Efficient Detection of Overlapping Communities in Large Graphs

Richard Millson

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science Mathematics and Statistics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

This thesis proposes an algorithm for the efficient detection of overlapping communities in large graphs. Only super-fast local algorithms like Louvain are really practical for very large datasets, but they tend to give hierarchical rather than overlapping partitions. We develop some techniques that let you get reasonable families of overlapping partitions while preserving most of the good properties of Louvain. We build off an advance in the efficient detection of separated communities, the multilevel Louvain method, and draw inspiration from the Wang-Landau efficiency improvement to Markov chain Monte Carlo sampling. Partitions are iteratively proposed by Louvain, with the internal edges of the best parts downweighted after each step. This suppresses the dominant parts in subsequent partitions, allowing alternative parts to appear. The result is an ensemble of parts describing the overlapping structure of the network.

Acknowledgement

I would first like to thank my supervisor Professor Aaron Smith. I am extremely grateful for your immense support, creative ideas, guidance, detailed explanations and instruction, and countless conversations and emails.

Thank you to Professors Dave Campbell and François Théberge for agreeing to evaluate this thesis; I look forward to hearing your thoughts.

I am thankful for the conversations I had with my uncle John Millson; explaining my work to a mathematician of another field, differential geometry, was an invaluable exercise in exposition. Thank you also for your support and encouragement.

I would like to thank my parents for their love and support throughout.

Finally I would like to thank my dearest Morgan; you are a continual source of strength and inspiration, especially so through the dark months of lockdowns and uncertainty.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Each of us belongs to a multitude of communities, whether through work, hobbies, family, as alumnus of our past schools, etc. If asked to reduce our identity to just one community, much about who we are and many of our relationships would be unexplained. Yet this is what most community detection algorithms do, uncovering only the dominant and separated communities. Instead we are interested in uncovering communities with overlapping structure. This property of overlapping communities is well known in sociology [30], but it is not just social networks that exhibit it; for instance many words in a word association network and proteins in a protein-protein interaction network have been shown to belong to more than one community [63]. Further, many proteins belong to several protein complexes that are used for different cellular functions [35]. Current algorithms developed for overlapping communities suffer from computational inefficiencies that prevent them from being applied to truly large networks. We build off an advance in the efficient detection of separated communities, the multilevel Louvain method, and draw inspiration from the Wang-Landau efficiency improvement to Markov chain Monte Carlo sampling to develop an efficient method for detecting overlapping communities.

We now describe a simple model of the underlying data generation process. Let V be a set of vertices and A and B be two partitions of V. We will construct a graph by considering each pair of vertices from V one at a time. If two vertices belong to the same part A_i of partition A, then we will form an edge between them with probability $p_A > 0$. Edges are similarly created for vertices that belong to the same part B_j of B with probability $p_B > 0$. Let E be the set of all edges created this way. Thus we get the graph $G = (V, E)$.

Given graph G, can we recover the partitions A and B that were used to generate it? Since the partitions determined edge formation, one strategy is to look for subsets that have dense internal edges and sparse external edges. A subset of vertices with this property is said to be **modular**. Thus we want many edges within each part, and few edges between the parts.

There exist graph partitioning algorithms for this task, however most algorithms return only a single partition. If A, B are similar-looking partitions with $p_A > p_B$, then this is likely to be a partition close to A as its parts are more modular. Modularity will be formally defined in Section 2.3.1 and can be used to determine which partitions are expected to be more attractive to a partitioning algorithm. If the partitioning algorithm is non-deterministic and hence has a stochastic component, then repeatedly asking it for partitions may eventually return both A and B. However this could require a prohibitive number of replicates, or may never occur. This is a problem of recovering multiple overlapping communities, as each vertex belongs to a part of both A and B.

This thesis proposes a method for efficiently detecting overlapping partitions. In a given graph there may be a dominant partition like A was above, whose communities have denser internal edges than the alternatives and that broadcasts a stronger signal to the partitioning algorithm. Some partitioning algorithms accept edge weights that give the strength of the relationship, with the definition of modularity naturally extending to trying to maximize the sum of weights within parts and minimize the sum between. Assigning a weight to each edge and downweighting the internal edges of the dominant parts then makes these parts less attractive to the partitioner, thus increasing the probability that inferior parts will be proposed. This procedure is described in Algorithm 1.

