of only one component are called **connected**.

For directed graphs, there is also a stricter version of communication. We say that two vertices u and v **communicate strongly**, when there is a directed path from u to v and

a second path back from v to u. If any two vertices communicate in this sense, we call the directed graph **strongly connected**. Note however, that the definition of components does not rely on strong communication, even for directed graphs. A directed graph may be connected, but at the same time not strongly connected.

To help analyze the structure of networks, we will later introduce a specific class of random processes defined on the vertex set of networks, whose trajectories are directed paths. To ensure that such random walks never get 'trapped' in some vertex without exit options, we are mostly interested in the special case of graphs where such vertices do not appear. The following definition gives a formal description.

Definition 6. In a network, any vertex with out-degree 0 is called a **sink** and any vertex with in-degree 0 is called a **source**.

A strongly connected graph is always free of sinks and sources. A typical source for large networks of that type are discrete stochastic models for dynamical systems. Assume that $T : S \to S$ is a continuous map on some finite-dimensional state space S. Given a sequence of observations of a system whose driving forces are partially unknown or too complicated for numerical treatment, one may be interested in approximations of statistical quantities based on a discrete approximation of the system.

An approach named after the Polish-American mathematician Stanisław Ulam has been employed and adapted over decades for a range of related problems [77]. The method has been extended to random interval maps, Lasota-Yorke maps with holes and non-uniformly expanding maps for instance [32, 52, 6].

The construction proposed by Ulam is based on a finite partition of the state space into connected sets $\{B_1, \dots, B_k\}$ and compute a transition matrix defined by

$$P_{ij} = \frac{|B_i \cap T^{-1}B_j|}{|B_j|}.$$

Given a sufficiently long trajectory $\{x_0, x_1, \dots, x_N\} = \{T^k(x_0)\}_{k=0}^N$ of a mixing system for an arbitrary starting point x_0, these entries can be approximated with arbitrary precision by counting the number of transitions from set B_i to set B_j for every pair (i, j) and then normalizing the matrix row-wise

$$P_{ij} \approx \tilde{P}_{ij} = \frac{\#\{k : \ x_k \in B_i \text{ and } x_{k+1} \in B_j\}}{\#\{k : \ x_k \in B_i\}}.$$

Matrix P defines a discrete stochastic process on the partition sets $\{B_1, \dots, B_k\}$, which are thoroughly introduced in 2.2.1. Alternatively, one may look at P as the weighted adjacency matrix of a network whose vertices are the partition sets. Directed edges are inserted between any two vertices that were successively visited by the trajectory and the weighting corresponds to how often that transition has occurred.

P takes the role of a discrete approximation of T, acting on density vectors on the discretized space $\{B_1, \dots, B_k\}$. Its left eigenvector $v = vP$ approximates T-invariant measures $\eta = \eta \circ T^{-1}$ and its spectrum allows for detecting the number and location of cyclic and metastable structures. These procedures shall be examined in the following section.

2.2 Probability theory basics

Markov processes are stochastic processes with a special property often dubbed 'memorylessness', which expresses intuitively that, at any point in time, future behavior of a process depends only on its present state, not on its entire history. To define precisely what 'future behavior' and dependencies are in the context of events that may occur with a probability as low as 0, we need a robust stochastic framework.

Typically, the first definition presented in textbooks on stochastics is that of a probability space, the basic ecosystem populated by random variables, probability distributions and stochastic processes. This section starts slightly differently, as the notion of probability spaces requires tools, namely structured families of sets which are called σ-algebras and measures defined thereupon.

Definition 7. A **σ-algebra** $\mathcal{A}$ on a given set Ω is a collection of subsets of Ω, which is closed under certain set-theoretic operations. In detail, $\mathcal{A}$ must comply with the following rules:

1. $\Omega \in \mathcal{A}$ (Ω itself is always contained),
2. $E \in \mathcal{A} \implies (\Omega \setminus E) \in \mathcal{A}$ (closed under complementation) and
3. $E_1, E_2, E_3 \cdots \in \mathcal{A} \implies \bigcup_{i\in\mathbb{N}} E_i \in \mathcal{A}$ (closed regarding countable unions)

Let $\mathbb{F}$ be a σ-algebra. Then any σ-algebra $\mathbb{G}$ contained in $\mathbb{F}$ is said to be a **sub-σ-algebra** of $\mathbb{F}$. A pair $(\Omega, \mathcal{A})$ of a set Ω equipped with some σ-algebra $\mathcal{A}$ is called a 'measurable space'.

Two basic examples: $\{\varnothing, \Omega\}$ is the 'smallest' possible σ-algebra on Ω in the sense that it is a sub-σ-algebra of any σ-algebra on Ω. The power set of Ω is the 'biggest' possible σ-algebra on Ω. Any other σ-algebra on Ω is a sub-σ-algebra of it.

Definition 8. A **measure** on a σ-algebra $\mathcal{A}$ is a function $\mu : \mathcal{A} \to \mathbb{R} \cup \{-\infty, +\infty\}$ which abides to the following conditions:

1. $\mu(A) \geq 0$ for all $A \in \mathcal{A}$,
2. $\mu(\varnothing) = 0$ and
3. $A_1, A_2, A_3 \cdots \in \mathcal{A}$ are disjoint $\implies \mu\left(\bigcup_{i\in\mathbb{N}} A_i\right) = \sum_{i\in\mathbb{N}} \mu(A_i)$.

When a measure on some measurable space $(\Omega, \mathcal{A})$ is also normalized in the sense that $\mu(\Omega) = 1$, which implies that it takes values between 0 and 1 only, it is called a **probability measure**. The three rules above then transform to what is known as 'Kolmogorov's axioms':

1. $\mu : \mathcal{A} \to [0, 1]$,
2. $\mu(\Omega) = 1$ and
3. $A_1, A_2, A_3 \cdots \in \mathcal{A}$ are disjoint $\implies \mu\left(\bigcup_{i\in\mathbb{N}} A_i\right) = \sum_{i\in\mathbb{N}} \mu(A_i)$.

Two other optional properties of measures will be of importance in the following sections. A measure μ on $(\Omega, \mathcal{A})$ is called a 'finite measure' if $\mu(\Omega) < \infty$. For example, probability measures are always finite measures. A much weaker condition is called 'σ-finiteness'. σ-finiteness requires merely that a countable covering of the entire set Ω exists, such that each set in the covering has a finite measure. For instance, the ubiquitous Lebesgue-measure on the set of real numbers is not finite, but σ-finite, since the reals can be covered with a countable collection of finite intervals, namely $\mathbb{R} \subset \bigcup_{k\in\mathbb{Z}} [k, k+1]$.

Definition 9. A **measure space** $(\Omega, \mathcal{A}, \mu)$ is a triplet consisting of a set Ω, a σ-algebra $\mathcal{A}$ on Ω and a measure μ on $\mathcal{A}$. When μ is even a probability measure, $(\Omega, \mathcal{A}, \mu)$ is called a **probability space** and we refer to its basic set Ω as the **observation space** and elements of $\mathcal{A}$ as **events**.

A reader unfamiliar with the above notion might wonder what probability spaces look like and hope for a simple example at this point. In practice however, probability spaces are not specified any further – purposefully. Instead, random experiments are modeled as measurable functions on some probability space, characterized by their possible outcomes and their respective likelihoods.

Definition 10. A **random variable** X is a measurable mapping from a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ to another space Ω' equipped with σ-algebra $\mathcal{A}'$. Here, **measurability** means that the pre-image of any open set in $(\Omega', \mathcal{A}')$ must be an open set in $(\Omega, \mathcal{A})$:

$$E \in \mathcal{A}' \implies X^{-1}(A) \in \mathcal{A}.$$

Due to the condition that random variables must be measurable, they allow to 'push-forward' the measure $\mathbb{P}$ to the target space, such that it becomes a probability space on its own when complemented by the $\mathcal{A}'$-measure $\mathbb{P}'$ defined via

$$\mathbb{P}'(A') = \mathbb{P}(X^{-1}(A')).$$

Measurability of X guarantees that the pre-image of any set A' in $\mathcal{A}'$ exists in $\mathcal{A}$ and thus makes $\mathbb{P}'$ a well-defined (probability) measure on $\mathcal{A}'$. Typically, the target space is chosen to have an easily manageable structure. In most cases, Ω' is either a subset of $\mathbb{N}$ or $\mathbb{R}^n$. Random variables mapping into $\mathbb{N}$ or some other denumerable set are called **discrete**, while real-valued ones are called **continuous**. The σ-algebra is usually chosen to be the Borel σ-algebra on Ω', defined as the 'smallest' σ-algebra which contains all intervals – or Cartesian products of intervals, in higher dimensions. In the discrete case, this is simply the power set, which contains every subset of the target space. When the σ-algebra of the target space is not specified, it is understood that it must contain all the images of open sets in Ω under X, in order not to violate the measurability constraint.

A real-valued random variable X has the property of being **almost surely finite** if $\mathbb{P}(X(\omega) < \infty) = 1$. More generally, we speak of events A to be **almost sure**, when $\mathbb{P}(A) = 1$. Thus, when 'x has property P' is an almost sure event, the property is said to apply **almost surely**.

Thinking of standard six-faced dice, the crucial information required to model them is that they may show one of their six faces, each with equal likelihood of one over six. It is not necessary to know every element of the observation space or an explicit description of the random variable to derive conclusions on the probability of an outcome of one or several

independent experiments such as 'How likely is it to throw an odd number?' or 'How likely is it to throw a six twice in a row?'. By our understanding, no deterministic algorithm can produce a randomized outcome. Instead of defining random variables explicitly by putting down an analytic formula, they are typically defined indirectly by their co-domain and their induced probability measure on it.

In the case of a standard die, the range of values would be $\Omega' = \{1, 2, 3, 4, 5, 6\}$, a finite subset of $\mathbb{N}$, and a short way of describing the induced measure $\mathbb{P}'$ would be $\mathbb{P}'(i) = \frac{1}{6}\ \forall i \in \Omega'$. It is not necessary to give values of $\mathbb{P}'$ for each element of $\mathcal{A}'$ in this finite case, as the third defining property of measures, as stated in Definition 8, allows to compute the value of $\mathbb{P}'$ for every set in the σ-algebra generated by the pre-images of the elements of Ω'. Stating the probabilities of each 'basic outcome' is sufficient.

Having introduced the basic notion of random variables, the first thing often looked at to help understand them better, are statistical values describing their mean behavior and spread of outcomes. Not all of them are required to dive into the theory of random walkers, so we hint at textbooks on stochastics and statistics to those who wish to learn more about these concepts [10]. One indispensable tool shall be noted here however, as it is essential for the introduction of conditional events and lead to the notion of random events and then random processes: the 'expectation'.

Definition 11. When a random variable X is $\mathbb{P}$-integrable (satisfying $\int_\Omega |X(\omega)| d\mathbb{P}(\omega) < \infty$) for some probability measure $\mathbb{P}$, the **expected value** or statistical **mean** $\mathbb{E}[X]$ is defined as the $\mathbb{P}$-weighted average of X over Ω, that is

$$\mathbb{E}[X] := \int_\Omega X(\omega) d\mathbb{P}(\omega).$$

When dealing with events A and B of positive probability, such events are called 'independent' of each other, when $\mathbb{P}(A \cap B) = \mathbb{P}(A) \cdot \mathbb{P}(B)$. When they are not, it is often of interest to consider the likelihood of one event, say A, under the condition that the other event, here B, occurs. This is called the conditional probability of A, conditioned on B, denoted by $\mathbb{P}(A|B)$ and is formally defined by the formula $\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$. If and only if A and B are independent events, $\mathbb{P}(A|B)$ coincides with $\mathbb{P}(A)$, as knowledge about the outcome of B has no impact on A. Note that conditional probability is well-defined only under the constraint that the conditioned event has strictly positive probability. In the following, we will need a tool that generalizes the concept of conditional probability to the general case, frequently found in continuous settings, that the conditioned events have a probability of 0. To this end, we shall introduce measurable functions known as conditional expectations, which act like the expectation of a random value, but averaged over each admissible event in a given σ-algebra, such as the one generated by another random variable Y. In fact, this property defines them uniquely, up to $\mathbb{P}$-null sets.

Definition 12. Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space. A 'conditional expectation' $\mathbb{E}(X|\mathcal{B})$ of a continuous random variable X with respect to a sub-σ-algebra $\mathcal{B}$ of $\mathcal{A}$ is any $\mathcal{B}$-measurable function defined by the property

$$\int_B \mathbb{E}(X|\mathcal{B}) d\mathbb{P} = \int_B X d\mathbb{P} \quad \forall B \in \mathcal{B}.$$

Note that the σ-algebra $\mathcal{B}$ controls the granularity of information about X that we may retrieve from $\mathbb{E}(X|\mathcal{B})$. When X itself is $\mathcal{B}$-measurable, X and $\mathbb{E}(X|\mathcal{B})$ coincide. The

existence of $\mathbb{E}(X|\mathcal{B})$ can be assured using the following result by Johann Radon and Otto Nikodym, known as the 'Radon–Nikodym theorem' [54]. It is sufficient to assume that the expected value of X exists. Proofs of widely known results are omitted here for brevity, unless the involved methods provide particularly important insights.

Theorem 2.2.1. *Let μ and ν be σ-finite measures on a measurable space $(\Omega, \mathcal{A})$. Now, let ν be 'absolutely continuous' with respect to μ, that is $\mu(A) = 0 \Rightarrow \nu(A) = 0 \quad \forall A \in \mathcal{A}$. Then there is a measurable function $\frac{d\nu}{d\mu} : \Omega \to [0, \infty)$, called 'Radon–Nikodym derivative' of ν with respect to μ, which satisfies*

$$\nu(A) = \int_A \frac{d\nu}{d\mu} d\mu \quad \forall A \in \mathcal{A}.$$

$\mathbb{E}(X|\mathcal{B})$ takes the form of a Radon–Nikodym derivative for suitably chosen measures as follows: Define a measure μ on $\mathcal{A}$ induced by X through $\forall A \in \mathcal{A} : \ \mu(A) = \int_A X d\mathbb{P}$. This measure is by construction absolutely continuous with respect to $\mathbb{P}$, as integrals over null sets always vanish. Likewise, $\mu|_{\mathcal{B}}$, the restriction of μ to $\mathcal{B}$, is absolutely continuous with respect to $\mathbb{P}|_{\mathcal{B}}$, the restriction of $\mathbb{P}$ to $\mathcal{B}$. Both $\mu|_{\mathcal{B}}$ and $\mathbb{P}|_{\mathcal{B}}$ are σ-finite measures on $(\Omega, \mathcal{B})$. Their σ-finiteness is inherited from $\mathbb{P}$, which is even a probability measure and therefore finite. Employing Theorem 2.2.1, we conclude the existence of a $\mathcal{B}$-measurable function $\frac{\mu|_{\mathcal{B}}}{\mathbb{P}|_{\mathcal{B}}}$, which we identify as $\mathbb{E}(X|\mathcal{B})$.

We have not specified yet, how to choose the sub-σ-algebra $\mathcal{B}$ that takes the role of a 'condition' here. The type of event we are usually interested in is the outcome of another random experiment, which may or may not have zero likelihood, a.k.a. events like $\{\omega : Y(\omega) = y\}$ or $\{\omega : Y(\omega) \in S\}$, involving some other random variable $Y : (\Omega, \mathcal{A}, \mathbb{P}) \to (\Omega', \mathcal{A}')$. The sought-after algebra is generated by the pre-image of $\mathcal{A}'$ with respect to Y, which always lies in $\mathcal{A}$ due to Y being a measurable function by definition. So we set $\mathcal{B} = Y^{-1}(\mathcal{A}')$ to define conditional expectation of X with respect to Y as

$$\mathbb{E}(X|Y) = \frac{d(\mu \circ Y^{-1})}{d(\mathbb{P} \circ Y^{-1})}.$$

Both $(\mu \circ Y^{-1})$ and $(\mathbb{P} \circ Y^{-1})$ are push-forwards of measures on $(\Omega, \mathcal{A})$ via Y and therefore defined on $(\Omega', \mathcal{A}')$, not $(\Omega, \mathcal{A})$. Their Radon–Nikodym derivative thus maps from Ω' to $[0, \infty)$, unlike $\mathbb{E}(X|Y^{-1}(\mathcal{A}'))$, which is defined on Ω. The link between them is easily established through Y:

$$\mathbb{E}(X|Y) \circ Y = \mathbb{E}(X|Y^{-1}(\mathcal{A}')).$$

We retrieve a function $\mathbb{E}(X|Y)$ defined on the range of Y which acts like X for any value Y can assume in the sense that

$$\int_\Omega \mathbb{E}(X|Y) f(Y) d\mathbb{P} = \int_\Omega X f(Y) d\mathbb{P} \quad \text{for every measurable real-valued function } f.$$

In the following, we will make use of abusive notation for the sake of brevity and readability, such as $\mathbb{E}(X|Y = y)$ instead of $\mathbb{E}(X|Y)(y)$. Likewise, formal expressions of the type $\mathbb{P}(X \in A|Y)$ are to be understood as conditional expectations $\mathbb{E}(\mathbb{1}_{X \in A}|Y)$. Where conditional probabilities are well-defined, the two coincide.

Theorem 2.2.1 guarantees existence of functions satisfying an integral formula, unique up to null sets, but tells nothing about how to compute such functions. Fortunately, we find that in some common cases the conditional expectation takes a familiar form that

matches our intuition of what it should express. When for example both X and Y are discrete random variables with respective ranges $\mathcal{X}$ and $\mathcal{Y}$, we may express $\mathbb{E}(X|Y=y)$ for some value $y \in \mathcal{Y}$ using standard conditional probabilities:

$$\mathbb{E}(X|Y=y) = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}(X=x|Y=y).$$

When both X and Y are continuous, this reads as

$$\mathbb{E}(X|Y=y) = \int_{\mathcal{X}} x \frac{f_{X,Y}(x,y)}{f_Y(y)} dx,$$

where $f_{X,Y}(x,y)$ is the joint density of X and Y and $f_Y(y)$ the density of Y.

The conditional expectation inherits several useful properties from the expected value, which we will wrap up in the following lemma:

Lemma 2.2.2. *Let X and Y be random variables on some probability space $(\Omega, \mathcal{A}, \mathbb{P})$, each with finite expectation. For any sub-σ-algebra $\mathcal{B}$ of $\mathcal{A}$ and any real numbers a, b we have*

$$\mathbb{E}(aX + bY|\mathcal{B}) = a\mathbb{E}(X|\mathcal{B}) + b\mathbb{E}(Y|\mathcal{B}) \quad \textit{(linearity).}$$

Moreover,

$$X \leq Y \Rightarrow \mathbb{E}(X|\mathcal{B}) \leq \mathbb{E}(Y|\mathcal{B}) \quad \textit{(monotonicity)}$$

and

$$X \geq 0 \Rightarrow \mathbb{E}(X|\mathcal{B}) \geq 0 \quad \textit{(positivity).}$$

Finally, for any convex real function f, we have

$$f(\mathbb{E}(X|\mathcal{B})) \leq \mathbb{E}(f(X)|\mathcal{B}) \quad \textit{(Jensen's inequality).}$$

2.2.1 Random processes

Definition 13. A **random process** $\mathbb{X} = (X_t)_{t \in T}$ is a family of random variables $X_t : \Omega \to \Omega'$ on some fixed probability space $(\Omega, \mathcal{A}, \mathbb{P})$ to a fixed state space (S, Σ). The totally ordered index set T is typically chosen as either $\mathbb{N}_0$ or $\mathbb{R}_+$ and the random process is accordingly called a **discrete** or **continuous** random process.

Random processes are often used to model the evolution of a system through time. We will look at random processes on finite state spaces, such as the vertex sets of given graphs, and omit the σ-algebra over the state space, which is simply its power set, for brevity.

A special class of random processes we are particularly interested in are discrete processes with a 'short memory', where each random variable may depend only on its direct predecessor, but not on its former history. Formally, this reads as follows:

Definition 14. A discrete random process $\mathbb{X}$ on a countable state space S is said to have the **Markov property** and thus is called a **Markov chain**, if and only if it fulfills

$$\mathbb{P}(X_{t+1} = x_{i_{t+1}} \mid X_t = x_{i_t}) \;=\; \mathbb{P}(X_{t+1} = x_{i_{t+1}} \mid X_t = x_{i_t}, \ldots, X_0 = x_{i_0})$$

for any time $t \in \mathbb{N}_0$ and any states $x_{i_0}, \ldots, x_{i_{t+1}} \in S$.

In a continuous setting, the formalization of the Markov property requires more sophisticated tools to describe the information gained from observing a random process. The probability that a random process follows a certain trajectory in continuous space is typically zero, as uncountably many trajectories are admissible. Therefore, probabilities conditioned on such events are ill-defined, which motivated the earlier introduction of conditional expectations.

We may 'condition' the expectation of a random variable on the outcome of another by using the σ-algebra generated by the pre-images of the latter. For random processes $\mathbb{X} = (X_t)_{t\in T}$ on state space (S, Σ), which are families of random variables, this allows for defining probabilities concerning the state which the process assumes at one point in time $t_1 \in T$, given knowledge about the process at another point in time $t_0 \in T$, e.g.

$$\mathbb{P}(X_{t_1} \in A | X_{t_0}) = \mathbb{E}(\mathbb{1}_{X_{t_1}\in A} | X_{t_0}) \quad \forall A \in \Sigma.$$

In order to introduce 'memoryless' processes next, we will need a notion for information acquired from observing a process up to a certain point in time, not only at that given moment. As the uncertainty about the trajectory of a process decreases the longer we observe it, the gain of information is described by time-indexed families of σ-algebras of increasing granularity, which we define here as follows:

Definition 15. For any probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and index set T, a T-indexed family $\mathbb{F} = (\mathcal{F}_t)_{t\in T}$ of sub-σ-algebras of $\mathcal{A}$ is called a **filtration**, if it is ascending in the following sense: If $t_1 < t_2$, then $\mathcal{F}_{t_1}$ is a sub-σ-algebra of $\mathcal{F}_{t_2}$.

Filtrations are used to model the time-dependent gain of knowledge about the trajectory of a process. Intuition dictates that observing a realization of a random process gradually decreases uncertainty about the outcome. Every sample $\omega \in \Omega$ is mapped to a trajectory $(x_t)_{t\in T}(\omega)$ about which we have only partial information at any point of time. At $t_0 \in T$, we know about the current and all former states the process has assumed, but lack information about the future. We may answer some statements of the type 'Does ω lie in event E?' with certainty, in some cases we have no certain knowledge, but the probability of an event to occur is influenced by the information already gained until t_0. Therefore, we express the information gained by observing a random process $\mathbb{X} = (X_t)_{t\in T}$ as a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in T}$ 'generated' by $\mathbb{X}$ in the following sense:

$$\mathcal{F}_t = \sigma(X_s; s \leq t). \tag{2.2.1}$$

The operator $\sigma(E)$ defines the smallest σ-algebra which contains an event E. In this case: the event that the process $\mathbb{X}$ has followed a specific trajectory until time t: $E = \bigcap_{s\leq t} X_s{-}1(E'_{i_s})\ \forall E' \in \mathcal{A}'$. Note that E' are events in the measurable space $(\Omega, \mathcal{A})$ in which the random variables $X_s, s \leq t$ assume values. As random variables are measurable functions, the pre-images of these events are guaranteed to be events of the basic probability space $(\Omega, \mathcal{A}, \mathbb{P})$ on which the random process is defined. Due to the structure of σ-algebras, the intersection of events is also an event, assuring existence of E as defined above for any trajectory the process can take.

A filtration encodes the information available at any moment in time t, namely the events contained F_t and by consequence all F_t-measurable random variables. A closely related concept is called 'adaptation'. It allows us to consider random processes $\mathbb{X} = (X_t)$ that can be evaluated in real time. That is: We fix some filtration $\mathbb{F} = (\mathcal{F}_t)$, which stands for the available information, and require X_t to be $\mathcal{F}_t$-measurable for every t.

Definition 16. Let $\mathbb{X} = (X_t)_{t\in T}$ be a T-indexed random process on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and $\mathbb{F} = (\mathcal{F}_t)_{t\in T}$ be a filtration sharing the index set with $\mathbb{X}$. $\mathbb{X}$ is said to be 'adapted' to $\mathbb{F}$, if X_t is $\mathcal{F}_t$-measurable for every $t \in T$.

Naturally, any process $\mathbb{X} = (X_t)$ with $X_t : (\Omega, \mathcal{A}) \to (S, \Sigma)$ is adapted to the filtration it generates (according to 2.2.1) and, by extension, every process $\mathbb{Y}$ derived from $\mathbb{X}$ by composing it with a measurable function g via $Y_t = g(X_t)$ is adapted to the same filtration.

Given the notion of conditional expectations and filtrations, we are finally ready to define the Markov property in a continuous setting and present a generalization of Markov chains.

Definition 17. Let $\mathbb{X} = (X_t)_{t\in T}$ be a (S, Σ)-valued random process on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and $\mathbb{F} = (\mathcal{F}_t)_{t\in T}$ the filtration generated by $\mathbb{X}$. $\mathbb{X}$ is said to possess the 'Markov property', if

$$\mathbb{P}(X_{t_2} \in A|\mathcal{F}_{t_1}) = \mathbb{P}(X_{t_2} \in A|X_{t_1}) \tag{2.2.2}$$

for each $A \in \Sigma$ and arbitrary $t_1, t_2 \in T$ with $t_1 < t_2$.

Even more general, one could state the Markov property as follows:

$$\mathbb{E}(f(X_{t_2})|\mathcal{F}_{t_1}) = \mathbb{E}(f(X_{t_2})|X_{t_1}) \tag{2.2.3}$$

for all $t_1, t_2 \in T$ with $t_1 < t_2$ and any measurable bounded $f : S \to \mathbb{R}$.

Since conditional probability is defined as a conditional expectation, satisfying $\mathbb{P}(X \in A|Y) = \mathbb{E}(\mathbb{1}_A(X)|Y)$, and $\mathbb{1}_A$ is such a bounded measurable function, Equation 2.2.3 implies 2.2.2.

To show that equivalence holds in the other direction too, recall that conditional expectation is defined by an integral property, i.e. $\int_B \mathbb{E}(f(X_{t_2})|X_{t_1})d\mathbb{P} = \int_B f(X_{t_2})d\mathbb{P}$. Here, $f \circ X_{t_2}$ takes the role of a real random variable and equality holds for any integration domain B which is measurable according to the σ-algebra generated by X_{t_1}. As f and X_{t_2} are measurable, their composition is measurable too. So its pre-images are measurable sets in $(\Omega, \mathcal{A})$ and the Lebesgue integral allows for splitting set B into intersections with these pre-images. We thus transform the coordinate ω to $y = f(X_{t_2}(\omega))$ and integrate over the range of f, thereby replacing the measure with the push-forward of $\mathbb{P}$ via $f \circ X_{t_2}$, restricted to B:

$$\int_B f(X_{t_2})d\mathbb{P} = \int_{\mathbb{R}} y d\mathbb{P}(B \cap X_{t_2}^{-1}(f^{-1}(y))),$$

where

$$\mathbb{P}(B \cap X_{t_2}^{-1}(f^{-1}(y))) = \int_B \mathbb{P}(X_{t_2}^{-1}(f^{-1}(y))|X_{t_1})d\mathbb{P},$$

which equals $\int_B \mathbb{P}(X_{t_2}^{-1}(f^{-1}(y))|\mathcal{F}_{t_1})d\mathbb{P}$ for any B in $\mathcal{F}_{t_1}$, according to Property 2.2.2. We may conclude that $\int_B \mathbb{E}(f(X_{t_2})|X_{t_1})d\mathbb{P} = \int_B f(X_{t_2})d\mathbb{P}$ holds not only for sets B in $\sigma(X_{t_1})$, but any $\mathcal{F}_{t_1}$-measurable set. Thus $\mathbb{E}(f(X_{t_2})|X_{t_1})$ coincides with $\mathbb{E}(f(X_{t_2})|\mathcal{F}_{t_1})$.

While the presented general definition of Markov chains and Markov processes allow for changing the dynamics over time, we want to restrict ourselves to a type of stochastic processes that is homogeneous in time in the sense that transition probabilities depend only on the current state of the process, not on time. This property encourages to investigate

statistical quantities such as invariant densities, mean hitting times or metastability, which we will introduce and exploit later to examine the structure of a given system.

Definition 18. A Markov chain $\mathbb{X}$ is called **homogeneous**, if its dynamical behavior is not only independent of the past, but also independent of time, that is:

$$\forall s,t \in T, x,y \in S: \quad \mathbb{P}(X_{t+1} = y \mid X_t = x) \;=\; \mathbb{P}(X_{s+1} = y \mid X_s = x).$$

Being independent of time, the transition rates of homogeneous Markov chains on finite state spaces can be written conveniently as a matrix $P = (p_{ij})_{i,j\in S}$ with $p_{ij} = \mathbb{P}(X_{s+1} = j \mid X_s = i)$. For continuous state spaces, that role is taken by a stochastic kernel, denoted by κ.

Definition 19. Let (S, Σ, μ) be a measure space. A **stochastic kernel** is a measurable function $\kappa: S \times S \to \mathbb{R}$, that is both

$$\kappa(x,y) \geq 0 \quad \text{(non-negative)}$$

and

$$\int_S \kappa(x,y)\mu(dx) = 1 \quad \text{(normalized)}.$$

For any measure space (S, Σ, μ), the function space L^1, short for $L^1(S, \Sigma, \mu)$, contains all Lebesgue-integrable real-valued functions on S and even fulfills the requirements of a vector space, using the L^1-norm

$$||f||_{L^1} = \int_S |f(x)|\mu(dx)$$

and the associated distance $d_{L^1}(f,g) = ||f-g||_{L^1}$. Elements of L^1 are considered equal, if they are point-wise equal almost everywhere, as the Lebesgue-integral over an almost-everywhere-zero function must vanish.

We may then define an integral operator $P: L^1 \to L^1$ acting on integrable functions via stochastic kernel κ according to

$$Pf(x) = \int_S \kappa(x,y)f(y)\mu(dy) \quad \forall f \in L^1.$$

By construction, this operator inherits linearity from the integral operator and non-negativity of the stochastic kernel. It is also norm-preserving non-negative L^1-functions $f \geq 0$, as

$$||Pf||_{L^1} = \int_S \left| \int_S \kappa(x,y)f(y)\mu(dy) \right| \mu(dx) = \int_S |f(y)|\mu(dy) \int_S \kappa(x,y)\mu(dx) = ||f||_{L^1}.$$

We call such operators **Markov operators**. For finite state spaces with a counting measure, P is a stochastic matrix and defines a homogeneous Markov chain.

2.2.2 Random walks and transfer operators

Within the function spaces defined above, we are interested in those functions describing a distribution of matter or probabilities on this space, normalized with respect to the given

measure. Densities which are invariant under propagation play an important role in the next chapters. We shall begin with a discrete setting, then look into systems continuous in space and time.

A most important example of random processes for the purpose of analyzing networks are 'random walkers'. The basic idea, rigorously formulated in the early 2000's [76, 55], is to define a homogeneous Markov chain on the vertex set of a given network, which in each time step moves along a randomly chosen incident edge to a neighboring node. In the case of directed networks, only outgoing edges may be chosen by the random walker. For undirected networks, such a process is always reversible. Intuitively, network clusters which have few connections to the remaining network act like a trap to such a walker, so suitable algorithms may exploit the correspondence between network clusters and metastable sets of random walker processes, which can be calculated from a spectral decomposition or hitting time analysis as described in the next chapter.

