Motif-Based Spectral Clustering of Weighted Directed Networks

William George Underwood

Department of Statistics

University of Oxford

Part C Dissertation in Mathematics & Statistics

Trinity 2019

Abstract

Clustering is an essential technique for network analysis, with applications in a diverse range of fields. Although spectral clustering is a popular and effective method, it fails to consider higher-order structure and can perform poorly on directed networks. We aim to address these shortcomings by exploring motif-based spectral clustering methods. We present new matrix formulae for motif adjacency matrices, and a motif-based approach for clustering bipartite networks. Comprehensive experimental results from both synthetic and real data demonstrate the effectiveness of our techniques on a variety of networks. We conclude that motif-based spectral clustering is a valuable tool for analysis of directed and bipartite weighted networks, which is also scalable and easy to implement.

Contents

List of Figures

List of Tables

Abbreviations

ARI	Adjusted Rand Index
BSBM	Bipartite stochastic block model
DSBM	Directed stochastic block model
MAM	Motif adjacency matrix
Ncut	Normalised cut

Chapter 1

Introduction

Networks are ubiquitous in modern society; from the internet and online blogs to protein interactions and human migration, we are surrounded by inherently connected structures [24]. The mathematical and statistical analysis of networks is therefore a very important area of modern research, with applications in a diverse range of fields including biology [3], chemistry [23], physics [32] and sociology [1].

A common problem in network analysis is that of *clustering* [35]. Network clustering refers to the division of a network into several parts so that objects in the same part are similar, while those in different parts are dissimilar.

Spectral methods for network clustering have a long and successful history [7,15,19], and have become increasingly popular in recent years. These techniques exhibit many attractive properties including generality, ease of implementation and scalability [41].

However traditional spectral methods have shortcomings, particularly involving their inability to consider higher-order network structures [6], and their insensitivity to edge direction [12]. These weaknesses can lead to unsatisfactory results, especially when working with directed networks. Motif-based spectral methods have proven more effective for clustering directed networks on the basis of higher-order structures [39], with the introduction of the *motif adjacency matrix* (MAM).

In this dissertation we will explore motif-based spectral clustering methods with a focus on addressing these shortcomings for weighted directed networks. Our main contributions include a collection of new matrix-based formulae for MAMs on weighted directed networks, and a motif-based approach for clustering bipartite networks. We also provide comprehensive experimental results both from synthetic data (stochastic block models) and from real-world network data.

Dissertation layout

In Chapter 2 we describe our graph-theoretic framework which provides a natural model for real-world weighted directed networks. We define motifs and instances, and then state and prove new matrix-based formulae for MAMs. In Chapter 3 we provide a summary of random-walk spectral clustering and discuss techniques for cluster extraction and evaluation. We state the algorithms for both traditional and motif-based spectral clustering. In Chapter 4 we introduce directed stochastic block models (DSBMs), a family of generative models for directed networks, and evaluate the performance of motif-based clustering both on synthetic data and on real data (US Political Blogs network, US Migration network). In Chapter 5 we propose a motif-based approach for clustering bipartite graphs and introduce bipartite stochastic block models (BSBMs), a family of generative models for bipartite networks. We again provide experimental results both on synthetic data and on real data (American Revolution network, Unicode Languages network). Finally in Chapter 6 we present our conclusions, along with a discussion about limitations and potential extensions of our work.

Chapter 2

Graphs and Motifs

We describe our graph-theoretic framework for network analysis and give matrix-based formulae for motif adjacency matrices (MAMs). In Section 2.1 we outline basic concepts relating to graphs and motifs. In Section 2.2 we define the adjacency and indicator matrices of a graph. In Section 2.3 we introduce MAMs and present the main results of this chapter, Proposition 2.1 and Proposition 2.2.

2.1 Graph definitions

Graph notation is notoriously inconsistent in the literature [46], so we begin by giving all of the relevant notation and definitions.

Definition 2.1 (Graphs)**.** A *graph* is a triple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ where $\mathcal{V}$ is the *vertex set*, $\mathcal{E} \subseteq \{(i,j) : i,j \in \mathcal{V}, i \neq j\}$ is the *edge set* and $W \colon \mathcal{E} \to (0,\infty)$ is the *weight map*.

Remark. We consider weighted directed graphs without self-loops or multiple edges. We can extend to undirected graphs by replacing undirected edges with bidirectional edges. Where it is not relevant, we may sometimes omit the weight map W.

Definition 2.2 (Underlying edges)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. Its *underlying edges* are $\bar{\mathcal{E}} := \big\{\{i,j\} : (i,j) \in \mathcal{E}\big\}$.

Definition 2.3 (Subgraphs)**.** A graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is a *subgraph* of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (write $\mathcal{G}' \leq \mathcal{G}$) if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. It is an *induced subgraph* (write $\mathcal{G}' < \mathcal{G}$) if further $\mathcal{E}' = \mathcal{E} \cap (\mathcal{V}' \times \mathcal{V}')$.

Definition 2.4 (Connected components)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. The *connected components* of $\mathcal{G}$ are the partition $\mathcal{C}$ generated by the transitive closure of the relation $\sim$ on $\mathcal{V}$ defined by $i \sim j \iff \{i,j\} \in \bar{\mathcal{E}}$. We say $\mathcal{G}$ is (weakly) *connected* if $|\mathcal{C}| = 1$.

Definition 2.5 (Graph isomorphisms)**.** A graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is *isomorphic* to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (write $\mathcal{G}' \cong \mathcal{G}$) if there exists a bijection $\phi\colon \mathcal{V}' \to \mathcal{V}$ with $(u, v) \in \mathcal{E}' \iff \big(\phi(u), \phi(v)\big) \in \mathcal{E}$. An isomorphism from a graph to itself is called an *automorphism*.

Definition 2.6 (Motifs and anchor sets)**.** A *motif* is a pair $(\mathcal{M}, \mathcal{A})$ where $\mathcal{M} = (\mathcal{V}_\mathcal{M}, \mathcal{E}_\mathcal{M})$ is a connected graph with $\mathcal{V}_\mathcal{M} = \{1, \ldots, m\}$ for some small $m \geq 2$, and $\mathcal{A} \subseteq \mathcal{V}_\mathcal{M}$ with $|\mathcal{A}| \geq 2$ is an *anchor set*. If $\mathcal{A} \neq \mathcal{V}_\mathcal{M}$ we say the motif is *anchored*, and if $\mathcal{A} = \mathcal{V}_\mathcal{M}$ we say it is *simple*.

Remark. Anchor sets [6] specify which rôles vertices play in the motif, and are crucial for defining the collider and expander motifs given in Section 5.1.1. When an anchor set is not given, it is assumed that the motif is simple. Figure 2.1 shows all simple motifs (up to isomorphism) on at most three vertices.

Definition 2.7 (Instances)**.** Let $\mathcal{G}$ be a graph and $(\mathcal{M}, \mathcal{A})$ a motif. We say that $\mathcal{H}$ is a *functional instance* of $\mathcal{M}$ in $\mathcal{G}$ if $\mathcal{M} \cong \mathcal{H} \leq \mathcal{G}$. We say that $\mathcal{H}$ is a *structural instance* of $\mathcal{M}$ in $\mathcal{G}$ if $\mathcal{M} \cong \mathcal{H} < \mathcal{G}$.

Definition 2.8 (Anchored pairs)**.** Let $\mathcal{G}$ be a graph and $(\mathcal{M}, \mathcal{A})$ a motif. Suppose $\mathcal{H}$ is an instance of $\mathcal{M}$ in $\mathcal{G}$. Define the *anchored pairs of the instance* $\mathcal{H}$ as

$$\mathcal{A}(\mathcal{H}) := \big\{\{\phi(i), \phi(j)\} : i, j \in \mathcal{A},\ i \neq j,\ \phi \text{ is an isomorphism from } \mathcal{M} \text{ to } \mathcal{H}\big\}.$$

Remark. Example A.1 demonstrates functional and structural instances. Note that $\{i, j\} \in \mathcal{A}(\mathcal{H})$ if and only if $\mathcal{H}$ appears in $\mathcal{G}$ as an instance of $\mathcal{M}$ with $i \neq j$ co-appearing in the image of $\mathcal{A}$ under isomorphism. The motivation for this is that clustering methods should avoid separating vertices which appear as an anchored pair.

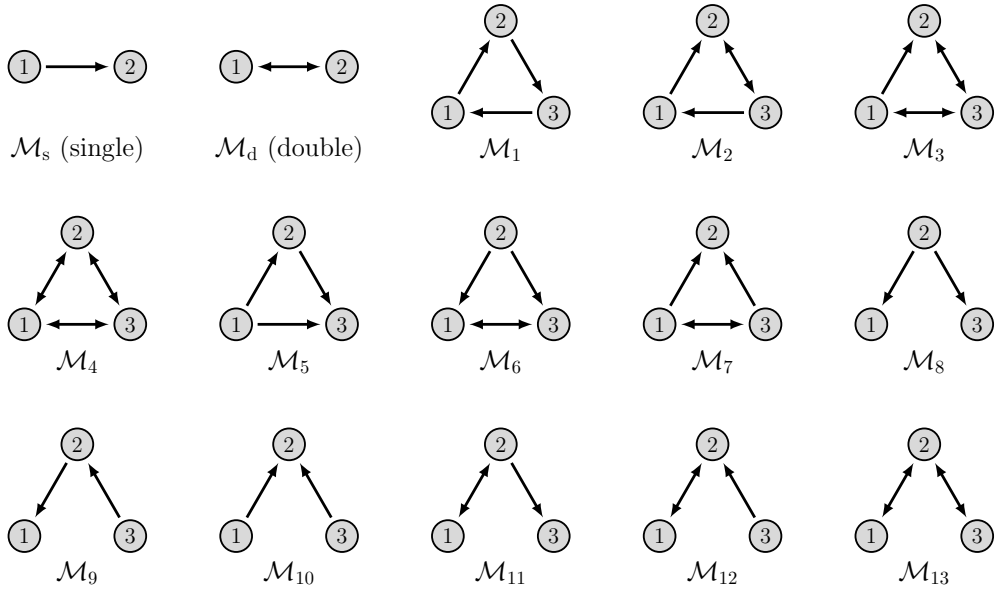


Figure 2.1: All simple motifs on at most three vertices

2.2 Adjacency and indicator matrices

Adjacency matrices provide a useful data structure for representing graphs and have many uses in calculating graph properties [5]. We define several variants of the adjacency matrix, which appear in Proposition 2.1 and Table B.1.

Definition 2.9 (Adjacency matrices)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ be a graph with vertex set $\mathcal{V} = \{1, \dots, n\}$. The *adjacency matrix, single-edge adjacency matrix* and *double-edge adjacency matrix* of $\mathcal{G}$ are respectively the $n \times n$ matrices

$$\begin{aligned} G_{ij} &:= W((i,j))\, \mathbb{I}\{(i,j) \in \mathcal{E}\}\,, \\ (G_{\mathrm{s}})_{ij} &:= W((i,j))\, \mathbb{I}\{(i,j) \in \mathcal{E} \text{ and } (j,i) \notin \mathcal{E}\}\,, \\ (G_{\mathrm{d}})_{ij} &:= \big(W((i,j)) + W((j,i))\big)\, \mathbb{I}\{(i,j) \in \mathcal{E} \text{ and } (j,i) \in \mathcal{E}\}\,. \end{aligned}$$

Definition 2.10 (Indicator matrices)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ be a graph with vertex set $\mathcal{V} = \{1, \dots, n\}$. The *indicator matrix, single-edge indicator matrix, double-edge indicator matrix, missing-edge indicator matrix* and *vertex-distinct indicator matrix* of $\mathcal{G}$ are respectively the $n \times n$ matrices

$$\begin{aligned} J_{ij} &:= \mathbb{I}\{(i,j) \in \mathcal{E}\}\,, \\ (J_{\mathrm{s}})_{ij} &:= \mathbb{I}\{(i,j) \in \mathcal{E} \text{ and } (j,i) \notin \mathcal{E}\}\,, \\ (J_{\mathrm{d}})_{ij} &:= \mathbb{I}\{(i,j) \in \mathcal{E} \text{ and } (j,i) \in \mathcal{E}\}\,, \\ (J_{0})_{ij} &:= \mathbb{I}\{(i,j) \notin \mathcal{E} \text{ and } (j,i) \notin \mathcal{E} \text{ and } i \neq j\}\,, \\ (J_{\mathrm{n}})_{ij} &:= \mathbb{I}\{i \neq j\}\,. \end{aligned}$$

2.3 Motif adjacency matrices

The central object in motif-based spectral clustering is the *motif adjacency matrix* (MAM) [6], which serves as a similarity matrix for spectral clustering (Chapter 3). We provide here our main results: Proposition 2.1 gives a computationally useful formula for MAMs, and Proposition 2.2 gives a complexity analysis of this formula.

2.3.1 Definitions

Definition 2.11 (Motif adjacency matrices)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ be a graph with n vertices and let $(\mathcal{M}, \mathcal{A})$ be a motif. The *functional* and *structural motif adjacency matrices* (MAMs) of $(\mathcal{M}, \mathcal{A})$ in $\mathcal{G}$ are respectively the $n \times n$ matrices

$$M_{ij}^{\text{func}} := \frac{1}{|\mathcal{E}_{\mathcal{M}}|} \sum_{\mathcal{M} \cong \mathcal{H} \leq \mathcal{G}} \mathbb{I}\big\{\{i,j\} \in \mathcal{A}(\mathcal{H})\big\} \sum_{e \in \mathcal{E}_{\mathcal{H}}} W(e)\,,$$
$$M_{ij}^{\text{struc}} := \frac{1}{|\mathcal{E}_{\mathcal{M}}|} \sum_{\mathcal{M} \cong \mathcal{H} < \mathcal{G}} \mathbb{I}\big\{\{i,j\} \in \mathcal{A}(\mathcal{H})\big\} \sum_{e \in \mathcal{E}_{\mathcal{H}}} W(e)\,.$$

Remark. Example A.2 gives a simple illustration of calculating an MAM. When $W \equiv 1$ and $\mathcal{M}$ is simple, the (functional or structural) MAM entry M_{ij} $(i \neq j)$ simply counts the (functional or structural) instances of $\mathcal{M}$ in $\mathcal{G}$ containing i and j. When $\mathcal{M}$ is not simple, M_{ij} counts only those instances with anchor sets containing both i and j. MAMs are always symmetric, since the only dependency on (i,j) is via the unordered set $\{i,j\}$.