Algorithm 1: Overlapping Community Detection (Sketch)

Input: graph $G = (V, E)$,
graph clustering algorithm $f : G \to \Omega$ where Ω is the space of all partitions of V
If G is unweighted, assign each edge a weight of 1
Set $t = -1$ and $G_0 = G$
repeat
  $t = t + 1$
  Sample a partition $C_t = f(G_t)$
  Compute how modular each part of C_t is
  Select some number of the most modular parts
  Downweight the edges of G_t that are internal to these parts
  (so that they are less likely to be chosen by the clusterer next time)
  Set G_{t+1} to be this downweighted graph
until C_t is near $\text{span}(C_1, \ldots, C_{t-1})$;
Cluster the parts of partitions $C_1, \ldots, C_t$, only keeping the best representative from each cluster
return the representative parts from $C_1, \ldots, C_t$

Consider the set of all partitions of a graph. We can construct a graph from

this set of partitions by creating an edge between two partitions if they differ by the membership of exactly one vertex. Local partitioning algorithms can be viewed as a walk over this graph as they change the community of one vertex at each step. Our algorithm seeks to influence this walk to steer a partitioner away from previously visited regions and make unexplored regions more attractive.

Proceeding in this manner gives a sequence of partitions of decreasing quality, probability and explanatory power. Section 3.3.1 develops a stopping criteria for determining when the algorithm stops providing previously unseen parts. Section 3.3.2 proposes a method for taking the final ensemble of partitions and producing a mixture of parts.

Previous results are covered in Section 2, specifically covering models with community structure (Section 2.2) and methods for their detection (Section 2.3). Overlapping communities are by definition multi-modal, i.e. having multiple competing partitions that each explain part of the observed community structure. This is likely to give rise to the related challenges of low ergodicity, when some partitions are unlikely to be sampled, and bottlenecks, when sampling algorithms are unable to move between competing partitions. These are discussed in greater generality in Section 2.4. Section 2.4.3 discusses the Wang-Landau sampling method that seeks to solve these problems, and whose core idea of decreasing the probability of sampling already seen states we applied to our algorithm's downweighting.

Overlapping	Efficient	Stable	Algorithm	Section
	X		Multilevel Louvain	2.3.2
	X	X	Ensemble Clustering for Graphs	2.3.3
X		X	Clique Percolation Method	2.3.4
			Spectral Clustering	2.3.5
X		X	Bayesian Inference	2.3.6
X	X	X	Ego-Splitting [28]	
X	X	X	Our Algorithm	3.1

Table 1.1: Informal Comparison of Community Detection Algorithms

Table 1.1 gives an informal comparison of existing algorithms and shows where our contribution fits in. It details whether they can detect and describe overlapping community structure, whether they are computationally efficient i.e. have a fast run time, and whether they are stable and so produce the same partitions repeatedly. The multilevel Louvain method is the fastest, however it is stochastic and so unstable. It also cannot describe overlapping communities as it returns only a single partition. Ensemble Clustering for Graphs uses multilevel Louvain to generate an ensemble of partitions that it uses to create a stable final partition. It also only proposes a single partition and so does not handle the overlapping case. Spectral clustering relies on a stochastic subalgorithm such as k-means clustering and so is not stable. It is

also inefficient as it involves computing eigenvectors of a large matrix. The Clique Percolation Method was developed specifically to detect overlapping communities, and is deterministic and hence stable, however it is also computationally inefficient. Bayesian inference has been proposed as a method to detect overlapping communities; however as a Markov chain Monte Carlo method it is inefficient. In theory it is stable if allowed to sample for long enough, however in practice may be unstable given different starting conditions and a finite run time.

It is in this context that we introduce a method for detecting overlapping communities. Our algorithm is computationally efficient as it uses the efficient multilevel Louvain to generate a small ensemble, and is stable as it enumerates the likeliest partitions when generating this ensemble. Additionally our reweighting method can be configured to upweight and converge on a stable partition if only a single partition is desired.

Section 3 contains our contributions. Section 3.2 develops formulas for determining the appropriate downweight and discusses the possible approaches. In Section 3.4.1 we show that for graphs generated from multiple overlapping partitions, multilevel Louvain will always choose only one of the constituent partitions and will never pick a more refined partition with more parts that captures all the partitions. In Section 3.4.2 we describe a nucleation phenomenon that biases Louvain and other greedy algorithms toward specific partitions. Section 4 applies the proposed algorithms to two datasets.