Definition 20. More generally, for any countable state space S, any stochastic matrix $P \in \mathbb{R}^{|S|} \times \mathbb{R}^{|S|}$ and any initial probability distribution p_0 on S, there is a Markov chain $(X_n)_{n\in\mathbb{N}}$ defined by these properties:

$$\mathbb{P}[X_0 = i] = p_0(i) \quad \text{and} \quad \mathbb{P}[X_{k+1} = j | X_k = i] = P(i,j) \quad \forall i, j \in S.$$

The transition probabilities are independent of time, so this Markov chain is homogeneous. Matrix P s called the **transition matrix** of (X_n).

For such processes, it is simple to define and, in the finite special case, compute the distribution of (X_n) for $n \geq 1$ from P and initial distribution p_0. P defines a linear operator $\mathcal{P}$ acting on the space of vectors with finite absolute sums $l^1 = \{v : S \to \mathbb{R} \mid \sum_{i\in S} |v(i)| < \infty\}$ by

$$(\mathcal{P}v)(j) = \sum_{i\in S} P(i,j)v(i).$$

The restriction to l^1 guarantees that $\mathcal{P}$ is well-defined, yet in applications with finite state spaces, such as random walkers on finite graphs, any vector on S lies in the range of $\mathcal{P}$. If X_0 is distributed according to some probability distribution v, it follows that X_1 is distributed according to $\mathcal{P}v$ since

$$(\mathcal{P}v)(j) = \sum_{i\in S} P(i,j)v(i) = \sum_{i\in s} \mathbb{P}[X_1 = j | X_0 = i]\mathbb{P}[X_0 = i] = \mathbb{P}[X_1 = j].$$

Likewise, due to (X_n) being a homogeneous process, higher powers of $\mathcal{P}$ allow to describe the distribution of a Markov chain initiated according to v_0 after k time steps as $\mathcal{P}^k v_0$. Thus, the operator $\mathcal{P}$ is known as **propagator**. It pushes probability distributions forward in time and therefore allows to define distributions that are invariant with respect to that propagation.

Definition 21. Let $\mathcal{P}$ be a propagator. Any distribution π which satisfies

$$\mathcal{P}\pi = \pi,$$

in other words: which are right-hand side eigenfunctions of $\mathcal{P}$, are called a **stationary distribution** or **invariant measure** of $\mathcal{P}$.

For a Markov chain on a finite state space given by transition matrix P with unique stationary distribution $\pi = \pi P$, we may define an operator $\mathcal{T}$ on the space of vectors

v, which are not in general probability distributions themselves, but must be probability vectors 'with respect to π', which means that v is non-negative and becomes a probability vector when weighted with π:

$$\sum_{x \in S} v(x)\pi(x) = 1.$$

$\mathcal{T}v$ may then be explicitly evaluated by the formula

$$(\mathcal{T}v)(y) = \frac{1}{\pi(y)} \sum_{x \in S} P(x, y)v(x)\pi(x).$$

Compared to $\mathcal{P}$, the operator $\mathcal{T}$ acts on a wider range of functions, as it propagates not only probability distributions, but any vector whose π-weighted sum of entries is finite.

While it is well-defined on the space $l^1(\pi) = \{v : S \to \mathbb{R} \mid \sum_{x \in S} |v(x)\pi(x)| < \infty\}$, we prefer to consider $\mathcal{T}$ as an operator on $l^2(\pi) = \{v : S \to \mathbb{R} \mid \sum_{x \in S} v(x)^2\pi(x) < \infty\}$, which has the added benefits of a Hilbert space when equipped with the π-weighted scalar product

$$\langle v, w \rangle_\pi = \sum_{x \in S} v(x)w(x)\pi(x).$$

In the special case of reversible Markov chains, a useful consequence is that $\mathcal{T}$, as an operator on $l^2(\pi)$, is self-adjoint, that is: $\langle \mathcal{T}v, w \rangle_\pi = \langle v, \mathcal{T}w \rangle_\pi$. It has real eigenvalues $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \dots$ contained in $[-1, 1]$ and corresponding eigenvectors $u_1, u_2, u_3, \dots$ which form an orthonormal basis of $l^2(\pi)$ [63]. We shall call $\mathcal{T} : l^2(\pi) \to l^2(\pi)$ a **transfer operator**.

Using this orthonormal basis, the spectral decomposition of the transfer operator can be written as

$$\mathcal{T}^k v = \sum_{i=1}^{\infty} \lambda_i^k \langle v, u_i \rangle u_i.$$

A notable consequence is that aperiodic and irreducible systems, which requires that no eigenvalue of $\mathcal{T}$ equals -1 and exactly one eigenvalue to be equal to 1, any initial distribution converges to the stationary distribution when propagated by $\mathcal{T}$ and the rate of convergence depends on the second-largest eigenvalue. In other words, any such system (X_n) has a steady state and the distribution of X_k converges to that stationary distribution as $k \to \infty$, no matter the initial conditions. If (X_n) starts with that stationary distribution, the random variables X_k are equally, but not independently, distributed.

Much of these insights translate into continuous settings, though several definitions need to be adapted accordingly. Transition kernels take the role of transition rate matrices and we need to distinguish between distribution functions and their densities.

Definition 22. Any non-negative and normalized L^1-function f of some measure space (S, Σ, μ), that is: any element of $D(S, \Sigma, \mu) = \{f \in L^1(S, \Sigma, \mu) : f \geq 0, ||f||_{L^1} = 1\}$ is called a **density** function.

Densities describing an equilibrium state are of special importance. Technically, they are defined by the property of staying invariant under the action of a given Markov operator.

Definition 23. When (S, Σ, μ) is a measure space and P is a Markov operator, a density f is called a **stationary density**, if

$$Pf = f.$$

Density functions describe the relation between an associated measure and some reference measure, usually the Lebesgue measure. Some publications use both measures and their density functions interchangeably, yet to avoid confusion, it shall be stated explicitly that a density f and its associated measure ν_f are related as

$$\nu_f(A) = \int_A f(x)dx.$$

One helpful tool to describe the evolution of probability distributions according to the dynamics of a given Markov chain is a kernel function which contains transition probabilities at any point in time t from any starting point $x \in S$ into any subset of the state space $B \in \Sigma$.

Definition 24. Let $\mathbb{X}$ be a homogeneous Markov chain on some state space S equipped with σ-algebra Σ. A stochastic kernel $\kappa : \ S \times S \to [0, 1]$ is called the **transition kernel** of $\mathbb{X}$ if

1. $x \mapsto \kappa(x, B)$ is Σ-measurable for any set $B \in \Sigma$,
2. $B \mapsto \kappa(x, B)$ is a probability measure on (S, Σ) for every fixed state $x \in S$ and
3. it describes the transition rules of the process: $\forall t \in T, x \in S, B \in \Sigma : \ \mathbb{P}(X_{t+1} \in B \mid X_t = x) = \kappa(x, B)$.

Using transition kernels, we can formulate the propagation of densities analogously to the discrete setting as

$$(\mathcal{P}_t f)(y) = \int_S p(t, x, y)f(x)dx.$$

When μ is a unique stationary density of $\mathcal{P}_t$ as defined above, we may properly define continuous-time Transfer Operator $\mathcal{T}_t$ on $L^2(\mu)$ by

$$(\mathcal{T}_t f)(y) = \frac{1}{\mu(y)} \int_S p(t, x, y)f(x)\mu(x)dx$$

where $L^2(\mu) = \{f : S \to \mathbb{R} \mid \int_S f^2(x)\mu(x)dx < \infty\}$ is again a Hilbert space with a μ-weighted scalar product:

$$\langle f, g \rangle_\mu = \int_S f(x)\mu(x)g(x)dx.$$

Unlike the discrete case, where a single transfer operator was sufficient to describe the evolution of the distribution of a system, we now have a semi-group of operators with $T_{s+t} = T_s T_t$ and $T_0 = Id$, that depends on the distance in time between two states of the system X_{t_0} and X_{t_0+t}.

However, when $\mathcal{T}_t$ is 'strongly continuous' in the sense that it perturbs less and less, the smaller t is chosen, or technically written:

$$\lim_{t\to 0} \mathcal{T}_t f = f,$$

then we may define an **infinitesimal generator** L of $\mathcal{T}_t$ as

$$Lf = \lim_{t\to 0} \frac{\mathcal{T}_t f - f}{t}.$$

For any $f \in L^2(\mu)$ where that limit exists, it also exists at $\mathcal{T}_t f$ [29] and the relation between $\mathcal{T}_t$ and L may be rewritten as

$$\mathcal{T}_t f = \exp(Lt) f.$$

As we see, knowledge of L is sufficient to describe the whole family of operators $\{\mathcal{T}_t\}_{t\in\mathbb{R}^+}$, hence the name 'generator'. The eigenvalues of L are thus exponentially related to those of $\mathcal{T}_t$ and we may deduce characteristics of the system from properties of L similar to the spectral decomposition of discrete transfer operators.

Having introduced the notion of information gained by observing a process in Definitions 15 and 16 earlier, we may define random variables that track when a process meets a pre-defined condition, based on information received from the observation. For example, one might be interested in when a random process hits a certain subset of the state space for the first time. As the actual point of time may depend on chance, the value could differ for every single realization of the process, but the expected values of such functions are statistical quantities that hold information about the averaged behavior of a process.

Definition 25. Given a random process $\mathbb{X}$ on index set T and the filtration $\mathbb{F}$ generated by $\mathbb{X}$, a **stopping time** τ is a T-valued random variable which satisfies $\{\omega \in \Omega : \ \tau(\omega) \leq t\} \in \mathcal{F}_t$.

In other words: For any point in time t, it is possible to decide whether the stopping condition has been met by a realization of the process based only on information obtained by observing it until time t. If so, its value is determined and must be lesser or equal to t. If not, its value is unknown, but definitely greater than t. A very common example we shall encounter again in a later section is the '(first) hitting time' τ_A of a subset A of the state space S of a random process (X_t)

$$\tau_A(\omega) = \inf\{t \in T \mid X_t(\omega) \in A\}.$$

A closely related notion shall be introduced here for the sake of completeness: The '(first) exit time' from a set $A \subset S$ is the hitting time $\tau_{S\setminus A}$ of the complement of A in S.

2.2.3 Stationarity, ergodicity, reversibility

Now, we turn to characteristics of random processes that we typically seek or presume, as they ensure favorable properties for numerical treatment or analytical results that can be exploited to learn about the structure of the state space and dynamics of the process.

Definition 26. A **random process** $\mathbb{X} = (X_t)_{t\in T}$ is called **(strictly) stationary** if for all natural numbers k and all time points $t_1, \ldots, t_k$ and any shift value τ the cumulative

distribution function $F_{\mathbb{X}}$ of the joint distribution of (X_t) at times $t_1 + \tau, \ldots, t_k + \tau$ does not depend on τ, thus fulfilling

$$F_{\mathbb{X}}(t_1 + \tau, \ldots, t_k + \tau) = F_{\mathbb{X}}(t_1, \ldots, t_k).$$

Definition 27. A stationary process $\mathbb{X} = (X_t)_{t \in T}$ is called **(mean-)ergodic** if the temporal mean of a trajectory converges almost surely to its ensemble mean, that is:

$$\frac{1}{T} \int_0^T X_t dt \xrightarrow[T \to \infty]{} \mathbb{E}\left[X_t\right].$$

Ergodicity is a property of a process that ensures that (almost) any infinitely long trajectory reproduces the expected value of the process weighted according to the inherent probability measure of the space. Ergodic processes cannot have non-trivial 'traps' or insurmountable barriers separating non-null sets, as (almost) any trajectory must have a non-zero chance to visit any non-null subset of the state space again.

An even stronger property demands that the process itself is indistinguishable from a copy of the process running backward in time. This is the continuous analogon to a finite discrete homogeneous process with transition probabilities $p(\cdot, \cdot)$ fulfilling the **detailed balance condition** with respect to a unique stationary probability vector μ:

$$\mu(x)p(x, y) = \mu(y)p(y, x) \quad \forall x, y \in S.$$

Definition 28. A stationary random process $\mathbb{X} = (X_t)_{t \in T}$ is called **time-reversible** if

$$\mathbb{P}(X_{t+\tau} \in B \mid X_t \in A) \;=\; \mathbb{P}(X_{t-\tau} \in B \mid X_t \in A)$$

for any time $t \in T$, any time shift $0 < \tau < t$ and any measurable sets A, B.

In the case of random walks on graphs, these properties are directly related to the structure of the graph. For any connected graph, the random walk is an irreducible process. On undirected graphs, random walks are reversible, which follows directly from the symmetry of their adjacency matrices. Random walks are also aperiodic, unless the graph is bipartite. When these properties coincide, namely on connected, undirected, non-bipartite graphs, the resulting random walk process is ergodic [23].

Two results of particular importance will make obvious why ergodicity is so relevant for many methods of network analysis. The first one ensures convergence of probability distributions of finite random walks towards a unique stationary distributions.

Theorem 2.2.3. *Let transition matrix P define a discrete random walk on state space S. There exists a unique stationary distribution μ such that*

$$\lim_{k \to \infty} P^k(x, y) = \mu(y) \quad \forall x, y \in S.$$

A proof can be found in textbooks such as [10]. The theorem guarantees convergence to μ independent of the starting point x and thus ensures that the system approaches a steady state no matter the initial probability distribution. Computing higher powers of transition matrix P even allows to approximate μ computationally without solving a possibly ill-posed linear problem, which is a feasible job when the state space is not

prohibitively large. The second result is based on a theorem by Oskar Perron and Georg Frobenius [57, 31] dating back to the early 20th century. This specific application was published in [72]. Note that it is not restricted to reversible Markov chains.

Theorem 2.2.4. *Let transition matrix P define an aperiodic irreducible Markov chain $(X_n)_{n\in\mathbb{N}}$ on finite state space S, then*

1. *P has a unique eigenvalue $\lambda_1 = 1$, corresponding left eigenvector $\mathbf{1} = (1, \ldots, 1)^T$ and right eigenvector μ with strictly positive entries.*
2. *All other eigenvalues of P are real-valued and have strictly smaller absolute values than λ_1.*

For the purpose of analyzing networks, particularly for the detection of clusters, a suitable structure is not guaranteed. However, when a graph is not connected, each of its components may be looked at individually as a connected subgraph. Bipartite graphs require special treatment, as standard discrete random walks are not guaranteed to be aperiodic. Possible workarounds include inserting artificial self-loops on each vertex to enforce aperiodicity or resorting to continuous random walk processes. With some restrictions, random walk based methods may be applied to nearly any undirected network which actually has a modular structure.

Once orientations are added to edges and random walkers are taking these into account, much of the required theory no longer holds. For instance, such random walks are not necessarily reversible. Their transfer operators are, in general, not self-adjoint, their spectra may include complex eigenvalues and the correspondence between their eigenvectors and graph clusters is no longer assured. Even the matter of how to define a cluster in a directed graph, is more ambiguous then in the undirected case. Several suggestions for definitions of modules or clusters only coincide in the special case of undirected graphs. The definition we use, based on the dynamics of a random walker process, may or may not agree with functional subunits of a system, or categories of similar items that specific application cases require.

In the following chapter, we shall highlight a few existing methods that exploit information encoded in the spectrum of transfer operators to identify invariant, cyclic and metastable structures.

3 | Clustering undirected graphs

3.1 Definition of clusters

Part of what makes the problem of clustering graphs so rich, is that even the definition of what a cluster is, is a matter of on-going discussion. Usually, it is seen as a property associated to edge-density. That is, a set of vertices of a graph is called a cluster, when its members are highly connected internally and are sparsely linked to the remainder of the graph.

Depending on the application, this description may or may not serve well in finding results of high quality. A graph on vertices representing people in a sociological context where edges indicate which people share friendship fits well to this definition of clusters.

On the other side, if we analyze the link structure of Wikipedia articles and want to group them based on topics, we might find that very similar articles, like those of different subspecies of a plant for instance, often link to the same articles, such as their classification family, their common biotope or clime, the articles 'Sun' and 'Water', but barely point to each other. Therefore, the authors of a recent paper propose to group vertices in directed graphs based on similarity of their neighborhoods [68]. Vertices both pointing to the same vertices or receiving links from the same vertices are considered similar and thus more likely to be grouped together.

As another example, let us take a citation network, where papers citing the same sources are likely to share a topic, while they might have been published too close together in time to cite one another. Furthermore, the homepages of enterprises settled in the same business field are highly unlikely to link to each other, even more so, the more similar their products or services are.

In the backgrounds selected by the authors to validate the superiority of their new approach and to reveal the flaws of other algorithms, this way of defining a similarity measure and identifying clusterings based on it shows to correspond better to intuition than grouping vertices based on edge-density.

However, there is a well-founded reason for the upper definition to be predominant from a random walk process based point of view. When networks on thousands of vertices are too large to analyze efficiently, some researchers are interested in simplified networks on a considerably smaller vertex set, resembling the dynamics of the original network as good as possible. Defining random walk processes on the graphs, we can measure the loss of information caused by the simplification of the network in a way we will discuss later on. We will recognize that dominant timescales of the random walk process can be well

preserved by milestoning processes based on clusterings in the sense of grouping densely connected vertices together. We shall return to random walk induced clusterings soon, but in the following section we commence with famous non-spectral clustering algorithms often used as reference due to their spread and frequent use.

3.2 Non-spectral approaches

3.2.1 Graclus

Graclus is a multi-level algorithm claimed to outperform pure spectral algorithms in terms of speed and memory usage [22]. Instead of employing expensive eigenvector calculations, it uses a kernel-based approach, exploiting a connection between vector clustering kernel k-means and graph clustering objectives.

We shall take a closer look, both because it is a widely used reference algorithm [67, 50, 68, 4] and it reveals an insightful relation between partitioning a graph and clustering a set of vectors in the real space. Out of many similar approaches, we will focus on one showcase method for each problem and prove that they can be expressed as a matrix trace maximization due to particular constraints and are therefore equivalent.

Vector clustering

First let us consider the problem to assign n vectors $a_1, \ldots, a_n$ of dimension m to k clusters $\pi_1, \ldots, \pi_k$ minimizing the objective function

$$\sum_{c=1}^{k} \sum_{a_i \in \pi_c} ||a_i - m_c||^2, \quad \text{where } m_c = \frac{\sum_{a_i \in \pi_c} a_i}{|\pi_c|}.$$

So the goal is to choose clusters such that the sum of the squared distances of their members to their mean is as small as possible. This implies that each vertex is assigned to the cluster whose mean is closest to that vertex. Therefore, the boarder separating any two clusters is always a hyperplane orthogonal to the straight line connecting their means, crossing this line right in the middle. This is a restriction inherent to the standard k-means we want to dispose of by generalizing the idea to so-called kernel k-means.

Let us assume that ϕ is a function mapping the vectors $a_1, \ldots, a_n$ to a higher dimensional space and apply standard k-means clustering there, then project the obtained clustering back to the original space. Depending on ϕ the separating hyperplanes in the higher dimensional space may or may not become linear separators after projection onto the original space. So we have gained a more general method which allows for non-linear separators and the choice of ϕ is still open. The objective function may be formalized as

$$\sum_{c=1}^{k} \sum_{a_i \in \pi_c} ||\phi(a_i) - m_c||^2, \quad \text{where } m_c = \frac{\sum_{a_i \in \pi_c} \phi(a_i)}{|\pi_c|}.$$

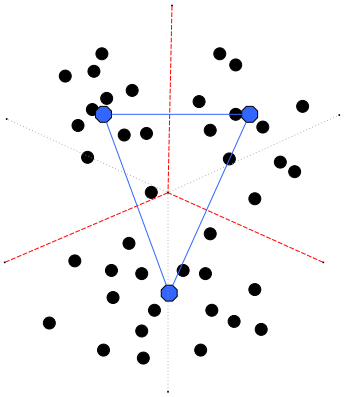


Figure 3.1: Standard k-means clustering of a set of two-dimensional vectors. Boarders separating any two clusters are always orthogonal to the line connecting their barycenters.

Note that the squared distance required may be rewritten using inner products only

$$||\phi(a_i) - m_c||^2 = \phi(a_i) \cdot \phi(a_i) - \frac{2\sum_{a_j \in \pi_c} \phi(a_i) \cdot \phi(a_j)}{|\pi_c|} + \frac{\sum_{a_j, a_l \in \pi_c} \phi(a_j) \cdot \phi(a_l)}{|\pi_c|^2}.$$

If we have a kernel matrix K providing us all required inner products $K_{ij} = \phi(a_i) \cdot \phi(a_j)$, we do not even need to know ϕ itself. Without a proof, we will exploit the fact that any positive semidefinite matrix can be regarded as a kernel matrix to some function ϕ. Popular choices are Polynomial Kernels $K_{ij} = (a_i \cdot a_j + c)^d$, Gaussian Kernels $K_{ij} = \exp(-||a_i - a_j||^2/2\alpha^2)$ and Sigmoid Kernels $K_{ij} = \tanh(c(a_i \cdot a_j) + \theta)$ [22].

Our next step is to generalize the objective function a bit further by introducing a non-negative weighting w, such that the weighted k-means objective function now looks like this

$$\sum_{c=1}^{k} \sum_{a_i \in \pi_c} w_i ||\phi(a_i) - m_c||^2, \quad \text{where } m_c = \frac{\sum_{a_i \in \pi_c} w_i \phi(a_i)}{\sum_{a_i \in \pi_c} w_i}.$$

Here, we want to quote an algorithm of complexity $\mathcal{O}(n^2)$ which, starting from an arbitrary initial clustering, optimizes the weighted objective function until a maximum number of iterations t_{max} or convergence is reached. The algorithm is guaranteed to converge monotonically for any positive semidefinite kernel matrix K [22].

We will show that this is in fact a matrix trace maximization problem. To this end, let $s_c := \sum_{a_i \in \pi_c} w_i$ and define the $n \times k$ matrix Z by

$$Z_{ic} := \begin{cases} 1/\sqrt{s_c} & \text{if } a_i \in \pi_c \\ 0 & \text{otherwise.} \end{cases}$$

As the clusters are disjoint, the columns of Z are mutually orthogonal. In addition, let Φ be the $m \times n$ matrix whose columns are the vectors $\phi(a_i)$ and W a diagonal matrix containing the weights $W_{ii} = w_i$. We may then rewrite the weighted kernel k-means objective function as

$$\sum_{c=1}^{k} \sum_{a_i \in \pi_c} w_i ||\phi(a_i) - m_c||^2 = \sum_{i=1}^{n} w_i ||\Phi_{:i} - (\Phi W Z Z^T)_{:i}||^2,$$

Algorithm 1 Kernel k-means Batch

Input: kernel matrix K, number of clusters k, weighting w, maximum number of iterations t_{max}, initial clustering $\{\pi_c^0\}_{c=1}^k$

Output: final clustering $\{\pi_c\}_{c=1}^k$

$t = 0$
repeat
  for $i = 1$ **to** n **do**
    for $c = 1$ **to** k **do**
      $d(a_i, m_c) = K_{ii} - \frac{2\sum_{a_j \in \pi_c^t} w_j K_{ij}}{\sum_{a_j \in \pi_c^t} w_j} + \frac{\sum_{a_j, a_l \in \pi_c^t} w_j w_l K_{jl}}{(\sum_{a_j \in \pi_c^t} w_j)^2}$
    end for
  end for
  for $i = 1$ **to** n **do**
    $c^*(a_i) = \arg\min_c d(a_i, m_c)$
  end for
  for $c = 1$ **to** k **do**
    $\pi_c^{t+1} = \{a | c^*(a_i) = c\}$
  end for
  $t = t + 1$
until $t = t_{max}$ or $\{\pi_c^t\}_{c=1}^k = \{\pi_c^{t-1}\}_{c=1}^k$
return $\{\pi_c^t\}_{c=1}^k$

where $A_{:i}$ denotes the i-th column of a matrix A. Since W is a diagonal non-negative matrix, the matrix exponential which we use in the following is well defined. We introduce the orthonormal matrix $\tilde{Y} := W^{1/2}Z$ replacing Z in the objective function

$$\sum_{i=1}^{n} w_i ||\Phi_{:i} - (\Phi W Z Z^T)_{:i}||^2 = \sum_{i=1}^{n} w_i ||\Phi_{:i} - (\Phi W^{1/2} \tilde{Y} \tilde{Y}^T W^{-1/2})_{:i}||^2$$
$$= \sum_{i=1}^{n} ||\Phi_{:i} w_i^{1/2} - (\Phi W^{1/2} \tilde{Y} \tilde{Y}^T)_{:i}||^2 = ||\Phi W^{1/2} - \Phi W^{1/2} \tilde{Y} \tilde{Y}^T||_F^2.$$

This is the squared Frobenius norm of the matrix $(\Phi W^{1/2} - \Phi W^{1/2} \tilde{Y} \tilde{Y}^T)$. Now we exploit the relation $||A||_F^2 = \text{trace}(A^T A)$

$$||\Phi W^{1/2} - \Phi W^{1/2} \tilde{Y} \tilde{Y}^T||_F^2 = \text{trace}((\Phi W^{1/2} - \Phi W^{1/2} \tilde{Y} \tilde{Y}^T)^T \cdot (\Phi W^{1/2} - \Phi W^{1/2} \tilde{Y} \tilde{Y}^T)),$$

process the outer transposition symbol

$$= \text{trace}((W^{1/2} \Phi^T - \tilde{Y} \tilde{Y}^T W^{1/2} \Phi^T) \cdot (\Phi W^{1/2} - \Phi W^{1/2} \tilde{Y} \tilde{Y}^T)),$$

multiply the terms in brackets out

$$= \text{trace}(W^{1/2} \Phi^T \Phi W^{1/2} - W^{1/2} \Phi^T \Phi W^{1/2} \tilde{Y} \tilde{Y}^T$$
$$- \tilde{Y} \tilde{Y}^T W^{1/2} \Phi^T \Phi W^{1/2} + \tilde{Y} \tilde{Y}^T W^{1/2} \Phi^T \Phi W^{1/2} \tilde{Y} \tilde{Y}^T)$$

and make use of $\text{trace}(A + B) = \text{trace}(A) + \text{trace}(B)$ and $\text{trace}(AB) = \text{trace}(BA)$

$$= \text{trace}(W^{1/2} \Phi^T \Phi W^{1/2}) - \text{trace}(\tilde{Y}^T W^{1/2} \Phi^T \Phi W^{1/2} \tilde{Y})$$
$$- \text{trace}(\tilde{Y}^T W^{1/2} \Phi^T \Phi W^{1/2} \tilde{Y}) + \text{trace}(I_k \tilde{Y}^T W^{1/2} \Phi^T \Phi W^{1/2} \tilde{Y}),$$

to finally simplify the whole term

$$= \operatorname{trace}(W^{1/2}\Phi^T\Phi W^{1/2}) - \operatorname{trace}(\tilde{Y}^T W^{1/2}\Phi^T\Phi W^{1/2}\tilde{Y})$$

into the constant part $\operatorname{trace}(W^{1/2}\Phi^T\Phi W^{1/2})$ and a negative part which may be optimized by choosing $\tilde{Y}$ appropriately. Note that $(\Phi^T\Phi)_{ij} = \phi(a_i)\cdot\phi(a_j) = K_{ij}$ and thus $\Phi^T\Phi = K$, so that we can reformulate the minimization of the weighted kernel k-means objective as

$$\max_{\tilde{Y}} \operatorname{trace}(\tilde{Y}^T W^{1/2} K W^{1/2}\tilde{Y}). \tag{3.2.1}$$

Graph clustering

We shall now look at the problem of clustering a graph G on a finite vertex set $V = \{1, \dots, n\}$ from another point of view. Here, we consider a weighted adjacency matrix A, whose positive entries are regarded as an edge weighting while zero-entries mean that there is no edge connecting the correspondent vertices. In addition, there is a vertex weighting w_i. Using the notations $w(S) := \sum_{i\in S} w_i$, $\operatorname{links}(S, S') := \sum_{i\in S, j\in S'} A_{ij}$ and just for simplification $\operatorname{cut}(S) := \operatorname{links}(S, V\setminus S) = \sum_{i\in S, j\notin S} A_{ij}$, we may formulate the Weighted Cut objective as

$$\operatorname{WCut}(G) = \min_{C_1,\dots,C_k} \sum_{c=1}^{k} \frac{\operatorname{cut}(C_c)}{w(C_c)}.$$

Since Ratio Cut and Normalized Cut, two other frequently used clustering objectives, are special cases of WCut with suitable vertex weightings, we content ourselves with considering the Weighted Cut objective only.

A similar clustering objective, the Weighted Association, seeks to maximize the total weight of connections within each cluster and reads as

$$\operatorname{WAssoc}(G) = \max_{C_1,\dots,C_k} \sum_{c=1}^{k} \frac{\operatorname{links}(C_c, C_c)}{w(C_c)}.$$

In a short while we will realize the immediate relation to WCut, but before we deviate a trace maximization interpretation of WAssoc using characteristic vectors of the clusters defined by

$$(\chi_c)_i = \begin{cases} 1 & \text{if } i \in C_c \\ 0 & \text{otherwise,} \end{cases}$$

rescaled using the diagonal weight matrix to $\tilde{\chi}_c = \chi_c \cdot (\chi_c^T W \chi_c)^{-1/2}$, and the matrix $Y = W^{1/2}\tilde{X}$. Via the matrix $\tilde{X}$, whose columns are the rescaled characteristic vectors of the clusters, the alignment of vertices to clusters is now contained within Y. Thus, we now gain

$$\begin{aligned} \operatorname{WAssoc}(G) &= \max_{C_1,\dots,C_k} \sum_{i=1}^{k} \frac{\operatorname{links}(C_i, C_i)}{w(C_i)} = \max_{\chi_1\cdots,\chi_k} \sum_{c=1}^{k} \frac{\chi_c^T A \chi_c}{\chi_c^T W \chi_c} = \max_{\tilde{\chi}_1\cdots,\tilde{\chi}_k} \sum_{c=1}^{k} \tilde{\chi}_c^T A \tilde{\chi}_c \\ &= \max_{Y} \operatorname{trace}(Y^T W^{-1/2} A W^{-1/2} Y). \end{aligned}$$

Using the same notation as above together with the diagonal matrix D of weighted vertex degrees, that is $D_{ii} = \sum_{j=1}^{n} A_{ij}$ and the **Laplacian** $L = D - A$ of the graph, we

can reformulate WCut in a similar fashion

$$\begin{aligned}\mathrm{WCut}(G) &= \min_{\chi_1,\dots,\chi_k} \sum_{c=1}^{k} \frac{\chi_c^T (D-A)\chi_c}{\chi_c^T W \chi_c} = \min_{\tilde{\chi}_1,\dots,\tilde{\chi}_k} \sum_{c=1}^{k} \tilde{\chi}_c^T L \tilde{\chi}_c \\ &= \min_{Y} \mathrm{trace}(Y^T W^{-1/2} L W^{-1/2} Y)\end{aligned}$$

and recognize a problem very similar to the trace maximization problem WAssoc. Moreover, exploiting

$$\mathrm{trace}(Y^T W^{-1/2}(W-L)W^{-1/2}Y) = k - \mathrm{trace}(Y^T W^{-1/2} L W^{-1/2} Y)$$

we realize that minimizing WCut is in fact equivalent to maximizing WAssoc on a graph whose adjacency matrix is given by $A = W - L$ and therefore both objectives may be rewritten as a matrix trace maximization problem identical to the weighted kernel k-means objective 3.2.1 since Y corresponds to $\tilde{Y}$ from the previous section. Given a weight matrix W, we may map from one problem to the other by setting $K = W^{-1}AW^{-1}$ or $A = WKW$.