2.3.2 Computation

In order to state Propositions 2.1 and 2.2, we need one more definition.

Definition 2.12 (Anchored automorphism classes)**.** Let $(\mathcal{M}, \mathcal{A})$ be a motif. Let $S_{\mathcal{M}}$ be the set of permutations on $\mathcal{V}_{\mathcal{M}} = \{1, \ldots, m\}$ and define the *anchor-preserving permutations* $S_{\mathcal{M},\mathcal{A}} = \{\sigma \in S_{\mathcal{M}} : \{1, m\} \subseteq \sigma(\mathcal{A})\}$. Let $\sim$ be the equivalence relation defined on $S_{\mathcal{M},\mathcal{A}}$ by: $\sigma \sim \tau \iff \tau^{-1}\sigma$ is an automorphism of $\mathcal{M}$. Finally the *anchored automorphism classes* are the quotient set $S^{\sim}_{\mathcal{M},\mathcal{A}} := S_{\mathcal{M},\mathcal{A}} \,/\, \sim$.

Proposition 2.1 (MAM formula)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ be a graph with vertex set $\mathcal{V} = \{1, \ldots, n\}$ and let $(\mathcal{M}, \mathcal{A})$ be a motif on m vertices. Then for any $i, j \in \mathcal{V}$ and with $k_1 = i$, $k_m = j$, the functional and structural MAMs of $(\mathcal{M}, \mathcal{A})$ in $\mathcal{G}$ are given by*

$$M_{ij}^{\text{func}} = \frac{1}{|\mathcal{E}_{\mathcal{M}}|} \sum_{\sigma \in S^{\sim}_{\mathcal{M},\mathcal{A}}} \sum_{\{k_2, \ldots, k_{m-1}\} \subseteq \mathcal{V}} J_{\mathbf{k},\sigma}^{\text{func}}\; G_{\mathbf{k},\sigma}^{\text{func}}\,, \tag{1}$$
$$M_{ij}^{\text{struc}} = \frac{1}{|\mathcal{E}_{\mathcal{M}}|} \sum_{\sigma \in S^{\sim}_{\mathcal{M},\mathcal{A}}} \sum_{\{k_2, \ldots, k_{m-1}\} \subseteq \mathcal{V}} J_{\mathbf{k},\sigma}^{\text{struc}}\; G_{\mathbf{k},\sigma}^{\text{struc}}\,, \tag{2}$$

where

$$\mathcal{E}_{\mathcal{M}}^{0} := \{(u,v) : 1 \leq u < v \leq m : (u,v) \notin \mathcal{E}_{\mathcal{M}}, (v,u) \notin \mathcal{E}_{\mathcal{M}}\}\,,$$
$$\mathcal{E}_{\mathcal{M}}^{\text{s}} := \{(u,v) : 1 \leq u < v \leq m : (u,v) \in \mathcal{E}_{\mathcal{M}}, (v,u) \notin \mathcal{E}_{\mathcal{M}}\}\,,$$
$$\mathcal{E}_{\mathcal{M}}^{\text{d}} := \{(u,v) : 1 \leq u < v \leq m : (u,v) \in \mathcal{E}_{\mathcal{M}}, (v,u) \in \mathcal{E}_{\mathcal{M}}\}\,,$$

are respectively the missing edges, single edges and double edges of $\mathcal{E}_{\mathcal{M}}$*, and*

$$\begin{aligned}
J^{\text{func}}_{\mathbf{k},\sigma} &:= \prod_{\mathcal{E}^0_{\mathcal{M}}} (J_{\text{n}})_{k_{\sigma u},k_{\sigma v}} \quad \prod_{\mathcal{E}^{\text{s}}_{\mathcal{M}}} J_{k_{\sigma u},k_{\sigma v}} \prod_{\mathcal{E}^{\text{d}}_{\mathcal{M}}} (J_{\text{d}})_{k_{\sigma u},k_{\sigma v}} \,, \\
G^{\text{func}}_{\mathbf{k},\sigma} &:= \sum_{\mathcal{E}^{\text{s}}_{\mathcal{M}}} G_{k_{\sigma u},k_{\sigma v}} \quad + \sum_{\mathcal{E}^{\text{d}}_{\mathcal{M}}} (G_{\text{d}})_{k_{\sigma u},k_{\sigma v}} \,, \\
J^{\text{struc}}_{\mathbf{k},\sigma} &:= \prod_{\mathcal{E}^0_{\mathcal{M}}} (J_0)_{k_{\sigma u},k_{\sigma v}} \quad \prod_{\mathcal{E}^{\text{s}}_{\mathcal{M}}} (J_{\text{s}})_{k_{\sigma u},k_{\sigma v}} \prod_{\mathcal{E}^{\text{d}}_{\mathcal{M}}} (J_{\text{d}})_{k_{\sigma u},k_{\sigma v}} \,, \\
G^{\text{struc}}_{\mathbf{k},\sigma} &:= \sum_{\mathcal{E}^{\text{s}}_{\mathcal{M}}} (G_{\text{s}})_{k_{\sigma u},k_{\sigma v}} + \sum_{\mathcal{E}^{\text{d}}_{\mathcal{M}}} (G_{\text{d}})_{k_{\sigma u},k_{\sigma v}} \,.
\end{aligned}$$

Proof. See Proof A.1. □

Proposition 2.2 (Complexity of MAM formula)**.** *Suppose that* $m \leq 3$*, and the adjacency matrix* G *of* $\mathcal{G}$ *is known. Then computing adjacency and indicator matrices and calculating an MAM using Equations* (1) *and* (2) *in Proposition 2.1 involves at most 18 matrix multiplications, 22 entry-wise multiplications and 21 additions of (typically sparse)* $n \times n$ *matrices.*

Proof. See Proof A.2. □

Hence for motifs on at most three vertices and with sparse adjacency matrices, Proposition 2.1 gives a fast and parallelisable matrix-based procedure for computing MAMs. In practice, additional symmetries of the motif often allow computation with even fewer matrix operations, demonstrated in Example A.3.

A list of such MAM formulae for all simple motifs on at most three vertices (up to isomorphism), as well as for the *collider* and *expander* motifs (Section 5.1.1), is given in Table B.1. These formulae are generalisations of those stated in Table S6 in the supplementary materials for [6], in an incomplete list of only *structural* MAMs of *unweighted* graphs. Note that the functional MAM formula for the two-vertex motif $\mathcal{M}_{\text{s}}$ yields the symmetrised adjacency matrix $M = G + G^\top$ which is used for traditional spectral clustering (Section 3.1). The question of whether to use functional or structural MAMs for motif-based spectral clustering will be addressed in Section 3.5.2.

Chapter 3

Spectral Clustering

We provide a summary of traditional random-walk spectral clustering and show how it applies to motif-based clustering. This chapter mostly follows the relevant sections in the tutorial by U. Von Luxburg [41], which provides further explanations and proofs. In Section 3.1 we give an overview of the spectral clustering procedure. In Section 3.2 we define the random-walk Laplacian and state some of its useful properties (Proposition 3.1). In Section 3.3 we introduce normalised cut (Ncut) as an objective function for graph partitioning. In Section 3.4 we explore methods of extracting clusters from $\mathbb{R}^l$-valued embeddings, and in Section 3.5 we present the algorithms for both traditional and motif-based random-walk spectral clustering.

3.1 Overview of spectral clustering

Suppose $x_1, \ldots, x_n$ are data points with some associated symmetric similarity matrix M with $M_{ij} = \text{similarity}(x_i, x_j)$. The intuitive aim of clustering is to find a partition $\mathcal{P}_1, \ldots, \mathcal{P}_k$ of $\{x_1, \ldots, x_n\}$ which places similar points in the same group and dissimilar points in different groups. Where other methods such as k-means++ [4] and GMM clustering [16] demand some further structure on x_i (such as taking values in $\mathbb{R}^l$), spectral clustering has no such requirements.

In the context of *undirected* graph clustering, the data points are the vertices of the graph, and a similarity matrix is provided by the graph's adjacency matrix G. To cluster directed graphs, the adjacency matrix must first be symmetrised, traditionally by the transformation $M = G + G^\top$ [29]. This symmetrisation ignores information about edge direction and higher-order structures; and can lead to poor performance, as will be seen in Section 4.1.2.

Spectral clustering consists of two steps. Firstly, eigendecomposition of a Laplacian matrix embeds the vertices into $\mathbb{R}^l$. The k clusters are then extracted from this space.

3.2 Graph Laplacians

The Laplacians of an undirected graph are a family of matrices which play a central rôle in spectral clustering. While many different graph Laplacians are available, we focus in this dissertation on just the *random-walk Laplacian*, for reasons concerning objective functions, consistency and computation [41, 42].

Definition 3.1. Let $\mathcal{G}$ be an undirected graph with (symmetric) adjacency matrix G. The *random-walk Laplacian matrix* of $\mathcal{G}$ is

$$L_{\text{rw}} := I - D^{-1}G$$

where I is the identity and $D_{ii} := \sum_j G_{ij}$ is the diagonal matrix of weighted degrees.

Remark. $D^{-1}G$ is the transition matrix of a random walk on the vertex set $\mathcal{V}$ where the probability of the transition $v_i \to v_j$ is proportional to G_{ij}.

Proposition 3.1 (Properties of the random-walk Laplacian)**.** *L_{rw} is positive semi-definite with eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_n$. The multiplicity k of the eigenvalue 0 is equal to the number of connected components $\mathcal{P}_1, \ldots, \mathcal{P}_k$ of $\mathcal{G}$. The eigenspace of the eigenvalue 0 is spanned by the indicator vectors on these components; $\mathbb{I}_{\mathcal{P}_1}, \ldots, \mathbb{I}_{\mathcal{P}_k}$.*

Proof. See [41]. □

3.3 Graph cuts

Graph cuts provide objective functions which we seek to minimise while clustering the vertices of a graph. We look at the normalised cut and its relationship with the random-walk Laplacian.

Definition 3.2. Let $\mathcal{G}$ be a graph. Let $\mathcal{P}_1, \ldots, \mathcal{P}_k$ be a partition of $\mathcal{V}$. Then the *normalised cut* [37] of $\mathcal{G}$ with respect to $\mathcal{P}_1, \ldots, \mathcal{P}_k$ is

$$\text{Ncut}_{\mathcal{G}}(\mathcal{P}_1, \ldots, \mathcal{P}_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{\text{cut}(\mathcal{P}_i, \bar{\mathcal{P}}_i)}{\text{vol}(\mathcal{P}_i)}$$

where $\text{cut}(\mathcal{P}_i, \bar{\mathcal{P}}_i) := \sum_{u \in \mathcal{P}_i,\, v \in \mathcal{V} \setminus \mathcal{P}_i} G_{uv}$ and $\text{vol}(\mathcal{P}_i) := \sum_{u \in \mathcal{P}_i} D_{uu}$.

Remark. More desirable partitions have a lower Ncut value; the numerators penalise partitions which cut a large number of heavily weighted edges, and the denominators penalise partitions which have highly imbalanced cluster sizes.

It can be shown [41] that minimising Ncut over partitions $\mathcal{P}_1, \ldots, \mathcal{P}_k$ is equivalent to finding the cluster indicator matrix $H \in \mathbb{R}^{n \times k}$ minimising

$$\mathrm{Tr}\big(H^\top (D - G) H\big)$$

subject to

$$H_{ij} = \mathrm{vol}(\mathcal{P}_j)^{-\frac{1}{2}}\, \mathbb{I}\{v_i \in \mathcal{P}_j\}\,, \qquad (\dagger)$$

$$H^\top D H = I\,.$$

Solving this problem is in general NP-hard [43]. However, by dropping the constraint (†) and applying the Rayleigh Principle [28], we find that the solution to this relaxed problem is that H contains the first k eigenvectors of L_{rw} as columns [41]. In practice, to find k clusters it is often sufficient to use only the first $l < k$ eigenvectors of L_{rw}.

3.4 Cluster extraction

Once Laplacian eigendecomposition has been used to embed the data into $\mathbb{R}^l$, the clusters may be extracted using a variety of methods. We propose k-means++ and eigenvector sweep as two appropriate techniques.

3.4.1 k-means++

k-means++ [4] is a popular clustering algorithm for data in $\mathbb{R}^l$. It aims to minimise the within-cluster sum of squares, based on the standard Euclidean metric on $\mathbb{R}^l$. This makes it a reasonable candidate for clustering spectral data, since the Euclidean metric corresponds to notions of 'diffusion distance' in the original graph [31].

3.4.2 Eigenvector sweep

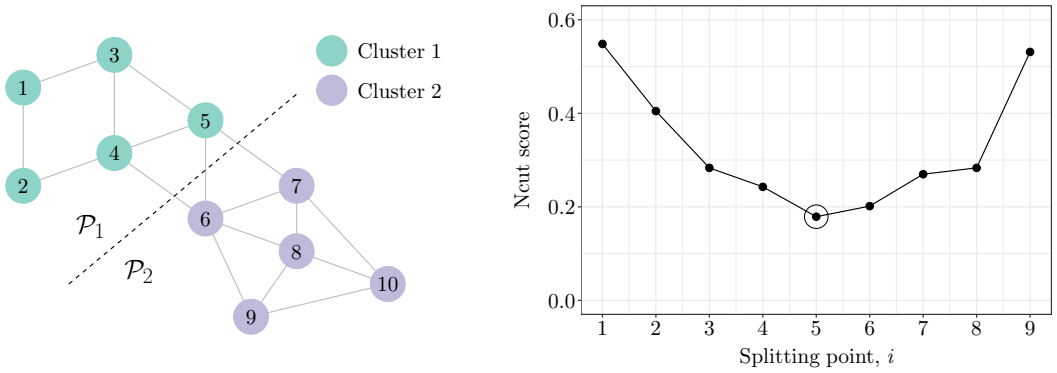
Eigenvector sweep (Algorithm 3.1) [37] offers a more principled technique for cluster extraction when $k = 2$ clusters are required, and a single eigenvector (usually the second eigenvector of L_{rw}) is available. It works by sorting the eigenvector and selecting a splitting point to minimise the Ncut score of the partition generated.