Chapter 2

Previous Results

2.1 Networks as Graphs

A **graph** is a mathematical construction that captures the relationship between objects. More formally, a graph is a pair $G = (V, E)$ where V is a set of objects called vertices, and E is a set of pairs of vertices called edges. For example the sets $V = \{a, b, c, d\}$ and $E = \{(a, b), (b, c)\}$ define the graph in Figure 2.1. A network that captures relationships between objects induces a graph where the objects in the network become the vertices and where an edge is placed between two objects if a relationship exists between them. For example in a social network, the people are the vertices and an edge exists between two people if they are friends.

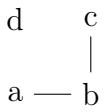

Figure 2.1: An example undirected graph. Vertex d forms its own disconnected component.

Graphs can be categorized by special named properties. Figure 2.1 is **disconnected** as there is no edge between vertex d and any other. It is also an **undirected** graph, as the edges between nodes were not given a direction. However there exist graphs where relationships are not reciprocated; for example, it is possible to follow someone on Twitter without them following you back. This would be an example of a directed graph. A **weighted** graph has numeric values assigned to each edge. For example in Figure 2.2, the vertices are a subset of Canadian cities with an edge placed connecting two cities if there exists a direct flight between them. The edges were then weighted with the corresponding flight time. As the flight time is not the same in each direction (e.g. due to the polar jet stream above Canada), each edge has a different weight.

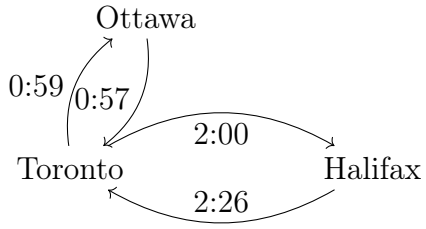

Figure 2.2: A weighted directed graph showing flight times between some Canadian cities.

Some graphs are famous and have been given special names. The **complete graph** K_n on n vertices is the graph where every pair of vertices is connected with an edge, for example Figure 2.3. A **subgraph** H of G is a graph formed from a subset of the vertices and edges of G with the condition that the vertex subset must contain all endpoints of the edge subset. A **clique** is a subgraph that is complete. A **tree** is a graph where any two vertices are connected by exactly one path, for example Figure 2.4. A **path graph** P_n has n vertices, 2 with degree 1 and $n-2$ with degree 2, and can be drawn so that all vertices and edges lie on a line.

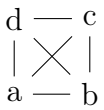

Figure 2.3: The complete graph on 4 vertices.

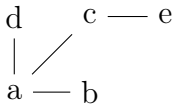

Figure 2.4: A tree on 5 vertices.

A **lattice graph** or **square grid graph** has vertices that correspond to integer coordinates in some d-dimensional space, with an edge exists between a pair of vertices if their corresponding points have a distance of 1. We invent some notation to describe them by counting the number of points on each dimension of the lattice. For example the [2, 3, 2]-block lattice depicted in Figure 2.5 has 2 points along first up-down axis, 3 along the second left-right axis, and 2 along the third in-out axis. It can also be thought of as the Cartesian product of path graphs $P_2 \square P_3 \square P_2$. More generally, the d-dimensional $[n_1, \ldots, n_d]$-block lattice will have vertices corresponding to points in $\mathbb{R}^d$ with first coordinate in the range $1, \ldots, n_1$, second coordinate in the range $1, \ldots, n_2$, etc. It is generated by the Cartesian product $P_{n_1} \square P_{n_2} \square \cdots \square P_{n_d}$.

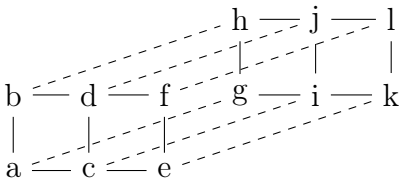

Figure 2.5: The [2, 3, 2]-block lattice graph.

Let $G = (V, E)$ be a graph and $V' \subseteq V$ a subset of vertices. The **induced subgraph** of graph G with respect to the subset of vertices V' is the graph with vertices V' and all the edges from E that connect pairs of vertices in V'. A k**-core** of a graph is the largest induced subgraph such that every vertex has degree at least k. The k-core can be computed by recursively removing all vertices of degree less than k, which changes the degrees of the remaining vertices, until no vertices can be removed. For example, the 2-core of Figure 2.4 is the empty graph, because a tree does not contain a subgraph where each vertex has at least degree two. This is despite the fact that vertex a has degree 3; vertices b and d have degree 1 and so must be removed, and once they are removed vertex a has a degree of only 1.