Many graph clustering objectives, such as Normalized, Ratio or Weighted Cut, no longer require eigenvector calculations as they can be taken as a weighted kernel k-means optimization problem and solved efficiently using algorithms such as kernel k-means Batch [1]. This result is key to the multi-level algorithm Graclus.

Graclus basic structure

Coarsening phase Starting from an initial graph G_0 to be clustered, the goal of the first phase is to generate a sequence of graphs $G_1, \dots, G_N$ on progressively smaller vertex sets $|V_0| > \cdots > |V_N|$, such that several vertices in V_i are combined into one vertex of V_{i+1} whose weight is the sum of weights of the vertices that were combined. At the same time, the edge weighting given by the adjacency matrix A is treated as well. Let us assume, the vertices $x_1, \dots, x_r \in V_i$ are to be combined into vertex $x^* \in V_{i+1}$ and $y_1, \dots, y_s \in V_i$ are selected for merger into $y^* \in V_{i+1}$ and there is at least one edge connecting the sets $\{x_1, \dots, x_r\}$ and $\{y_1, \dots, y_s\}$. Then the vertex x^* must share an edge with y^* in G_{i+1} and its weight is the sum of the weights of all edges between $\{x_1, \dots, x_r\}$ and $\{y_1, \dots, y_s\}$ in G_i. So if A_i is the weighted adjacency matrix of G_i, we may compute the weight of the new edge as

$$A_{i+1}(x^*, y^*) = \sum_{j=1}^{r} \sum_{k=1}^{s} A_i(x_j, y_k).$$

A non-deterministic way to coarsen a graph is to go through all vertices in a random order, select an adjacent vertex fitting best for merger according to some criterion, mark both as dispatched and ignore marked vertices until the algorithm finishes and all vertices are marked. When a suitable neighbor for vertex x is to be selected, the user may implement a similarity or distance measure of his choice. The criterion should match the clustering objective, such that reverting the coarsening procedure preserves the quality of a clustering to some extend. A method called max-cut coarsening procedure respects the

edge and vertex weightings by selecting the unmarked adjacent vertex y which maximizes

$$\frac{A(x,y)}{w(x)} + \frac{A(x,y)}{w(y)}.$$

Base-clustering phase Once the original graph has been sufficiently coarsened, some basic clustering algorithm, like Metis [37] or the random walk based method which we present in Section 3.4, is used to obtain a clustering of the coarsest graph G_N. Note that Metis heavily tends towards building clusters of nearly equal size and the quality of its results may depend on the randomly selected initial vertices that grow into clusters. Though it is seen as the main advantage of Graclus not to require eigenvector calculations, since G_N is considerably smaller than the original graph G_0, spectral methods may be used without concern to generate this initial clustering.

Refinement phase The clustering constructed in the second phase is used to initialize the refinement method for G_N. The clustering obtained in this way induces a clustering on G_{N-1} in a very natural way: If a certain vertex $x \in V_N$ has been assigned to some cluster then all vertices in V_{N-1} that have been merged into x during the coarsening process shall be assigned to the same cluster. In every subsequent step, the clustering induced by the refined clustering of the previous step initializes the refinement algorithm for the current step. This procedure usually finishes quickly, as each step is starting from a good initial clustering.

The practical implementation incorporates some optimizations which improve stability and speed. A diagonal shift guarantees positive semidefiniteness of the kernel used in the k-means batch algorithm, thus ensuring monotonic convergence. In addition, Graclus alternates between steps of the batch algorithm and incremental updates based on a local search to avoid getting stuck in poor local minima. Pre-computation of weights and constants as well as restricting the refinement on cluster boundaries considerably speed up the computation. More details and performance tests can be found in literature [22] and the software is available under the terms of the GNU Public License[1].

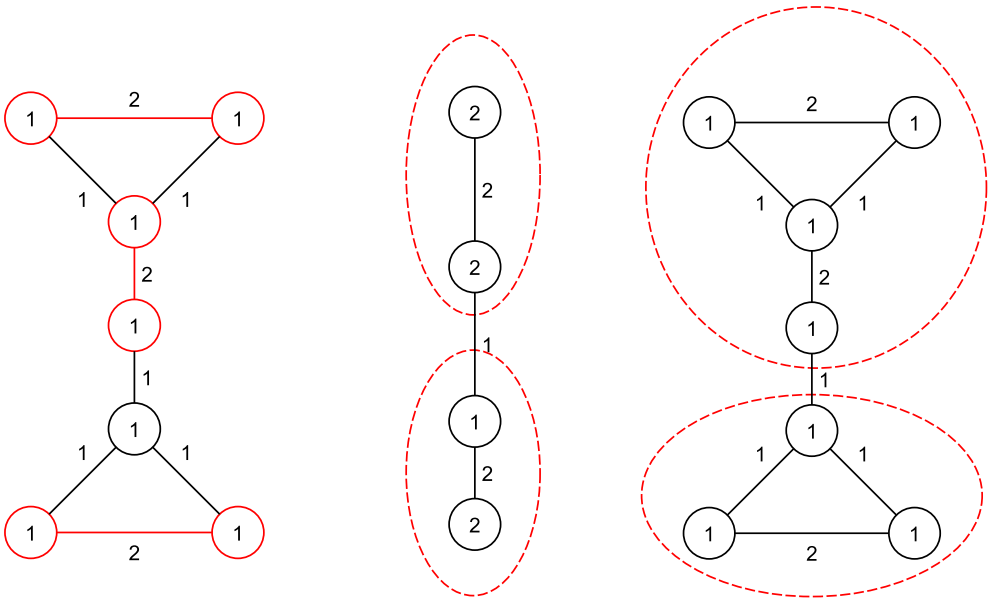
3.2.2 Markov clustering

Here, we want to present a well-known clustering algorithm by van Dongen [78] and an altered version invented by Satuluri et al. [67]. The **Markov clustering (MCL)** employs simple matrix calculations only and abandons spectral properties and eigenvector computations altogether.

The original algorithm is based on the idea of gradually emphasizing edges in well-connected communities while fading out other edges, such that a subgraph remains, where each component corresponds to a cluster in the original graph. D is the diagonal matrix containing the degree numbers of the vertices. The operator '$.^$' means entry-wise exponentiation and r is the inflation rate, usually set to 2.

While this approach is compelling due to its simpleness and robustness, its performance might be insufficient when applied to large scale networks, as it tends to deliver an inflated

[1] http://www.cs.utexas.edu/users/dml/Software/graclus.html



(a) **(b)** **(c)**

Figure 3.2: Simplified illustration of the phases of a multi-level clustering algorithm, numbers denote vertex and edge weights.
Pairs of vertices are matched as candidates for mergence during the coarsening process (a). A base-clustering of the coarsened graph (b) is sought which induces a clustering of the original graph (c) by expanding previously merged vertices, delivering the final clustering after refinement.

number of clusters. Therefore, Satuluri et al. have developed a modification known as **regularized Markov clustering (R-MCL)** which prevents the divergence between neighboring vertices and invented a multi-level implementation (MLR-MCL) frequently used until present day [67].

The basic concept is to coarsen the graph progressively, run the R-MCL algorithm on these coarsened graphs in the opposite order, beginning with the most simplified version of the graph, where it converges considerably faster than it would on the original graph, and then use the resulting matrix M of each step recursively to initialize the flow matrix used to cluster the less coarsened graphs up to the final step, which returns a clustering of the original graph.

Algorithm 2 Markov clustering - MCL

Input: adjacency matrix A
Output: cluster matrix M
1: initialize transition matrix $M := (A + Id)D^{-1}$
2: **repeat**
3: expand $M := M * M$
4: inflate $M := M.\hat{}r$
5: re-normalize columns of M
6: prune entries close to zero
7: **until** M converges
8: **return** M

Algorithm 3 Regularized Markov clustering - R-MCL

Input: adjacency matrix A
Output: cluster matrix M
1: initialize transition matrix and save a copy $M := M_G := (A + Id)D^{-1}$
2: **repeat**
3: regularize $M := M * M_G$
4: inflate $M := M.\hat{}r$
5: re-normalize columns of M
6: prune entries close to zero
7: **until** M converges
8: **return** M

MLR-MCL outline

1. generate a finite sequence $G = G_0, G_1, \ldots, G_N$ of increasingly coarse graphs on vertex sets V_i such that $|V_0| > \cdots > |V_N|$

2. apply R-MCL on G_N to gain a $|V_N| \times |V_N|$ cluster matrix M_N

3. project M_N onto $|V_{N-1}| \times |V_{N-1}|$ by restoring the vertices merged during the coarsening of G_{N-1} to G_N

4. initialize R-MCL on G_{N-1} using the projection of M_N

5. repeat steps 3 and 4, lowering all indices by one in every iteration step until a clustering of G is acquired

Although MLR-MCL overcomes the fragmentation problem and delivers faster and higher-quality results, unfortunately there is no direct generalization to directed graphs. As a remedial measure, various symmetrization methods can be applied to the adjacency matrix of the graph to obtain an undirected graph inheriting the cluster structure of the original graph, such that clustering algorithms like MLR-MCL can be implemented. Some symmetrization methods will be covered in Section 4.2.2.

Performance

An extensive comparison of quality and speed of clustering algorithms is not part of this work as others have already covered that topic in detail. See a comprehensive benchmark of spectral algorithms by Verma et al. [79] or a comparison of several symmetrization methods and spectral/non-spectral clustering algorithms by Satuluri et al. [68].

In general, state-of-the-art random walk inspired spectral algorithms show good quality if clusters are expected to be densely connected subgraphs. They are inapplicable to very large data sets however, due to the time-consuming eigenvector calculations they require.

Unless future work provides new, considerably faster ways to compute eigenvectors, algorithms such as Multi-Level Regularized Markov Clustering (MLR-MCL), Graclus or Metis outperform currently known spectral algorithms on large networks [68].

3.3 Spectral clustering of symmetric matrices

Several approaches to cluster directed graphs are based on symmetrizing its adjacency matrix and then applying methods originally invented for clustering undirected graphs.

The first ideas, borrowed from the field of bibliometrics, were to consider symmetrized matrices like $A + A^T$ or $A^T A$ [50]. However, it turns out that in many frequent cases these techniques can blur the cluster structure of the original matrix and deliver unreliable results due to the loss of information incurred by the symmetrization [56].

More sophisticated approaches we will discuss later build up on this idea nonetheless and in order to understand how they work, we must first look into spectral clustering on symmetric matrices, such as the transition matrix of a random walk process $(X_n)_{n\in\mathbb{N}}$ on an undirected graph.

We shall assume that the reader is familiar with the theory of Markov processes and the representation of time-discrete homogeneous Markov chains on a finite state space S of size n via a stochastic transition matrix $P = (p_{ij})_{1<=i,j<=n}$ containing the probabilities to switch from state i into state j in a single time-step

$$p_{ij} = \mathbb{P}(X_{n+1} = j | X_n = i).$$

The required background information on stochastic processes can be found in any elaborate textbook on the subject such as the one written by D. Stroock [75]. As we focus on random walk processes on graphs, the state space will usually be identical to the set of vertices $S := V$ and the transition matrix is induced by the edge set and, if applicable, the edge weighting.

Notation: Whenever the range of integration is omitted, it shall be set to the whole state space $\int := \int_S$.

Intuitively, random walk approaches divide the graph into sections that are highly transient each, but 'difficult' to leave once the walker has entered one of them. This way, subgraphs shall be identified that consist of densely connected vertices but are weakly connected to the remainder of the graph. The vertex set of such a subgraph is what we call a cluster or community. Here, we are not interested in full decompositions of the graph, but will first identify a set of vertices that do not belong to any cluster, then perform a full clustering of the remaining graph and finally associate the unclustered vertices using affiliation functions constructed from the partial clustering.

3.3.1 Committor functions

Committor functions are the central objects of Transition Path Theory [27] and will prove to be valuable helpers for fuzzy clusterings. Given a random walk (X_t) on a graph on state space S and a set of nonempty disjoint subsets $C_1, \ldots, C_K$ of S, the **forward committor** $q_i^+(x)$ gives the probability that the random walk enters the subset C_i before entering any other $C_j, j \neq i$ when starting at vertex x.

It follows directly from this definition that $q_i^+(x) = 1$ for any $x \in C_i$ and $q_i^+(x) = 0$ if $x \in C_j, j \neq i$. Let τ_{C_i} be the hitting time of C_i and τ_{M_i} be the hitting time of $M_i = \bigcup_{j\neq i} C_j$,

then we may define $q_i^+(x)$ by

$$q_i^+(x) = \mathbb{P}[\tau_{C_i}(x) < \tau_{M_i}(x)].$$

In analogy, we want to define the **backward committor** $q_i^-(x)$ that gives the probability that the random walk last visited the set C_i. This coincides with the forward committor of the process (X_t) running backward in time. Thus, when $\overleftarrow{\tau}_{C_i}$ and $\overleftarrow{\tau}_{M_i}$ are hitting times of the reversed process (X_{-t}), the backward committor $q_i^-(x)$ of (X_t) is given by

$$q_i^-(x) = \mathbb{P}[\overleftarrow{\tau}_{C_i}(x) < \overleftarrow{\tau}_{M_i}(x)].$$

The committor functions are uniquely given if the invariant measure is unique and does not vanish on all the sets $C_1, \ldots, C_K$. If (X_t) is even a time-reversible process, i.e. the invariant measure μ satisfies $\mu(x)p(x,y) = \mu(y)p(y,x)$, such as a random walk on an undirected graph, forward and backward committors are the same and simply called *the* committor $q_i(x)$.

Committors will be useful both as affiliation functions to assign vertices in $S \setminus \bigcup_{i=1}^K C_i$ to clusters when $C_1, \ldots, C_K$ do not form a full partition of the state space and to construct subspaces of L^2 that will be of particular interest when we consider milestoning processes and assess the quality of a clustering.

3.3.2 Reversible transfer operators

In the following we want to examine (T_t), a semi-group of operators associated to a time-continuous Markov process (X_t), which we define by its action on certain distribution functions. Let μ be the unique stationary distribution. For any fixed t, T_t is a linear operator acting on the μ-weighted Hilbert space

$$L^2_\mu := \left\{ f : S \to \mathbb{R} \quad | \quad f \text{ is measurable, } ||f||^2_\mu = \langle f, f \rangle_\mu < \infty \right\}$$

of quadratically μ-integrable distribution functions equipped with the scalar product $\langle f, g \rangle_\mu = \int f(x)g(x)\mu(x)dx$. Then we may define $(T_t f)$ as the conditional expectation

$$(T_t f)(x) = \mathbb{E}_x(f(X_t)) = \int f(y)\mathbb{P}[X_t = y | X_0 = x]dy.$$

T_t fulfills another prominent role. The adjoint operator T_t^* describes the propagation of distribution functions by the dynamics of the process within time t. That is, when the process is initially distributed $X_0 \sim \pi_0$ for some distribution function $\pi_0 \in L^2_\mu$, then $X_t \sim T_t^* \pi_0$. In the case of reversible Markov processes, which we are going to stick to for the moment, these two operators coincide.

Theorem 3.3.1. *For any reversible Markov process (X_t) the operators (T_t) are self-adjoint on L^2_μ.*

Proof. This follows directly from the fact that a time-continuous reversible process satisfies the detailed balance condition 3.4.4, which we will explain in the following section.

$$\begin{aligned}\langle T_t f, g\rangle_\mu &= \int (T_t f)(x) g(x) \mu(x) dx = \int\int f(y) p_t(x,y) g(x) \mu(x) dy dx \\ &= \int\int f(y) p_t(y,x) g(x) \mu(y) dy dx = \int f(y) (T_t g)(y) \mu(y) dy = \langle f, T_t g\rangle_\mu\end{aligned}$$

□

Now, we take a closer look at $T := T_\tau$ for a fixed **lag time** τ, examine its properties and what we can learn from them about the structure of the underlying system. In the following we assume the transfer operator T to meet the following key property.

Assumption 1. T has $K+1$ real eigenvalues satisfying

$$\lambda_0 = 1 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K,$$

with associated orthonormal eigenvectors $u_1, \ldots, u_K$ and the remaining spectrum of T lies within a ball of radius $R < \lambda_K$.

Note, that $\lambda_0 = 1 > \ldots$ implies the uniqueness of the left eigenvector $\mu T = \mu$ and the right eigenvector $T\mathbb{1} = \mathbb{1} = u_0$ associated to $\lambda_0 = 1$. It follows that the corresponding process has no invariant or cyclic classes and its dominant dynamics can be approximated well by a coarse-grained milestoning process. In practice, we hope for the **spectral gap** Δ between λ_K and the smallest possible radius of a ball containing the remainder of the spectrum to be as large as possible, as this would indicate the existence of a K-clustering into sets of 'high' metastability in the sense we will specify in the next section. It is a frequent problem in real life applications, that the spectral gap is not clearly visible and we do not know in advance how many metastable sets a given random walk process has. So-called soft decompositions, which we will present in Section 3.4.4, circumvent this problem by restricting the random walk process to the non-transient part of the state space, often resulting in a more clearly arranged spectrum of the transfer operator.

3.4 Markov state models

When we look at an arbitrary trajectory of some reversible Markov process, such as a random walk process on a graph, for a very long time, we usually observe the following behavior: There are areas which, once entered by the process, take very long time to leave while others are quite transient in comparison. The first are called metastable sets of the state space and can be seen as a generalization of invariant subsets, which is also a hint at how we can identify them, exploiting their relation to eigenvalues of the transfer operator.

When starting in a metastable region, on a short time-scale, it is likely that the process would not leave it. Starting in a transient state, it quickly enters a metastable set and probably stays there. On a large time-scale, the process mainly jumps between metastable sets. This is our motivation to approximate the original process by a time-discrete process on a finite state space, a so-called **Markov state model**, losing as few information as possible in the sense of Section 4.3.

Employing spectral methods, we identify metastable sets of the original process. Each of these sets shall be represented by one vertex of the simplified state space, such that the coarse-grained process describes movement between metastable sets only. The discretization of time is realized by taking snapshots of the process in uniform time steps. The lag time chosen determines the predominant time-scale we focus on.

First, we will present an MSM approach from Sarich et al. [65] to find metastable sets of a reversible Markov process based on eigenvectors associated to the largest eigenvalues of the transition matrix. Then, we will discuss the advantages of certain time-continuous random walks over discrete ones and briefly introduce the idea of fuzzy MSM clustering, following [23].

3.4.1 Identification of metastable sets

Let (X_t) be a time-continuous Markov process on state space S and (T_t) the associated semi-group of **transfer operators** which describe the propagation of distribution functions. We assume that the process is sufficiently ergodic, implying that there are no invariant or cyclic subclasses and a unique stationary measure μ exists, such that $T_t\mu = \mu \quad \forall t$.

For some fixed lag time $\tau > 0$ and a full partition $S = \bigcup_{i=1}^{K} C_i$ of the state space, the continuous process (X_t) is coarse-grained space- and time-wise by the discrete process $(\hat{X}_k)$ defined on $\hat{S} = \{1, \dots, K\}$ by $\hat{X}_k = i \iff X_{k\tau} \in C_i$. Since this process is not Markovian in general, we define the **MSM process** $(\tilde{X}_k)$ as an approximation of this snapshot process by choosing its transition matrix $\tilde{P}$ as

$$\tilde{P}_{ij} = \mathbb{P}_\mu[\hat{X}_1 = j | \hat{X}_0 = i] = \mathbb{P}_\mu[X_\tau \in C_j | X_0 \in C_i].$$

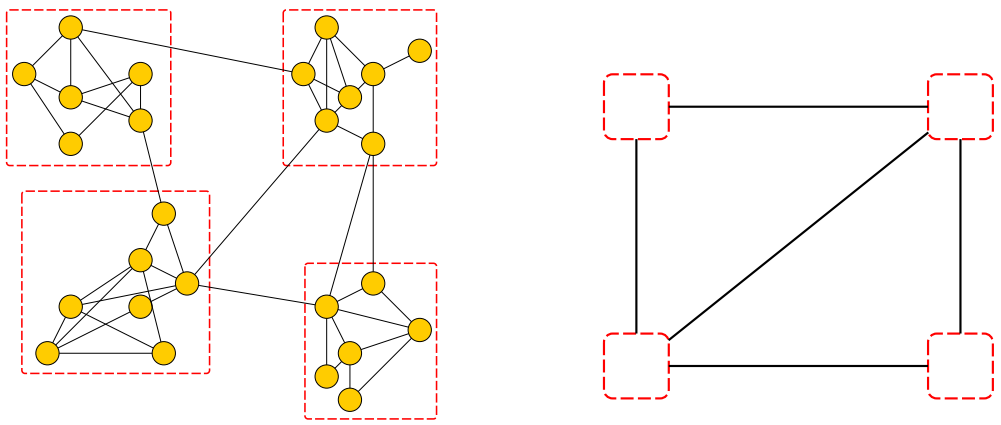
In Section 4.3.4, after introducing an appropriate distance measure for transfer operators, we will discuss that this process approximates the original process well concerning long-term dynamical behavior, if and only if the partition $C_1, \dots, C_K$ corresponds to sets of high metastability.

We have stated already that we may specify a time-scale to focus on by choosing the lag time appropriately, but we did not yet clarify how to identify the full partition we need for the Markov state model from the spectrum of the transfer operator $T := T_\tau$. Let us remind our assumption 1 from the previous section, postulating the existence of $K + 1$ 'dominant' near-one eigenvalues $\lambda_1 \geq \cdots \geq \lambda_K$, and an associated system of orthonormal eigenvectors $u_1, \dots, u_K$. We claim that these vectors, suitably normalized, are almost **piecewise constant (PC)** with respect to a partition of the state space into K metastable sets.

Definition 29. A vector $v = [v_1, \dots, v_k]^T$ is called piecewise constant (PC) with respect to sets $C_1, \dots, C_m$ if

$$v_i = v_j \qquad \forall i, j \in C_r \quad \forall r \in 1, \dots, m.$$

We substantiate our claim by the following observation: Given some process X on some state space S with exactly r minimal invariant subclasses $I_1, \dots, I_r \subset S$, the spectrum of the transfer operator contains the eigenvalue 1 exactly r-fold. Since none of the invariant subclasses can be left once entered by the process, there are r stationary distributions



(a) **(b)**

Figure 3.3: The main principle of Markov state modeling.
The original process, e.g. the random walk process on a graph (a), is approximated by a reduced version of itself, where metastable sets of the process, corresponding to clusters of the graph, are merged into single states each (b). The loss of information with respect to the original process measures the quality of the clustering.

$\mu_1, \ldots, \mu_r$, satisfying $\text{support}(\mu_i) = I_i$. They are eigenvectors to the eigenvalue one and orthogonal due to their disjoint supports, therefore spanning a r-dimensional eigenspace associated to one. Thus the eigenvalue one occurs r times in the spectrum of the transfer operator.

Assuming that we already knew these stationary distributions, we could easily identify the communication classes of the process as their support sets and the transient region D, since

$$D := \{x \in S | \quad \mu_1(x) = \cdots = \mu_r(x) = 0\}.$$

From a computational point of view, we might find that an eigenvalue solver yields eigenvectors $\eta_1, \ldots, \eta_r$ which do not have disjoint supports in general and are possibly not even non-negative. In this case, we must first construct an maximal stationary distribution of the process, identify and separate the transient region of the state space and normalize the computed eigenvectors in order to obtain piecewise constant vectors which reveal the invariant classes of the process. To this end, we decompose each η_k into its positive and negative parts

$$\eta_k^+(x) = \begin{cases} \eta_k(x) & , \eta_k(x) > 0 \\ 0 & , \text{ else} \end{cases}, \quad \eta_k^-(x) = \begin{cases} -\eta_k(x) & , \eta_k(x) < 0 \\ 0 & , \text{ else} \end{cases},$$

which are eigenvectors associated to one as well and satisfy $\eta_k = \eta_k^+ - \eta_k^-$.

This enables us to avoid cancellation of entries when we calculate a maximal stationary distribution π by

$$\pi = \frac{1}{Z} \sum_{i=1}^{r} (\eta^+ + \eta^-),$$

where $Z := \sum_x \sum_{i=1}^r (\eta^+(x) + \eta^-(x))$ is just a normalization factor. Using π, we may now

compute the transient region by

$$D = \{x \in S | \pi(x) = 0\}.$$

In the following, we restrict ourselves to the non-transient part of the state space, assuming $D = \emptyset$, which grants us $\pi > 0$ everywhere.

Since $\mu_1, \ldots, \mu_r$ form a basis of the eigenspace to one, the stationary distribution μ of the process may be written as a linear combination $\pi = \sum_{i=1}^{r} a_i \mu_i$. Similarly, any of the eigenvectors $\eta_1, \ldots, \eta_r$ can be represented as $\eta_j = \sum_{i=1}^{r} a_i^j \mu_i$ uniquely determined by the weights $a_1^j, \ldots, a_r^j$. In order to construct piecewise constant vectors, we may simply normalize the eigenvectors, dividing them entry-wise by π, since for any invariant set I_i we have

$$u_j(x) := \frac{\eta_j(x)}{\pi(x)} = \frac{a_i^j \mu_i}{a_i \mu_i} = \frac{a_i^j}{a_i} =: c_i^j.$$

Given these vectors $u_1, \ldots, u_r$, piecewise constant on each invariant subset, we observe that the function $f : x \mapsto (u_1(x), \ldots, u_r(x))$ assumes exactly r different values $\gamma_1, \ldots, \gamma_r$ and that we may now identify invariant subsets simply by calculating

$$I_i = \{x \in S | f(x) = \gamma_i\}.$$

Metastable sets are to be regarded as a generalization of invariant subclasses, being not impossible but very difficult to leave by the process. The associated transfer operator, which in the case of reversible processes is self-adjoint and therefore also describes the propagation of density functions according to Section 3.3.2, almost does not permit the flow of density out of a metastable set. Very similar to how an invariant subclass of a reducible process corresponds to an eigenvalue equal to one, a metastable set of a reversible process corresponds to a near-one eigenvalue of the transfer operator.

The normalized eigenvectors associated to those near-one eigenvalues are not piece-wise constant in general, but we can expect them to be nearly constant on metastable sets, permitting an identification of these sets similar to the identification process of invariant sets described above. The more pronounced the metastable behavior of the process is, the closer are the normalized eigenvectors to being piecewise constant and the more precisely can the metastable sets be detected [21].

3.4.2 Standard random walk

We have seen already how to construct an MSM process from a given reversible Markov process and now we will apply the procedure to find a clustering on an arbitrary simple and connected graph $G = (V, E)$ on a finite vertex set $|V| = n$.

The main idea is the perception of a 'random walker' traveling the vertices of the graph, choosing randomly at each step which edge to follow. For any undirected graph, this turns out to be a reversible Markov chain as the adjacency matrix of the graph is symmetric. Metastable sets of this process, corresponding to subgraphs where the random walker spends much time in statistically, are then identified as clusters of G. As we assume graphs to be connected and are interested in properties of infinite trajectories, it does

not matter in the long run which vertex the process starts from. The opening of such a trajectory is illustrated in Figure 3.4.

Therefore, let us define the n by n **transition matrix** $P = (p(x,y))_{x,y\in V}$ of the **standard random walk process** (X_k) by

$$p(x,y) = \begin{cases} 1/d(x), & \forall (x,y) \in E \\ 0, & \text{otherwise,} \end{cases} \tag{3.4.1}$$

where $d(x)$ is the degree of vertex x. By construction, the random walk process given by this stochastic matrix is a time-discrete homogeneous Markov chain. We even know the unique **stationary distribution** μ, which is given by

$$\mu(x) = \frac{d(x)}{\sum_{y\in V} d(y)}.$$

In case a positive real-valued edge weighting $w : E \mapsto \mathbb{R}_+$ is given, the transition probabilities can be adapted to respect the weighting by favoring high-valued edges over others. Here, the weights can be taken as measure of attractiveness of an edge. To this end, we choose

$$p(x,y) = \begin{cases} w(x,y)/\sum_{z\in\delta(x)} w(x,z), & \forall (x,y) \in E \\ 0, & \text{otherwise,} \end{cases} \tag{3.4.2}$$

where $\delta(x)$ denotes the neighborhood of the vertex x, the set of vertices adjacent to x. When we set w uniformly to one, we gain the same transition matrix as in the unweighted case, since 3.4.1 and 3.4.2 coincide. Hence the name 'standard' random walk.

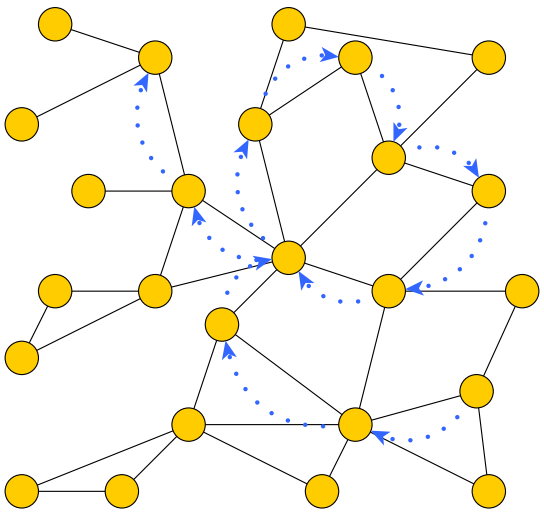


Figure 3.4: Section of the trajectory of a random walk process. Starting from an arbitrary vertex the random walker decides in each step, based on a given edge weighting or chosen uniformly at random, which adjacent vertex to visit next.

Because of the symmetry of the adjacency matrix, the random walk process is **reversible** in time

$$\mathbb{P}[X_k = x, X_{k+1} = y] = \mathbb{P}[X_k = y, X_{k+1} = x],$$

which implies that the discrete **detailed balance condition** holds

$$\mu(x)p(x,y) = \mu(y)p(y,x). \tag{3.4.3}$$

In the more general case of a time-continuous reversible process, this property reads as

$$\mu(x)p_t(x,y) = \mu(y)p_t(y,x), \tag{3.4.4}$$

where $(p_t(x,y))_{x,y\in S}$ contains the probabilities to travel from vertex x to vertex y within time t

$$p_t(x,y) = \mathbb{P}[X_t = y | X_0 = x].$$

We may then apply the construction of the MSM process as described in the section above where we can skip the choice of a lag time since the process is already discrete in time.

3.4.3 Markov jump processes

Time-continuous Markov jump processes were introduced to overcome the problem of e.g. long bridges in a graph being identified as highly metastable sets of standard random walks and therefore as clusters of the graph, despite their simple structure. To reduce the time a random walker would spend in such parts of a graph, the idea was developed to adapt the speed of the random walk process according to the number of adjacent vertices. The less neighbors a vertex has, the quicker the process would leave it, whereas it consumes more time to make a choice when many adjacent vertices are available. A time-continuous variant of the random walk approach was necessary to allow for such fluid changes of speed.

We define a family of random walk processes by its **infinitesimal generator**

$$L_p(x,y) = \begin{cases} -\frac{1}{d(x)^p}, & x = y \\ \frac{k(x,y)}{k(x)d(x)^p}, & x \neq y, (x,y) \in E \\ 0, & otherwise, \end{cases} \tag{3.4.5}$$

where p is a real-valued parameter, $k(x,y)$ are non-negative edge weights and $k(x)$ is short for $\sum_y k(x,y)$, the sum of weights of all edges incident to x. It is called a generator, because the whole family P_t of transition matrices containing the possibilities to switch from one state to another within a certain time $t \geq 0$ can be generated by

$$P_t = \exp(tL_p).$$

Since the **waiting time** of a vertex x, the average time it takes a random walk process defined by L_p to leave x when starting from there, is proportional to $d(x)^p$, the choice of p allows us to control the significance of vertex degrees on the speed of the process. The higher p is set, the quicker does a random walker leave loosely connected sets such as bridges and the more emphasis is put on highly connected metastable regions.