Algorithm 3.1: Eigenvector sweep

Input: Graph $\mathcal{G}$, eigenvector x
Output: Partition $\mathcal{P}_1, \mathcal{P}_2$

function `EigenvectorSweep`$(\mathcal{G}, x)$**:**
  $\hat{x} \leftarrow$ `sort`(x)
  $\text{Score}_{\text{best}} \leftarrow \infty$
  for i **in** $1, \ldots, n-1$ **do**
    $\mathcal{P} \leftarrow \{\hat{x}_1, \ldots \hat{x}_i\}$
    $\text{Score} \leftarrow \text{Ncut}_{\mathcal{G}}(\mathcal{P}, \mathcal{V} \setminus \mathcal{P})$
    if $\text{Score} < \text{Score}_{\text{best}}$ **then**
      $\mathcal{P}_{\text{best}} \leftarrow \mathcal{P}$
      $\text{Score}_{\text{best}} \leftarrow \text{Score}$
    end
  end
  $\mathcal{P}_1 \leftarrow \mathcal{P}_{\text{best}}$
  $\mathcal{P}_2 \leftarrow \mathcal{V} \setminus \mathcal{P}_{\text{best}}$
  return $\mathcal{P}_1, \mathcal{P}_2$

Figure 3.1a shows a small network with vertices labelled by position in the sorted second eigenvector $\hat{x}$ of L_{rw}. Figure 3.1b shows the 'sweep profile' of Ncut scores, which is minimised at the splitting point $i = 5$. Hence eigenvector sweep chooses the final partition $\mathcal{P}_1 = \{1, \ldots, 5\}$, $\mathcal{P}_2 = \{6, \ldots, 10\}$; as indicated by the vertex colours and dashed line in Figure 3.1a.



(a) A small network

(b) Sweep profile of the network

Figure 3.1: Eigenvector sweep selects a partition by minimising Ncut

3.4.3 Cluster evaluation

When a graph has been clustered, we assign a score to the partition. If the ground-truth clustering is available, we can compare it to our clustering using the *adjusted Rand index* (ARI) [22]. The ARI between two clusterings has expected value 0 under random cluster assignment, and maximum value 1 denoting perfect agreement between the clusterings. A larger ARI indicates a more similar clustering.

If the ground-truth clustering is not available, we can use the objective function Ncut. Clusterings with lower Ncut values partition the graph more agreeably.

3.5 Spectral clustering algorithms

We present the full random-walk spectral clustering algorithm and show how it can be applied to motif-based random-walk spectral clustering.

3.5.1 Random-walk spectral clustering

Algorithm 3.2 gives random-walk spectral clustering [41], which takes a symmetric connected adjacency matrix as input. We use k-means++ rather than eigenvector sweep as the cluster extraction method, due to its superior flexibility and computational speed. We drop the first column of H (the first eigenvector of L_{rw}) since although it should be constant and uninformative (Proposition 3.1), numerical imprecision may give unwanted artefacts. It is worth noting that although the relaxation used in Section 3.3 is reasonable and often leads to good approximate solutions of the Ncut problem, there are cases where it performs poorly [20]. The Cheeger inequality [8] gives a bound on the error introduced by this relaxation.

Algorithm 3.2: Random-walk spectral clustering

Input: Symmetric adjacency matrix G, number of clusters k, dimension l
Output: Partition $\mathcal{P}_1, \ldots, \mathcal{P}_k$

function `RWSpectClust`(G, k, l)**:**
  Construct the weighted degree matrix $D_{ii} \leftarrow \sum_j G_{ij}$
  Construct the random walk Laplacian matrix $L_{\text{rw}} \leftarrow I - D^{-1}G$
  Let H have the first l eigenvectors of L_{rw} as columns
  Drop the first column of H
  Run k-means++ on the rows of H with k clusters to produce $\mathcal{P}_1, \ldots, \mathcal{P}_k$
  return $\mathcal{P}_1, \ldots, \mathcal{P}_k$

3.5.2 Motif-based random-walk spectral clustering

Algorithm 3.3 gives motif-based random-walk spectral clustering. Note that although $\mathcal{G}$ may be a connected graph, there is no guarantee that the MAM is connected too. Hence M is restricted to its largest connected component C before spectral clustering is applied. While this may initially seem to be a flaw with motif-based spectral clustering (since not all vertices are assigned to a cluster), in fact it can be very useful; restriction of M can remove vertices which are in some sense not 'well connected' to the rest of the graph, which means that only a 'core' set of vertices are clustered. This can result in Algorithm 3.3 making fewer misclassifications with motif-based methods than with traditional spectral clustering, as seen in Section 4.2.

There is ambiguity in whether to use functional or structural MAMs. While the authors in [6] opt for structural MAMs, we propose to use functional MAMs, for a few reasons. Firstly, note that $0 \leq M_{ij}^{\text{struc}} \leq M_{ij}^{\text{func}}$ for all $i, j \in \mathcal{V}$. This implies that the largest connected component of M^{func} is always at least as large as that of M^{struc}, meaning that often more vertices can be assigned to a cluster. Secondly, we argue that functional instances are of more interest than structural motifs, since they specify only 'existence' rather than 'non-existence' of edges. For consistency we will therefore use functional MAMs throughout our experiments.

The most computationally expensive part of Algorithm 3.3 is the calculation of the MAM using a formula from Table B.1. We found this to be feasible for graphs with up to around $n \approx 10\,000$ vertices. General notes on hardware and software are given in Section C.1.1, and timings for MAM computation across a range of graph sizes and sparsities are available in Section C.1.2.

Algorithm 3.3: Motif-based random-walk spectral clustering

Input: Graph $\mathcal{G}$, motif $\mathcal{M}$, number of clusters k, dimension l
Output: Partition $\mathcal{P}_1, \ldots, \mathcal{P}_k$

```
function MotifRWSpectClust(G, M, k, l):
    Construct the motif adjacency matrix M of the graph G with motif M
    Let M̃ be M restricted to its largest connected component, C
    P_1, ..., P_k ← RWSpectClust(M̃, k, l)
    return P_1, ..., P_k
```

Chapter 4

Motif-Based Clustering

We analyse the performance of motif-based random-walk spectral clustering on both synthetic and real data. In Section 4.1 we propose a family of stochastic block models and perform experiments with a variety of motifs and parameters. In Section 4.2 we analyse the US Political Blogs network and in Section 4.3 we present results from the US Migration network.

4.1 Directed stochastic block models

We begin by describing *directed stochastic block models* (DSBMs), a broad class of generative models for directed graphs. A DSBM is characterised by a block count k, a list of block sizes $(n_i)_{i=1}^k$ and a sparsity matrix $F \in [0,1]^{k\times k}$. We define the cumulative block sizes $N_i = \sum_{j=1}^{i} n_j$ with $N_0 = 0$, and the total graph size $N = N_k$. These are used to construct the expected adjacency matrix $A \in [0,1]^{N\times N}$ given by $A_{ij} = F_{rs}\ \mathbb{I}\{i \neq j\}$ where $N_{r-1} < i \leq N_r$ and $N_{s-1} < j \leq N_s$. Finally a graph $\mathcal{G}$ is generated with adjacency matrix entries $G_{ij} \sim \mathrm{Ber}(A_{ij})$ sampled independently. We say that a DSBM is *symmetric* if F is a symmetric matrix.

This DSBM definition is similar to that given by [12], although we impose independence between all entries of the adjacency matrix, allowing for bidirectional edges.

4.1.1 Symmetric two-block DSBMs

We define the *symmetric two-block DSBM* as the DSBM with $k = 2$, $n_1 = n_2 = n$ and $F = \left(\begin{smallmatrix} p & q \\ q & p \end{smallmatrix}\right)$ where $p > q$. Figure 4.1 illustrates the block structure and sparsity matrix of this model. Thicker lines indicate existence of edges with higher probability.

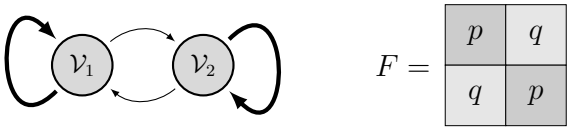
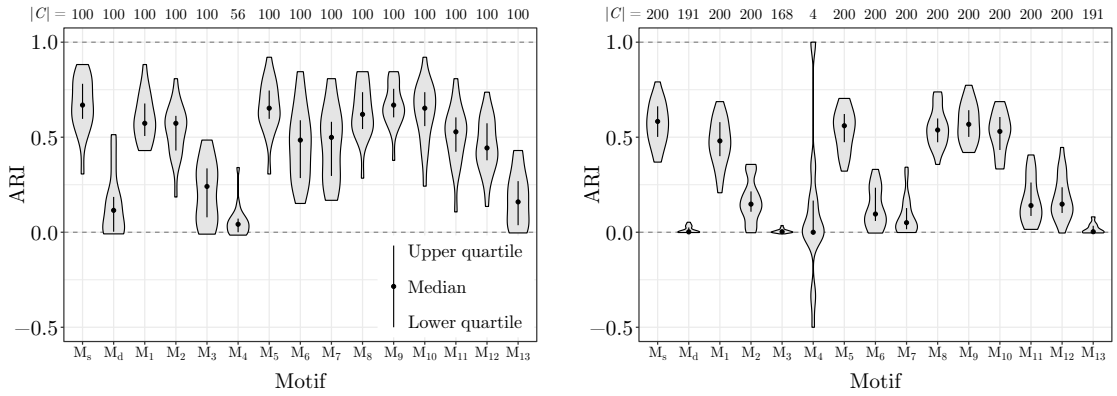


Figure 4.1: Symmetric two-block DSBM block structure and sparsity matrix

We test the performance of Algorithm 3.3 across various motifs with parameters $k = l = 2$ on this model. Figure 4.2 shows violin plots over 20 trials of ARI against motif, for different sets of parameters n, p, q. Also shown is $|C|$, the average size of the largest connected component of each MAM. It can be seen that several motifs (such as $\mathcal{M}_5$ and $\mathcal{M}_9$) achieve a similar ARI to the traditional spectral clustering technique given by the symmetrised adjacency matrix $M = G + G^\top$ generated by the motif $\mathcal{M}_\mathrm{s}$ (Table B.1). However the strongly connected motifs (particularly $\mathcal{M}_4$) generate MAMs with small connected components, especially when $\mathcal{G}$ is sparse, and hence only cluster a subset of the vertices of $\mathcal{G}$.



(a) $n = 50$, $p = 0.3$, $q = 0.2$ (b) $n = 100$, $p = 0.15$, $q = 0.1$

Figure 4.2: ARI violin plots for the symmetric two-block DSBM

4.1.2 Asymmetric two-block DSBMs

We define the *asymmetric two-block DSBM* as the DSBM with $k = 2$, $n_1 = n_2 = n$ and $F = \left(\begin{smallmatrix} p & q_1 \\ q_2 & p \end{smallmatrix}\right)$ where $q_1 > q_2$ and $p = \frac{1}{2}(q_1 + q_2)$. Figure 4.3 shows this model.

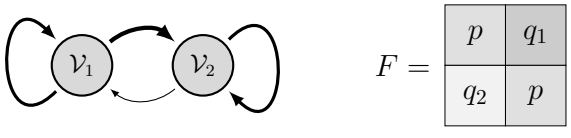
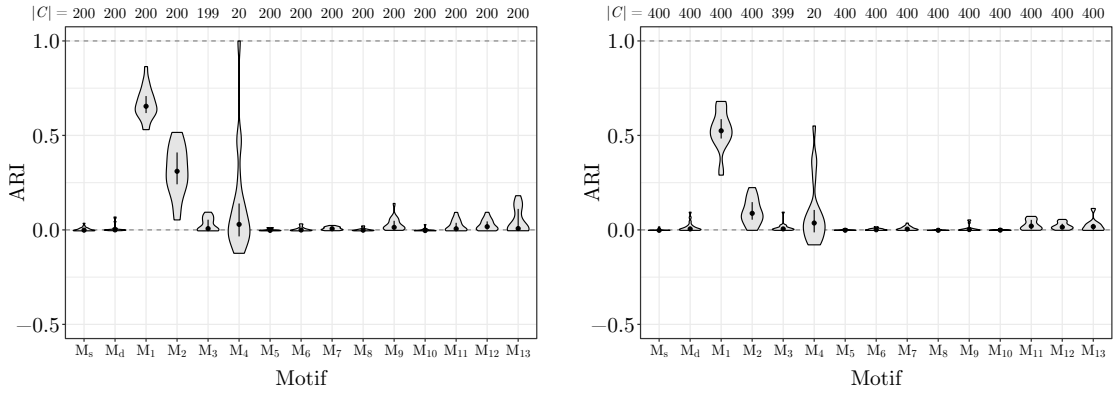


Figure 4.3: Asymmetric two-block DSBM block structure and sparsity matrix

We test the performance of Algorithm 3.3 across various motifs with parameters $k = l = 2$ on this model. Figure 4.4 shows violin plots over 20 trials of ARI against motif, for different sets of parameters n, p, q_1, q_2, and $|C|$ is shown. It is apparent that motif-based clustering with $\mathcal{M}_1$ is the best method, consistently achieving the highest ARI and keeping $|C|$ at its maximum value of $2n$. It is unsurprising that $\mathcal{M}_1$ (feed-back loop) performs well on this model; large p makes feed-back loops within clusters likely, and small q_2 makes feed-back loops spanning the clusters unlikely. Motif $\mathcal{M}_2$ also performs reasonably well since it contains $\mathcal{M}_1$ as a submotif. Furthermore, the constraint $p = \frac{1}{2}(q_1 + q_2)$ ensures that the naïve symmetrisation $M = G + G^\top$ produces indistinguishable clusters, and hence the traditional method performs extremely poorly.



(a) $n = 100$, $p = 0.2$, $q_1 = 0.35$, $q_2 = 0.05$ (b) $n = 200$, $p = 0.15$, $q_1 = 0.25$, $q_2 = 0.05$

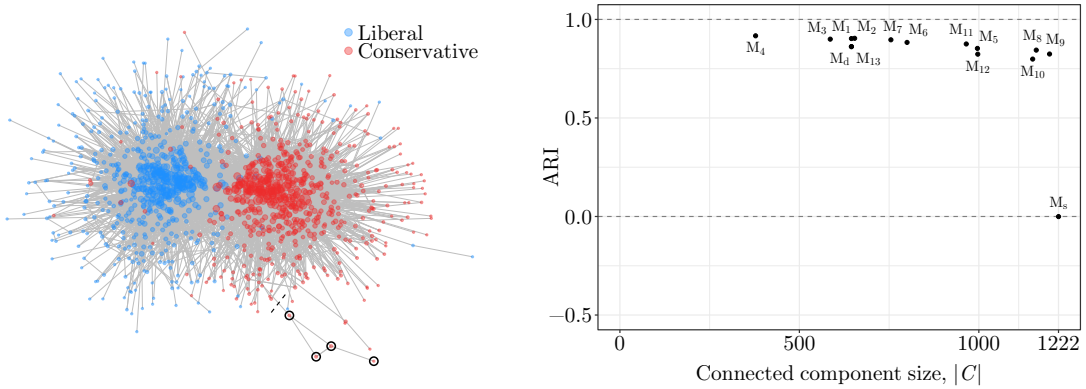
Figure 4.4: ARI violin plots for the asymmetric two-block DSBM

4.2 US Political Blogs network

Our first real data set is the US Political Blogs network [1], consisting of data collected two months before the 2004 US election. Vertices represent blogs, and are labelled by their political leaning ('liberal' or 'conservative'). Weighted directed edges represent the number of citations from one blog to another. After preprocessing (Section C.2) there are 536 liberal blogs, 636 conservative blogs (total 1222) and 19 024 edges. The network is plotted in Figure 4.5a.

We test the performance of Algorithm 3.3 across various motifs with parameters $k = l = 2$ on this network. Figure 4.5b plots ARI against component size $|C|$. There is an apparent trade-off between ARI and connected component size. Motif $\mathcal{M}_9$ clusters many vertices with $|C| = 1197$ and an ARI of 0.82, while the more strongly connected $\mathcal{M}_4$ only clusters 378 vertices, with an improved ARI of 0.92. Finally, the poor performance of traditional spectral clustering is due to a small number of very

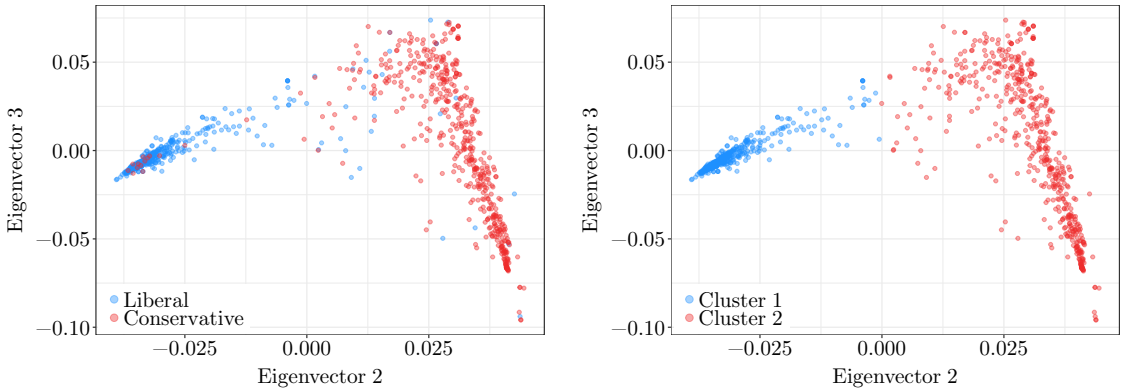
weakly connected vertices being partitioned off, indicated by the dashed line and circled vertices in Figure 4.5a.



(a) The US Political Blogs network (b) ARI against $|C|$ across motifs

Figure 4.5: Plots relating to the US Political Blogs network

Figure 4.6 shows the embedding given by eigenvectors 2 and 3 of the random-walk Laplacian of the MAM generated by motif $\mathcal{M}_{12}$. An instance of this motif in the network indicates the presence of a pair of mutually citing blogs with an incoming citation from a third (see Figure 2.1). Colourings are provided for Figure 4.6a by the truth labels and for Figure 4.6b by the k-means++ clustering of eigenvector 2. The clusterings are very similar, giving an ARI of 0.82.



(a) Colouring by truth label (b) Colouring by k-means++ cluster

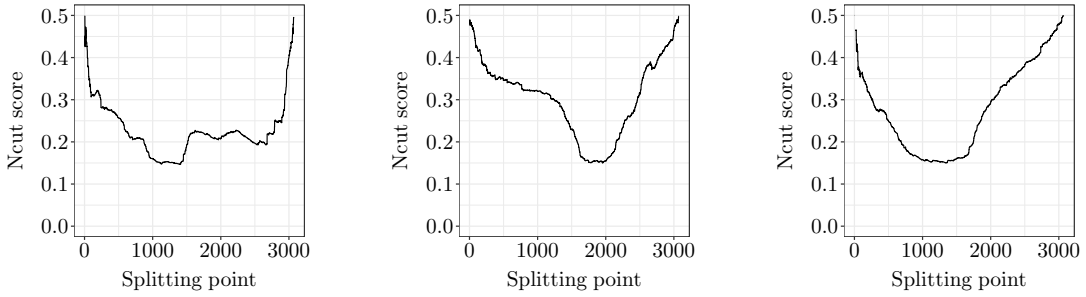
Figure 4.6: Eigendecomposition embedding of the US Political Blogs network

4.3 US Migration network

The next data set is the US Migration network [40], consisting of data collected during the US Census in 2000. Vertices represent the 3075 counties in 49 contiguous states (excluding Alaska and Hawaii, and including the District of Columbia). The 721 432 weighted directed edges represent the number of people migrating from county to county, capped at 10 000 (the 99.9th percentile) to control large entries, as in [12].

We test the performance of Algorithm 3.3 with three selected motifs: $\mathcal{M}_{\mathrm{s}}$, $\mathcal{M}_6$ and $\mathcal{M}_9$ (see Figure 2.1). $\mathcal{M}_{\mathrm{s}}$ gives the traditional spectral clustering method with naïve symmetrisation. $\mathcal{M}_6$ represents a pair of counties exchanging migrants, with both also receiving migrants from a third. $\mathcal{M}_9$ is a path of length two, allowing counties to be deemed similar if there is migration between them via another.

Firstly, we plot sweep profiles of the graph using the second eigenvector of the random-walk Laplacian of the MAM associated with each motif, in Figure 4.7. Note that all three display clear minima, indicating that these motifs produce well-defined clusters. The two-part clusterings produced by eigenvector sweep are somewhat similar across the three motifs, with pairwise ARIs equal to $\mathrm{ARI}(\mathcal{M}_{\mathrm{s}}, \mathcal{M}_6) = 0.67$, $\mathrm{ARI}(\mathcal{M}_{\mathrm{s}}, \mathcal{M}_9) = 0.92$ and $\mathrm{ARI}(\mathcal{M}_6, \mathcal{M}_9) = 0.73$.



(a) $\mathcal{M}_{\mathrm{s}}$ (b) $\mathcal{M}_6$ (c) $\mathcal{M}_9$

Figure 4.7: Sweep profiles of the US Migration network

Next, Figure 4.8 plots maps of the US, with counties coloured initially by the first six non-trivial eigenvectors $x_2, \ldots, x_7$ of the random-walk Laplacian of the associated MAM, and then by the clustering C obtained by Algorithm 3.3 with $k = l = 7$.

For the eigenvector colourings, note how the coloured regions often line up with state boundaries, indicating that many migrants stay within the same state. It is also apparent that the motifs $\mathcal{M}_6$ and $\mathcal{M}_9$ produce 'noisier' embeddings than traditional spectral clustering, due to their reliance on three-vertex motifs. Eigenvector 2 approximately differentiates counties by longitude, although $\mathcal{M}_9$ achieves a clearer division between east and west, while $\mathcal{M}_{\mathrm{s}}$ and $\mathcal{M}_6$ colour California (CA, see Figure C.1) more

similarly to the East Coast. Eigenvector 3 tends to differentiate by latitude, though $\mathcal{M}_{\mathrm{s}}$ and $\mathcal{M}_6$ particularly isolate the states of North Dakota (ND), South Dakota (SD), Minnesota (MN), Wisconsin (WI) and Michigan (MI). Further structure is visible across all three motifs for eigenvectors 4–7.

The clusterings C partition the counties into $k = 7$ regions, and there are some interesting differences between the motifs. Since there is no ground-truth clustering, we record the Ncut score associated with each clustering. It is apparent that motifs $\mathcal{M}_6$ and $\mathcal{M}_9$ give a similar partition, although with some differences: $\mathcal{M}_6$ clusters the East Coast together with western Florida (FL) and the counties containing Los Angeles (CA), San Diego (CA), Las Vegas (NV), Phoenix (AZ), Tucson (AZ), Denver (CO), Chicago (IL) and Nashville (TN). $\mathcal{M}_6$ favours a larger 'central' region, which includes significant parts of Colorado (CO), Oklahoma (OK), Arkansas (AR) and Illinois (IL). $\mathcal{M}_{\mathrm{s}}$ gives a somewhat different partition, with one of the clusters allocated to Michigan (MI) and Wisconsin (WI) rather than Mississippi (MS), Alabama (AL), Georgia (GA) and Tennessee (TN). As with the eigenvectors, the clustering is smoother for $\mathcal{M}_{\mathrm{s}}$ than for $\mathcal{M}_6$ and $\mathcal{M}_9$.

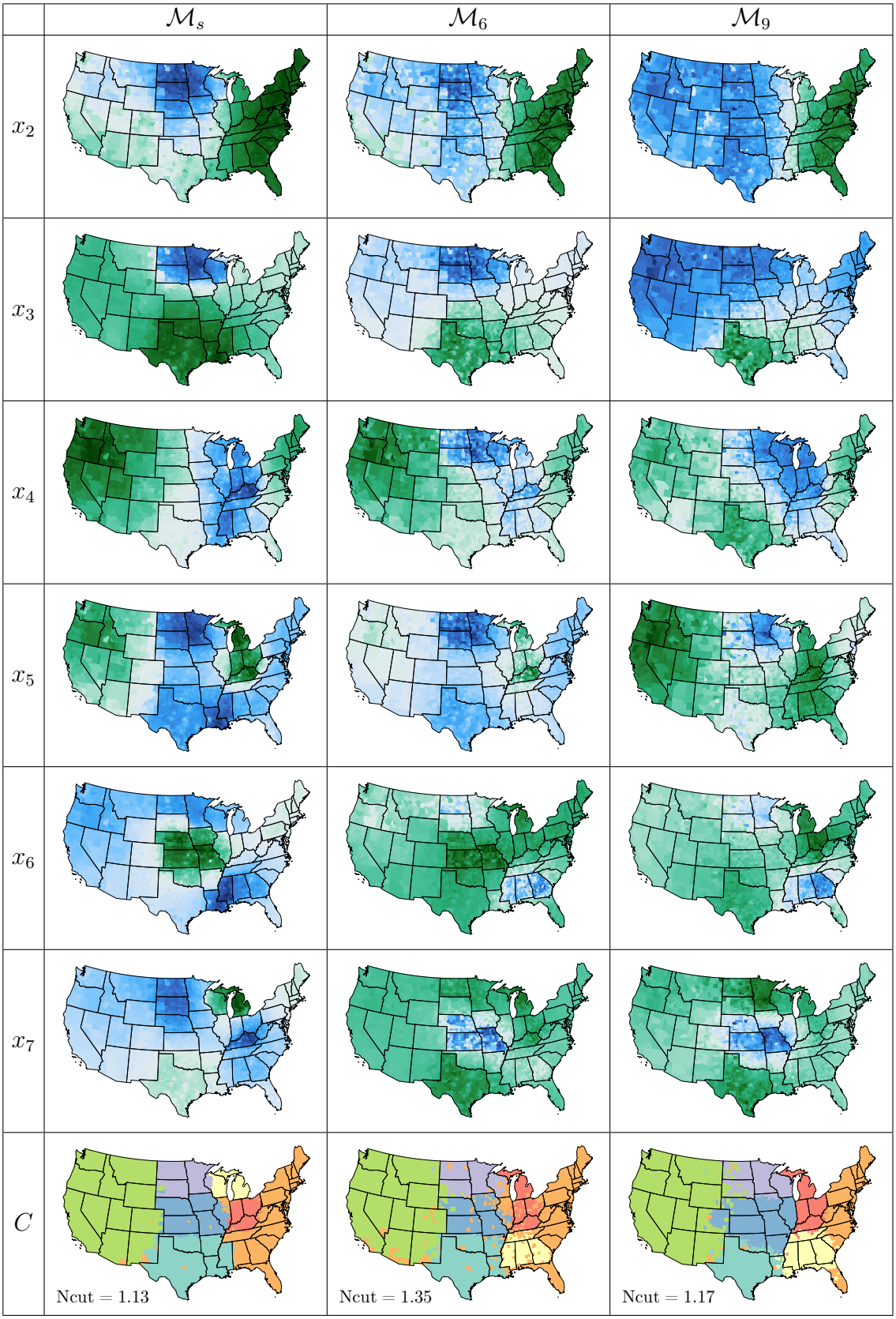


Figure 4.8: Motif-based colourings of the US Migration network

Chapter 5

Bipartite Clustering

We propose a technique for spectral clustering of bipartite graphs and test its performance on both real and synthetic data. In Section 5.1 we define bipartite graphs and present our clustering technique. In Section 5.2 we propose a bipartite stochastic block model (BSBM) and perform experiments with varying parameters. In Section 5.3 we demonstrate our method using the American Revolution network. In Section 5.4 we analyse the Unicode Languages network.