The invariant measure of a Markov jump process (X_t) defined in such a way is

$$\mu(x) = \frac{1}{Z} d(x)^p k(x),$$

where $Z = \sum_{x \in S} d(x)^p k(x)$ is just a normalization constant to guarantee that the non-negative entries of μ sum up to one, such that μ is a unique probability measure. We can now easily prove that the detailed balance condition 3.4.4 holds for the jump process defined by the generator L_p since

$$\begin{aligned}
\mu(x)L_p(x,y) &= \mu(y)L_p(y,x), \quad x = y \\
\mu(x)L_p(x,y) = 0 &= \mu(y)L_p(y,x), \quad x \neq y \wedge (x,y) \notin E \\
\mu(x)L_p(x,y) = \frac{1}{Z} d(x)^p k(x) \frac{k(x,y)}{k(x)d(x)^p} &= \frac{k(x,y)}{Z} \\
= \frac{k(y,x)}{Z} = \frac{1}{Z} d(y)^p k(y) \frac{k(y,x)}{k(y)d(y)^p} &= \mu(y)L_p(y,x), \quad x \neq y \wedge (x,y) \in E.
\end{aligned}$$

We can conclude that the Markov jump process (X_t) is reversible and that its generator L_p is a self-adjoint operator on the space L^2_μ and therefore has real-valued eigenvalues featuring orthonormal eigenvectors. Moreover, for any connected, non-bipartite graph, the largest eigenvalue is zero and has multiplicity one, such that the spectrum looks like

$$0 = \Lambda_0 > \Lambda_1 \geq \cdots \geq \Lambda_m. \tag{3.4.6}$$

There is an immediate connection to the properties of an embedded snapshot processes. For any fixed time $\tau > 0$ the discrete process $(X_{n\tau}), n \in \mathbb{N}$ is a reversible Markov chain and has the transition matrix $P = \exp(\tau L_p)$ with eigenvalues $\lambda_i = \exp(\tau \Lambda_i)$ which can be ordered in the same fashion

$$1 = \lambda_0 > \lambda_1 \geq \cdots \geq \lambda_m.$$

We want to highlight two possible choices for p and k which let us gain valuable insight into the connection between these Markov jump processes to standard random walk processes and their advantages over the discrete version. When we choose p as one and $k(x,y) = a(x,y)$, where $a(x,y)$ are the entries of the adjacency matrix A such that $k(x) = d(x)$, the generator simplifies to

$$L(x,y) = \begin{cases} -\frac{1}{d(x)}, & x = y \\ \frac{1}{d(x)^2}, & x \neq y, (x,y) \in E \\ 0, & otherwise. \end{cases}$$

Then, the waiting time for each vertex is equal to its degree and the invariant measure is given as $\mu(x) = \frac{1}{Z} d(x)$. Since all edges are weighted uniformly, the adjacent vertex the process goes next is chosen uniformly at random. Therefore, the embedded Markov chain for $\tau = 1$ is the standard random walk process from Section 3.4.2.

We noted already, that we can run into problems with our standard random walk approach such that the spectral gap is sometimes hard to spot and near-one eigenvalues lead to the identification of loosely connected subgraphs as clusters. With the time-continuous random walk approach we can overcome these downsides. Let us anticipate

an idea we will analyze in detail when we look at symmetrization methods in the next chapter, **bibliometric similarity**. This is, we consider two vertices x and y as closely related the more links to other vertices they have in common, expressed by the scalar product $\langle a_x, a_y \rangle$ of the x-th and y-th rows of the adjacency matrix. To realize this, let $p = 1$ and $k(x, y) = a(x, y) \cdot (1 + \langle a_x, a_y \rangle)$.

The probability to jump from vertex x to vertex y is no longer identical for any neighbor of x, but depends on the similarity of the neighborhoods of x and y. This increases the likelihood that two vertices which are elements of the same highly connected subset of S are grouped together by clustering algorithms employing this time-continuous approach. One such method, presented in [23], is highlighted in the next section.

3.4.4 Soft clusterings

In general, the clusters do not form a full partition of the whole state space. There might be vertices which clearly belong to a cluster, while others are part of intersections, bridges or operate as hub vertices between several clusters. The normalized eigenvectors often feature nearly constant sections, which can be clearly related to a metastable set and rather fluctuating parts, where an assignment is difficult.

To address these obstacles, a technique was developed which does not construct a full decomposition of the state space directly, but identifies clear 'core' clusters first, which shall be called **modules** and includes soft assignment functions on the remaining vertices, allowing for a more detailed description of a graph's cluster structure. The idea of a 'soft clustering' or 'fuzzy decomposition' was already mentioned in the early eighties [3], the concept presented here was recently introduced in [23].

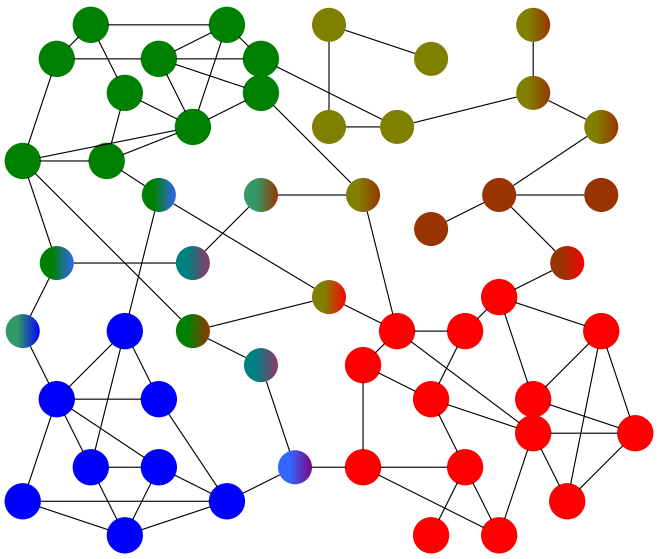


Figure 3.5: Soft clustering of an undirected graph. Single-colored red, green or blue vertices were identified as full members of modules, mixed-colored vertices show affiliation to more than one module.

Again, we want modules to be subsets of the state space which are quickly entered but difficult to leave by a random walker, but this time they do not need to form a full partition of the state space. That is, we first need to separate the modules, which

correspond to metastable sets of the transfer operator associated to the random walk process. Second, we reduce the process to the part of the state space containing the modules using a special milestoning process. Third, a full decomposition of the reduced process by standard Markov state modeling grants us modules of the original process. Then we apply committor functions as introduced in Section 3.3.1 as soft assignment functions interpreted as probability for a given vertex to belong to a certain module.

Identifying the module ensemble

Assume, we could control the metastable behavior of a process by some free parameter to design a slightly more metastable version of the original process. Then we calculate the stationary distribution of this altered process and propagate it a small time step using the transfer operator of the original process. We could then identify the 'attractive' part of the state space by comparing the calculated density function to the stationary distribution.

Using the recently developed approach outlined in Section 3.4.3, we can implement this heuristic by taking the invariant measure μ^* of a time-continuous random walk process generated by the generator L_0 defined according to 3.4.5, such that waiting times are uniform and the process no longer adapts its speed depending on vertex degrees. This function now serves as the initial distribution of a process generated by L_1, whose transition matrix for a fixed lag time $\tau > 0$ is given as $P_\tau = \exp(\tau L_1)$. We hereby increase the attractiveness of metastable sets of the random walk process and watch the impact on the stationary distribution. The vertices which accumulate more density upon the incitement of metastable behavior form the module ensemble

$$M^\tau = \{x \in S | (P_\tau^T \mu^*)(x) > \mu^*(x)\}.$$

This way, we may now exploit the relation between the lag time and the timescales at which the process enters the modules. By shifting τ, we can control how metastable a set of vertices has to be in order to be part of M^τ. When the spectrum of the generator, ordered as in 3.4.6, shows a clear spectral gap after eigenvalue Λ_m, we can ensure that the metastable sets corresponding to the dominant eigenvalues are taken into account by setting the gap time according to $1/|\Lambda_m| > \tau > 1/|\Lambda_{m+1}|$.

Decomposing the module ensemble

Our next step is to reduce the process to the set M^τ, using a special kind of committor functions with respect to a total clustering of M^τ, where every vertex y of M^τ is regarded as a cluster on its own. Then we create a full decomposition using standard Markov state modeling and identify the obtained clusters as core clusters of the original process. As we have chosen τ such that M^τ contains the metastable sets corresponding to certain dominant eigenvalues, we know the number of desired clusters in advance, facilitating the decomposition.

In accordance with Section 3.3.1, we consider committor functions $q_y(z)$ containing the probabilities for each vertex y in M^τ and each vertex z of the whole state space V, that the random walk process, when starting in z, hits y on its first entrance into M^τ. These can

be calculated by solving sparse linear systems, which can be done efficiently in the sense, that the computational effort is usually dominated by the subsequent full decomposition of the reduced process [23].

With these special committors at hands, the reduced process can be defined by setting the entries of its transition matrix for any $x, y \in M^\tau$ to

$$\hat{P}_\tau(x, y) = \sum_{x \in V} P(x, z) q_y(z).$$

In simple words, the reduced process skips any movement outside the module ensemble and immediately jumps back to a vertex in M^τ according to the probabilities to go there via an ordinary path in the original graph.

As long as the random walker does not leave the set M^τ it behaves exactly the same as under the dynamics of the original process, since the committor functions $q_y(z)$ are equal to δ_{yz} whenever z is inside the module ensemble and thus $\hat{P}_\tau(x, y) = P(x, y) \quad \forall x, y \in M^\tau$. Due to this, it seems reasonable to assume, that a full decomposition $C_1, \ldots, C_m$ of the subgraph induced by M^τ under the dynamics of the reduced process, corresponds to core clusters of the original graph.

Now, as our goal was to construct a soft clustering, we would like to answer the question, how high the affiliation of the vertices outside the module ensemble is to each of the identified modules. More precisely, we want to state the probability of each vertex to belong to one of the clusters, such that these numbers add up to one. Here, we employ committor functions again, yet this time with respect to the clustering $C_1, \ldots, C_m$. These affiliation functions $q_1(x), \ldots, q_m(x)$ necessarily satisfy the constraints $q_i(x) = 1 \quad \forall x \in C_i$ and $q_i(x) = 0 \quad \forall x \in M^\tau \setminus C_i$ by construction.

Unlike a full decomposition, the fuzzy decomposition yields core clusters only, corresponding to the most metastable sets of the time-continuous random walk process. This can be an advantage, when a graph has a large transient region where vertices are easily mis-assigned The affiliation functions contain much more information about the remaining vertices than a binary assignment to one of the clusters. If the desired application requires a minimization of unassigned vertices, the clusters can be enlarged with little effort by selecting a threshold value $0 < \epsilon < 0,5$ and accepting additional vertices into a cluster when their affiliation towards it is high enough. Depending on the tolerance ϵ, we receive a new clustering $\mathcal{C}^\epsilon = \{\tilde{C}_1, \ldots, \tilde{C}_m\}$ by selecting the new sets according to

$$\tilde{C}_i := \{x \in V | q_i(x) > 1 - \epsilon\}.$$

This method can even be generalized to obtain a full decomposition again by choosing

$$\bar{C}_i := \{x \in V | q_i(x) \geq q_j(x) \quad \forall j = 1, \ldots, m\},$$

yet uniqueness can only be assured when every vertex is most affiliated to exactly one cluster, such that ties are excluded.

Unfortunately, this procedure can not be generalized to directed graphs in a straightforward manner, for the transfer operators associated to random walks on directed graphs are not reversible in general and even the existence of real eigenvalues can not be guaran-

teed. Therefore, we will take a closer look at clustering methods specifically designed for directed graphs and ways of symmetrizing directed graphs in the next chapter.

4 | Clustering directed graphs

4.1 Non-reversible transfer operators

One open question about analyzing directed graphs is how to define modules and hubs in the sense of directed networks. There are several known techniques based on the idea of tracing back to the identification of modules in undirected graphs by simply ignoring the directions of the edges or by constructing a self-adjoint transfer operator from the two non-self-adjoint transfer operators associated to the forward and backward running random walker processes. Such approaches shall be further described in the upcoming sections.

However, whether or not these sets are indeed representing functional units or important sections of the underlying system depends on the desired application.

What keeps us from using the known MSM approaches on directed graphs is the non-reversibility of a random walk process, which implies non-self-adjointness of the associated transfer operator and thus non-real eigenvalues in the spectrum of the transition matrix.

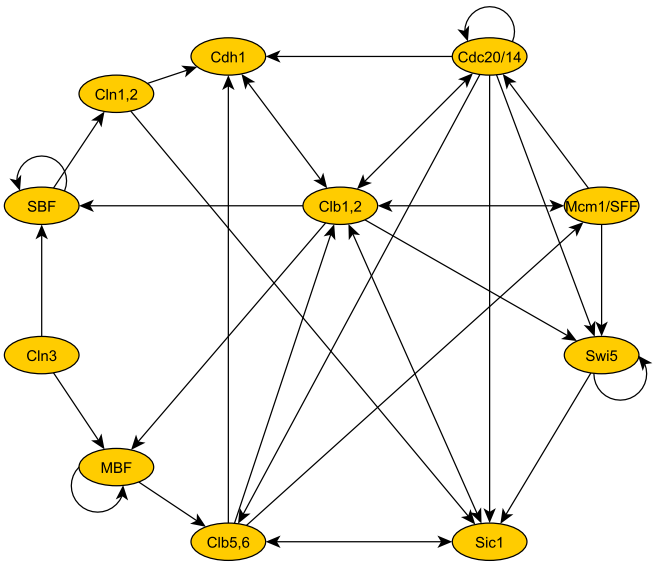


Figure 4.1: An example of a directed graph with self-loops, depicting the cell cycle network in budding yeast [43].

4.2 Non-spectral approaches

4.2.1 Cuts for directed graphs

Many clustering criteria can be formulated as some kind of weighted cut minimization, some of which seek to minimize the combined normalized cut of all clusters. We define the normalized cut of a subset $S \subset V$ by

$$\mathrm{NCut}(S) = \frac{\sum_{i\in S, j\notin S} A(i,j)}{\sum_{i\in S} deg(i)} + \frac{\sum_{i\in S, j\notin S} A(i,j)}{\sum_{j\notin S} deg(j)}, \tag{4.2.1}$$

where A is the symmetric adjacency matrix of an undirected graph. If $(X_n)_{n\in\mathbb{N}}$ is a standard random walk process on the graph in the spirit of Section 3.4.2, this coincides with the probability to leave the set $S \subset V$ or enter it from the complementary $S^C = V \setminus S$ within one time-step when starting in the stationary distribution [68]

$$\mathrm{NCut}(S) = \frac{\mathbb{P}(X_{n+1} \notin S | X_n \in S)}{\mathbb{P}(X_n \in S)} + \frac{\mathbb{P}(X_{n+1} \in S | X_n \notin S)}{\mathbb{P}(X_n \notin S)}.$$

This can be extended to directed graphs using the concept of random walks via the transition matrix P and its associated stationary distribution π. When the process is distributed according to π, the drain of density from vertex i to vertex j via the directed edge $\overrightarrow{ij}$ is given by $\pi(i)P(i,j)$, so we may write

$$\mathrm{NCut}_{dir}(S) = \frac{\sum_{i\in S, j\notin S} \pi(i)P(i,j)}{\sum_{i\in S} \pi(i)} + \frac{\sum_{i\in S, j\notin S} \pi(j)P(j,i)}{\sum_{j\notin S} \pi(j)}. \tag{4.2.2}$$

We wish to further generalize that concept by introducing an arbitrarily weighted directed cut measure such that we can later identify certain clustering objectives as special cases of cut minimization problems, which can be solved by standardized algorithms both with or without spectral methods.

This weighted class of cut measures is due to Meila et al. [50]

$$\mathrm{WCut}(S) = \frac{\sum_{i\in S, j\notin S} T'(i)\tilde{A}(i,j)}{\sum_{i\in S} T(i)} + \frac{\sum_{i\in S, j\notin S} T'(j)\tilde{A}(j,i)}{\sum_{j\notin S} T(j)}.$$

The introduction of three parameters, namely the vectors T, T' and the matrix $\tilde{A}$, offers a lot of flexibility. Both NCut and NCut_{dir} can be recovered from WCut by choosing these parameters appropriately. To restore formula 4.2.1, choose $T' = \mathbb{1}$, $(T(x)) = (deg(x))$ and $\tilde{A} = A$ or set $T' = T = \pi$ and $\tilde{A} = P$ to restore formula 4.2.2.

4.2.2 Symmetrizations

In the following, we discuss an approach from Satuluri et al. [68] to cluster a directed graph, based on symmetrizing its adjacency matrix and then using an arbitrary clustering procedure designed for undirected graphs, such as those presented in Chapter 3. To this end, let A be the adjacency matrix of a given directed graph, unsymmetrical in general,

whose non-zero entries are perceived as a weighting on the corresponding edges. The goal is to construct a symmetric matrix U of the same size, the adjacency matrix of an undirected graph, inheriting the cluster structure of the original graph.

One of the simplest attempts, $U = A + A^T$, adds up the weights of the arcs $\overrightarrow{uv}$ and $\overrightarrow{vu}$, merging them into one undirected edge $\overline{uv}$. If we ignore weightings altogether and simply focus on existence or non-existence of an edge, this approach simplifies to abandoning the orientation of the arcs. While this idea might seem quite plain, we will find later that some other, more sophisticated symmetrization techniques lead to the same non-zero-structure, but with different weights.

One symmetrization technique of particular interest to us, preserves the NCut of any vertex set S, in the sense that $\mathrm{NCut}_{dir}(S)$ in the original directed graph is equal to $\mathrm{NCut}(S)$ in the undirected graph given by adjacency matrix

$$U = \frac{\Pi P + P^T \Pi}{2}.$$

Here, P is the transition matrix of the forward random walk according to 4.4.11 and Π is the diagonal matrix, where the main diagonal entries, regarded as a vector, are the stationary distribution of P. This way we can trace back the problem of finding a NCut_{dir}-minimizing clustering of a directed graph to the already familiar problem of finding a NCut-minimizing clustering of an undirected graph. Note however, that since P is obtained by normalizing the rows of A and Π is a diagonal matrix, the undirected matrix U computed this way has the same non-zero structure as $A + A^T$ and will therefore be subject to the same drawbacks, depending on what kind of result we expect from our clustering method [35].

The authors of a recent paper claim, that the usual concept of clusters as densely connected vertex sets in a graph is not appropriate to cover categories of 'similar' vertices in many real-life applications of directed graphs [68]. As highlighted in Section 3.1 common linking behavior of two vertices might tell more about their similarity in certain contexts than an edge or a short directed path connecting them. This thought led to the bibliometric symmetrization $U = AA^T + A^T A$, a compromise respecting both the bibliometric coupling $B = AA^T$ introduced by Kessler [38] and the co-citation $C = A^T A$ due to Small [74]. While $B(u, v)$ gives the number of vertices both u and v point to, $C(u, v)$ counts the number of vertices which point to both u and v.

The practical implementation revealed a flaw in the bibliometric symmetrization caused by the fact that vertex degrees are not accounted for. Hub vertices with exceedingly high degree tend to share many links with other vertices simply because of the sheer number of incident edges. Therefore, many vertices can show similarity to these hubs leading to their relative overvaluation and a dense similarity structure. In order to counter computational overhead, low non-zero entries of U are usually truncated in practice, yet we face the dilemma that, due to the high value assigned to hub vertices, a low threshold would not affect sufficiently many entries while a high threshold might delete similarities which are considered important, whereas rows representing a hub keep a lot of entries. Satuluri et al. [68] opposed this inherent flaw by designing a degree-discounted version of the bibliometric symmetrization

$$U = D_{out}^{-1/2} A D_{in}^{-1/2} A^T D_{out}^{-1/2} + D_{in}^{-1/2} A^T D_{out}^{-1/2} A D_{in}^{-1/2},$$

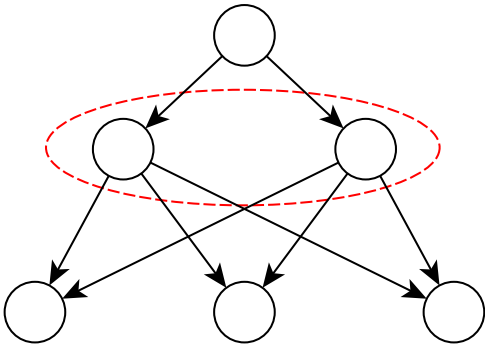


Figure 4.2: Clustering based on common linkage behavior.
The two vertices in the second line are pointing at the same vertices and both have an incoming arc from the vertex above. The bibliometric symmetrization identifies them as highly similar, even though they are not directly connected.

where D_{in} and D_{out} are diagonal matrices containing the in- and out-degrees of all vertices on their main diagonals. The more incident edges a vertex has, the less value is assigned to a common link with another vertex. This degrades big hubs, such that a wisely chosen threshold yields a sparse matrix, holding computational effort at bay, while vertices sharing a considerable ratio of their links are identified as highly similar.

Beside offering the possibility to choose a similarity measure more appropriate to citation networks and other such data sets, the symmetrization phase is independent from the algorithm used to identify clusters in the symmetrized matrix and thus allows the user to combine it with his clustering algorithm of choice. Some applicable methods to cluster undirected graphs were presented in the preceding chapter.

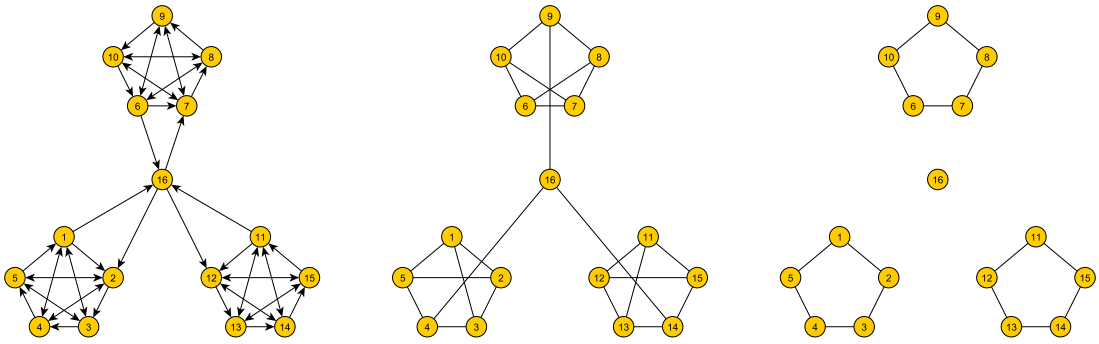
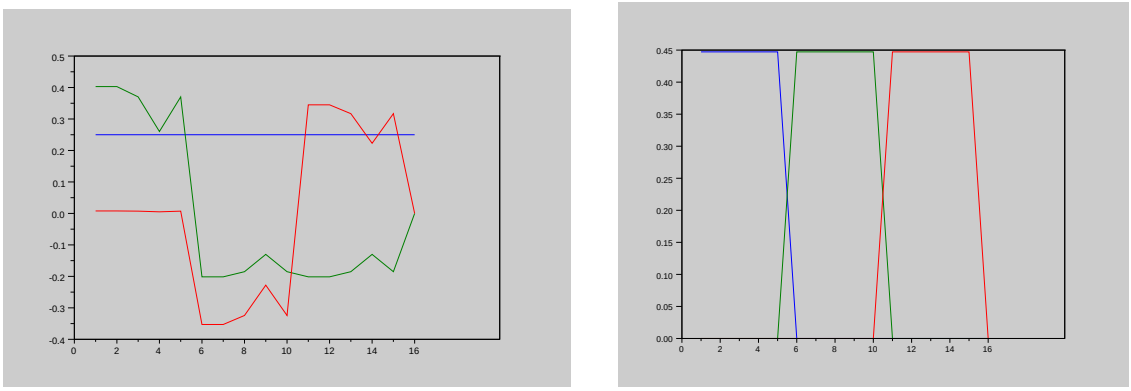


Figure 4.3: An example motivating degree-discounted bibliometric symmetrization.
a: A directed graph featuring a dedicated hub vertex – vertex 16
b: Undirected graph obtained by bibliometric symmetrization after truncation
c: Properly truncated graph resulting from degree-discounted symmetrization
Due to the lower value assigned to hub connections, truncation eliminates insignificant edges to the hub vertex in the graph obtained from degree-discounted symmetrization, whereas a higher threshold would cause the extinction of all edges in the undiscounted case.
Note: Edge weightings were omitted in favor of clearness. Weights were chosen constantly equal to 1 in the original directed graph and are almost uniform in the symmetrized graphs after truncation.



(a) **(b)**

Figure 4.4: Two drawings, each depicting three eigenvectors corresponding to the dominant eigenvalues of the transfer operators of random walks on graphs 4.3b and 4.3c respectively. Though barely visible in the process induced by standard bibliometric symmetrization (a), the piece-wise constant eigenvectors obtained after degree-discounted symmetrization illustrate the cluster structure of the original graph notably clearer (b).

4.3 Evaluating clusterings

As we discussed before, it is highly debatable, context-sensitive and user dependent, what kind of result is desired of a clustering algorithm. So in order to evaluate the quality of clustering methods in an impartial manner, an objective clustering based on some natural categorization or an arbitrary handmade assignment is needed.

We want to present three methods of evaluating clusterings of finite networks based on their deviation from the objective clustering, namely the clustering error [79], the variation in information [49] and the F-measure [68].

4.3.1 Clustering error

The clustering error counts the number of 'misclassified' vertices. The matrix $\text{Confusion}(k, k') = |C_k \cap C'_{k'}|$ contains the number of points that are assigned to cluster k in the objective clustering $\mathcal{C} = C_1, \ldots, C_m$ and to cluster k' of the clustering $\mathcal{C}' = C'_1, \ldots, C'_m$ to be evaluated. Necessarily, the number of clusters must be the same for both clusterings. We may then define

$$\text{CE}(\mathcal{C}, \mathcal{C}') = \frac{1}{|V|} \left(\sum_k \sum_{k' \neq k} \text{Confusion}(k, k') \right).$$

Note that two clusterings which are identical up to a permutation of their indices would receive a very bad rating just because a cluster would not be recognized if its index is not the same for both clusterings. Therefore, the minimum over all possible relabellings is usually taken as the actual **clustering error**. As the number of clusters is usually much smaller than the total number of vertices, the necessary count of iterations does not have a significant impact on the computational effort.

4.3.2 Variation of information

To define this measure derived from information theory, we must explain two key notions that were first introduced by Claude E. Shannon in 1948 [73]. $\mathcal{C} = C_1, \ldots, C_m$ and $\mathcal{C}' = C'_1, \ldots, C'_{m'}$ are clusterings, n_k gives the size of C_k and we introduce $P(k) := n_k/|V|$, the formal probability to select a vertex in cluster C_k when choosing a vertex uniformly at random. We define $n'_k := |C'_k|$, $P'(k) := n'_k/|V|$ in analogy and finally the formal probability that a uniformly chosen vertex is a member of cluster C_k and cluster C'_k at the same time

$$P(k, k') := \frac{|C_k \cap C'_{k'}|}{|V|}.$$

These abstract probabilities allow us to express two important things. First, we can assess the amount of information contained in one clustering in the sense of how many possibilities there are to distribute vertices among the clusters. A trivial clustering consisting of only one cluster equal to the whole vertex set contains no information at all, for we know in advance that all vertices are members of the same cluster and therefore no information is encoded in the cluster-membership of vertices. The more clusters a clustering consists of and the more balanced their sizes are, the more possibilities there are to distribute vertices among the clusters and thus, the amount of information encoded in such a clustering is higher.

Second and more importantly, we are now able to define how much information two given clusterings share, i.e. the amount of information contained in both clusterings. The similarity of two clusterings can be expressed by the ratio of information they share and conversely the distance between two clusterings can be defined via the amount of unknown information they still contain, when we know nothing but the information they share. This is what the following definitions express formally.

Definition 30. The **entropy** of a clustering $\mathcal{C} = C_1, \ldots, C_m$ is defined as

$$H(\mathcal{C}) = -\sum_{k=1}^{m} P(k) \log P(k)$$

and measures the information contained in the partition given by $\mathcal{C}$. The **mutual information**

$$I(\mathcal{C}, \mathcal{C}') = \sum_{k=1}^{m} \sum_{k'=1}^{m'} P(k, k') \log \frac{P(k, k')}{P(k)P'(k')}$$

assesses the amount of shared information contained in both clusterings $\mathcal{C}$ and $\mathcal{C}'$. Given these, we may regard the **variation of information** as the amount of information contained in either, but not mutually contained in both clusterings

$$\mathrm{VI}(\mathcal{C}, \mathcal{C}') = H(\mathcal{C}) + H(\mathcal{C}') - 2I(\mathcal{C}, \mathcal{C}').$$

When it comes to evaluating the output of a clustering algorithm or compare the results of two similar methods, usually ground truth data is available, providing a native clustering of a given network which should ideally be recreated by a clustering algorithm. It would stand to reason to measure the quality of a clustering by its clustering error or its variation of information with respect to the native clustering. As an example, Meila and Pentney [50] have employed both methods to analyze the results of their **BestWCut**

algorithm to reason its superiority in comparison to other approaches.

4.3.3 F-measure

Satuluri et al. present another evaluation method called F-measure based on the notion of precision and recall [68]. Let $\mathcal{C} = C_1, \ldots, C_m$ be some clustering to be evaluated with respect to the ground truth clustering $\mathcal{C}' = C'_1, \ldots, C'_m$. First, we define the **precision** and **recall** of a cluster C_i against a ground truth cluster C'_j by

$$\mathrm{Prec}(C_i, C'_j) := \frac{|C_i \cap C'_j|}{|C_i|} \text{ and } \mathrm{Rec}(C_i, C'_j) := \frac{|C_i \cap C'_j|}{|C'_j|}.$$

Second, we define $F(C_i, C'_j)$ as the harmonic mean of precision and recall. That is

$$F(C_i, C'_j) := \frac{2 * \mathrm{Prec}(C_i, C'_j) * \mathrm{Rec}(C_i, C'_j)}{\mathrm{Prec}(C_i, C'_j) + \mathrm{Rec}(C_i, C'_j)}.$$

Third, we set the F-measure of a cluster C_i as $F(C_i) := \max_j F(C_i, C'_j)$, since we do not know in advance which cluster in the objective clustering is best represented by cluster C_i and we want the measure to be independent of index permutations. Finally, the F-measure of the clustering $\mathcal{C}$ is chosen as a size-weighted average of the F-measures of its clusters

$$F(\mathcal{C}) := \frac{\sum_i |C_i| * F(C_i)}{\sum_i |C_i|}.$$

This measure features several desirable treats due to its subtle definition. A perfect rating of 1 can only be assumed by the objective clustering itself. Additionally, a naive clustering where each vertex is identified as a cluster would yield highest possible precision, but very low recall values, unless the objective clustering is highly fragmented, and a naive clustering consisting of just one cluster containing the whole state space would yield perfect recall but, in general, low precision values. Both would end up with a modest F-rating, as the harmonic mean of a 'very high' and a 'very low' number is close to the 'very low' number. Even if the objective clustering has some lone vertices that form clusters on their own, which are perfectly matched by a naive clustering, these would not have a large impact on the overall rating due to the size-based weighting of clusters. Thus, in order to receive a good F-rating, a clustering must resemble the level of fragmentation of the objective clustering.