5.1 Bipartite graphs

Definition 5.1. A *bipartite graph* is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ can be partitioned into $\mathcal{V} = \mathcal{S} \sqcup \mathcal{D}$ such that $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{D}$. That is, every edge starts in $\mathcal{S}$ and ends in $\mathcal{D}$. We refer to $\mathcal{S}$ as the *source vertices* and to $\mathcal{D}$ as the *destination vertices*.

5.1.1 Collider and expander motifs

Our method for clustering bipartite graphs revolves around two *anchored* motifs; the *collider* and the *expander* (Figure 5.1). For each motif the anchor set is $\mathcal{A} = \{1, 3\}$.

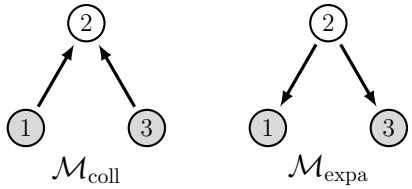


Figure 5.1: The collider and expander motifs

These motifs are useful for bipartite clustering because of Proposition 5.1, which states that their restricted MAMs are the adjacency matrices of the projections [24] of the graph $\mathcal{G}$. In particular they can be used as similarity matrices for the source

and destination vertices respectively. The similarity of two distinct source (resp. destination) vertices is the sum over their mutual neighbours of the average weights of their edges to (resp. from) that neighbour.

Proposition 5.1 (Colliders and expanders in bipartite graphs)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ be a directed bipartite graph. Let M_{coll} and M_{expa} be the structural or functional MAMs of $\mathcal{M}_{\text{coll}}$ and $\mathcal{M}_{\text{expa}}$ respectively in $\mathcal{G}$. Then*

$$(M_{\text{coll}})_{ij} = \mathbb{I}\{i \neq j\} \sum_{\substack{k \in \mathcal{D} \\ (i,k),(j,k) \in \mathcal{E}}} \frac{1}{2}\Big[W((i,k)) + W((j,k))\Big], \tag{1}$$

$$(M_{\text{expa}})_{ij} = \mathbb{I}\{i \neq j\} \sum_{\substack{k \in \mathcal{S} \\ (k,i),(k,j) \in \mathcal{E}}} \frac{1}{2}\Big[W((k,i)) + W((k,j))\Big]. \tag{2}$$

Proof. See Proof A.3. □

5.1.2 Bipartite spectral clustering algorithm

Algorithm 5.1 gives our procedure for clustering a bipartite graph. The algorithm uses the collider and expander motifs to create similarity matrices for the source and destination vertices respectively (as in Section 5.1.1), and then applies random-walk spectral clustering (Algorithm 3.2) to produce the partitions.

Algorithm 5.1: Bipartite random walk spectral clustering

Input: Bipartite graph $\mathcal{G}$, source clusters $k_{\mathcal{S}}$, destination clusters $k_{\mathcal{D}}$, source dimension $l_{\mathcal{S}}$, destination dimension $l_{\mathcal{D}}$

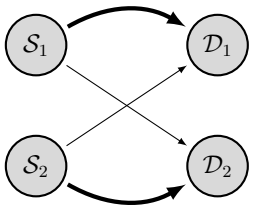
Output: Source partition $\mathcal{S}_1, \ldots, \mathcal{S}_{k_{\mathcal{S}}}$, destination partition $\mathcal{D}_1, \ldots, \mathcal{D}_{k_{\mathcal{D}}}$

function `BipartiteRWSpectClust`$(\mathcal{G}, k_{\mathcal{S}}, k_{\mathcal{D}}, l_{\mathcal{S}}, l_{\mathcal{D}})$**:**
- Construct the collider motif adjacency matrix M_{coll} of the graph $\mathcal{G}$
- Construct the expander motif adjacency matrix M_{expa} of the graph $\mathcal{G}$
- $M_{\text{coll}} \leftarrow M_{\text{coll}}[\mathcal{S}, \mathcal{S}]$ ▷ restrict rows and columns of M_{coll} to $\mathcal{S}$
- $M_{\text{expa}} \leftarrow M_{\text{expa}}[\mathcal{D}, \mathcal{D}]$ ▷ restrict rows and columns of M_{expa} to $\mathcal{D}$
- $\mathcal{S}_1, \ldots, \mathcal{S}_{k_{\mathcal{S}}} \leftarrow$ `RWSpectClust`$(M_{\text{coll}}, k_{\mathcal{S}}, l_{\mathcal{S}})$
- $\mathcal{D}_1, \ldots, \mathcal{D}_{k_{\mathcal{D}}} \leftarrow$ `RWSpectClust`$(M_{\text{expa}}, k_{\mathcal{D}}, l_{\mathcal{D}})$
- **return** $\mathcal{S}_1, \ldots, \mathcal{S}_{k_{\mathcal{S}}}$ and $\mathcal{D}_1, \ldots, \mathcal{D}_{k_{\mathcal{D}}}$

5.2 Bipartite stochastic block models

We define the *bipartite stochastic block model* (BSBM) [17] as the DSBM with $k = 4$, $n_1 = \cdots = n_4 = n$ and $F = \begin{pmatrix} 0 & 0 & p & q \\ 0 & 0 & q & p \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ where $p > q$. Figure 5.2 illustrates the block

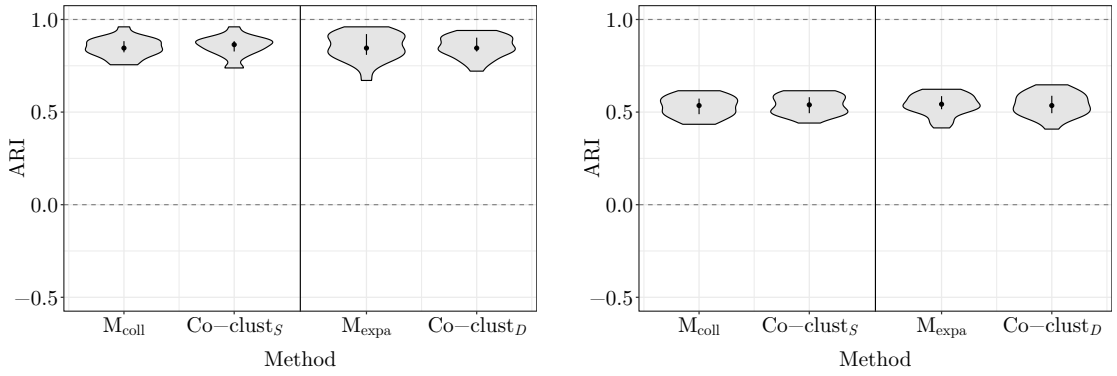
structure and sparsity matrix of this model. This model partitions the source vertices as $\mathcal{S} = \mathcal{S}_1 \sqcup \mathcal{S}_2$ and the destination vertices as $\mathcal{D} = \mathcal{D}_1 \sqcup \mathcal{D}_2$. Edges exist with high probability from $\mathcal{S}_1$ to $\mathcal{D}_1$ and from $\mathcal{S}_2$ to $\mathcal{D}_2$.



$$F = \begin{array}{|c|c|c|c|} \hline 0 & 0 & p & q \\ \hline 0 & 0 & q & p \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Figure 5.2: BSBM block structure and sparsity matrix

We test the performance of Algorithm 5.1 with parameters $k_{\mathcal{S}} = k_{\mathcal{D}} = l_{\mathcal{S}} = l_{\mathcal{D}} = 2$ on this model. For comparison we implement the co-clustering method from [14], which is based on random-walk spectral clustering of the symmetrised adjacency matrix $G + G^\top$. Figure 5.3 shows violin plots over 20 trials of ARI against method, for different sets of parameters n, p, q. Note that if a bipartite graph is connected, then so are M_{coll} and M_{expa}, so we need not consider the largest connected component size $|C|$. Performance of the two methods is very similar, for source and destination vertices.



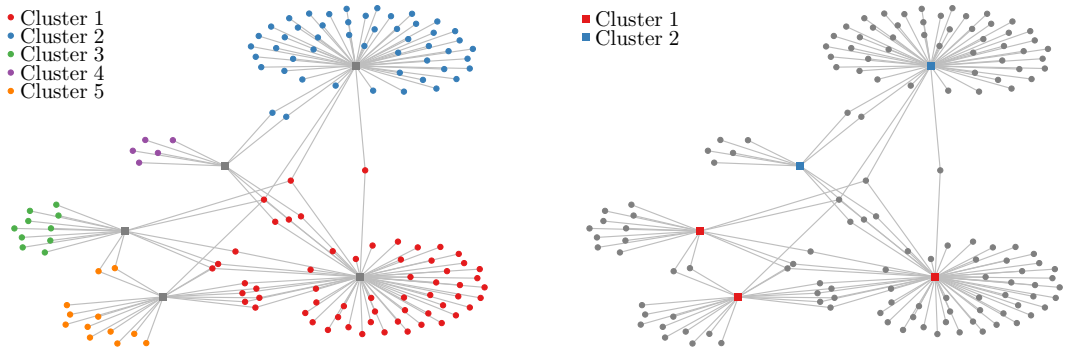
(a) $n = 100$, $p = 0.2$, $q = 0.1$ (b) $n = 200$, $p = 0.1$, $q = 0.06$

Figure 5.3: ARI violin plots for the BSBM

5.3 American Revolution network

As an example of application of our bipartite clustering method to real data, we consider the American Revolution network [25]. This consists of data collected from before the American Revolution. Source vertices are people, and destination vertices are organisations. Edges represent membership of a person to an organisation. There are 136 people, 5 organisations and 160 edges.

Algorithm 5.1 is run on the American Revolution network, with parameters $k_{\mathcal{S}} = l_{\mathcal{S}} = 5$ and $k_{\mathcal{D}} = l_{\mathcal{D}} = 2$. Figure 5.4a plots the network with people coloured by source cluster, and Figure 5.4b plots the network with organisations coloured by destination cluster. The algorithm succeeds in clustering people based on their common memberships, and in clustering organisations based on their common members.



(a) Grouping people into 5 clusters (b) Grouping organisations into 2 clusters

Figure 5.4: Bipartite clustering of the American Revolution network

5.4 Unicode Languages network

The final data set is the Unicode Languages network [26], consisting of data collected in 2014 on languages spoken around the world. Source vertices are territories, and destination vertices are languages. Weighted directed edges from territory to language indicate the number of inhabitants in that territory who speak the specified language (territory population data taken from [10]). After preprocessing (Section C.2) there are 155 territories, 270 languages and 705 edges.

We test Algorithm 5.1 with parameters $k_{\mathcal{S}} = l_{\mathcal{S}} = k_{\mathcal{D}} = l_{\mathcal{D}} = 6$ on this network. For the source vertices, Figure 5.5 plots maps of the world with territories coloured by the clustering obtained. The top 20 territories (by population) in each cluster are given in Table 5.1. Cluster 1 is by far the largest cluster, and includes a wide variety of territories, of which many but not all speak some English. Cluster 2 contains the Persian-speaking territories of Iran and Afghanistan, the Arabic territories of Saudi Arabia and Syria, and the African French-speaking DR Congo, Côte d'Ivoire, Burkina Faso, Niger and others. It also includes Haiti, another French-speaking territory. Cluster 3 mostly captures Spanish-speaking territories in the Americas and also contains Equatorial Guinea, another Spanish-speaking territory in Africa. Cluster 4 includes the Slavic territories of Russia and some of its neighbours. The absence of Kazakhstan may be due to the 981 760 Kazakhs who speak German which

is not a Slavic or Turkic language. Cluster 5 covers China, Hong Kong, Mongolia and some of South-East Asia. The inclusion of Panama might be due to the 6821 Panamanians who speak Chinese. Cluster 6 is the smallest cluster and contains only Japan and the Koreas, which are connected by the 636 440 Japanese who speak Korean.

There are a few territories and languages which are not contained in the large connected component of the network due to their linguistic isolation. These territories are Laos, Norway and Timor-Leste, and the languages are Lao, Norwegian Bokmål and Norwegian Nynorsk.

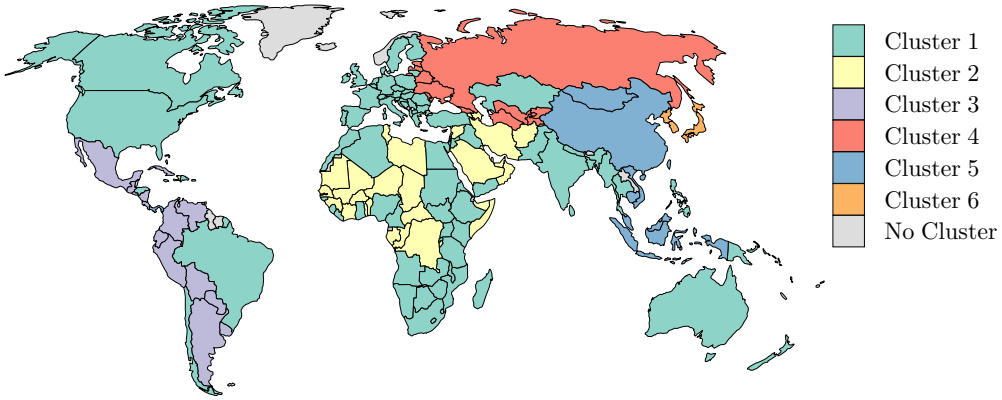


Figure 5.5: Clustering the territories from the Unicode Languages network

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
India	Iran	Mexico	Russia	China	Japan
United States	DR Congo	Colombia	Ukraine	Indonesia	S. Korea
Brazil	Afghanistan	Argentina	Uzbekistan	Vietnam	N. Korea
Pakistan	Saudi Arabia	Peru	Belarus	Malaysia	
Bangladesh	Syria	Venezuela	Tajikistan	Taiwan	
Nigeria	Côte d'Ivoire	Ecuador	Kyrgyzstan	Cambodia	
Philippines	Burkina Faso	Guatemala	Turkmenistan	Hong Kong	
Ethiopia	Niger	Cuba	Georgia	Singapore	
Germany	Mali	Bolivia	Moldova	Panama	
Egypt	Senegal	Paraguay	Latvia	Mongolia	
Turkey	Tunisia	El Salvador	Estonia		
Thailand	Chad	Nicaragua			
France	Guinea	Costa Rica			
United Kingdom	Somalia	Uruguay			
Italy	Burundi	Eq. Guinea			
Myanmar	Haiti				
South Africa	Benin				
Spain	Azerbaijan				
Tanzania	Togo				
Kenya	Libya				
...	...				
|Cluster 1| = 87	|Cluster 2| = 29	|Cluster 3| = 15	|Cluster 4| = 11	|Cluster 5| = 10	|Cluster 6| = 3

Table 5.1: Clustering the territories from the Unicode Languages network

For the destination vertices, we present the six clusters obtained by Algorithm 5.1. Table 5.2 contains the top 20 languages (by number of speakers) in each cluster. Cluster 1 is the largest cluster and contains the European languages of Spanish, Portuguese and French, as well as dialects of Arabic. Cluster 2 is also large and includes English as well as several South Asian languages such as Hindi, Bengali, Urdu and Punjabi. Cluster 3 consists of many indigenous African languages such as Swahili, Kinyarwanda and Somali. Cluster 4 captures languages from South-East Asia, mostly spoken in Indonesia and Malaysia. Cluster 5 identifies several varieties of Chinese and a few other Central and East Asian languages such as Kazakh and Uighur. Interestingly Korean is also placed in this group and not with Japanese, even though the Koreas are clustered together with Japan in Table 5.1. Cluster 6 captures more South-East Asian languages, this time from Thailand, Myanmar and Cambodia. Pattani Malay is in this cluster because despite its name it is spoken more in Thailand than in Malaysia.

Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Spanish	English	Swahili	Indonesian	Chinese	Thai
Arabic	Hindi	Kinyarwanda	Javanese	Wu Chinese	N.E. Thai
Portuguese	Bengali	Somali	Malay	Korean	Khmer
French	Urdu	Luba-Lulua	Sundanese	Xiang Chinese	N. Thai
Russian	Punjabi	Kikuyu	Madurese	Hakka Chinese	S. Thai
Japanese	Telugu	Congo Swahili	Minangkabau	Minnan Chinese	Shan
German	Marathi	Luyia	Betawi	Gan Chinese	Pattani Malay
Turkish	Vietnamese	Ganda	Balinese	Kazakh	
Persian	Tamil	Luo	Buginese	Uighur	
Italian	Lahnda	Sukuma	Banjar	Sichuan Yi	
Egyptian Arabic	Filipino	Kalenjin	Achinese	Mongolian	
Polish	Gujarati	Lingala	Sasak	Zhuang	
Nigerian Pidgin	Kannada	Nyankole	Makasar	Tibetan	
Ukrainian	Pushto	Gusii	Lampung Api		
Dutch	Malayalam	Kiga	Rejang		
Algerian Arabic	Oriya	Soga			
Moroccan Arabic	Burmese	Luba-Katanga			
Hausa	Bhojpuri	Meru			
Azerbaijani	Amharic	Teso			
Uzbek	Oromo	Nyamwezi			
...	...	...			
|Cluster 1| = 120	|Cluster 2| = 90	|Cluster 3| = 25	|Cluster 4| = 15	|Cluster 5| = 13	|Cluster 6| = 7

Table 5.2: Clustering the languages from the Unicode Languages network

Chapter 6

Conclusion

With this dissertation we have introduced a graph-theoretic framework for analysis of weighted directed networks, and presented new matrix-based formulae for MAMs (Chapter 2). We have summarised the method of random-walk spectral clustering and shown how it can be used with motif-based techniques (Chapter 3). We have presented results from the application of a motif-based method both to synthetic data (DSBMs) and to real data (US Political Blogs network, US Migration network). We have demonstrated that this technique outperforms traditional spectral clustering methods on several occasions (Chapter 4). We have introduced a motif-based spectral method for clustering bipartite graphs and presented results both from synthetic data (BSBMs) and from real data (American Revolution network, Unicode Languages network).

In particular we have shown that motif-based spectral clustering is a valuable tool for clustering weighted directed networks, which is scalable and easy to implement. Superior performance has been demonstrated especially with asymmetric DSBMs in Section 4.1.2, and with the US Political Blogs network in Section 4.2.

Limitations

There are limitations to our work. While our matrix-based formulae for MAMs are simple to implement and moderately scalable, they are computationally unwieldy for large networks (see Section C.1 for details). As mentioned in [6], fast triangle enumeration algorithms [13, 44, 45] offer increased performance, at the expense of methodological simplicity. Another shortcoming of the matrix-based formulae is that unlike motif detection algorithms such as [45], they do not extend to motifs on four or more vertices.

Future work

There is plenty of scope for methodological investigation related to our work. Simple extensions could involve an analysis of the differences between clustering methods based on functional and structural MAMs respectively. One could also experiment with the effects of replacing the random-walk Laplacian with the unnormalised Laplacian or symmetric normalised Laplacian [41]. Similarly one might try replacing Ncut with RatioCut [21]. We note that although our methods apply to weighted graphs, we have only discussed unweighted DSBMs. Therefore it would be interesting to investigate weighted DSBMs (perhaps following the exponential family method detailed in [2]) and to use them for evaluation of motif-based spectral clustering procedures.

Further experimental work is also desirable. We would like to conduct experiments on more real data, and suggest that collaboration networks such as [27], and bipartite preference networks such as [9] could be interesting. Comparison with other clustering methods could also be insightful; the Hermitian matrices method in [12], the PageRank method in [47] and TECTONIC from [39] may give suitable benchmarks for performance.

Appendix A

Proofs and Examples

A.1 Proofs

Proof A.1 (Proposition 2.1, MAM formula)**.** Consider (1). We sum over functional instances $\mathcal{M} \cong \mathcal{H} \leq \mathcal{G}$ such that $\{i,j\} \in \mathcal{A}(\mathcal{H})$. This is equivalent to summing over $\{k_2, \ldots, k_{m-1}\} \subseteq \mathcal{V}$ and $\sigma \in S^{\sim}_{\mathcal{M},\mathcal{A}}$, such that k_u are all distinct and

$$(u,v) \in \mathcal{E}_{\mathcal{M}} \implies (k_{\sigma u}, k_{\sigma v}) \in \mathcal{E}\,. \qquad (\dagger)$$

This is because the vertex set $\{k_2, \ldots, k_{m-1}\} \subseteq \mathcal{V}$ indicates which vertices are present in the instance $\mathcal{H}$, and σ describes the mapping from $\mathcal{V}_{\mathcal{M}}$ onto those vertices: $u \mapsto k_{\sigma u}$. We take $\sigma \in S^{\sim}_{\mathcal{M},\mathcal{A}}$ to ensure that $\{i,j\} \in \mathcal{A}(\mathcal{H})$ (since $i = k_1,\ j = k_m$), and that instances are counted exactly once. The condition (†) is to check that $\mathcal{H}$ is a functional instance of $\mathcal{M}$ in $\mathcal{G}$. Hence

$$\begin{aligned} M^{\text{func}}_{ij} &= \frac{1}{|\mathcal{E}_{\mathcal{M}}|} \sum_{\mathcal{M} \cong \mathcal{H} \leq \mathcal{G}} \mathbb{I}\big\{\{i,j\} \in \mathcal{A}(\mathcal{H})\big\} \sum_{e \in \mathcal{E}_{\mathcal{H}}} W(e) \\ &= \frac{1}{|\mathcal{E}_{\mathcal{M}}|} \sum_{\{k_2,\ldots,k_{m-1}\}} \sum_{\sigma \in S^{\sim}_{\mathcal{M},\mathcal{A}}} \mathbb{I}\big\{k_u \text{ all distinct, } (\dagger)\big\} \sum_{e \in \mathcal{E}_{\mathcal{H}}} W(e)\,. \end{aligned}$$

For the first term, by conditioning on the types of edge in $\mathcal{E}_{\mathcal{M}}$:

$$\begin{aligned} \mathbb{I}\big\{k_u \text{ all distinct, } (\dagger)\big\} &= \prod_{\mathcal{E}^0_{\mathcal{M}}} \mathbb{I}\{k_{\sigma u} \neq k_{\sigma v}\} \\ &\quad \times \prod_{\mathcal{E}^{\text{s}}_{\mathcal{M}}} \mathbb{I}\{(k_{\sigma u}, k_{\sigma v}) \in \mathcal{E}\} \\ &\quad \times \prod_{\mathcal{E}^{\text{d}}_{\mathcal{M}}} \mathbb{I}\{(k_{\sigma u}, k_{\sigma v}) \in \mathcal{E} \text{ and } (k_{\sigma v}, k_{\sigma u}) \in \mathcal{E}\} \\ &= \prod_{\mathcal{E}^0_{\mathcal{M}}} (J_{\text{n}})_{k_{\sigma u}, k_{\sigma v}} \prod_{\mathcal{E}^{\text{s}}_{\mathcal{M}}} J_{k_{\sigma u}, k_{\sigma v}} \prod_{\mathcal{E}^{\text{d}}_{\mathcal{M}}} (J_{\text{d}})_{k_{\sigma u}, k_{\sigma v}} \\ &= J^{\text{func}}_{\mathbf{k},\sigma}\,. \end{aligned}$$

Assuming $\{k_u \text{ all distinct, } (\dagger)\}$, the second term is

$$\begin{aligned}\sum_{e\in\mathcal{E}_{\mathcal{H}}} W(e) &= \sum_{\mathcal{E}^{\mathrm{s}}_{\mathcal{M}}} W((k_{\sigma u}, k_{\sigma v})) + \sum_{\mathcal{E}^{\mathrm{d}}_{\mathcal{M}}} \big(W((k_{\sigma u}, k_{\sigma v})) + W((k_{\sigma v}, k_{\sigma u}))\big) \\ &= \sum_{\mathcal{E}^{\mathrm{s}}_{\mathcal{M}}} G_{k_{\sigma u}, k_{\sigma v}} + \sum_{\mathcal{E}^{\mathrm{d}}_{\mathcal{M}}} (G_{\mathrm{d}})_{k_{\sigma u}, k_{\sigma v}} \\ &= G^{\mathrm{func}}_{\mathbf{k},\sigma}\end{aligned}$$

as required. For (2), we simply change (†) to (‡) to check that an instance is a *structural* instance:

$$(u, v) \in \mathcal{E}_{\mathcal{M}} \iff (k_{\sigma u}, k_{\sigma v}) \in \mathcal{E} \qquad (\ddagger)$$

Now for the first term:

$$\begin{aligned}\mathbb{I}\{k_u \text{ all distinct, } (\ddagger)\} &= \prod_{\mathcal{E}^0_{\mathcal{M}}} \mathbb{I}\{(k_{\sigma u}, k_{\sigma v}) \notin \mathcal{E} \text{ and } (k_{\sigma v}, k_{\sigma u}) \notin \mathcal{E}\} \\ &\quad \times \prod_{\mathcal{E}^{\mathrm{s}}_{\mathcal{M}}} \mathbb{I}\{(k_{\sigma u}, k_{\sigma v}) \in \mathcal{E} \text{ and } (k_{\sigma v}, k_{\sigma u}) \notin \mathcal{E}\} \\ &\quad \times \prod_{\mathcal{E}^{\mathrm{d}}_{\mathcal{M}}} \mathbb{I}\{(k_{\sigma u}, k_{\sigma v}) \in \mathcal{E} \text{ and } (k_{\sigma v}, k_{\sigma u}) \in \mathcal{E}\} \\ &= \prod_{\mathcal{E}^0_{\mathcal{M}}} (J_0)_{k_{\sigma u}, k_{\sigma v}} \prod_{\mathcal{E}^{\mathrm{s}}_{\mathcal{M}}} (J_{\mathrm{s}})_{k_{\sigma u}, k_{\sigma v}} \prod_{\mathcal{E}^{\mathrm{d}}_{\mathcal{M}}} (J_{\mathrm{d}})_{k_{\sigma u}, k_{\sigma v}} \\ &= J^{\mathrm{struc}}_{\mathbf{k},\sigma}\,.\end{aligned}$$

Assuming $\{k_u \text{ all distinct, } (\ddagger)\}$, the second term is

$$\begin{aligned}\sum_{e\in\mathcal{E}_{\mathcal{H}}} W(e) &= \sum_{\mathcal{E}^{\mathrm{s}}_{\mathcal{M}}} W((k_{\sigma u}, k_{\sigma v})) + \sum_{\mathcal{E}^{\mathrm{d}}_{\mathcal{M}}} \big(W((k_{\sigma u}, k_{\sigma v})) + W((k_{\sigma v}, k_{\sigma u}))\big) \\ &= \sum_{\mathcal{E}^{\mathrm{s}}_{\mathcal{M}}} (G_{\mathrm{s}})_{k_{\sigma u}, k_{\sigma v}} + \sum_{\mathcal{E}^{\mathrm{d}}_{\mathcal{M}}} (G_{\mathrm{d}})_{k_{\sigma u}, k_{\sigma v}} \\ &= G^{\mathrm{struc}}_{\mathbf{k},\sigma}\,.\end{aligned}$$

□

Proof A.2 (Proposition 2.2, Complexity of MAM formula)**.** Suppose $m \leq 3$ and consider M^{func}. The adjacency and indicator matrices of $\mathcal{G}$ are

(1) $J = \mathbb{I}\{G > 0\}$,
(2) $J_0 = \mathbb{I}\{G + G^\top = 0\} \circ J_{\text{n}}$,
(3) $J_{\text{s}} = J - J_{\text{d}}$,
(4) $G_{\text{d}} = (G + G^\top) \circ J_{\text{d}}$,
(5) $J_{\text{n}} = \mathbb{I}\{I_{n\times n} = 0\}$,
(6) $J_{\text{d}} = J \circ J^\top$,
(7) $G_{\text{s}} = G \circ J_{\text{s}}$,

and are computed using four additions and four element-wise multiplications. $J^{\text{func}}_{\mathbf{k},\sigma}$ is a product of at most three factors, and $G^{\text{func}}_{\mathbf{k},\sigma}$ contains at most three summands, so

$$\sum_{k_2 \in \mathcal{V}} J^{\text{func}}_{\mathbf{k},\sigma}\ G^{\text{func}}_{\mathbf{k},\sigma}$$

is expressible as a sum of at most three matrices, each of which is constructed with at most one matrix multiplication (where $\{k_{\sigma r}, k_{\sigma s}\} \neq \{i, j\}$) and one entry-wise multiplication (where $\{k_{\sigma r}, k_{\sigma s}\} = \{i, j\}$). This is repeated for each $\sigma \in S^{\sim}_{\mathcal{M},\mathcal{A}}$ (at most six times) and the results are summed. Calculations are identical for M^{struc}.