4.3.4 Milestoning process

Another way to assess the quality of a clustering uses a random walk based approach presented in a recent publication [23]. The basic idea is to check how well the original process can be approximated by a coarse-grained version of itself, where every cluster is represented by a single vertex and the transient part of the state space is omitted.

Given a reversible Markov process (X_t), the unique invariant measure μ and a clustering $\mathcal{C} = C_1, \ldots, C_m$ of the state space, we use the stopping time $\sigma(t) = \sup_{s \leq t}\{X_s \in \bigcup_{k=1}^{m} C_k\}$

to define the **milestoning process** $(\hat{X}_t)$ by $\hat{X}_t = i \iff X_{\sigma(t)} \in C_i$.

Let $T : L^2_\mu \to L^2_\mu$ be the transfer operator associated to (X_t) with

$$(Tf)(y)\mu(y) = \sum_x p(x,y)f(x)\mu(x)$$

and $q_i^-, i = 1, \ldots, m$ be the backward committors related to T as defined in Section 3.3.1. Then, the invariant measure $\hat{\mu}$ of the milestoning process may be given in terms of committors of the original process by

$$\hat{\mu}(i) = \mathbb{P}_\mu(\hat{X}_n = i) = \sum_{x \in S} \mathbb{P}_\mu(\hat{X}_n = i, X_n = x) = \sum_{x \in S} q_i^-(x)\mu(x).$$

In order to evaluate the quality of our clustering, we compare how probability densities are propagated by the random walk and the milestoning process. Let P be the L^2-representation of T given by $P = J^{-1}TJ$, where $J : L^2 \to L^2_\mu$ is the linear transformation

$$(J\tilde{f})(x) = \frac{\tilde{f}(x)}{\sqrt{\mu(x)}} = f(x).$$

We may then define L^2-representations $\tilde{q}_i^+ = J^{-1}q_i^+$ and $\tilde{q}_i^- = J^{-1}q_i^-$ of the forward and backward committors related to T and orthogonal projections Q_1 and Q_2 onto the subspaces $D_1 = span\{\tilde{q}_1^+, \ldots, \tilde{q}_m^+\}$ and $D_2 = span\{\tilde{q}_1^-, \ldots, \tilde{q}_m^-\}$ by

$$Q_1 v = \sum_{i,j=1}^m S_{ij}^{-1}(v, \tilde{q}_j^+)\tilde{q}_i^+ \qquad \text{and} \qquad Q_2 v = \sum_{i,j=1}^m R_{ij}^{-1}(v, \tilde{q}_j^-)\tilde{q}_i^-,$$

with matrices $S_{ij} = (\tilde{q}_i^+, \tilde{q}_j^+)$ and $R_{ij} = (\tilde{q}_i^-, \tilde{q}_j^-)$.

Now we can give a formula for the maximal possible error after k time-steps between the original and the coarse-grained process under the assumption that both were initially equally distributed by

$$E(k) = ||Q_1 P^k Q_2 - (Q_1 P Q_2)^k||.$$

In the case of a sufficiently ergodic time-continuous process approximated by a Markov state model (MSM) process based on a hard clustering of the state space, Satuluri et al. were able to give an upper bound for $E(k)$ [65].

Remember that reversible Markov processes feature some very desirable properties. The associated semi-group of transfer operators (T_t) is self-adjoint in L^2_μ and due to the reversibility of the process, forward and backward committors as defined in Section 3.3.1 coincide $q_i(x) := q_i^+(x) = q_i^-(x)$. In this case, as the clustering is a full partition of the state space, the committor functions are merely characteristic functions of the clusters $q_i(x) = \chi_{C_i}(x)$. Therefore, the orthogonal projections Q_1 and Q_2 in the sense of milestoning processes are identical, projecting onto the m-dimensional space spanned by step functions $D_m = span\{\chi_{C_1}(x), \ldots, \chi_{C_m}(x)\}$.

We find that the projected transfer operator $QT_\tau Q : D_m \to D_m$ is a linear operator on a finite-dimensional space and its matrix representation with respect to the basis (ψ_i)

given by

$$\psi_i = \frac{\chi_{C_i}}{\mu(C_i)}$$

is a familiar sight:

$$\begin{aligned} QT_\tau Q\psi_i = QT_\tau\psi_i &= \sum_{j=1}^{m} \frac{\langle T\psi_i, \chi_{C_j}\rangle}{\mu(C_j)}\chi_{C_j} = \sum_{j=1}^{m} \frac{\langle T\chi_{C_i}, \chi_{C_j}\rangle}{\mu(C_i)}\psi_j \\ &= \sum_{j=1}^{m} \left(\int_{C_i} \mathbb{P}[X_\tau \in C_j | X_0 = x] d\mu(x) \right) \psi_j = \sum_{j=1}^{m} \mathbb{P}_\mu[X_\tau \in C_j | X_0 \in C_i]\psi_j. \end{aligned}$$

So we conclude that a matrix representation of the projected transfer operator is given by the transition matrix $\tilde{P}$ of the MSM process as defined in Section 3.4 and therefore we may now give a simpler formula for the approximation error by

$$E(k) = ||QT_\tau^k Q - (QT_\tau Q)^k|| = ||QT_\tau^k Q - \tilde{P}^k||.$$

While we do not wish to go into full detail on the error estimation, which has been covered by Sarich et al. [65], we content ourselves with the key result, that the maximal possible error decays exponentially and that the leading constant, though also depending on the spectral gap, can be made smaller than any tolerance by selecting $C_1, \ldots, C_m$ and the lag time τ appropriately. While these results can be generalized to some non-reversible processes which are sufficiently ergodic and have a dominant part of T that is nearly self-adjoint [23], an error estimation applicable to all non-reversible processes is still open.

4.4 Random walk clustering

The generalization of the random walk approach from the previous chapter to directed graphs does not work in a straightforward manner, because a standard random walk process respecting the direction of arcs cannot be reversible except for trivial cases. Non-reversible processes do not possess self-adjoint transfer operators in general, which do not have a real-valued spectrum in consequence. When we learn how eigenvectors corresponding to complex eigenvalues are related to metastable sets of the random walk process, new clustering methods might be developed exploiting these relations. Up to now, we have no proof that there is any such relation and therefore recent attempts of random walk based clustering algorithms for directed graphs focus on assembling a self-adjoint transfer operator applicable for Markov state modeling from the transfer operators associated to forward and backward running random walk processes. Here, an approach developed by Nataša Đurđevac shall be presented [23].

Let us start by considering two Markov processes (X_t) and (Y_t) on a continuous state space S, that do not necessarily represent forward and backward running random walkers. Unless otherwise stated, the index parameter i shall be an element of $\{1, 2\}$. Let $p_i : V \times V \to [0, 1]$ be the associated transition functions of the above mentioned processes and let two positive measures μ_i satisfy

$$\mu_2(y) = \int p_1(x, y)\mu_1(x)dx \quad \text{and} \quad \mu_1(y) = \int p_2(x, y)\mu_2(x)dx.$$

Then, let us define two scalar products

$$\langle f, g\rangle_i = \int f(x)g(x)\mu_i(x)dx$$

in Hilbert spaces determined by these scalar products $H_i = L^2_{\mu_i}$ and finally two transfer operators

$$(T_{12}f)(y)\mu_2(y) = \int p_1(x,y)f(x)\mu_1(x)dx \quad \text{and} \quad (T_{21}f)(y)\mu_1(y) = \int p_2(x,y)f(x)\mu_2(x)dx.$$

We would like to find representations of these operators in L^2. For simplicity, we introduce two linear transformations

$$J_i : L^2 \to H_i, g_i = J_i g, \quad (J_i g)(x) = \frac{g(x)}{\sqrt{\mu_i(x)}}.$$

These transformations preserve the respective norms of H_i:

$$||g||^2 = (g,g) = (J_i^{-1}g_i, J_i^{-1}g_i) = \int g_i(x)^2\mu_i(x)dx = \langle g_i, g_i\rangle_i = ||g_i||_i^2.$$

Here and onwards, $(\cdot,\cdot)$ shall denote the standard scalar product of L^2.

Both operators are compatible with the unit measure of their respective spaces $T_{21}\mathbb{1} = \mathbb{1}, T_{12}\mathbb{1} = \mathbb{1}$. Additionally, they possess representations in L^2, such as

$$(P_{21}f)(y)\sqrt{\mu_1(y)} = \int p_2(x,y)f(x)\sqrt{\mu_2(x)}dx,$$

satisfying the equations

$$\langle g_1, T_{21}f_2\rangle_1 = \int\int g(y)\frac{p_2(x,y)}{\sqrt{\mu_1(y)}}f(x)\sqrt{\mu_2(x)}dxdy = (g, P_{21}f) \tag{4.4.1}$$

and

$$\langle T_{12}g_1, f_2\rangle_2 = (P_{12}g, f) \tag{4.4.2}$$

respectively. We may give these representations in the kernel form

$$(P_{21}f)(y) = \int \pi_{21}(x,y)f(x)dx,$$

where π_{21} is the kernel given by

$$\pi_{21}(x,y) = \sqrt{\frac{\mu_2(x)}{\mu_1(x)}}p_2(x,y).$$

In terms of the linear transformations introduced above, we may express the representation P_{21} of T_{21} as

$$(g, P_{21}f) = \langle g_1, T_{21}f_2\rangle_1 = (J_1^{-1}g_1, J_1^{-1}T_{21}f_2) = (g, J_1^{-1}T_{21}J_2 f)$$

and thus

$$P_{21} = J_1^{-1}T_{21}J_2. \tag{4.4.3}$$

In analogy we get $\langle T_{12}g_1, f_2\rangle_2 = (P_{12}g, f)$ and

$$P_{12} = J_2^{-1}T_{12}J_1 \tag{4.4.4}$$

for P_{12}, the L^2-representation of T_{12}.

4.4.1 Extended detailed balance condition

Now we have the means to prove an extension of the detailed balance condition for time-reversible Markov processes. Later we will see that it coincides with the original formulation when the considered transfer operators are equal, i.e. when they are associated to a time-reversible process running forward or backward in time.

Theorem 4.4.1. *Operators P_{12} and P_{21} are adjoint in L^2 with standard scalar product if and only if*

$$\mu_1(x)p_1(x,y) = \mu_2(y)p_2(y,x)$$

is satisfied.

Proof. Consider P_{21} and P_{12} in their kernel forms. Then $(P_{12}f, g) = (f, P_{21}g)$ is equivalent to $\pi_{12}(x,y) = \pi_{21}(y,x)$ and thus equivalent to the condition above. □

For time-reversible processes, the formula occurring in the theorem reduces to the standard detailed balance condition 3.4.3, stating that the associated transfer operator is self-adjoint. There are examples of non-reversible processes which fulfill the extended detailed balance condition none the less.

Let us now assume that the extended detailed balance condition is satisfied. Due to (4.4.1), (4.4.2) and the adjointness of P_{21} and P_{12} it is evident that

$$\langle T_{12}f_1, g_2\rangle_2 = \langle f_1, T_{21}g_2\rangle_1 \tag{4.4.5}$$

and

$$\langle T_{21}f_2, g_1\rangle_1 = \langle f_2, T_{12}g_1\rangle_2. \tag{4.4.6}$$

Using these operators we construct an operator suitable for our MSM approach.

Theorem 4.4.2. *The operator $\mathcal{T} = T_{21}T_{12} : H_1 \to H_1$ is self-adjoint on H_1.*

Proof. For any $f, g \in L^2$ with $f_1 = J_1 f \in H_1$ and $g_1 = J_1 g \in H_1$ we also have $T_{12}f_1 \in H_2$ and $T_{12}g_1 \in H_2$. Using (4.4.5) and (4.4.6) we have

$$\langle f_1, \mathcal{T}g_1\rangle_1 = \langle f_1, T_{21}T_{12}g_1\rangle_1 = \langle T_{12}f_1, T_{12}g_1\rangle_2 = \langle T_{21}T_{12}f_1, g_1\rangle_1 = \langle \mathcal{T}f_1, g_1\rangle_1.$$

□

In analogy $\mathcal{B} = T_{12}T_{21} : H_2 \to H_2$ is self-adjoint as an operator on H_2.

4.4.2 Singular value decomposition

We will now restrict ourselves to discrete state spaces for simplicity. Then the self-adjoint operator $\mathcal{T}$ can be written as

$$\mathcal{T} = T_{21}T_{12} = \sum_k \lambda_k^2 \langle \cdot, \phi_k \rangle_1 \phi_k,$$

where $\{\phi_k\}$ are orthonormal eigenvectors of $\mathcal{T}$ with respect to $\langle \cdot, \cdot \rangle_1$ corresponding to eigenvalues λ_k^2, so

$$\mathcal{T}\phi_k = T_{21}T_{12}\phi_k = \lambda_k^2 \phi_k.$$

Now we introduce $\psi_k = \lambda_k^{-1} T_{12}\phi_k$, such that

$$\langle \psi_k, \psi_l \rangle_2 = \lambda_k^{-1}\lambda_l^{-1}\langle T_{12}\phi_k, T_{12}\phi_l \rangle_2 \quad = \lambda_k^{-1}\lambda_l^{-1}\langle \phi_k, T_{21}T_{12}\phi_l \rangle_1 = \lambda_k^{-1}\lambda_l \langle \phi_k, \phi_l \rangle_1 = \delta_{kl}.$$

So $\{\psi_k\}$ are orthonormal with respect to $\langle \cdot, \cdot \rangle_2$. But furthermore, they are eigenvectors of $\mathcal{B} = T_{12}T_{21}$ corresponding to eigenvalues λ_k^2, as we may easily check

$$\mathcal{B}\psi_k = T_{12}T_{21}\psi_k = \lambda_k^{-1}T_{12}T_{21}T_{12}\phi_k = \lambda_k^{-1}\lambda_k^2 T_{12}\phi_k = \lambda_k T_{12}\phi_k = \lambda_k^2 \psi_k$$

Now we can write the so called dispersion relations for transfer operators T_{12} and T_{21}

$$T_{12}\phi_k = \lambda_k \psi_k, \qquad T_{21}\psi_k = \lambda_k \phi_k. \tag{4.4.7}$$

Without a proof we will also use these singular value decomposition of the two transfer operators given by

$$T_{12} = \sum_k \lambda_k \langle \cdot, \phi_k \rangle_1 \psi_k \quad \text{and} \quad T_{21} = \sum_k \lambda_k \langle \cdot, \psi_k \rangle_2 \phi_k. \tag{4.4.8}$$

We will use our newly gained knowledge to find a matrix representation of our transfer operators in L^2. To do so, the linear transformations introduced in the last section will come in handy. Let us define the vectors $\{\hat{\phi}_k\}$ and $\{\hat{\psi}_k\}$ by

$$\hat{\phi}_k = J_1^{-1}\phi_k \quad \text{and} \quad \hat{\psi}_k = J_2^{-1}\psi_k.$$

Now using (4.4.4) and the dispersion relation for T_{12} (4.4.7) we get

$$T_{12}\phi_k = J_2 P_{12} J_1^{-1} J_1 \hat{\phi}_k = J_2 P_{12}\hat{\phi}_k = \lambda_k J_2 \hat{\psi}_k.$$

That is $P_{12}\hat{\phi}_k = \lambda_k \hat{\psi}_k$ and in analogy $P_{21}\hat{\psi}_k = \lambda_k \hat{\phi}_k$. Therefore, we may formulate dispersion relations for P_{12} and P_{21} very similar to those above

$$P_{12}\hat{\phi}_k = \lambda_k \hat{\psi}_k \qquad \text{and} \qquad P_{21}\hat{\psi}_k = \lambda_k \hat{\phi}_k. \tag{4.4.9}$$

As J_1 and J_2 are linear transformations preserving their respective norms, $\{\hat{\phi}_k\}$ and $\{\hat{\psi}_k\}$ are sets of orthonormal vectors in L^2.

We are now close to finding a matrix representation of P_{12}. We use (4.4.4) and (4.4.8)

to provide a bi-orthogonal decomposition of P_{12} in L^2 by

$$P_{12}f = J_2^{-1}T_{12}J_1 f = \sum_k \lambda_k \langle J_1 f, \phi_k\rangle_1 J_2^{-1}\psi_k = \sum_k \lambda_k (f, J_1^{-1}\phi_k)\hat{\psi}_k = \sum_k \lambda_k (f, \hat{\phi}_k)\hat{\psi}_k,$$

and in analogy

$$P_{21}f = \sum_k \lambda_k (f, \hat{\psi}_k)\hat{\phi}_k.$$

Using $P_{12} = P_{21}^T$, the dispersion relation for P_{12} (4.4.9) and the insight that $\{\hat{\psi}_k\}$ are normed, we find that $P_{21}^T\hat{\phi}_k = \lambda_k\hat{\psi}_k$ and thus $\hat{\phi}_k^T P_{21}^T\hat{\phi}_k = \lambda_k$.

Let us define $\hat{\phi} = [\hat{\phi}_1 \dots \hat{\phi}_n]$ as the matrix which has $\hat{\phi}_i$ as its i-th column and similarly $\hat{\psi} = [\hat{\psi}_1 \dots \hat{\psi}_n]$, then we get

$$\hat{\psi}^T P_{12}\hat{\phi} = D,$$

where $D = diag(\lambda_i)$ is the diagonal matrix (D_{ij}) given by $D_{ii} = \lambda_i$.

Using the invertibility of $\hat{\phi}$ and $\hat{\psi}$ due to the orthogonality of their columns and that $(\hat{\psi}_k, \hat{\psi}_l) = \delta_{kl} = (\hat{\phi}_k, \hat{\phi}_l)$ we finally receive matrix representation of P_{12}

$$P_{12} = (\hat{\psi}^T)^{-1} D \hat{\phi}^{-1}.$$

In analogy we get $P_{21} = (\hat{\phi}^T)^{-1} D \hat{\psi}^{-1}$.

4.4.3 Coarse graining

Let us now focus on one of the two non-reversible processes we have discussed above, for example (X_t) and assume that this process has an unique invariant measure μ. Then, we introduce the transfer operator $T : L^2_\mu \to L^2_\mu$, with

$$(Tf)(y)\mu(y) = \sum_x p(x,y) f(x)\mu(x).$$

For a linear transformation

$$J : L^2 \to L^2_\mu, \qquad (J\tilde{f})(x) = \frac{\tilde{f}(x)}{\sqrt{\mu(x)}} = f(x)$$

arbitrary $\tilde{f}, \tilde{g} \in L^2$ and $f = J\tilde{f}, g = J\tilde{g} \in L^2_\mu$, it follows that

$$\langle f, Tg\rangle_\mu = \sum_y f(y)(Tg)(y)\mu(y) = \sum_{x,y} \frac{\sqrt{\mu(x)}}{\sqrt{\mu(y)}} \tilde{f}(y) p(x,y) \tilde{g}(x) = (\tilde{f}, P\tilde{g}),$$

where P is the representation of T in L^2

$$(Pf)(y)\sqrt{\mu(y)} = \sum_x p(x,y) f(x) \sqrt{\mu(x)},$$

which can be written in the typical kernel form

$$(Pf)(y) = \sum_x \pi(x,y) f(x) \qquad \text{with} \qquad \pi(x,y) = \frac{\sqrt{\mu(x)}}{\sqrt{\mu(y)}} p(x,y).$$

Furthermore, from

$$\langle f, Tg\rangle_\mu = (J^{-1}f, J^{-1}Tg) = (\tilde{f}, J^{-1}TJ\tilde{g}), \tag{4.4.10}$$

we see that

$$P = J^{-1}TJ.$$

We will now follow an approach based on MSM and use it in order to find dominant metastable sets of the observed process (X_t). Let us assume that the state space S contains m disjoint sets $C_1, \dots, C_m \subset S$ that do not form a full partition of S in general. Then, for the process (X_t) with the transfer operator T, we can define the milestoning process $(\hat{X}_t)$ in the following way

$$(\hat{X}_t) = i \iff X_{\sigma(t)} \in C_i, \text{ with } \sigma(t) = \sup_{s\le t}\left\{X_s \in \bigcup_{k=1}^{m} C_k\right\}.$$

Let q_i^+ and q_i^- denote forward and backward committors related to T. For backward committors q_i^-, let us introduce $\tilde{q}_i^- \in L^2$ with $\tilde{q}_i^- = J^{-1}q_i^-$, such that for $x \notin \bigcup_{j=1}^m C_j$

$$(Tq_i^-)(x) = (JPJ^{-1}q_i^-)(x) = (JP\tilde{q}_i^-)(x) = q_i^-(x) = (J\tilde{q}_i^-)(x),$$

where we used that $(Tq_i^-)(x) = q_i^-(x)$, for all $x \notin \bigcup_{j=1}^m C_j$. Therefore, functions $\tilde{q}_i^-$ are representations of q_i^- in L^2

$$(P\tilde{q}_i^-)(x) = \tilde{q}_i^-(x), \quad \forall x \notin \bigcup_{j=1}^{m} C_j.$$

Similarly, we define the set of $\tilde{q}_i^+ \in L^2$ with $\tilde{q}_i^+ = J^{-1}q_i^+$ that are representations of q_i^+ in L^2.

Theorem 4.4.3. *For a time-discrete process (X_n), the entries of the discrete generator $\tilde{L}_d$ of the milestoning process $(\tilde{X}_n)$ are given by*

$$\tilde{l}_d(i,j) = \frac{1}{\tilde{\mu}(i)}(\tilde{q}_j^+, L_d\tilde{q}_i^-),$$

where $\tilde{\mu}(i) = \sum_x \tilde{q}_i^-(x)\sqrt{\mu(x)}$, $L_d = P - Id$ and P is the representation of T in L^2.

Proof. The entries $\tilde{l}_d(i,j)$ are

$$\tilde{l}_d(i,j) = \frac{1}{\tilde{\mu}(i)}\langle q_j^+, \mathcal{L}_d q_i^-\rangle_\mu,$$

where using (4.4.10) it follows that

$$\langle q_j^+, \mathcal{L}_d q_i^-\rangle_\mu = \langle q_j^+, (T-Id)q_i^-\rangle_\mu = (\tilde{q}_j^+, P\tilde{q}_i^-) - (\tilde{q}_j^+, \tilde{q}_i^-) = (\tilde{q}_j^+, (P-Id)\tilde{q}_i^-).$$

The invariant measure of the milestoning process $\tilde{\mu}$ is

$$\tilde{\mu}(i) = \mathbb{P}_\mu(\hat{X}_n = i) = \sum_x \mathbb{P}_\mu(\hat{X}_n = i, X_n = x) = \sum_x q_i^-(x)\mu(x) = \sum_x \tilde{q}_i^-(x)\sqrt{\mu(x)}.$$

□

4.4.4 Random walk on directed graphs

We will now show how the results of the proceeding sections lead to the construction of self-adjoint transfer operators eligible for Markov state modeling. In simple terms, we define forward and backward transfer operators, which describe the propagation of density functions on the graph under the dynamics of a random walk process running forward or backward in time. By composing these operators, we gain an operator associated to a reversible process which emerges from the original process when the directions of the edges switch persistently.

To this end, define the time-forward random walk, a discrete process on the directed graph $G = (V, E)$ which chooses uniformly in each step which outgoing arc to follow. Let $d_{out}(x)$ be the out-degree of a vertex x, the number of directed edges pointing from x to some other vertices. Then we set the entries of the transition matrix $P^+ = (p^+(x, y))_{x,y\in V}$ to

$$p^+(x, y) = \begin{cases} 1/d_{out}(x), & (x, y) \in E \\ 0, & (x, y) \notin E. \end{cases} \tag{4.4.11}$$

Assuming that the network has no sources or sinks, P^+ is a well-defined stochastic matrix. In analogy we define the time-backward random walk process by the transition matrix $P^- = (p^-(x, y))_{x,y\in V}$ with entries

$$p^-(x, y) = \begin{cases} 1/d_{in}(x), & (y, x) \in E \\ 0, & (y, x) \notin E, \end{cases}$$

where $d_{in}(x)$ is the in-degree of vertex x. If we perceive an undirected edge as both an in- and an out-going arc, the forward and backward random walks on an undirected graph coincide. We can therefore see this as a true generalization of the random walk approach on undirected graphs.

Let us define the two following probability measures

$$\mu^+(x) = \frac{d_{out}(x)}{vol(G)} \text{ and } \mu^-(x) = \frac{d_{in}(x)}{vol(G)}, \tag{4.4.12}$$

where $vol(G) := \sum_{y\in V} d_{out}(y) = \sum_{y\in V} d_{in}(y)$. The extended detailed balance condition as given in Section 4.4.1 is satisfied as the following conversion shows:

$$\mu^+(x)p^+(x, y) = \frac{d_{out}(x)}{vol(G)} \frac{1}{d_{out}(x)} = \frac{1}{vol(G)} = \frac{d_{in}(x)}{vol(G)} \frac{1}{d_{in}(x)} = \mu^-(x)p^-(x, y).$$

Note however, that μ^+ and μ^- are in general not stationary distributions with respect to P^+ or P^-, but they satisfy

$$({\mu^+}^T P^+)(y) = \sum_{x\in V} \frac{d_{out}(x)}{vol(G)} \frac{1}{d_{out}(x)} \chi_E(x, y) = \frac{1}{vol(G)} d_{in}(y) = \mu^-(y)$$

and in analogy ${\mu^-}^T P^- = \mu^+$.

With these measures we can define two transfer operators

$$(T_{12}f)(y)\mu^-(y) = \sum_{x\in V} \mu^+(x)f(x)p^+(x,y) \tag{4.4.13}$$

and

$$(T_{21}f)(y)\mu^+(y) = \sum_{x\in V} \mu^-(x)f(x)p^-(x,y) \tag{4.4.14}$$

and notice that they are adjoint in the sense of Equation 4.4.5

$$\langle T_{12}f, g\rangle_{\mu^-} = \langle f, T_{21}g\rangle_{\mu^+}.$$

Consequently, the composed operator $\mathcal{T} = T_{21}T_{12}$ is self-adjoint and therefore eligible for Markov state modeling in order to identify core clusters of the directed graph as metastable sets of the random walk process and assign the remaining vertices in a fuzzy clustering on the lines of the undirected case. In the next section we will take a closer look at ways to play out this concept and open questions concerning its direct implementation.

4.5 Module identification in directed graphs

The attempt to generalize the random walk approach to directed graphs presented in the previous section provided us a self-adjoint transfer operators $\mathcal{T}$. One might be tempted to use it to construct a clustering of the directed graph by fuzzy MSM in the same way we would use the unique self-adjoint random walk transfer operator of an undirected graph. At second glance this is not as straightforward as it appears. Instead of $\mathcal{T} = T_{21}T_{12}$ we could choose $\mathcal{B} = T_{12}T_{21}$, which is self-adjoint as well, to find a clustering. Why should we favor one over the other? Can we incorporate both into a unique clustering and would any of these help us identify modules which correspond to what we expect intuitively? We want to discuss these matters in this section and conclude with a proposal on how to cluster directed graphs using a symmetrization step.

We have been looking at clustering methods in the previous sections already and this might appear to be a late time to question the definition of clusters, but we have seen several clustering approaches which were designed on different ideas of what clusters are and hence, how to identify them. In the undirected case the perception of clusters being highly connected subsets of the vertex set has rarely been in dispute. The colloquial meaning of the word 'cluster', suggests an accumulation of vertices where two connected vertices are considered to be 'close-by'. In that sense, the usual definition fits well to intuition.

In the case of directed graphs however, the generalization of this idea is not as obvious. We could ignore the direction of arcs altogether and use the same clustering algorithms as in the undirected case. From a random walk perspective, this causes contradictions to common expectations. A highly connected subgraph might be directed in a way which forces a random walker to leave it quickly, but would be identified as a metastable set of a standard random walk process if we ignore directions, such as in the situation illustrated by Figure 4.5.

The previous section led us to an operator which takes the direction of edges into

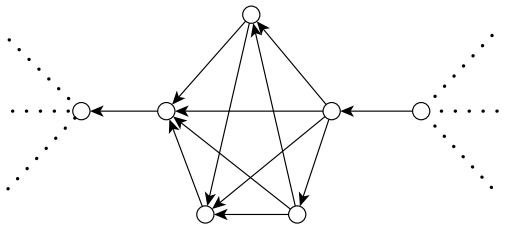


Figure 4.5: A K_5 subgraph. Due to the orientation of arcs a random walker would leave it within a few steps, although it is densely connected.

consideration, yet it might yield odd results as it allows densities to dissipate backwards through a network. As it seemingly dismisses edge directions, one might question whether it is appropriate for the construction of clusterings with respect to the special nature of directed graphs. We substantiate this claim by a simple graph example shown in Figure 4.6a. The associated random walk transition matrices are

$$P^+ = P^- = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \end{pmatrix}.$$

In the spirit of formulas 4.4.12 we introduce the measures

$$\mu^+ = \mu^- = (1/4, 1/4, 1/4, 1/4)^T,$$

allowing us to calculate T_{12} and T_{21}. Eventually, we end up with a matrix representation of $\mathcal{T}$, such that we can write

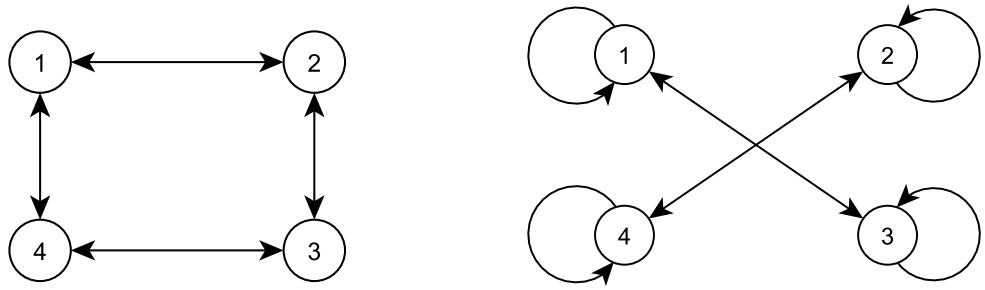
$$(\mathcal{T}f)(x) = \begin{pmatrix} 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \end{pmatrix} f(x).$$

From the block structure of the transformation matrix we recognize that $\mathcal{T}$ denies any interaction between the vertex sets $\{1, 3\}$ and $\{2, 4\}$. So, composing a forward step and a backward step of a random walk in such a way does not only permit traveling arcs in the wrong direction, but may even consume existing edges and create new ones, leading to a completely different process as illustrated in Figure 4.6b and conceivably deliver clusterings which do not resemble the original graph structure at all.

Assuming a **strongly connected** directed graph, that is any two vertices are connected in both directions by a directed path, the measures μ^+ and μ^- are nowhere-zero and therefore we may rewrite formulas 4.4.13 and 4.4.14 in vector/matrix notation as

$$T_{12}f = ((\mu^-)^T Id(\mu^+)^{-1})(P^+)^T f \quad \text{and} \quad T_{21}f = ((\mu^+)^T Id(\mu^-)^{-1})(P^-)^T f,$$

where $(\mu^+)^{-1}$ and $(\mu^+)^{-1}$ denote entry-wise inversion of the two vectors. Then the matrix representation of $\mathcal{T}$ can be simplified to $(P^+P^-)^T$, as the pre-factors cancel out. Note that, as $\mathcal{T}$ is a self-adjoint operator by construction, its real-valued matrix representation



(a) Small example of a directed graph.

(b) Visualization of the dynamics described by operator $\mathcal{T}$.

Figure 4.6: An illustration of how the operator $\mathcal{T} = T_{21}T_{12}$ can distort the dynamics of a random walk process.

is symmetric $(P^+P^-)^T = P^+P^-$. This equation can be derived directly from the extended detailed balance condition as well. This result substantiates the remark above, that the propagation of density functions by the operator $\mathcal{T}$ can be grasped as alternating forward and backward time-steps of a random walk process.

Consequently, when two vertices x and y are connected by a bi-directional arc, such as in the example shown in Figure 4.6a, there must be a loop at vertex x in the graph representation of $\mathcal{T}$ since

$$\mathcal{T}(x,x) = \sum_{z \in V} p^+(x,z)p^-(z,x) \geq p^+(x,y)p^-(y,x) > 0.$$

The same applies to y which also gains a loop in the process. In the same spirit we observe that two vertices x_1 and x_2 which both have outgoing edges pointing at another vertex x_3, will be directly connected in the graph representing the dynamics of $\mathcal{T}$.

Another thing we learn from this consideration is that this random walk approach to clustering directed graphs may as well be regarded as a special case of the graph symmetrizations explained in Section 4.2.2, where the symmetric adjacency matrix B of an undirected graph is constructed from the adjacency matrix A of the original graph by $B = AA^T$, which is known as the bibliometric coupling. The matrix representation of $\mathcal{T}$ has the very same non-zero structure as B, since P^+ and P^- are simply derived from A and A^T by normalizing their columns. Therefore the symmetrized graph obtained is the same up to different edge weights.

Next, we want to go further and look at clustering methods that do not require any symmetrization which possibly sacrifices information about cyclic structures and one-way connections in directed networks that an undirected network can not preserve. The next chapter details a method based on analyzing hitting times instead of eigenvectors, which allows to work with non-reversible Markov chains like random walks on directed networks.

5 | Clustering via Hitting Times

In recent years, techniques for finding metastable or almost invariant sets in dynamical systems have attracted a lot of attention, for deterministic systems [15] and set-oriented numerics [14, 16] as well as stochastic systems [70] with applications in molecular dynamics [71, 9]. These techniques allow to identify metastability by discretization of the transfer operator or Frobenius–Perron operator of the underlying dynamics which results in a stochastic matrix and an associated Markov chain. Metastable or almost invariant sets are then identified via spectral approaches: the dominant eigenvectors of the matrix contain essential information about the invariant measure of the system and about the decomposition of the system into metastable sets.

There is a long list of articles on finding metastable decompositions of Markov chains, via spectral approaches [20, 36, 21, 30, 17, 41, 71] and Markov chain aggregation techniques [12, 48] as well as via exit time approaches [7, 8, 71]. However, by far most results are only available for reversible Markov chains, or do not result in robust algorithms for finding metastable decompositions with more than two sets.

In addition to the discussion about metastability in dynamical systems, Markov chain decompositions are key to finding good partitions of networks into modules: When wanting to find the strongly connected modules of a network, one can do this via the metastable sets of an appropriately defined random walk on the network, see [44, 64]. The Markov chain associated with the random walk is reversible as long as the network is undirected; directed network, in general, lead to non-reversible chains. Therefore many of the powerful algorithms for identifying metastability cannot be used for finding modules in directed networks.

Different recent approaches have led to working algorithms for community detection in graphs – such as the method introduced by Newman and Girvan [53] exploiting 'betweenness' scores of edges, a multi-level algorithm based on optimizing modularity [5] developed by a group of researchers then seated in Louvain [13], or the Infomap method based on the map equation [60]. Some are restricted to reversible dynamics, while others are not applicable to very big problems due to computational cost rising too quickly with network size. Most methods yield full decompositions of the state space, sometimes even a hierarchical cluster structure, but do not allow for overlapping clusters or separating truly metastable sets from the transient part of the state space.

The algorithmic approach we present herein is not based on completely new concepts, see for example results from Donsker and Varadhan [25] examining the relation between principal eigenvalues of infinitesimal generators and hitting times as well as the work of Betz and Le Roux [2], who build metastable representations of irreducibly perturbed

Markov chains exploiting committor functions, hitting and return times. The latter group has also presented an algorithm to compute the asymptotic stationary distribution of systems with multi-scale dynamics. To our knowledge, no one has exploited hitting times for the detection of metastable sets in a more general setting yet.

This situation calls for a generalization of the theory for decomposing metastable Markov chains to non-reversible chains, which is presented in this chapter. We show that the well-known statement about optimal metastable decompositions of reversible chains can be reformulated for non-reversible chains if one switches from a spectral approach to an exit time approach: We can get a lower bound to the metastability index of a decomposition by considering the exit time functions from metastable subsets such that decompositions with strong metastability can be found by lower bound maximization. Furthermore, we will see that the expected hitting times of test sets are almost constant on any metastable set with sharp jumps between sets; consequently we can use this property for identifying metastable sets. After establishing the underlying theory we will show how to use the results algorithmically and demonstrate the performance of the resulting algorithms in some numerical examples.

Remark: This chapter is based on collaborative work with Marco Sarich and Christof Schütte. The key ideas of the method presented herein are the foundation of an article to be published by the Journal of Computational Dynamics. A pre-print can be found in the database of Zuse Institute Berlin [66]. The author of this thesis developed the algorithm from a blueprint that existed before, implemented it in Matlab, contributed the analysis of computational properties, adapted a random network generator for that purpose and added several example cases on real world data.

Figures included in this chapter were used before in [66] and a follow-up paper accepted by the Journal of Computational Dynamics, pending publication at the time of writing.

5.1 Basics

In this section, we consider an irreducible aperiodic homogeneous Markov chain (X_j) on a finite state space $\mathbb{X} = \{1, \ldots, n\}$ with transition matrix P and transition probability $p(x, y)$ for each pair of states $x, y \in \mathbb{X}$. We label the invariant measure of the chain as μ, a left-hand side eigenvector of P. This invariant measure defines the μ-weighted scalar product $\langle u, v\rangle_\mu = \sum_{x\in\mathbb{X}} u(x)v(x)\mu(x)$ and an associated weighted 2-norm $\|\cdot\|_\mu$. For any given set $A \subset \mathbb{X}$ we define the probability to stay in A during one step by

$$P(A) = \mathbb{P}_\mu \left[X_1 \in A | X_0 \in A\right],$$

where $\mathbb{P}$ is a probability measure given by the probability space on which (X_j) is defined (see Definition 9 in Section 2.2). The index μ indicates that X_0, the initial distribution of (X_j) is distributed according to μ. Using the weighted scalar product $\langle u, v\rangle_\mu$, the probability to stay in A can be written as

$$P(A) = \frac{\langle \mathbf{1}_A, P\mathbf{1}_A\rangle_\mu}{\mu(A)}$$

where $\mu(A) = \sum_{x \in A} \mu(x) = \langle \mathbf{1}_A, \mathbf{1}_A \rangle_\mu$ is the invariant measure of some set $A \in \mathbb{X}$ and $\mathbf{1}_A$ denotes the indicator function of set A.

Metastability Intuitively, any metastable set are such sets A with $P(A) \approx 1$. The closer $P(A)$ is to 1, the more metastable is A. Thus, we speak of a metastable decomposition $\mathcal{D} = \{A_1, \ldots, A_m\}$ of the space $\mathbb{X}$ into m disjoint sets $\{A_1, \ldots, A_m\}$ with $\cup_i A_i = \mathbb{X}$ if the metastability index $\mathcal{M}(\mathcal{D})$ of the decomposition, defined as

$$\mathcal{M}(\mathcal{D}) = \frac{1}{m} \left(P(A_1) + \ldots + P(A_m) \right),$$

is close to one. An alternative definition of metastability is based on the notion of exit times, a special case of stopping times, as introduced in Definition 25, Section 2.2.2. If a set is metastable, we intuitively expect that the exit time of the chain from the set is large in comparison to other sets. Recall that the hitting time of a set $B \subset \mathbb{X}$ is defined as $\tau(B) = \min\{k : \ X_k \in B\}$. $\tau(B)$ is a random variable whose outcome may differ for every single realization of the Markov chain. As we are interested in a statistical average independent of chance, we turn to its expectation value $\mathbb{E}_x(\tau(B)) = \mathbb{E}(\tau(B)|X_0 = x)$ when starting from $x \notin B$, propagating forward w.r.t. the law of (X_j). Then, the expected exit time from A is given by the hitting time $\tau(A^c)$ of the complement $A^c = \mathbb{X} \setminus A$, starting somewhere in A, and takes the form

$$\mathbb{E}_x(\tau(A^c)) = \mathbb{E}(\tau(A^c)|X_0 = x), \qquad x \in A.$$

Networks Consider a strongly connected finite graph/network $G = (V, E)$, where V is the set of n vertices and E the set of directed edges of the graph. We denote the adjacency matrix of the network by $(a(x, y))_{x,y \in V}$ and the out-degree of a vertex x by $d(x) = \sum_{y \in V} a(x, y)$, in agreement with Definitions 2 and 3 in Section 2.1. If the network has weighted edges, then $a(x, y)$ denotes the weight of edge $(x, y) \in E$ and $a(x, y) = 0$ if $(x, y) \notin E$. When edges are unweighted, we simply set $a(x, y) = 1$ for any $(x, y) \in E$. Whenever a network is undirected, its adjacency matrix is symmetric. In network clustering based on random walks, we focus on the standard random walk defined on the network, i.e., the Markov chain with one-step transition matrix directly given by the adjacency structure / weights,

$$p(x, y) = \frac{a(x, y)}{d(x)}. \tag{5.1.1}$$

Note that P may be computed very directly from A by normalizing its row sums. This construction guarantees that P is a stochastic matrix, but requires row sums of A to be non-zero. In graph theory terminology, the network may not contain any sinks – thus the restriction to strongly connected graphs.

As discussed in Section 2.2.3, the associated Markov chain is irreducible and aperiodic, and if reversible, it has invariant measure $\mu(x) = d(x)/\sum_x d(x)$. That is, vertices with relatively high out-degree are visited often, i.e., subnetworks with strong internal connectedness are metastable sets of the associated chain. However, other random walks have been considered also, where $p(x, y)$ is defined in terms of $a(\cdot, \cdot)$ in a different way [64]. Another approach worth mentioning in this context are Markov jump processes with waiting times depending on each vertex' weight or out-degree to emphasize highly populated subgraphs over long bridges or circles in community detection [23].

5.2 Metastable decomposition of reversible Markov chains

Whenever a Markov chain is reversible, its metastability can be analyzed based on the leading eigenvalues of its transition matrix P. The chain (X_j) is said to be reversible, if the discrete detailed balance condition $\mu(x)p(x,y) = \mu(y)p(y,x)$, introduced in Section 3.4.2, holds. Then, P is symmetric with respect to $\langle \cdot, \cdot \rangle_\mu$ and thus all its eigenvalues are real-valued, as follows from Theorem 2.2.4. The standard random walk for any undirected network is reversible.

For reversible Markov chains, several mathematical statements relating dominant eigenvalues, corresponding eigenfunctions and a decomposition of their state space into metastable subsets are available [20, 36, 21, 41, 8, 71]. In this section, we consider the statement based on the metastability index $\mathcal{M}(\mathcal{D})$ of some decomposition $\mathcal{D} = \{A_1, \ldots, A_m\}$ of state space $\mathbb{X}$, see [36], and for a more general version [71]:

Theorem 5.2.1. *Let P denote a reversible $n \times n$ transition matrix with lowest eigenvalue $a = \min \sigma(P) > -1$. Let $\lambda_m \leq \ldots \leq \lambda_2 < \lambda_1 = 1$ be its eigenvalues, possibly counted according to multiplicity. Denote by $v_m, \ldots, v_1$ the corresponding eigenfunctions, normalized to $\|v_k\|_2 = 1$. Let Q be the orthogonal projection of $\mathbb{R}^n$ onto $\operatorname{span}\{\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_m}\}$. The metastability of an arbitrary decomposition $\mathcal{D} = \{A_1, \ldots, A_m\}$ of the state space $\mathbb{X}$ can be bounded from above by*

$$P(A_1) + \ldots + P(A_m) \leq 1 + \lambda_2 + \ldots + \lambda_m,$$

while it is bounded from below according to

$$1 + \kappa_2\lambda_2 + \ldots + \kappa_m\lambda_m + c \leq P(A_1) + \ldots + P(A_m)$$

where

$$\kappa_j = \|Qv_j\|_\mu^2 = \sum_{k=1}^{m} \frac{1}{\mu(A_k)} \langle v_j, \mathbf{1}_{A_k} \rangle_\mu^2.$$

and $c = a\,(1-\kappa_2)\ldots(1-\kappa_n)$.

Theorem 5.2.1 states that the metastability of an arbitrary decomposition $\mathcal{D}$ cannot be larger than $1 + \lambda_2 + \ldots + \lambda_m$, while it is at least $1 + \kappa_2\lambda_2 + \ldots + \kappa_m\lambda_m + c$, which is close to the upper bound whenever $\kappa_j \approx 1$ for all $j = 2, \ldots, n$. While the upper bound is dependent only on eigenvalues of P, the lower bound includes weights κ_j, which quantify how well eigenfunctions of P may be approximated by step functions that are constant on each element of the decomposition $\mathcal{D}$.

Thus the metastability index $\mathcal{M}$ of a decomposition $\mathcal{D} = \{A_1, \ldots, A_m\}$ will be high if

- the eigenvalues $\lambda_2 \geq \ldots \geq \lambda_m$ of P are all close to 1 and
- the corresponding dominant eigenfunctions $v_2, \ldots, v_m$ are almost constant on the metastable subsets $A_1, \ldots, A_m$, implying $\kappa_j \approx 1$.

The product c can be interpreted as a small correction term whenever $a \approx 0$ or $\kappa_j \approx 1\ \forall j$. In [36] it is demonstrated that the bounds given above are sharp and asymptotically exact.

Theorem 5.2.1 also highlights the strong relation between state space decompositions into metastable subsets and dominant eigenvalues. The closer the leading eigenvalues are to 1, the higher a decomposition's metastability index can be. To approach that upper bound, the decomposition must match as closely as possible the state space subsets where eigenvectors are nearly constant. Studies on the relation between nearly uncoupled Markov chains and dominant eigenvalues yield similar statements, cf. [48, 20, 8, 51].

In view of this theorem, the question arises, whether there is an *optimal* decomposition with highest possible metastability index. Several algorithms have been proposed to solve the associated optimization problem. However, even if there exists a unique optimal decomposition, the problem of finding it might be ill-conditioned [71]. The reason for this is that there may be extended transition regions between the metastable core sets for which a distinct assignment to one of the metastable core sets does not make sense. This problem can be resolved by relaxing the optimization to finding the core sets *after* identification and extraction of the transition region [71]. In Section 3.4.4, it was outlined how to identify the non-transient core sets using Markov jump processes.

5.3 Metastable decomposition of general Markov chains

Unlike undirected networks, the random walks associated to directed networks are, in general, not reversible. For non-reversible random walks, the eigenvalues and eigenvectors of the transition matrix may be complex-valued and the theorem presented in the last subsection is not applicable. We aim at a reformulation in terms of exit times, a special type of stopping times introduced in Definition 25, instead of spectral elements, in order to generalize the results above to directed networks.

5.3.1 Exit times from metastable sets

First, we aim at a result that connects the probability to stay in a set A to the mean exit time from it. Recall that the (first) exit time from a set $A \subset \mathbb{X}$ is the hitting time $\tau_{\mathbb{X}\setminus A}$ of the complement of A in X.

Lemma 5.3.1. *For every set $A \subset \mathbb{X}$, we have that*

$$\frac{\mathbb{E}_A[\tau(A^c)] - 1}{\max\limits_{x\in A} \mathbb{E}_x[\tau(A^c)]} \leq P(A),$$

where

$$\mathbb{E}_A[\tau(A^c)] = \frac{1}{\mu(A)} \sum_{x\in A} \mu(x)\mathbb{E}_x[\tau(A^c)].$$

denotes the exit time from A, weighted by the invariant measure μ.

Proof. Define the mean exit time from set A, conditioned on starting in vertex $x \in \mathbb{X}$ by $m(x) = \mathbb{E}_x[\tau(A^c)]$. Then for any $x \in A$ it follows from the definition that the exit time of x depends on the exit times of its neighboring vertices via

$$m(x) = 1 + \sum_{y\in A} p(x, y)m(y).$$

Inserting this into the definition of $\mathbb{E}_A[\tau(A^c)]$ we have

$$\mathbb{E}_A[\tau(A^c)] = 1 + \frac{1}{\mu(A)} \sum_{x,y \in A} \mu(x)p(x,y)m(y).$$

Dividing by $M := \max_{x \in A} \mathbb{E}_x[\tau(A^c)]$ yields

$$\begin{aligned} \frac{\mathbb{E}_A[\tau(A^c)] - 1}{M} &= \frac{1}{\mu(A)} \sum_{x,y \in A} \mu(x)p(x,y) \underbrace{\frac{m(y)}{M}}_{\leq 1} \\ &\leq \frac{1}{\mu(A)} \sum_{x,y \in A} \mu(x)p(x,y) = P(A). \end{aligned}$$

□

Lemma 5.3.1 exhibits that $P(A)$ will be close to 1 if

- the mean exit time $\mathbb{E}_A[\tau(A^c)]$ from A is large and
- $\mathbb{E}_x[\tau(A^c)]$ is almost constant for all $x \in A$, such that $\max_{x \in A} \mathbb{E}_x[\tau(A^c)] \approx \mathbb{E}_A[\tau(A^c)]$.

As a direct consequence of this lemma, we get the following lower bound to the metastability index of an arbitrary decomposition:

Theorem 5.3.2. *Let $A_1, ..., A_m$ be a decomposition of $\mathbb{X}$. Then,*

$$\hat{\lambda}_1 \hat{\kappa}_1 + \cdots + \hat{\lambda}_n \hat{\kappa}_n \leq \sum_{i=1}^{m} P(A_i)$$

where

$$\hat{\lambda}_i = 1 - \frac{1}{\mathbb{E}_{A_i}[\tau(A_i^c)]}, \qquad \hat{\kappa}_i = \frac{\mathbb{E}_{A_i}[\tau(A_i^c)]}{\max_{x \in A_i} \mathbb{E}_x[\tau(A_i^c)]}.$$

and $\mathbb{E}_{A_i}[\tau(A_i^c)]$ is the μ-averaged exit time from A_i, as defined in Lemma 5.3.1.

Proof. Define $m_i(x) = \mathbb{E}_x[\tau(A_i^c)]$ and $M_i := \max_{x \in A_i} \mathbb{E}_x[\tau(A_i^c)]$ and it follows immediately that

$$\frac{\mathbb{E}_{A_i}[\tau(A_i^c)] - 1}{M_i} = \hat{\lambda}_i \hat{\kappa}_i.$$

Using Lemma 5.3.1 for every set A_i and summing up the products $\hat{\lambda}_i \hat{\kappa}_i$ over $i = 1, ..., m$ completes the proof. □

According to this result, a decomposition $A_1, ..., A_m$ has a high metastability index if

- all $\hat{\lambda}_i$ are close to 1 – that is, if the expected exit times of the sets A_i are large on average with respect to the invariant measure μ and
- all $\hat{\kappa}_i$ are close to 1 – that is, if the μ-averaged expected exit time of each set is close to the largest possible exit time from single points within the respective set.

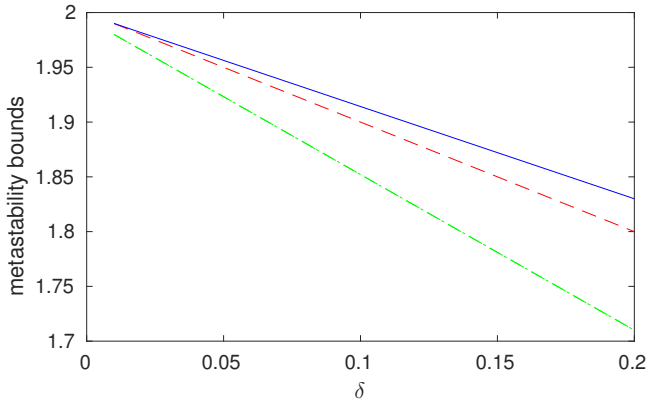


Figure 5.1: Metastable decomposition of the reversible Markov chain discussed in the example. Solid line: $P(A_1) + P(A_2)$ as a function of δ. Dashed line: Spectral bound $1 + \lambda_2\kappa_2$ as a function of δ. Dashed-dotted line: Hitting time bound $\hat{\lambda}_2\hat{\kappa}_1 + \hat{\lambda}_2\hat{\kappa}_2$ as a function of δ.
The figure is reprinted from [66].

The latter condition is certainly fulfilled if each hitting time function $m_i(x) = \mathbb{E}_x[\tau(A_i^c)]$ is almost constant on the respective set A_i, since

$$\hat{\kappa}_i = \frac{1}{\max\limits_{x\in A_i} m_i(x)} \frac{1}{\mu(A_i)} \sum_{x\in A_i} m_i(x)\mu(x).$$

Note that $\hat{\kappa}_i$ only measures the deviation of the μ-average of expected exit time on the set A_i to its maximal value, but there is no explicit relation to the minimal value. That is, the existence of states with rather small exit time from set A_i does not affect the value of $\hat{\kappa}_i$ much, as long as they are insignificant to the average exit time, in the sense that the invariant measure on these states is low enough.

Summarizing, it is apparent that $\hat{\lambda}_i$ and $\hat{\kappa}_i$ take the roles of λ_i and κ_i in Theorem 5.2.1.

Example: For reversible chains, the lower bounds given in Theorems 5.2.1 and 5.3.2 can be compared directly. To this end, we consider the reversible 4-state Markov chain with transition matrix

$$P = \begin{pmatrix} 0.4 & 0.6 & 0 & 0 \\ 0.5 & 0.5-\delta & \delta & 0 \\ 0 & \delta & 0.4-\delta & 0.6 \\ 0 & 0 & 0.3 & 0.7 \end{pmatrix},$$

where $0 \leq \delta < 0.4$ acts as a free parameter. For small δ the sets $A_1 = \{1, 2\}$ and $A_2 = \{3, 4\}$ form a metastable decomposition. Figure 5.1 shows the dependence of the metastability index on δ in comparison to the two lower bounds of Theorems 5.2.1 and 5.3.2.

5.3.2 Hitting times of test sets

Next, we will present conditions that ensure $\hat{\kappa}_i$, as introduced in Theorem 5.3.2, is close to 1, i.e., the function $m_i(x) = \mathbb{E}_x[\tau(A_i^c)]$ of exit times is almost constant on the respective set A_i.

Theorem 5.3.3. *Consider a Markov chain on some finite state space $\mathbb{X}$, two states $x, y \in \mathbb{X}$ and a test set $D \subset \mathbb{X}$. Then, the following two inequalities hold:*

(i)
$$\mathbb{E}_x[\tau(D)] \leq \mathbb{E}_x[\tau(y)] + \mathbb{E}_y[\tau(D)]$$

(ii)
$$\frac{|\mathbb{E}_x[\tau(D)] - \mathbb{E}_y[\tau(D)]|}{\max\{\mathbb{E}_x[\tau(D)], \mathbb{E}_y[\tau(D)]\}} \leq \max\left\{\frac{\mathbb{E}_x[\tau(y)]}{\mathbb{E}_x[\tau(D)]}, \frac{\mathbb{E}_y[\tau(x)]}{\mathbb{E}_y[\tau(D)]}\right\}.$$

Proof. To prove the first part, we follow the idea of Lemma 3.2 from a paper by Betz and Le Roux, which states a comparable result for return times [2].

We expand $\mathbb{E}_x[\tau(D)]$ as

$$\mathbb{E}_x[\tau(D)] = \mathbb{E}_x[\tau(D)]\mathbb{P}[\tau(D) \leq \tau(y)] + \mathbb{E}_x[\tau(D)]\mathbb{P}[\tau(D) > \tau(y)],$$

then exploit linearity of expectation values

$$\begin{aligned}\mathbb{E}_x[\tau(D)] = \mathbb{E}_x[\tau(D)]\mathbb{P}[\tau(D) \leq \tau(y)] + \mathbb{E}_x[\tau(y)]\mathbb{P}[\tau(D) > \tau(y)] \\ + \mathbb{E}_x[\tau(D) - \tau(y)]\mathbb{P}[\tau(D) > \tau(y)].\end{aligned}$$

The first two addends may be contracted to $\mathbb{E}_x[\min(\tau(D), \tau(y))]$, while the last part describes the expected time which the process, starting in x, takes to hit D after hitting y. This value must not depend on where the process came from before hitting y according to the strong Markov property (see Definition 17) and therefore is equal to $\mathbb{E}_y[\tau(D)]\mathbb{P}_x[\tau(D) > \tau(y)]$. This implies the first assertion, as

$$\begin{aligned}\mathbb{E}_x[\tau(D)] &= \mathbb{E}_x[\min(\tau(D), \tau(y))] + \mathbb{E}_y[\tau(D)]\mathbb{P}_x[\tau(D) > \tau(y)] \\ &\leq \mathbb{E}_x[\tau(y)] + \mathbb{E}_y[\tau(D)].\end{aligned}$$

Hence,

$$\mathbb{E}_x[\tau(D)] - \mathbb{E}_y[\tau(D)] \leq \mathbb{E}_x[\tau(y)].$$

By switching x and y, the same calculation also yields

$$\mathbb{E}_y[\tau(D)] - \mathbb{E}_x[\tau(D)] \leq \mathbb{E}_y[\tau(x)].$$

Now we consider two cases.

1. Assume $\mathbb{E}_x[\tau(D)] > \mathbb{E}_y[\tau(D)]$. Then,
$$\frac{|\mathbb{E}_x[\tau(D)] - \mathbb{E}_y[\tau(D)]|}{\max\{\mathbb{E}_x[\tau(D)], \mathbb{E}_y[\tau(D)]\}} = \frac{\mathbb{E}_x[\tau(D)] - \mathbb{E}_y[\tau(D)]}{\mathbb{E}_x[\tau(D)]} \leq \frac{\mathbb{E}_x[\tau(y)]}{\mathbb{E}_x[\tau(D)]}.$$

2. Is $\mathbb{E}_x[\tau(D)] < \mathbb{E}_y[\tau(D)]$, we have
$$\frac{|\mathbb{E}_x[\tau(D)] - \mathbb{E}_y[\tau(D)]|}{\max\{\mathbb{E}_x[\tau(D)], \mathbb{E}_y[\tau(D)]\}} = \frac{\mathbb{E}_y[\tau(D)] - \mathbb{E}_x[\tau(D)]}{\mathbb{E}_y[\tau(D)]} \leq \frac{\mathbb{E}_y[\tau(x)]}{\mathbb{E}_y[\tau(D)]}.$$

Putting both cases together proves the second assertion. □

Note that Theorem 5.3.3 (i) gives the triangle inequality for the distance $d(x,y) = \mathbb{E}_x[\tau(y)]$, if we consider test set D to consist of a single element. So $d(x,y)$ describes an asymmetric distance measure and the symmetrization $m(x,y) = 1/2(d(x,y) + d(y,x))$ is a metric. Theorem 5.3.3 (ii) has an interesting consequence in the case that $x, y \in M$ are elements of a metastable set M and D is an arbitrary test set with $D \cap M = \emptyset$. In this case, both ratios

$$\frac{\mathbb{E}_x[\tau(y)]}{\mathbb{E}_x[\tau(D)]} \qquad \text{and} \qquad \frac{\mathbb{E}_y[\tau(x)]}{\mathbb{E}_y[\tau(D)]}$$

should be very small, since the expected time to travel between elements within a metastable set M should be much smaller than the expected time to leave the metastable set and travel to another set D. Then, Theorem 5.3.3 implies that the hitting time function

$$m_D(x) = \mathbb{E}_x[\tau(D)]$$

should be almost constant on metastable sets, and this property should be robust against the choice of test set D.

5.3.3 Lower bounds using core sets

Next, we are interested in a quantitative estimate to our more qualitative argument regarding the constancy of exit times on metastable sets presented above. To this end, we study the exit time function from a metastable set A,

$$f(x) = \mathbb{E}_x[\tau(A^c)].$$

There may be states in A that belong to a transition region between the remaining graph and the most densely connected and isolated part of A, i.e., for which $f(x)$ is not really large. Perhaps, function f is not even nearly constant in the neighborhood of these states. However, assume that there is a core set $C \subset A$ with the property that C contains most of the μ-weight of A, and communication of states in C is much quicker on average than the exit from A, i.e.,

(A1) there is a small $\epsilon > 0$, such that $\frac{\mu(C)}{\mu(A)} = 1 - \epsilon$ and

(A2) there are constants $\delta > 0$ and $M > 1$ with $\delta/M \ll 1$, such that

- (a) $\mathbb{E}_x[\tau(y)] \leq \delta$ for $x, y \in C$ and
- (b) $f(x) \geq M$ for all $x \in C$.

We then derive the following statement regarding the value of the quantity

$$\hat{\kappa}(A) = \frac{1}{f_{max}} \frac{1}{\mu(A)} \sum_{x \in A} f(x)\mu(x), \qquad f_{max} = \max_{x \in A} f(x).$$

associated with set A, as introduced in Theorem 5.3.2:

Theorem 5.3.4. *Assume that set A satisfies the conditions (A1) and (A2) given above for a specific subset $C \subset A$. Furthermore, assume that there is a constant value $K > 0$,*

such that for all $x \in A \setminus C$ *we have*

$$\max_{x \in A \setminus C} \left\{ \frac{\mathbb{E}_x[\tau(y)]}{f(x)}, \frac{\mathbb{E}_y[\tau(x)]}{f(y)} \right\} \leq K,$$

for any $y \in A$ *with* $f(y) = f_{max}$*, and* $K = 0$ *if* $A = C$*. Then*

$$1 - \hat{\kappa}(A) \leq \frac{\delta}{M}(1 - \epsilon) + K\epsilon,$$

such that $1 - \hat{\kappa}(A) \ll 1$*, as long as* $K\epsilon \ll 1$*. Moreover we get the estimate*

$$1 - \hat{\lambda}(A) \leq \frac{1}{M(1 - \epsilon)}$$

for

$$\hat{\lambda}(A) = 1 - \frac{1}{\mathbb{E}_A[\tau(A^c)]}.$$

Proof. From Theorem 5.3.3 we get with $D = A^c$ for all $x \in A$ that

$$\frac{f_{max} - f(x)}{f_{max}} \leq \max \left\{ \frac{\mathbb{E}_x[\tau(y)]}{f(x)}, \frac{\mathbb{E}_y[\tau(x)]}{f_{max}} \right\}.$$

which implies that

$$\begin{aligned} 1 - \hat{\kappa}(A) &= \frac{1}{f_{max}} \left[\frac{1}{\mu(A)} \sum_{x \in A} (f_{max} - f(x))\, \mu(x) \right] \\ &\leq \frac{1}{\mu(A)} \sum_{x \in A} \max \left\{ \frac{\mathbb{E}_x[\tau(y)]}{f(x)}, \frac{\mathbb{E}_y[\tau(x)]}{f_{max}} \right\} \mu(x). \end{aligned}$$

For $x \in C$ condition (A2) yields

$$\max_{x \in C} \left\{ \frac{\mathbb{E}_x[\tau(y)]}{f(x)}, \frac{\mathbb{E}_y[\tau(x)]}{f_{max}} \right\} \leq \frac{\delta}{M},$$

while for $x \in A \setminus C$ we have

$$\max_{x \in A \setminus C} \left\{ \frac{\mathbb{E}_x[\tau(y)]}{f(x)}, \frac{\mathbb{E}_y[\tau(x)]}{f_{max}} \right\} \leq K.$$

With these estimates we get with condition (A1) that

$$1 - \hat{\kappa}(A) \leq \frac{1}{\mu(A)} \left(\mu(C) \frac{\delta}{M} + K\mu(A \setminus C) \right) = \frac{\delta}{M}(1 - \epsilon) + K\epsilon.$$

Using condition (A2) again, we find for $\mathbb{E}_A[\tau(A^c)] = (\sum_{x \in A} f(x)\mu(x))/\mu(A)$ that

$$\mathbb{E}_A[\tau(A^c)] \geq \frac{1}{\mu(A)}(M\mu(C) + \mu(A \setminus C) \cdot 1) = M(1 - \epsilon) + \epsilon \geq M(1 - \epsilon),$$

which implies the second assertion since $1 - \hat{\lambda}(A) = 1/\mathbb{E}_A[\tau(A^c)]$. □

For a decomposition $\mathcal{D} = \{A_1, \ldots, A_m\}$ in which every set A_i contains a core set C_i

such that $\mu(C_i) \approx \mu(A_i)$, and mixing within C is fast compared to exit times from A_i starting from inside C_i, this theorem asserts that $\mathcal{D}$ is a metastable partition in the sense that metastability index $\mathcal{M}(\mathcal{D})$ is very close to 1.