□

Proof A.3 (Proposition 5.1, Colliders and expanders in bipartite graphs)**.** Consider (1) and the collider motif $\mathcal{M}_{\text{coll}}$. Since $\mathcal{G}$ is bipartite, $M^{\text{func}}_{\text{coll}} = M^{\text{struc}}_{\text{coll}} =: M_{\text{coll}}$, and by Table B.1, $M_{\text{coll}} = \frac{1}{2} J_{\text{n}} \circ (JG^\top + GJ^\top)$. Hence

$$\begin{aligned}
(M_{\text{coll}})_{ij} &= \frac{1}{2}(J_{\text{n}})_{ij}\ (JG^\top + GJ^\top)_{ij} \\
&= \mathbb{I}\{i \neq j\} \sum_{k \in \mathcal{V}} \frac{1}{2}\Big(J_{ik}G_{jk} + G_{ik}J_{jk}\Big) \\
&= \mathbb{I}\{i \neq j\} \sum_{k \in \mathcal{V}} \frac{1}{2}\,\mathbb{I}\Big\{(i,k),(j,k) \in \mathcal{E}\Big\}\Big[W((i,k)) + W((j,k))\Big] \\
&= \mathbb{I}\{i \neq j\} \sum_{\substack{k \in \mathcal{D} \\ (i,k),(j,k) \in \mathcal{E}}} \frac{1}{2}\Big[W((i,k)) + W((j,k))\Big].
\end{aligned}$$

Similarly for the expander motif, $M_{\text{expa}} = \frac{1}{2} J_{\text{n}} \circ (J^\top G + G^\top J)$ so

$$\begin{aligned}
(M_{\text{expa}})_{ij} &= \frac{1}{2}(J_{\text{n}})_{ij}\ (J^\top G + G^\top J)_{ij} \\
&= \mathbb{I}\{i \neq j\} \sum_{\substack{k \in \mathcal{S} \\ (k,i),(k,j) \in \mathcal{E}}} \frac{1}{2}\Big[W((k,i)) + W((k,j))\Big].
\end{aligned}$$

□

A.2 Examples

Example A.1 (Functional and structural instances)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph with $\mathcal{V} = \{1, 2, 3, 4\}$ and $\mathcal{E} = \{(1,2), (1,3), (1,4), (2,3), (3,4), (4,3)\}$. Let $(\mathcal{M}, \mathcal{A})$ be the anchored motif with $\mathcal{V}_\mathcal{M} = \{1, 2, 3\}$, $\mathcal{E}_\mathcal{M} = \{(1,2), (1,3), (2,3)\}$ and $\mathcal{A} = \{1, 3\}$ as defined in Figure A.1.

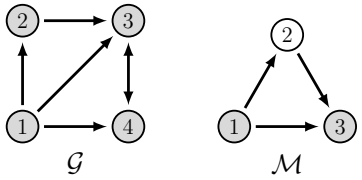


Figure A.1: The specified graph $\mathcal{G}$ and anchored motif $\mathcal{M}$

There are three functional instances of $\mathcal{M}$ in $\mathcal{G}$, shown in Figure A.2. However there is just one structural instance of $\mathcal{M}$ in $\mathcal{G}$, given by $\mathcal{H}_1$. This is because the double edge $3 \leftrightarrow 4$ in $\mathcal{G}$ prevents the subgraphs on $\{1, 3, 4\}$ from being induced subgraphs.

$$\begin{aligned}
\mathcal{H}_1: &\quad \mathcal{V}_1 = \{1,2,3\}; \quad \mathcal{E}_1 = \{(1,2),(2,3),(1,3)\}; \quad \mathcal{A}(\mathcal{H}_1) = \big\{\{1,3\}\big\}, \\
\mathcal{H}_2: &\quad \mathcal{V}_2 = \{1,3,4\}; \quad \mathcal{E}_2 = \{(1,3),(1,4),(3,4)\}; \quad \mathcal{A}(\mathcal{H}_2) = \big\{\{1,4\}\big\}, \\
\mathcal{H}_3: &\quad \mathcal{V}_3 = \{1,3,4\}; \quad \mathcal{E}_3 = \{(1,3),(1,4),(4,3)\}; \quad \mathcal{A}(\mathcal{H}_3) = \big\{\{1,3\}\big\}.
\end{aligned}$$

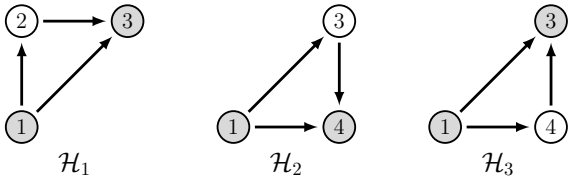
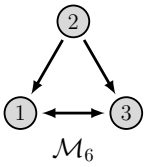


Figure A.2: Functional instances $\mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$

Example A.2 (Motif adjacency matrices)**.** Let $\mathcal{G}$ and $(\mathcal{M}, \mathcal{A})$ be as in Example A.1, and suppose $\mathcal{G}$ has weight map $W((i,j)) := i + j$. Then using Definition 2.11 directly, the functional and structural MAMs of $(\mathcal{M}, \mathcal{A})$ in $\mathcal{G}$ are respectively

$$M^{\text{func}} = \begin{pmatrix} 0 & 0 & 28 & 16 \\ 0 & 0 & 0 & 0 \\ 28 & 0 & 0 & 0 \\ 16 & 0 & 0 & 0 \end{pmatrix}, \qquad M^{\text{struc}} = \begin{pmatrix} 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Example A.3 (Calculating an explicit formula for an MAM)**.** Consider the functional MAM of the simple motif $\mathcal{M}_6$ (Figure A.3).



$\mathcal{M}_6$

Figure A.3: The motif $\mathcal{M}_6$

We use Equation (1) in Proposition 2.1. Firstly, $m = |\mathcal{V}_\mathcal{M}| = 3$ and $|\mathcal{E}_\mathcal{M}| = 4$. The automorphism group of $\mathcal{M}_6$ has order 2, corresponding to swapping vertices 1 and 3. Hence $|S^{\sim}_{\mathcal{M},\mathcal{A}}| = |S_m|/2 = 6/2 = 3$, and suitable representatives from $S^{\sim}_{\mathcal{M},\mathcal{A}}$ are

$$S^{\sim}_{\mathcal{M},\mathcal{A}} = \left\{ \sigma_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \sigma_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}, \sigma_3 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \right\}.$$

So by Proposition 2.1, with $i = k_1$ and $j = k_3$, and writing k for k_2:

$$M_{ij}^{\text{func}} = \frac{1}{4} \sum_{\sigma \in S^{\sim}_{\mathcal{M},\mathcal{A}}} \sum_{k \in \mathcal{V}} J_{\mathbf{k},\sigma}^{\text{func}} \; G_{\mathbf{k},\sigma}^{\text{func}}$$

where since there are no missing edges in $\mathcal{M}_6$:

$$\begin{aligned} J_{\mathbf{k},\sigma}^{\text{func}} &= \prod_{\mathcal{E}^{\text{s}}_\mathcal{M}} J_{k_{\sigma u}, k_{\sigma v}} \prod_{\mathcal{E}^{\text{d}}_\mathcal{M}} (J_{\text{d}})_{k_{\sigma u}, k_{\sigma v}} \,, \\ G_{\mathbf{k},\sigma}^{\text{func}} &= \sum_{\mathcal{E}^{\text{s}}_\mathcal{M}} G_{k_{\sigma u}, k_{\sigma v}} + \sum_{\mathcal{E}^{\text{d}}_\mathcal{M}} (G_{\text{d}})_{k_{\sigma u}, k_{\sigma v}} \,. \end{aligned}$$

Writing out the sum over σ:

$$\begin{aligned} M_{ij}^{\text{func}} &= \frac{1}{4} \sum_{k=1}^{n} J_{\mathbf{k},\sigma_1}^{\text{func}} \; G_{\mathbf{k},\sigma_1}^{\text{func}} + \frac{1}{4} \sum_{k=1}^{n} J_{\mathbf{k},\sigma_2}^{\text{func}} \; G_{\mathbf{k},\sigma_2}^{\text{func}} + \frac{1}{4} \sum_{k=1}^{n} J_{\mathbf{k},\sigma_3}^{\text{func}} \; G_{\mathbf{k},\sigma_3}^{\text{func}} \\ &= \frac{1}{4} \sum_{k=1}^{n} J_{ji} J_{jk} (J_{\text{d}})_{ik} \big(G_{ji} + G_{jk} + (G_{\text{d}})_{ik} \big) \\ &\quad + \frac{1}{4} \sum_{k=1}^{n} J_{ij} J_{ik} (J_{\text{d}})_{jk} \big(G_{ij} + G_{ik} + (G_{\text{d}})_{jk} \big) \\ &\quad + \frac{1}{4} \sum_{k=1}^{n} J_{ki} J_{kj} (J_{\text{d}})_{ij} \big(G_{ki} + G_{kj} + (G_{\text{d}})_{ij} \big) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{4} J_{ij}^{\top} \sum_{k=1}^{n} (J_{\mathrm{d}})_{ik} J_{kj}^{\top} \big(G_{ij}^{\top} + (G_{\mathrm{d}})_{ik} + G_{kj}^{\top} \big) \\
&\quad + \frac{1}{4} J_{ij} \sum_{k=1}^{n} J_{ik} (J_{\mathrm{d}})_{kj} \big(G_{ij} + G_{ik} + (G_{\mathrm{d}})_{kj} \big) \\
&\quad + \frac{1}{4} (J_{\mathrm{d}})_{ij} \sum_{k=1}^{n} J_{ik}^{\top} J_{kj} \big((G_{\mathrm{d}})_{ij} + G_{ik}^{\top} + G_{kj} \big) \,,
\end{aligned}
$$

and writing this as a sum of entry-wise and matrix products:

$$
\begin{aligned}
M^{\mathrm{func}} = \frac{1}{4} \Big[J^{\top} \circ (J_{\mathrm{d}} G^{\top}) + J^{\top} \circ (G_{\mathrm{d}} J^{\top}) + G^{\top} \circ (J_{\mathrm{d}} J^{\top}) \Big] \\
+ \frac{1}{4} \Big[J \circ (J G_{\mathrm{d}}) + J \circ (G J_{\mathrm{d}}) + G \circ (J J_{\mathrm{d}}) \Big] \\
+ \frac{1}{4} \Big[J_{\mathrm{d}} \circ (J^{\top} G) + J_{\mathrm{d}} \circ (G^{\top} J) + G_{\mathrm{d}} \circ (J^{\top} J) \Big]
\end{aligned}
$$

where $A \circ B$ is an entry-wise product and AB is a matrix product. Finally, setting

$$
C = J \circ (J G_{\mathrm{d}}) + J \circ (G J_{\mathrm{d}}) + G \circ (J J_{\mathrm{d}}) + J_{\mathrm{d}} \circ (J^{\top} G) \,,
$$

and

$$
C' = G_{\mathrm{d}} \circ (J^{\top} J) \,,
$$

then we have that

$$
M^{\mathrm{func}} = \frac{1}{4} \big(C + C^{\top} + C' \big) \,.
$$

as in Table B.1, achieved with just five matrix multiplications, nine entry-wise multiplications and nine matrix additions (including the four entry-wise multiplications and four additions needed to construct the adjacency and indicator matrices).

Appendix B

Motif Adjacency Matrix Formulae

We give explicit matrix-based formulae for functional motif adjacency matrices M^{func} for all simple motifs $\mathcal{M}$ on at most three vertices, along with the anchored motifs $\mathcal{M}_{\text{coll}}$ and $\mathcal{M}_{\text{expa}}$. For structural motif adjacency matrices, simply replace J_{n}, J and G with J_0, J_{s} and G_{s} respectively. Entry-wise products are denoted by $\circ$.