Example: Consider the 7-state Markov chain with the following transition matrix:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0.99 & 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.02 & 0 & 0.98 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

The chain is non-reversible and has an invariant measure

$$\mu^\intercal = (0.2210, 0.2210, 0.2232, 0.0045, 0.1116, 0.1094, 0.1094).$$

There are two metastable sets with state 4 being a transition state, more strongly attached to states 5 to 7 than 1 to 3. We therefore choose $A_1 = \{1, 2, 3\}$ and $A_2 = \{4, 5, 6, 7\}$. This choice leads to metastability index

$$\mathcal{M}(\{A_1, A_2\}) = 0.995.$$

Furthermore, direct computation of the quantities defined in Theorem 5.3.2 yields $\hat{\lambda}_1 = 0.997$, $\hat{\kappa}_1 = 0.997$, $\hat{\lambda}_2 = 0.997$ and $\hat{\kappa}_2 = 0.990$, so that the lower bound to the metastability index according to Theorem 5.3.2 is

$$\frac{1}{2}\left[\hat{\lambda}_1\hat{\kappa}_1 + \hat{\lambda}_2\hat{\kappa}_2\right] = 0.990.$$

In order to compute the lower bound resulting from Theorem 5.3.4, we choose the core sets $C_1 = \{1, 2, 3\}$ and $C_2 = \{5, 6, 7\}$, such that $\delta(A_1) = 3.53$, $M(A_1) = 298$, $\delta(A_2) = 8.14$, $M(A_2) = 298$, $\epsilon(A_2) = 0.013$ and $K(A_2) = 2.05$, which results in the estimate

$$\frac{1}{2}\left[\hat{\lambda}_1\hat{\kappa}_1 + \hat{\lambda}_2\hat{\kappa}_2\right] \geq 0.982.$$

We conclude that for those cases where some vertices can not be clearly assigned to any one metastable set, the existence of core sets which dominate the state space with respect to their μ-weight is sufficient to ensure the existence of a metastable decomposition.

Moreover, it indicates a path to find such a decomposition, starting with the identification of core sets based on exit times. In a second step, the remaining vertices may be assigned to their nearest core set, together forming metastable sets. This is, if a full decomposition of the state space is required.

Some Markov chains exhibit a modular structure, but also contain a high number of states that are difficult to assign to any single metastable set. In such cases, it can be favorable to detect core sets and leave the remaining states unassigned, yielding a decomposition into a number of highly metastable core sets and a transient region of low weight.

5.3.4 Algorithmic considerations

The spectral result stated in Theorem 5.2.1 can be exploited to algorithmically identify a metastable partition of a reversible Markov chain. First, we need to calculate the dominant spectrum that contains eigenvalues $1 = \lambda_1 > \lambda_2 \geq ... \geq \lambda_d$ that are close to 1 and corresponding eigenvectors $u_1, ..., u_d$. Then, we want to find a partition of state space into sets, such that the dominant eigenvectors are as constant as possible on these sets.

This problem can be ill-conditioned due to the existence of states in a transition region that cannot be clearly assigned to any metastable set. The transition region should be a set of low μ-weight, that the process typically leaves quickly for one of the core sets [71]. In [64] it was discussed how – for reversible chains – we can first identify the transition region, such that we can next restrict the problem to partitioning the core sets only: we assume that we are provided with a reference measure μ_r that allows us to find the transition region via

$$\mathcal{T} = \{x \in \mathbb{X} \mid (P^\intercal)^\alpha \mu_r(x) < \mu_r(x)\},$$

where $\alpha \in \mathbb{N}$ is the approximate average timescale on which the transition region is left.

Such a measure exists, e.g., when a Markov chain depends on a parameter that controls the metastability of the process. Say, we have the original Markov chain with transition matrix P and another Markov chain with reduced metastability, transition matrix P_r, and unique stationary distributions μ and μ_r, see, e.g., [64] for random walks on networks, or [71] for molecular dynamics with temperature embedding.

If a Markov chain is non-reversible, we assume that the transition region is left quickly either forward or backward in time. That is, if $\forall x \in \mathbb{X} : \ \mu(x) > 0$, we can define the transition matrix P_b of the time-reversed Markov chain with entries

$$p_b(x, y) = \frac{\mu(y)}{\mu(x)} p(y, x)$$

and the transition region as

$$\mathcal{T} = \{x \in \mathbb{X} \mid (P^\intercal)^\alpha \mu_r(x) < \mu_r(x) \text{ or } (P_b^\intercal)^\alpha \mu_r(x) < \mu_r(x)\}. \tag{5.3.1}$$

This notion implies that core sets are attractive sets in forward and backward time. If we do not have a parameter for controlling the metastability of the process, we simply choose μ_r to be a uniform distribution on the state space.

Having restricted the clustering problem to $C = \mathbb{X} \setminus \mathcal{T}$, the state space without transition region, there are optimization-based algorithmic strategies for finding partitions on which the dominant eigenvectors are almost constant [20, 36, 21]. We propose to associate with every state $x \in C$ the vector $u(x) = (u_1(x), ..., u_d(x)) \in \mathbb{R}^d$ and apply the k-means algorithm to the points $(u(x))_{x \in C}$ in $\mathbb{R}^d$.

In the general case of a non-reversible Markov chain, the eigenvalues and eigenvectors might not be real-valued and it is not clear how to use spectral information for clustering in this case. In the next section, we further discuss this issue using a simple network example. Our mathematical results motivate to replace the d leading eigenvectors by d hitting time functions with respect to suitably chosen test sets. Here, 'suitable' means that the test sets should not lead to redundant information in terms of hitting times, i.e.,

the sets should not be too close to each other.

In short, the proposed approach is to compute for d test sets $D_1, ..., D_d$ hitting time functions $m_i := m_{D_i}$ and use the algorithmic strategies from [20, 36, 21] on the functions m_i instead of the eigenvectors u_i. Later, we discuss the clustering of the points $(m(x))_{x\in C}$ in $\mathbb{R}^d$ with $m(x) = (m_1(x), ..., m_d(x))$ using k-means and some alternatives.

If the transition matrix P of the Markov chain is known, a hitting time function m_i can be computed as the solution of the following linear system

$$\begin{aligned} (P - Id)m_i &= -1 \qquad \text{on } \mathbb{X} \setminus D_i, \\ m_i &= 0 \qquad\quad\;\; \text{on } D_i. \end{aligned} \tag{5.3.2}$$

Summarizing, we have the following algorithmic blueprint:

1. Choose a number d of test sets and a metastability parameter $\alpha \in \mathbb{N}$.
2. Given P, P_b and μ_r, compute transition region $\mathcal{T}$ according to Equation 5.3.1.
3. Consider a test set $D_1 = \{x_1\}$ consisting of a randomly chosen single state $x_1 \in C = \mathbb{X} \setminus \mathcal{T}$ and compute hitting time function m_1.
4. For $i = 2, \ldots, d$, choose the test set $x_i = \arg\max_{x\in C} \min_{j=1,..,i-1} m_j(x)$ and compute hitting time function m_i with respect to $D_i = \{x_i\}$.
5. Normalize the functions m_i and partition C by clustering the points $m(x) = (m_1(x), ..., m_d(x)) \in \mathbb{R}^d$ with $x \in C$, using k-means for example.

When d is unknown or not specified beforehand, the number of clusters present may be estimated on-the-fly by tracking the values of the objective function used for selecting test sets in step 4. Once the state space is sufficiently covered with test sets, any candidate test vertex will have a short hitting time to at least one existing test set. Therefore, the values achieved by further test vertices typically drop sharply when the number of test sets exceeds the number of clusters. For undirected networks, we consider this to be a hitting time analogon to the spectral gap.

When a reference measure μ_r is not available, the metastability parameter α is unknown and there is no obvious way to identify the transition region a priori, one can run this procedure for $\mathcal{T} = \varnothing$ and separate cores from the transition region by applying fuzzy C-means on the generated point set $\{m(x)\}_{x\in\mathbb{X}}$ instead of k-means. Fuzzy C-means yields d-dimensional affiliation vectors $u_x(i)$ for each vertex x, which are non-negative and normalized as $\sum_{i=1}^{d} u_x(i) = 1\ \forall x$. Using these, vertices can be mapped to a core C_{i^*}, if their affiliation vectors take values close to one at index i^*, that is

$$C_{i^*} = \{x \mid u_x(i^*) > 1 - \epsilon\}$$

for some small threshold $1 \gg \epsilon > 0$. Then, cores can be expanded to metastable sets based on a more generous threshold $1 > \delta > \epsilon$. For any $\delta < 0.5$, such assignments to metastable sets are unambiguous, due to the normalization of assignment vectors. Larger threshold values may lead to overlapping modules.

A full decomposition $\mathcal{D} = \{A_1, \ldots, A_d\}$ can be derived by assigning each vertex to the set, to which it has the highest affiliation. The maxima might not be unique for transition vertices, so the decomposition is in general not unambiguous. Alternatively, one can apply k-means instead of fuzzy C-means clustering on the hitting time vectors to create a full decomposition.

Generalized network clustering algorithm

Given those amendments, one can modify the aforementioned blueprint for the general case where the transition region can not be determined in advance. This version does not require prior knowledge of a reference measure or metastable timescales, so it may be applied to directed networks from any source.

The benefit of being more generally applicable comes at a cost, however. As the transition region is not detected beforehand, there is no guarantee that each test set sits in a cluster core, making it harder to guess the number of clusters from the values of the objective function as explained above.

For undirected networks, this would translate to steadily declining eigenvalues showing no clear spectral gap, giving no hint on the number of clusters to identify. Under suitable conditions, when communication between clusters is slow, while mixing within each cluster is much faster, the generalized algorithm has proven to be robust towards the choice of d and the randomized selection of the first test set. Thus, we refer to an implementation of this blueprint during the runtime and accuracy evaluation following in the next section.

1. Choose d, the desired number of modules.
2. Consider a test set $D_1 = \{x_1\}$ consisting of a randomly chosen single state $x_1 \in \mathbb{X}$ and compute hitting time function m_1.
3. For $i = 2, \ldots, d$, choose test set $x_i = \arg\max_{x \in \mathbb{X}} \min_{j=1,..,i-1} m_j(x)$ and compute hitting time function m_i with respect to $D_i = \{x_i\}$.
4. Normalize the functions m_i and decompose $\mathbb{X}$ by clustering the points $m(x) = (m_1(x), \ldots, m_d(x)) \in \mathbb{R}^d$, using fuzzy C-means.

5.4 Numerical examples

In this section, we will employ the algorithmic strategy proposed above to identify metastable sets of random walks on networks, see Section 5.1. First, let us illustrate the relation between spectral and hitting time based approach, using an undirected network that is shown in Figure 5.2, where the induced random walk is a reversible Markov chain.

5.4.1 Reversible chain

The first step of the algorithm identifies the transition region as depicted in Figure 5.2. In Figure 5.3, the values of the three dominant eigenvectors of the random walk on the

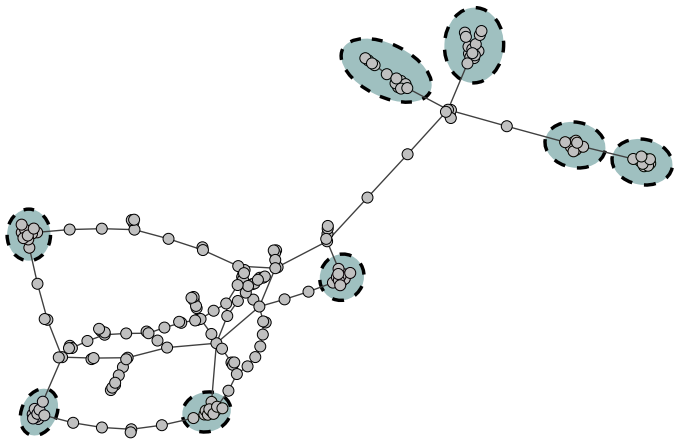


Figure 5.2: Artificially created undirected network with 8 strongly connected modules and a loosely connected transition region. Cluster cores indicated by shaded areas. [66]

core sets $C = \mathbb{X} \setminus \mathcal{T}$ are shown in comparison to three hitting time functions with respect to test sets generated by the algorithm. We restrict these functions to the non-transient region $C = \mathbb{X} \setminus \mathcal{T}$ that we want to partition.

As expected from Theorem 5.2.1 and Theorem 5.3.3, eigenvectors and hitting time functions are almost constant on the metastable sets. In both cases, the k-means method described in the previous section delivers the clustering that is visualized in Figure 5.4.

5.4.2 Non-reversible chain

The second example should underline the difficulty of extending spectral methods to non-reversible cases, where loops and cyclic structures are present. Consider the directed network that is illustrated in Figure 5.5.

The network consists of three cycles of length 5 that are connected via one transition vertex. Additionally, the two upper cycles in Figure 5.5 share one undirected edge. Applying the proposed method based on hitting time functions for finding metastable clusters on the network, we get the result for two and three sets illustrated in Figure 5.6.

Calculating the values

$$\hat{\lambda}_i = 1 - \frac{1}{\mathbb{E}_{A_i}[\tau(A_i^c)]}$$

from Theorem 5.3.2 for the three cycles that had been identified as clusters as shown in Figure 5.6, we obtain

i	$\hat{\lambda}_i$	color	line shape
1	0.7529	red	dashed
2	0.7529	green	dotted
3	0.8696	blue	straight

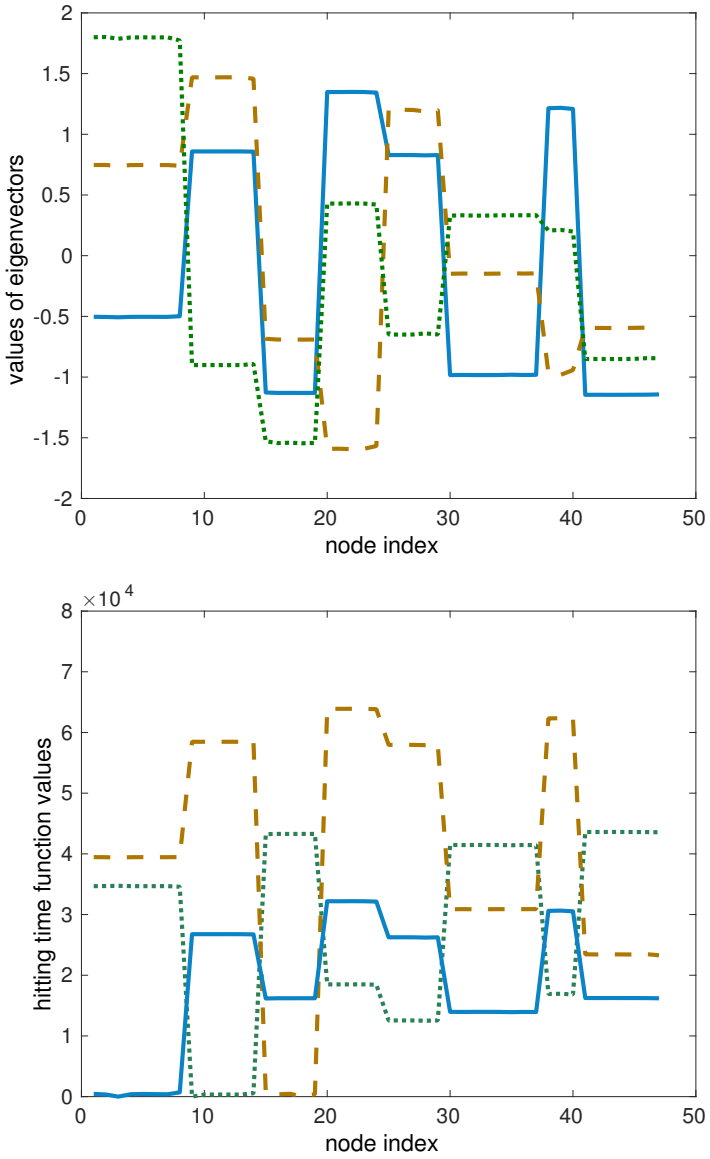


Figure 5.3: Upper panel: The 3 dominant non-trivial eigenvectors, shown in the reduced state space $C = \mathbb{X} \setminus \mathcal{T}$. Lower panel: 3 hitting time functions in C as generated by the algorithm. [66]

On the other hand, eigenvalues of the random walk on the network are complex and eigenvalues with the largest real part are given by

i	λ_i
1	1
2	0.7156 + 0.0053i
3	0.7156 - 0.0053i
4	0.3593 + 0.8579i
5	0.3593 - 0.8579i
6	0.3260 + 0.6374i
7	0.3260 - 0.6374i

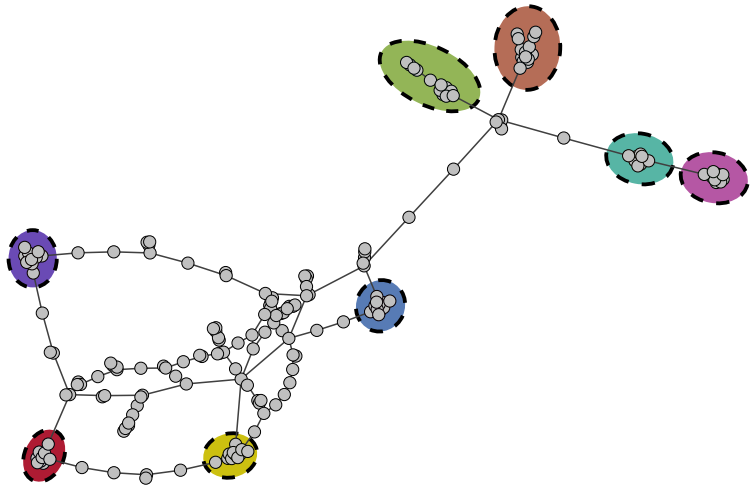


Figure 5.4: Modules of an undirected network identified by applying k-means on an embedding of either 3 eigenvectors or 3 hitting time functions into $\mathbb{R}^3$. Both results coincide. [66]

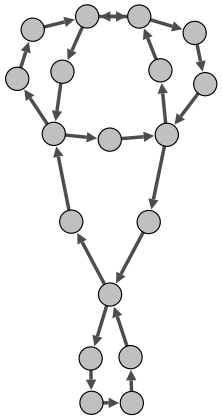


Figure 5.5: Directed network with 3 cycles that are metastable for the standard random walk. This is a toy example designed to highlight the detection of cyclic structures via eigenvectors and hitting time functions. [66]

That is, we have one complex conjugate pair of eigenvalues that has a dominating real part not far from 1; this agrees with the metastability of the three cycles. The following table shows the values of the real part of the dominant eigenvector on the cluster and the transition region found by the hitting time approach.

Real part of first non-trivial eigenvector on blue cluster

x	1	2	3	4	5
$Re(u_2(x))$	-0.0534	-0.0196	0.0035	0.0185	-0.1011

Real part of first non-trivial eigenvector on red cluster

x	6	7	8	9	10
$Re(u_2(x))$	0.3340	0.2626	0.2039	0.1561	0.1177

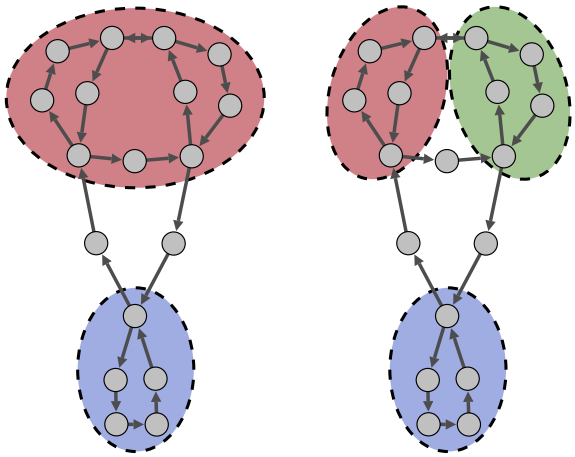


Figure 5.6: Two possible clusterings computed using hitting time functions. Left: Clustering into two sets. Right: Clustering into three sets. [66]

Real part of first non-trivial eigenvector on green cluster

x	11	12	13	14	15
$Re(u_2(x))$	-0.1596	-0.1362	-0.2329	-0.2092	-0.1843

Real part of first non-trivial eigenvector on transition region

x	16	17	18
$Re(u_2(x))$	-0.2357	-0.2357	-0.2357

We can see that the sign structure of the real part of the eigenvector does not coincide with the clustering [20, 36]. In particular, the eigenvector changes its sign on the most metastable set (blue in Figure 5.6) and assumes its minimal value $\min(Re(u_2(x))) = -0.2357$ in the transition region, whereas values close to 0 are expected for reversible Markov chains.

This example emphasizes the difficulty of spectral clustering for non-reversible Markov chains that show a strongly cyclic behavior within and between metastable sets. On the other hand, as motivated by the theoretical results and demonstrated by this example, the approach based on hitting time analysis is well applicable in such cases.

5.4.3 Dynamical system example

We turn toward an application example that stems from simulated trajectories of a well-documented dynamical system: the Arnold–Beltrami–Childress flow [1]. It describes an incompressible fluid flow in three dimensions that may exhibit chaotic trajectories, with parameters chosen as follows:

$$\begin{aligned}\dot{x} &= \sqrt{3}\sin(2\pi z) + \cos(2\pi y)\\ \dot{y} &= \sqrt{2}\sin(2\pi x) + \sqrt{3}\cos(2\pi z)\\ \dot{z} &= \sin(2\pi y) + \sqrt{2}\cos(2\pi x)\end{aligned}$$

In order to employ numerical analysis, the state space, a three-dimensional torus, was divided into 50-by-50-by-50 uniform boxes and transitions from one box to another were counted during the observation period according to Ulam's method [77]. This yields a 125000-by-125000 transition matrix with a highly sparse structure – less than 4 million of its 15.625 billion entries are non-zero.

The directed network that describes such an observation of a dynamical system can not have more than one sink and one source by construction. When there is a non-trivial metastable structure in the system and observations are sufficiently long to catch large-timescale transitions between metastable sets, the system is highly likely to eventually return to a state it has visited before. Cutting off the beginning of a given trajectory until the first recurrent state, one can even guarantee, for sufficiently long observations, that the resulting recurrence network is entirely free of sinks and sources.

With the given set of parameters, the flow has non-trivial metastable sets [24]. Froyland and co-workers identified these sets numerically as plateaus of the eigenvector corresponding to the second-largest eigenvalue of the approximated generator of the continuous flow [33]. As Figure 5.7 shows, we were able to reproduce those results with the hitting time approach.

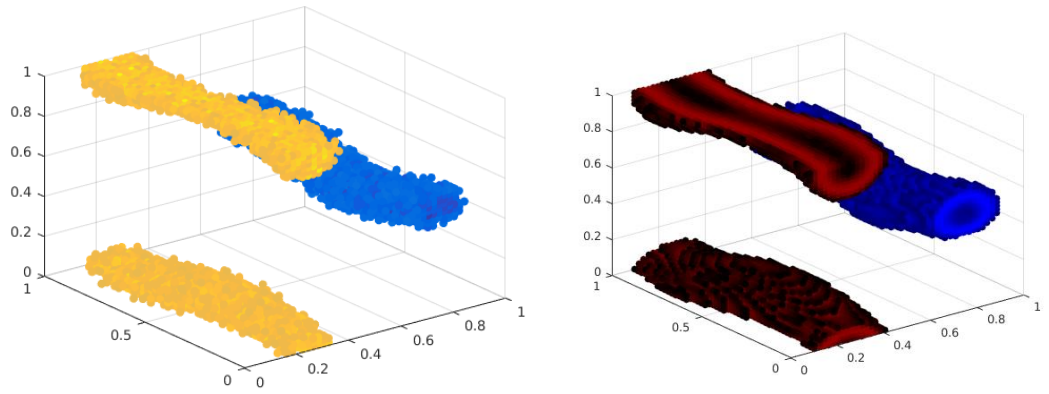


Figure 5.7: Arnold–Beltrami–Childress flow on the 3-D torus – Cores of two metastable sets
Left: identified via 2nd eigenfunction of the generator
Right: identified via hitting time functions

The massive size and sparse structure of transition matrices arising from high-resolution state space discretization call for methods that exploit sparsity and scale well with large networks, where spectral decompositions require prohibitive computational effort. The example shows that recurrence networks of dynamical systems can be a suitable area of application for hitting time based cluster analysis.

5.5 Analysis of computational properties

The algorithm as detailed in Section 5.3.4 was tested on both artificial and real data to measure its runtime for very large sparse networks and evaluate the accuracy of its results. Note that few other methods are applicable to directed networks and that test networks need to be free of sinks and sources to apply the hitting time approach without further adaptations. These restrictions severely limited the available options, both for the choice

of example networks as well as reference methods for an impartial comparison.

5.5.1 Building directed test networks

In order to check the predicted scaling behavior of the runtime of the presented algorithm, a number of directed networks of similar structure spanning orders of magnitudes in size were needed. These networks should exhibit a clear cluster structure, preferably one that is known beforehand to serve as reference in the evaluation of accuracy of results.

A. Lancichinetti and S. Fortunato have proposed benchmark algorithms for community detection algorithms that are weighted, directed and highly modular by construction [42]. We observed that networks generated by their method typically exhibit sinks and sources, rendering them unsuitable to test the proposed algorithm, as the hitting time calculation does not converge if there are no directed paths from every potential starting vertex to the target set. As the algorithm relies on mean hitting times of a random walker process to identify the module structure of a network, the iterative solvers approximating those hitting times must converge to finite values. Thus, strong connectivity of the adjacency matrix is a required condition to allow for a free choice of test sets. That is: any two vertices must be connected by directed paths in both directions.

The Network Analysis Toolbox provided by the Strategic Engineering Research Group, MIT[1], contains a function that generates random modular graphs. Given a desired number of vertices, number of clusters, overall probability of attachment and a module bias value, the function divides the set of vertices into equally sized clusters – up to integer rounding, then inserts edges randomly, biased towards pairs of vertices within the same module. M.E.J. Newman and M. Girvan have used such networks to benchmark their community identification method [53]. We adapted the function to insert directed edges instead. The function also saves full information about the module structure, allowing us to evaluate the quality of calculated clusterings by comparing output of our algorithm to the actual set of modules. Figure 5.8 shows a directed network generated with this method and the network's adjacency matrix, which highlights a pronounced block structure, each block corresponding to one of the three clusters.

We have generated networks spanning from 50 to 25600 vertices with an average degree of 10 and module bias 0.9, which ensures that, on average, 90% of all edges are placed within modules. Since edges are placed randomly, it is not assured that networks generated by this method are strongly connected. To ensure compatibility of test networks to Algorithm 5.3.4, we checked every candidate network for connectivity and replaced those which did not meet the condition. The closer the module bias is to the maximum of 100%, the more likely generated networks are unconnected. However, the lower this value is chosen, the less pronounced the module structure is. We found empirically that a module bias of 0.9, paired with an average degree of 10, rarely yields unconnected networks for the size range mentioned above.

These test networks are not meant to be considered true-to-life models of real-world networks. For instance, they do not exhibit power-law distributed vertex degrees, have no transition regions and show no variation in cluster size. They are designed only for the purpose of testing the runtime of algorithms on similarly built networks over a broad

[1] strategic.mit.edu

range of vertex counts and to check their precision in a best-case setting with strongly pronounced, non-overlapping modules. To evaluate the accuracy of our algorithm on more realistic problems, we turn to real-world benchmarks in the next section.

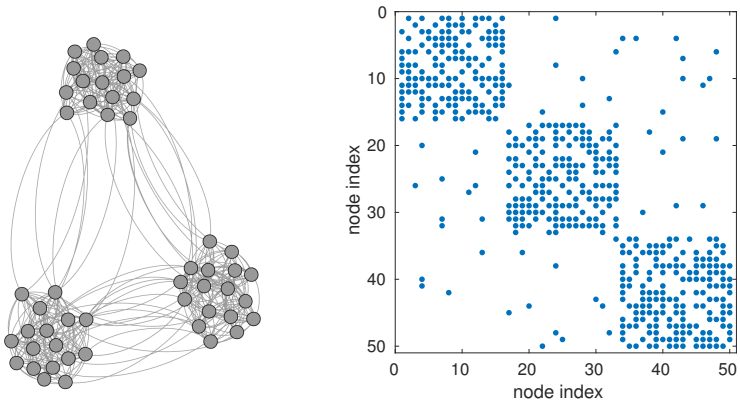


Figure 5.8: Left: Exemplary random network on 50 vertices with three clusters, an average vertex degree of 10 and module bias of 0.9. Right: Visualization of the non-zero structure of the associated adjacency matrix.

5.5.2 Accuracy evaluation

To evaluate results of the presented method on a given network, we use a reference classification and a measure of similarity. In the case of randomly generated test networks as explained in Section 5.5.1, the actual clustering is known by construction.

For some popular test networks, such as V. Krebs' graph of political books often copurchased at the online retailer Amazon according to its own statistics, a subjective classification is given by manual assignment – here based on M. E. J. Newman's analysis of book descriptions and costumer reviews [40]. Since political alignment of the books can not be perfectly expressed by just a binary mapping, Newman assigned some of the books to a 'neutral' category. Figure 5.9 shows the fuzzy clustering computed with Algorithm 5.3.4 in comparison to the manual classification by Newman.

We selected Normalized Mutual Information (NMI) as a measure of similarity between ground truth and a clustering found by the algorithm to evaluate its accuracy. NMI quantifies congruence of information in the sense of Shannon's information theory and the normalization penalizes the generation of too many clusters [73]. Note that there are competing definitions of NMI, depending on how mutual information is 'normalized' [47]. We adhere to the version presented in the textbook of C. D. Manning et al., which divides mutual entropy of two clusterings by the average entropy of the individual clusterings [46]. Another option we explored was the Rand index [59]. We did not observe striking discrepancies between the two ratings during our test runs and chose to stick with NMI, as it has been proposed and used before to benchmark accuracy of community detection algorithms [39, 45].

A series of test runs of Algorithm 5.3.4 has been performed on randomly generated networks (see Section 5.5.1 for construction details). All test networks share an expected average degree of 10, a module bias value of 0.9 and consist of three clusters of equal size (up to integer rounding) like the example network in Figure 5.8. The smallest network

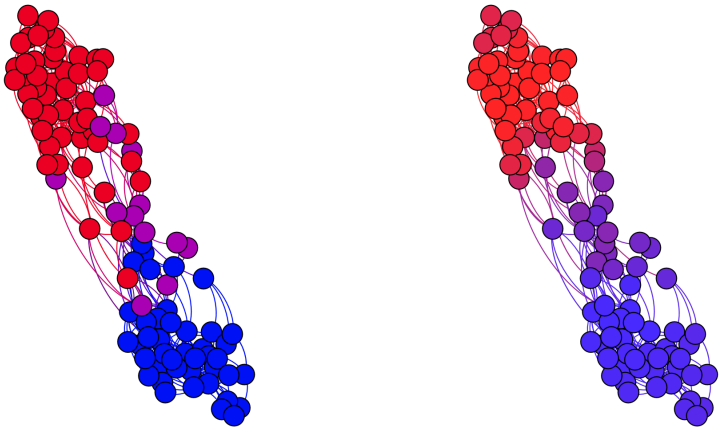


Figure 5.9: Undirected graph of books about US politics co-purchased on Amazon.com, compiled by V. Krebs [40].
Left: Reference clustering – conservative (red), neutral (blue), liberal (violet). Right: Computed fuzzy decomposition into 2 clusters in shades from red to blue.

has 50 vertices, the bigger ones grow successively by a factor of 2, such that the largest network holds 25600 vertices. The affiliation vectors generated by the algorithm were then exploited to assign vertices to clusters, as described in Section 5.3.4, yielding a hard clustering of the state space which was finally compared to the actual cluster structure, determined by construction.

The upper plot in Figure 5.10 shows the accuracy of results from these test runs. For the nine smaller test networks on up to 12800 vertices, both the NMI value and Rand index remain above 80%. For even larger networks of this type, accuracy of results apparently derails. A possible cause for this behavior is the fixed average degree leading to ever sparser clusters, the more total vertices the network (and thus each cluster) has. This leads to increasingly smaller relative differences between average hitting times for vertices in- and outside the cluster where the target set is situated. Fluctuations of hitting time vectors within clusters at some point overshadow the differences between clusters, rendering the identification of clusters based on hitting time vector plateaus impossible. It remains to be investigated if that behavior shows also for large networks from real world examples.

5.5.3 Total runtime

We assume given networks to be sparse in the sense that the average vertex degree is independent of the number of vertices and therefore the expected number of non-zero-entries of adjacency matrices is a fixed multiple of the total number of vertices. Furthermore, let the number of clusters and the maximum number of iteration steps for approximating solutions to linear systems be fixed independent of total vertex count.

Runtime of the algorithm presented in Section 5.3.4 is dominated by the calculation of hitting time vectors, which loops over the desired number of clusters, selects test sets depending on previously computed hitting times and requires the iterative solution of a sparse linear system to approximate mean hitting times. Assuming that the linear solver chosen to solve system 5.3.2 converges to the vector of expected hitting times w.r.t. any chosen test set using at most a fixed number of iteration steps and the computational

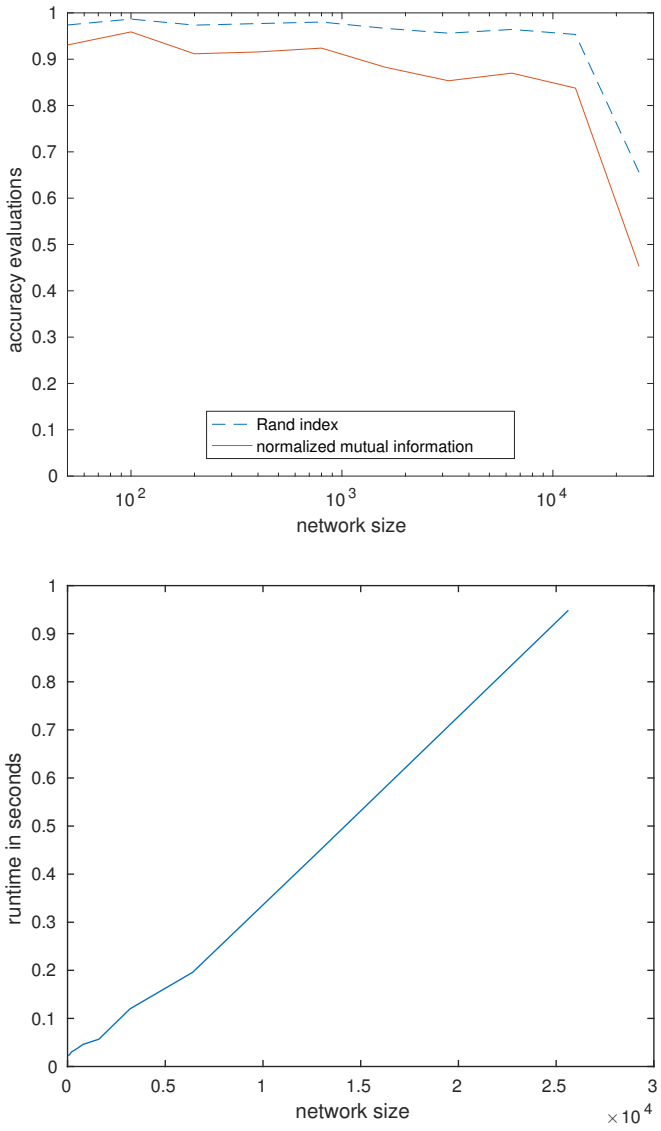


Figure 5.10: Results of test runs showing Algorithm 5.3.4 with fuzzy C-means clustering applied to test networks of varying size, randomly generated according to Section 5.5.1.
Upper panel: Accuracy of computed clusterings.
Lower panel: Total runtime of the algorithm on a single workstation.

effort of each iteration step is scaling linearly with the network size, time complexity of hitting time calculations also grows linearly with the number of vertices.

The final step, which derives affiliation vectors from the hitting time vectors, depends on which method is chosen for geometric clustering. In numerical experiments on randomly generated networks as described in the previous section, we have considered four options:

- fuzzy C-means [26, 3],
- a self-made implementation of k-means,
- an accelerated implementation of k-means presented by C. Elkan [28] and
- a naive clustering procedure

The latter method simply assigns each vertex v to the cluster holding the test set of shortest average hitting time, starting from v, that is: The state space is fully partitioned into supposedly metastable sets $A_1, \ldots, A_d$ defined as

$$A_i = \{x \in \mathbb{X} \mid m_i(x) = \min_{j=1,\ldots,d} m_j(x)\}.$$

If the hitting times do not show a unique minimum, one of the minimizing indices is chosen arbitrarily. Our implementation selects the first among these indices.

The time complexity of k-means and fuzzy C-means are known to scale linearly with the number of points and their dimension, see [34] for a comparative analysis of both algorithms. In our case, the number of points to be clustered is given by the number of vertices of the network and their dimension is determined by the number of selected test sets. Elkan's implementation of k-means performs even faster, making use of the triangle inequality to speed up the calculation of distances, which dominates the runtime. The lower plot in Figure 5.10 shows the total runtime of Algorithm 5.3.4, including geometric clustering using fuzzy C-means, which we found to be the slowest of the above mentioned options. We observe a linear relation between the number of vertices in the network and total calculation time. Note that the test networks are highly sparse by construction, which is exploited by our implementation of the algorithm. Non-sparse networks of the same size demand more computational effort.

5.6 Summary

In this chapter, we discussed the relation of metastable sets of a Markov chain to spectral properties of its transition matrix on the one hand and to exit times from these sets on the other. Both characterizations lead to numerical approaches of identifying metastable sets which coincide when both are well-defined. While the spectral approach is restricted to reversible Markov chains, the newly proposed hitting time approach may also be applied to non-reversible chains.

This generalization is highly useful for the use case of detecting clusters in networks, as the associated random walk of a directed network is, in general, not reversible. Furthermore, if a Markov chain is reversible and both approaches are applicable, we highlighted computational advantages of the hitting time approach.

As a network decomposition tool, the hitting time approach is restricted to strongly connected networks. This limitation is a consequence of the necessity for the random walker process to be ergodic, such that ensemble averages may be derived from long trajectories. Future work should expand the method to a more general setting, for example by making use of a modified random walker process actively avoiding sinks or performing random teleportation steps to ensure mixing behavior.

We illustrated the theoretical results and algorithmic ideas on several network examples. The first example was an undirected network where results from both approaches coincided, underlining the close relation between the two different characterizations of metastable sets in the reversible case.

The second example was a directed network with loops and strongly cyclic structures

where the spectral decomposition does not allow for identifying clusters in the usual way due to complex-valued eigenvectors without visible plateaus, whereas clusters can be detected from appropriate hitting time functions as demonstrated.

Third, a network based on simulated trajectories of a chaotic flow with metastable sets shows that directed networks with strong connectivity may arise from dynamical system use cases, when observations are sufficiently long and proper mixing is taking place.

Next, an example from book sales data showed how a fuzzy clustering derived from hitting time functions allows to capture information about overlapping modules and transition regions, where binary assignments are not suitable to account for soft transitions.

Finally, a series of test runs on random directed networks verified that the computational cost of the presented cluster identification algorithm using hitting times grows linearly with the size of sparse networks. While random walk based module identification algorithms have been proposed and examined before, the author is not aware of existing methods exploiting hitting times to identify modules of directed networks with such low computational effort.

6 | Conclusion

The detection of metastable subsets of a system's state space has been a recurring task in different fields such as bio-informatics, computational chemistry or social sciences, as these sets may correspond to functional subunits, stable states or homogeneous groups of people inside the investigated system. Modeling such systems as graphs allows to identify metastable sets in the original system as clusters of the model graph, enabling the usage of graph theoretical tools and methods.

When relations or interactions within the modeled system are not symmetrical or when a recurrence network is built from chronologically ordered observation data, the obtained graph may contain directed edges. Moreover, state space discretization of increasingly high resolution – which can be necessary for the analysis of continuous dynamical systems – potentially lead to very sparse, yet giant networks. These use cases require clustering methods adapted to large and sparse networks, suitable for directed and undirected graphs alike.

Following the idea to identify network clusters based on a dynamical system point-of-view, we have focused on a selection of approaches based on random walks. We examined existing methods suitable for undirected networks and presented a novel approach for clustering directed networks. Three key features of our approach are:

1. Edge directions and weights are taken into account.
2. It does not resort to a symmetrization step which reduces the problem to clustering an undirected network.
3. When a sparse network is given, that sparsity is exploited to reduce computational effort substantially.

The proposed method is based on hitting times with respect to specific target sets. The hitting time of a set is the expected number of steps a random walk would take before entering that set and obviously depends on where the random walk starts. The hitting time of a set can thus be seen as a function on the vertex set, assigning a non-negative number to each vertex that describes a notion of distance to the target set, induced by the dynamics of a standard random walk.

These hitting time functions can be computed easily by solving a linear system which is sparse whenever the network under consideration has a sparse structure. Metastable sets of the random walk are then identified as plateaus of the hitting time functions, that is:

groups of vertices with similar values. By applying k-means or fuzzy-C-means algorithms to these hitting time functions, hard or fuzzy clusterings of the network can be derived.

We investigated how to select target sets of hitting time computations in order to explore the whole network structure efficiently, gaining as much new information as possible from each additional hitting time function and estimating the number of clusters on-the-fly. Also, we evaluated the runtime of the proposed clustering algorithm, using a family of randomly generated sparse networks. These tests verified that computational effort of the algorithm scales linearly with the size of sparse networks, keeping the average vertex degree and the number of clusters fixed.

For undirected networks, where spectral decompositions as well as hitting time based clustering are applicable, both approaches aim at identifying metastable regions of standard random walks that correspond to network clusters and both deliver comparable results. Clustering via hitting times also works on strongly connected directed networks and takes into consideration that the orientation of edges has an impact on how quickly a state space subset may be entered or left and should not be neglected in the context of metastable sets.

The hitting time approach can thus be seen as an extension of random walk based network clustering to directed networks with strong connectivity. Information contained in hitting time vectors can be re-used to identify transient areas of a network, to generate hard or fuzzy decompositions and to quantify distances between clusters or between individual vertices, directly induced by the dynamics of a random walk. Therefore, analyzing hitting times provides new possibilities to reveal key structures in large modular networks.

Outlook

Some aspects remain open for investigation, regarding theory and implementation of random walk approaches. The prime question is how to deal with directed networks containing sinks and sources, which is the general case. The fundamental issue with random walks on such networks is, that without further adaptation, they never return to sources and get stuck in sinks eventually. With every trajectory ending in some sink, there are no more transitions between metastable sets in the long run – ergodicity is violated. The basic assumption that network clusters correspond to metastable sets of a random walk can not be ensured and the computation of hitting times does not converge for some target set.

The limitation to strongly connected networks is a necessary consequence of the requirement that random walks need to be aperiodic and ergodic to ensure that long trajectories show transitions between metastable sets. For directed graphs without strong connectivity, it is necessary to either adapt the random walk or regularize the graph such that the required conditions are met. In contrast, random walks on connected undirected graph are guaranteed to be ergodic and reversible, so this important subclass is eligible for random walk methods using hitting times.

Inspired by Google's PageRank algorithm [11], one way to ensure ergodicity of a random walk process, is to implement, in each step, a small chance of jumping to any other vertex in the network. In practice, this could be achieved by adding an ε slightly greater than 0 to every entry of the network's adjacency matrix, then normalizing it row-wise to obtain the random walk's transition matrix. This perturbed random walk, running on the original

network, acts the same way as a standard random walk on a perturbed network with additional, artificial edges. The graph theoretic interpretation is that edges were added to the original network between any two vertices, each new edge weighted by ε.

Such a regularization of the network – or adaptation of the random walk from another point-of-view – would allow to apply random walk based methods to any graph. Limiting the impact of this perturbation by setting ε small enough, clusters of the original network remain clusters after perturbation. Sinks of the original network are no longer dead ends, but allow the random walk to re-enter at a random point. Still, several difficulties need to be resolved: Even with newly-introduced edges carrying extremely low weights, each perturbation might have an undesired impact on the modular structure of a network. It needs to be verified if metastability of a decomposition is preserved under perturbation for sufficiently small ε. Additionally, a perturbation as described above fills every entry of the transition matrix with non-zeros, sacrificing the computational benefits of a sparse network structure.

To achieve a similar effect without losing sparsity altogether, a similar regularization could be considered, where one artificial vertex is added to the network, which is connected through bi-directional ε-weighted edges to all other vertices. This special vertex would not be considered a part of the network, but rather a 'resting state', from where a random walker can re-enter the network at a random point. Apart from one additional intermediate step, this has the same effect as adding a small chance to move directly to any other vertex as described before. However, instead of filling the transition matrix with near-zero entries, the matrix only needs to be extended by one additional column and row of non-zero entries. Sparse networks remain sparse with this modification, as each vertex gains only one additional neighbor.

These adaptations are currently in the early stages of research and our experimental implementation, improvised to test the clustering algorithm on networks without strong connectivity, needs further refinement. It is not yet fully understood what impact such a modification has on hitting time vectors in the limit of ε going to zero. Future research in this direction might lead to the development of hitting time based methods applicable to any directed graph.

Bibliography

[1] V. I. Arnold, *Sur la topologie des écoulements stationnaires des fluides parfaits*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 15–18.

[2] V. Betz and S. L. Roux, *Multi-scale metastable dynamics and the asymptotic stationary distribution of perturbed markov chains*, Stochastic Processes and their Applications, (2016).

[3] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[4] H. Bisgin, N. Agarwal, and X. Xu, *Investigating homophily in online social networks*, in Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01, WI-IAT '10, Washington, DC, USA, 2010, IEEE Computer Society, pp. 533–536.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment, 10 (2008), p. 10008.

[6] C. Bose, G. Froyland, C. González-Tokman, and R. Murray, *Ulam's method for lasota–yorke maps with holes*, SIAM Journal on Applied Dynamical Systems, 13 (2014), pp. 1010–1032.

[7] A. Bovier, M. Eckhoff, V. Gayrard, and M. Klein, *Metastability and small eigenvalues in Markov chains*, J. Phys. A, 33 (2000), pp. L447–L451.

[8] A. Bovier, M. Eckhoff, V. Gayrard, and M. Klein, *Metastability and low lying spectra in reversible Markov chains*, Comm. Math. Phys., 228 (2002), pp. 219–255.

[9] G. R. Bowman, V. S. Pande, and F. Noé, eds., *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*, vol. 797 of Advances in Experimental Medicine and Biology, Springer, 2014.

[10] P. Bremaud, *Markov chains, Gibbs fields, Monte Carlo simulation, and queues*, vol. 31 of Texts in Applied Mathematics, Springer-Verlag, New York, NY, USA, 1999.

[11] S. Brin and L. Page, *The anatomy of a large-scale hypertextual web search engine*, Computer networks and ISDN systems, 30 (1998), pp. 107–117.

[12] G. E. Cho and C. D. Meyer, *Aggregation/disaggregation methods for nearly uncoupled Markov chains*, Technical Report NCSU no. 041600-0400, North Carolina State University, (1999).

[13] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, *Generalized louvain method for community detection in large networks*, in Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on, Nov 2011, pp. 88–93.

[14] M. Dellnitz, G. Froyland, and O. Junge, *The algorithms behind GAIO - set oriented numerical methods for dynamical systems*, in Ergodic theory, analysis, and efficient simulation of dynamical systems, Springer, 2001, pp. 145–174.

[15] M. Dellnitz and O. Junge, *On the approximation of complicated dynamical behavior*, SIAM J. Num. Anal., 36 (1999), pp. 491–515.

[16] M. Dellnitz and O. Junge, *Set oriented numerical methods for dynamical systems*, in Handbook of dynamical systems, North-Holland, Amsterdam, 2002, pp. 221–264.

[17] M. Dellnitz and R. Preis, *Congestion and almost invariant sets in dynamical systems*, in Symbolic and Numerical Scientific Computation, LNCS, 2003, pp. 183–209.

[18] J. Demmel, I. Dumitriu, and O. Holtz, *Fast linear algebra is stable*, Numerische Mathematik, 108 (2007), pp. 59–91.

[19] P. Deuflhard, W. Huisinga, A. Fischer, and C. Schütte, *Identification of almost invariant aggregates in reversible nearly uncoupled markov chains*, Linear Algebra and its Applications, 315 (2000), pp. 39 – 59.

[20] P. Deuflhard, W. Huisinga, A. Fischer, and C. Schütte, *Identification of almost invariant aggregates in reversible nearly uncoupled Markov chains*, Lin. Alg. Appl., 315 (2000), pp. 39–59.

[21] P. Deuflhard and M. Weber, *Robust Perron cluster analysis in conformation dynamics*, Linear Algebra and its Applications, 161 (2005). 398 Special issue on matrices and mathematical biology.

[22] I. S. Dhillon, Y. Guan, and B. Kulis, *Weighted graph cuts without eigenvectors a multilevel approach*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1944–1957.

[23] N. Djurdjevac, *Methods for analyzing complex networks using random walker approaches*, PhD thesis, Freie Universität Berlin, 2012.

[24] T. Dombre, U. Frisch, J. M. Greene, M. Hénon, A. Mehr, and A. M. Soward, *Chaotic streamlines in the abc flows*, Journal of Fluid Mechanics, 167 (1986), pp. 353–391.

[25] M. D. Donsker and S. R. S. Varadhan, *On the principal eigenvalue of second-order elliptic differential operators*, Communications on Pure and Applied Mathematics, 29 (1976), pp. 595–621.

[26] J. C. Dunn, *A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters*, Journal of Cybernetics, 3 (1973), pp. 32–57.

[27] W. E and E. Vanden-Eijnden, *Transition-path theory and path-finding algorithms for the study of rare events*, Annual Review of Physical Chemistry, 61 (2010), pp. 391–420.

[28] C. Elkan, *Using the triangle inequality to accelerate k-means*, in Proceedings of the Twentieth International Conference on Machine Learning: August 21-24, 2003, Washington, DC, USA, T. Fawcett and N. Mishra, eds., Menlo Park, California, 2003, The AAAI Press, pp. 147–153.

[29] S. N. Ethier and T. G. Kurtz, *Markov Processes - Characterization and Convergence*, John Wiley and Sons, 2005.

[30] D. Fritzsche, V. Mehrmann, D. B. Szyld, and E. Virnik, *An SVD approach to identifying meta-stable states of Markov chains*, Electronic Transactions on Numerical Analysis, 29 (2008), pp. 46–69.

[31] F. G. Frobenius, *Über matrizen aus nicht negativen elementen*, Sitzungsberichte der Preuss. Akad. Wiss., (1912), pp. 456–477.

[32] G. Froyland, *Ulam's method for random interval maps*, Nonlinearity, 12 (1999), p. 1029.

[33] G. Froyland, O. Junge, and P. Koltai, *Estimating long-term behavior of flows without trajectory integration: The infinitesimal generator approach*, SIAM Journal on Numerical Analysis, 51 (2013), pp. 223–247.

[34] S. Ghosh and S. K. Dubey, *Comparative analysis of k-means and fuzzy c-means algorithms*, International Journal of Advanced Computer Science and Applications, 4 (2013), pp. 35–39.

[35] D. F. Gleich, *Hierarchical directed spectral graph partitioning.* Information Networks, Stanford University, Final Project, 2005, 2006.

[36] W. Huisinga and B. Schmidt, *Metastability and dominant eigenvalues of transfer operators*, in Advances in Algorithms for Macromolecular Simulation, C. Chipot, R. Elber, A. Laaksonen, B. Leimkuhler, A. Mark, T. Schlick, C. Schütte, and R. Skeel, eds., Lecture Notes in Computational Science and Engineering 49, Springer, 2002.

[37] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal of Scientific Computing, 20 (1998), pp. 359–392.

[38] M. M. Kessler, *Bibliographic coupling between scientific papers*, American Documentation, 14 (1963), pp. 10–25.

[39] G. Keziban Orman, V. Labatut, and H. Cherifi, *On Accuracy of Community Structure Discovery Algorithms*, ArXiv e-prints, (2011).

[40] V. Krebs, *unpublished.* http://www.orgnet.com/.

[41] S. Lafon and A. B. Lee, *Diffusion maps and coarse-graining: A unified framework for dimensionality reduction*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006), pp. 1393–1403.

[42] A. Lancichinetti and S. Fortunato, *Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities*, ArXiv e-prints, (2009).

[43] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, *The yeast cell-cycle network is robustly designed*, Proceedings of the National Academy of Sciences of the United States of America, 101 (2004), pp. 4781–4786.

[44] T. Li, W. E, and E. Vanden-Eijnden, *Optimal partition and effective dynamics of complex networks*, Proc. Nat. Acad. Sci., 105 (2008).

[45] F. D. Malliaros and M. Vazirgiannis, *Clustering and community detection in directed networks: A survey*, Physics Reports, 533 (2013), pp. 95 – 142. Clustering and Community Detection in Directed Networks: A Survey.

[46] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, Cambridge, 2008. available via http://www-nlp.stanford.edu/IR-book/.

[47] A. F. McDaid, D. Greene, and N. Hurley, *Normalized Mutual Information to evaluate overlapping community finding algorithms*, ArXiv e-prints, (2011).

[48] E. Meerbach, C. Schütte, and A. Fischer, *Eigenvalue bounds on restrictions of reversible nearly uncoupled Markov chains*, Lin. Alg. Appl., 398 (2005).

[49] M. Meila, *Comparing clusterings by the variation of information*, in Learning Theory and Kernel Machines, B. Schölkopf and M. Warmuth, eds., vol. 2777 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2003, pp. 173–187.

[50] M. Meila and W. Pentney, *Clustering by weighted cuts in directed graphs*, in SDM, SIAM, 2007.

[51] C. D. Meyer, *Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems*, SIAM Rev., 31 (1989).

[52] R. Murray, *Ulam's method for some non-uniformly expanding maps*, Discrete and Continuous Dynamical Systems, 26 (2010), pp. 1007–1018.

[53] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E, 69 (2004), p. 026113.

[54] O. Nikodym, *Sur une généralisation des intégrales de M. J. Radon. Errata et remarques.*, Fundam. Math., 15 (1930), p. 358.

[55] J. D. Noh and H. Rieger, *Random walks on complex networks*, Phys. Rev. Lett., 92 (2004), p. 118701.

[56] W. Pentney and M. Meila, *Spectral clustering of biological sequence data*, in Proceedings of the 20th national conference on Artificial intelligence - Volume 2, AAAI'05, AAAI Press, 2005, pp. 845–850.

[57] O. Perron, *Zur theorie der matrices*, Mathematische Annalen, 64 (1907), pp. 248–263.

[58] G. Peters and J. H. Wilkinson, *The Calculation of Specified Eigenvectors by Inverse Iteration*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1971, pp. 418–439.

[59] W. M. Rand, *Objective criteria for the evaluation of clustering methods*, Journal of the American Statistical Association, 66(336) (1971), pp. 846–850. JSTOR 2284239.

[60] M. Rosvall and C. T. Bergstrom, *Maps of random walks on complex networks reveal community structure*, Proceedings of the National Academy of Sciences of the United States of America, 105 (2008), pp. 1118–1123.

[61] Y. Saad, *Iterative methods for sparse linear systems*, vol. 82, siam, 2003.

[62] Y. Saad and M. H. Schultz, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on scientific and statistical computing, 7 (1986), pp. 856–869.

[63] M. Sarich, *Projected Transfer Operators*, PhD thesis, Free University Berlin, 2011.

[64] M. Sarich, N. Djurdjevac, S. Bruckner, T. O. F. Conrad, and C. Schütte, *Modularity revisited: A novel dynamics-based concept for decomposing complex networks*, Journal of Computational Dynamics, (2013).

[65] M. Sarich, F. Noe, and C. Schütte, *On the approximation quality of markov state models*, Multiscale Modeling and Simulation, 8 (2010), pp. 1154–1177.

[66] M. Sarich and C. Schütte, *Utilizing hitting times for finding metastable sets in non-reversible markov chains*, Tech. Rep. 14-32, ZIB, Takustr.7, 14195 Berlin, 2014.

[67] V. Satuluri and S. 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, KDD '09, New York, USA, 2009, ACM, pp. 737–746.

[68] V. Satuluri and S. Parthasarathy, *Symmetrizations for clustering directed graphs*, in Proceedings of the 14th International Conference on Extending Database Technology, EDBT/ICDT '11, New York, NY, USA, 2011, ACM, pp. 343–354.

[69] C. Schütte, *Conformational dynamics: Modelling, theory, algorithm, and application to biomolecules*, 1998. Habilitation Thesis.

[70] C. Schütte, A. Fischer, W. Huisinga, and P. Deuflhard, *A direct approach to conformational dynamics based on hybrid Monte Carlo*, J. Comput. Phys., 151 (1999), pp. 146–168. Special Issue on Computational Biophysics.

[71] C. Schütte and M. Sarich, *Metastability and Markov State Models in Molecular Dynamics: Modeling, Analysis, Algorithmic Approaches*, vol. 24 of Courant Lecture Notes, American Mathematical Society, December 2013.

[72] E. Seneta, *Non-negative matrices and Markov chains*, Springer-Verlag, New York, NY, USA, 1981.

[73] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, Univ of Illinois Press, 1949.

[74] H. Small, *Co-citation in the scientific literature: A new measure of the relationship between documents*, Journal of the American Society for Information Science, (1973), pp. 24:265–269.

[75] D. Stroock, *An Introduction to Markov Processes*, Graduate Texts in Mathematics, Springer, 2005.

[76] B. Tadic, *Exploring complex graphs by random walks*, in Modeling Complex Systems, P. L. Garrido and J. Marro, eds., vol. 661 of AIP Conference Proceedings, American Institute of Physics, 2003, p. 24.

[77] S. M. Ulam, *A collection of mathematical problems*, no. 8 in Interscience Tracts in Pure and Applied Mathematics, Interscience Publishers, 1960.

[78] S. van Dongen, *MCL - Graph Clustering by Flow Simulation*, PhD thesis, University of Utrecht, 2000.

[79] D. Verma and M. Meila, *A comparison of spectral clustering algorithms*, U Washington Tech Report, 1 (2003), pp. 1–18.

Zusammenfassung

Ein häufig wiederkehrendes Problem bei der Arbeit mit großen Systemen unterschiedlicher Natur, etwa zellularen, elektrischen oder sozialen Netzwerken, ist die Erkennung funktionaler Untereinheiten, die eine hohe Eigenständigkeit vom Rest des Systems auszeichnet.

Dazu bietet es sich an, das zu untersuchende System als Graphen zu modellieren und diesen mittels Clusteranalyse in Teile zu zerlegen, die zueinander wenige Verbindungen aufweisen, relativ zur Kantendichte innerhalb der Cluster. Ein bekannter Ansatz dazu basiert auf erinnerungslosen Zufallsprozessen, die sich mit jedem Zeitschritt von einer Ecke entlang einer anliegenden Kante zu einer zufällig gewählten benachbarten Ecke fortbewegen. Die Motivation hinter diesem Ansatz ist die Vorstellung, dass eine Zerlegung der Eckenmenge des Graphen in disjunkte, „metastabile“ Mengen existiert, die nahezu stabile Zustände des Prozesses beschreiben, also nur sehr selten gewechselt werden.

In dieser Dissertation wird die Theorie von Graphen und Zufallsprozessen eingeführt, einige bekannte Methoden der Clusteranalyse zusammen gefasst und schließlich eine neue Methode vorgeschlagen und analysiert, die ebenfalls auf Zufallsprozessen basiert, aber ohne die aufwändige Berechnung von Eigenvektoren auskommt. Stattdessen werden „Stoppzeiten“ genutzt, die beschreiben, wie lange ein Zufallsprozess, der auf dem Netzwerk wandert, im Durchschnitt benötigt, um eine fest gewählte Zielmenge zu erreichen.

Es wird aufgezeigt, dass diese neue Methode einen klaren Vorteil in der Skalierung des Rechenaufwandes zeigt, da die nötigen Rechenschritte zur Berechnung der Stoppzeiten im ungünstigsten Fall quadratisch mit der Zahl der Ecken ansteigt. Für dünn besetzte Graphen, in denen jede Ecke eine feste Zahl an Nachbarn nicht übersteigen darf, zeigen hier präsentierte numerische Experimente an großen Zufallsgraphen, dass die nötige Rechenzeit sogar näherungsweise linear mit der Eckenzahl der Graphen ansteigt. Im Gegensatz dazu skalieren klassische Zerlegungsalgorithmen, die eine Berechnung von Eigenvektoren erfordern, deutlich höher, nämlich mehr als quadratisch, mit der Eckenzahl.

Die präsentierte neue Methode unterliegt jedoch einer Einschränkung: Die eingesetzten Stoppzeiten haben nur dann garantiert endliche Werte, wenn die gewählte Zielmenge sich von jeder Ecke des Graphen tatsächlich erreichen lässt und der Prozess in endlicher Zeit fast sicher jede Ecke wieder besucht. Das setzt voraus, dass der Graph stark zusammenhängend, insbesondere frei von Quellen und Senken sein muss. Dies schließt viele gerichtete Graphen aus, stellt aber immer noch eine echte Erweiterung gegenüber der Klasse der ungerichteten Graphen dar. Ansätze zur Überwindung dieser Einschränkung werden in der Auswertung skizziert.