Motif	C	C'	M^{func}
$\mathcal{M}_{\text{s}}$			$G + G^\top$
$\mathcal{M}_{\text{d}}$			$\frac{1}{2} G_{\text{d}}$
$\mathcal{M}_1$	$J^\top \circ (JG) + J^\top \circ (GJ) + G^\top \circ (JJ)$		$\frac{1}{3}(C + C^\top)$
$\mathcal{M}_2$	$J^\top \circ (J_{\text{d}}G) + J^\top \circ (G_{\text{d}}J) + G^\top \circ (J_{\text{d}}J)$ $+ J^\top \circ (JG_{\text{d}}) + J^\top \circ (GJ_{\text{d}}) + G^\top \circ (JJ_{\text{d}})$ $+ J_{\text{d}} \circ (JG) + J_{\text{d}} \circ (GJ) + G_{\text{d}} \circ (JJ)$		$\frac{1}{4}(C + C^\top)$
$\mathcal{M}_3$	$J \circ (J_{\text{d}}G_{\text{d}}) + J \circ (G_{\text{d}}J_{\text{d}}) + G \circ (J_{\text{d}}J_{\text{d}})$ $+ J_{\text{d}} \circ (J_{\text{d}}G) + J_{\text{d}} \circ (G_{\text{d}}J) + G_{\text{d}} \circ (J_{\text{d}}J)$ $+ J_{\text{d}} \circ (JG_{\text{d}}) + J_{\text{d}} \circ (GJ_{\text{d}}) + G_{\text{d}} \circ (JJ_{\text{d}})$		$\frac{1}{5}(C + C^\top)$
$\mathcal{M}_4$	$J_{\text{d}} \circ (J_{\text{d}}G_{\text{d}}) + J_{\text{d}} \circ (G_{\text{d}}J_{\text{d}}) + G_{\text{d}} \circ (J_{\text{d}}J_{\text{d}})$		$\frac{1}{6}C$
$\mathcal{M}_5$	$J \circ (JG) + J \circ (GJ) + G \circ (JJ)$ $+ J \circ (JG^\top) + J \circ (GJ^\top) + G \circ (JJ^\top)$ $+ J \circ (J^\top G) + J \circ (G^\top J) + G \circ (J^\top J)$		$\frac{1}{3}(C + C^\top)$
$\mathcal{M}_6$	$J \circ (JG_{\text{d}}) + J \circ (GJ_{\text{d}}) + G \circ (JJ_{\text{d}}) + J_{\text{d}} \circ (J^\top G)$	$G_{\text{d}} \circ (J^\top J)$	$\frac{1}{4}(C + C^\top + C')$
$\mathcal{M}_7$	$J \circ (J_{\text{d}}G) + J \circ (G_{\text{d}}J) + G \circ (J_{\text{d}}J)$	$J_{\text{d}} \circ (JG^\top) + J_{\text{d}} \circ (GJ^\top) + G_{\text{d}} \circ (JJ^\top)$	$\frac{1}{4}(C + C^\top + C')$
$\mathcal{M}_8$	$J \circ (GJ_{\text{n}}) + G \circ (JJ_{\text{n}})$	$J_{\text{n}} \circ (J^\top G) + J_{\text{n}} \circ (G^\top J)$	$\frac{1}{2}(C + C^\top + C')$
$\mathcal{M}_9$	$J \circ (J_{\text{n}}G^\top) + G \circ (J_{\text{n}}J^\top) + J_{\text{n}} \circ (JG)$ $+ J_{\text{n}} \circ (GJ) + J \circ (G^\top J_{\text{n}}) + G \circ (J^\top J_{\text{n}})$		$\frac{1}{2}(C + C^\top)$
$\mathcal{M}_{10}$	$J \circ (J_{\text{n}}G) + G \circ (J_{\text{n}}J)$	$J_{\text{n}} \circ (JG^\top) + J_{\text{n}} \circ (GJ^\top)$	$\frac{1}{2}(C + C^\top + C')$
$\mathcal{M}_{11}$	$J_{\text{d}} \circ (GJ_{\text{n}}) + G_{\text{d}} \circ (JJ_{\text{n}}) + J_{\text{n}} \circ (J_{\text{d}}G)$ $+ J_{\text{n}} \circ (G_{\text{d}}J) + J \circ (G_{\text{d}}J_{\text{n}}) + G \circ (J_{\text{d}}J_{\text{n}})$		$\frac{1}{3}(C + C^\top)$
$\mathcal{M}_{12}$	$J_{\text{d}} \circ (J_{\text{n}}G) + G_{\text{d}} \circ (J_{\text{n}}J) + J_{\text{n}} \circ (JG_{\text{d}})$ $+ J_{\text{n}} \circ (GJ_{\text{d}}) + J \circ (J_{\text{n}}G_{\text{d}}) + G \circ (J_{\text{n}}J_{\text{d}})$		$\frac{1}{3}(C + C^\top)$
$\mathcal{M}_{13}$	$J_{\text{d}} \circ (G_{\text{d}}J_{\text{n}}) + G_{\text{d}} \circ (J_{\text{d}}J_{\text{n}}) + J_{\text{n}} \circ (J_{\text{d}}G_{\text{d}})$		$\frac{1}{4}(C + C^\top)$
$\mathcal{M}_{\text{coll}}$	$J_{\text{n}} \circ (JG^\top)$		$\frac{1}{2}(C + C^\top)$
$\mathcal{M}_{\text{expa}}$	$J_{\text{n}} \circ (J^\top G)$		$\frac{1}{2}(C + C^\top)$

Table B.1: Functional motif adjacency matrix formulae

Appendix C

Further Notes

C.1 Computation

C.1.1 Hardware and software

The hardware used for computation was an *Intel Core i7-4790* CPU at 3.60 GHz, with 32 GB of RAM. The software used was R 3.5.1 [34], along with several R packages:

- **igraph** [11] for plotting networks
- **LICORS** [18] for an implementation of k-means++
- **mclust** [36] for an implementation of ARI
- **rnaturalearth** [38] for world territory boundary data
- **RSpectra** [33] for eigendecomposition of sparse matrices
- **USAboundaries** [30] for US county and state boundary data

C.1.2 Timings for MAM computations

We record timings (in seconds) for the MAM formulae given in Table B.1. We test on DSBMs (Section 4.1) with $k = 1$, and vary the graph size n and sparsity parameter p.

p	$\mathcal{M}_{\mathrm{s}}$	$\mathcal{M}_{\mathrm{d}}$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$	$\mathcal{M}_7$	$\mathcal{M}_8$	$\mathcal{M}_9$	$\mathcal{M}_{10}$	$\mathcal{M}_{11}$	$\mathcal{M}_{12}$	$\mathcal{M}_{13}$
0.0001	0.013	0.012	0.017	0.029	0.034	0.015	0.028	0.022	0.022	0.019	0.030	0.019	0.042	0.021	0.016
0.001	0.013	0.011	0.016	0.035	0.027	0.017	0.028	0.024	0.023	0.026	0.027	0.018	0.021	0.022	0.016
0.01	0.013	0.012	0.024	0.028	0.028	0.016	0.028	0.022	0.032	0.021	0.026	0.020	0.023	0.023	0.017
0.1	0.014	0.019	0.019	0.031	0.029	0.019	0.033	0.025	0.032	0.023	0.028	0.023	0.026	0.025	0.019

Table C.1: Timings for MAM computation with $n = 100$

p	$\mathcal{M}_\mathrm{s}$	$\mathcal{M}_\mathrm{d}$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$	$\mathcal{M}_7$	$\mathcal{M}_8$	$\mathcal{M}_9$	$\mathcal{M}_{10}$	$\mathcal{M}_{11}$	$\mathcal{M}_{12}$	$\mathcal{M}_{13}$
0.0001	0.13	0.14	0.14	0.32	0.14	0.13	0.14	0.31	0.13	0.21	0.22	0.21	0.20	0.34	0.16
0.001	0.30	0.13	0.15	0.16	0.16	0.14	0.16	0.32	0.14	0.48	0.37	0.29	0.31	0.29	0.17
0.01	0.11	0.14	0.17	0.19	0.14	0.13	0.21	0.18	0.18	0.64	0.73	0.89	0.46	0.56	0.18
0.1	0.23	0.22	0.60	1.1	0.57	0.24	1.4	0.86	0.69	1.5	2.3	1.6	1.6	1.6	0.67

Table C.2: Timings for MAM computation with $n = 1000$

p	$\mathcal{M}_\mathrm{s}$	$\mathcal{M}_\mathrm{d}$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$	$\mathcal{M}_7$	$\mathcal{M}_8$	$\mathcal{M}_9$	$\mathcal{M}_{10}$	$\mathcal{M}_{11}$	$\mathcal{M}_{12}$	$\mathcal{M}_{13}$
0.0001	11	12	12	12	12	12	12	12	12	41	55	37	38	34	15
0.001	13	12	13	13	12	12	13	12	12	61	89	54	56	48	15
0.01	13	13	36	36	14	13	82	36	36	150	230	130	130	99	36
0.1	33	31	170	260	160	53	410	210	210	700	1100	520	760	580	150

Table C.3: Timings for MAM computation with $n = 10\,000$

C.2 Data preprocessing

All real networks were preprocessed by restriction to their largest connected component. The Unicode Languages network in Section 5.4 was also preprocessed to remove territories with under one million inhabitants and languages with under one million speakers. Vertex and edge counts of all networks are stated *after* this preprocessing.

C.3 US map

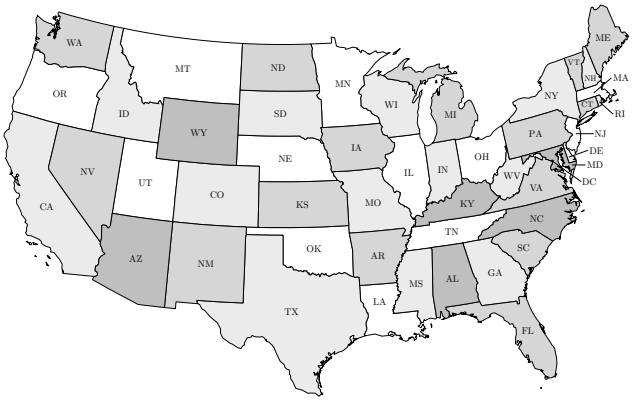


Figure C.1: US map with state boundaries and state abbreviations

C.4 Word count

The word count of this dissertation is 6231, obtained using TEXcount by running

```
texcount -relaxed -inc -0 -sum=1,1,1,0,0,0,0 <dissertation.tex>.
```

References

[1] L. A. Adamic and N. Glance. The political blogosphere and the 2004 US election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43. ACM, 2005.

[2] C. Aicher, A. Z. Jacobs, and A. Clauset. Adapting the stochastic block model to edge-weighted networks. *arXiv e-prints*, 2013. arXiv:1305.5782, `http://arxiv.org/abs/1305.5782v1`.

[3] R. Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118(21):4947–4957, 2005.

[4] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[5] R. B. Bapat. *Graphs and Matrices*, volume 27. Springer, London, 2010.

[6] A. R. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[7] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. In *Proceedings of the Princeton Conference in Honor of Professor S. Bochner*, 1969.

[8] F. Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, 2005.

[9] A. Clauset, E. Tucker, and M. Sainz. Filmtipset user movie ratings. The Colorado Index of Complex Networks. `https://icon.colorado.edu/`. Accessed on 15/04/2019.

[10] Creative Commons. GeoNames. `https://www.geonames.org/`. Accessed on 24/03/2019.

[11] G. Csárdi and T. Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[12] M. Cucuringu, H. Li, H. Sun, and L. Zanetti. Hermitian matrices for clustering directed graphs: Insights and applications. *Submitted*, 2019.

[13] S. Demeyer, T. Michoel, J. Fostier, P. Audenaert, M. Pickavet, and P. Demeester. The index-based subgraph matching algorithm (ISMA): Fast subgraph enumeration in large networks using optimized search trees. *PLOS ONE*, 8(4):1–15, 2013.

[14] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274. ACM, 2001.

[15] W. E. Donath and A. J. Hoffman. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 15(3):938–944, 1972.

[16] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[17] L. Florescu and W. Perkins. Spectral thresholds in the bipartite stochastic block model. In *Conference on Learning Theory*, pages 943–959, 2016.

[18] G. M. Goerg. *LICORS: Light Cone Reconstruction of States – Predictive State Estimation from Spatio-Temporal Data*, 2013. R package version 0.2.0.

[19] S. Guattery and G. L. Miller. On the performance of spectral graph partitioning methods. In *SODA*, volume 95, pages 233–242, 1995.

[20] S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19(3):701–719, 1998.

[21] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[22] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[23] P.-M. Jacob and A. Lapkin. Statistics of the network of organic chemistry. *Reaction Chemistry & Engineering*, 3(1):102–118, 2018.

[24] E. D. Kolaczyk and G. Csárdi. *Statistical Analysis of Network Data with R*, volume 65. Springer, New York, 2014.

[25] KONECT: The Koblenz Network Collection. American Revolution network dataset. `http://konect.cc/networks/brunson_revolution`. Accessed on 24/03/2019.

[26] KONECT: The Koblenz Network Collection. Unicode Languages network dataset. `http://konect.cc/networks/unicodelang`. Accessed on 24/03/2019.

[27] J. Leskovec and A. Krevl. Astrophysics collaboration network, SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data/ca-AstroPh.html`. Accessed 15/04/2019.

[28] H. Lütkepohl. *Handbook of Matrices*, volume 1. Wiley, Chichester, 1996.

[29] M. Meilă and W. Pentney. Clustering by weighted cuts in directed graphs. In *SDM*, 2007.

[30] L. A. Mullen and J. Bratt. USAboundaries: Historical and contemporary boundaries of the United States of America. *Journal of Open Source Software*, 3:314, 2018.

[31] B. Nadler, S. Lafon, I. Kevrekidis, and R. R. Coifman. Diffusion maps, spectral clustering and eigenfunctions of Fokker–Planck operators. In *Advances in Neural Information Processing Systems*, pages 955–962, 2006.

[32] M. Newman. The physics of networks. *Physics Today*, 61(11):33–38, 2008.

[33] Y. Qiu and J. Mei. *RSpectra: Solvers for Large-Scale Eigenvalue and SVD Problems*, 2018. R package version 0.13-1.

[34] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, 2018.

[35] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[36] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: Clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):205–233, 2017.

[37] J. Shi and J. Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[38] A. South. *rnaturalearth: World Map Data from Natural Earth*, 2017. R package version 0.1.0.

[39] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1451–1460. International World Wide Web Conferences Steering Committee, 2017.

[40] U.S. Census Bureau. County-to-county migration flow files. `https://www.census.gov/population/www/cen2000/ctytoctyflow/index.html`, 2002. Accessed on 02/03/2019.

[41] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[42] U. Von Luxburg, O. Bousquet, and M. Belkin. On the convergence of spectral clustering on random samples: The normalized case. In *Learning Theory*, pages 457–471. Springer, Berlin, Heidelburg, 2004.

[43] D. Wagner and F. Wagner. Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. Springer, Berlin, Heidelburg, 1993.

[44] S. Wernicke. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(4):347–359, 2006.

[45] S. Wernicke and F. Rasche. FANMOD: A tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.

[46] D. B. West. Introduction to graph theory home page. `https://faculty.math.illinois.edu/~west/igt/`. Accessed on 01/04/2019.

[47] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564. ACM, 2017.