A graph can be represented as an **adjacency matrix**. Suppose $G = (V, E)$ is an unweighted graph with vertices are labelled $1, 2, \ldots, |V|$ and adjacency matrix A. Then the (i, j)-th position of A is one if there is an edge from vertex i to vertex j, and zero if there is not. More formally,

$$A_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

If the graph is undirected, the adjacency matrix is symmetric. If the graph is weight, the value at A_{ij} is replaced with the weight assigned to edge (i, j). The graph from Figure 2.1 has the adjacency matrix given by Figure 2.6.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 2.6: Adjacency matrix of Figure 2.1.

The **degree** of a vertex v is the number of vertices that v shares an edge with. A k**-regular** graph is one where each vertex has the same degree k. In a directed

graph, the **in degree** of v is the number of edges coming into v and the **out degree** is the number of edges going out from v. In a weighted graph, the **weighted degree** of v is the sum of all edge weights for edges adjacent to v. In Figure 2.2, Toronto has an in degree and out degree of 2 while Ottawa and Halifax have in degree and out degrees of 1.

2.2 Network Modelling

A model for a network graph is a collection

$$\{\mathbb{P}_\theta(G),\ G \in \mathcal{G},\ \theta \in \Theta\}$$

where $\mathcal{G}$ is an ensemble of possible graphs, $\mathbb{P}_\theta$ is a probability distribution on $\mathcal{G}$, and θ a vector of parameters from the space of all possible parameters Θ. In some cases only a generative procedure will be specified and the probability distribution this induces will not be explicitly stated. Csárdi and Kolaczyk broadly define two classes of network models; mathematical models that lend themselves to mathematical analysis, and statistical models that lend themselves to model fitting and assessment [23]. Both can be used to analyze graph data as well as generate random graphs and differ in how they specify $\mathbb{P}$. Robins and Morris articulate the difference as, "a good [statistical network graph] model needs to be both estimable from data and a reasonable representation of that data, to be theoretically plausible about the type of effects that might have produced the network, and to be amenable to examining which competing effects might be the best explanation of the data" [78].

In this thesis we are interested in recovering the community structure parameter that created an observed graph. Of especial interest in solving this problem is the stochastic block model (SBM), one of the simplest statistical models that gives rise to community structure. We first introduce some mathematical graph models before moving on to the SBM and some of its generalizations, including those that create overlapping communities.

2.2.1 Mathematical Models

The simplest random graph model is the **Erdös-Rényi random graph** $G(n, p)$ with $0 < p < 1$. Under it each of the $n(n-1)/2$ possible edges between the n vertices are included independently with probability p.

Unsatisfied with the ability of the Erdös-Rényi random graph to describe real-world networks, Watts and Strogatz introduced the idea of **small-world networks** that possess two properties [86]. The first is the **average shortest path length** or **characteristic path length** which gives how close we expect any two randomly chosen vertices to be to each other. A small average shortest path length is sometimes

(confusingly) called the small-world effect. A graph is said to possess the small-world effect if the average shortest path length is proportional to the logarithm of the number of vertices. Erdös-Rényi random graphs along with most real-world networks have small average shortest path lengths, while lattices have large values. The second property is the clustering coefficient, which they introduced to measure the extent that vertices in a graph can be clustered together such that there is a high edge density within these clusters. In social networks, it measures whether all the friends of an individual know each other. The coefficient for vertex v is the number of edges between the neighbours of v divided by the total possible number of edges $\binom{k}{2} = n(n-1)/2$ which would occur if the neighbours of v formed a clique. The clustering coefficient for the graph is then the average of all individual clustering coefficients. Erdös-Rényi random graphs have a small clustering coefficient whereas real-world networks have a large value.

Small-world networks capture the colloquial idea of six degrees of separation, i.e. where any two people are six social connections or less away from each other. Watts and Strogatz showed that small-world networks appear in the neural network of worms, the power grid of the western United States, and the collaboration graph of actors appearing in the same film [86]. The co-occurrence of consecutive words in sentences has been shown to be a small-world network [15]. A small average shortest path length of 2.63 was observed and roughly agreed with the expected average shortest path length of 3.03 in a random graph of the same number of edges and vertices. A large clustering coefficient of 0.687 relative to the expected value of 0.000155 was also reported. Note that [82] claim the prevalence of small-world networks in the real world has been overstated as they argue the clustering coefficient should be compared to an equivalent lattice graph instead of random graph.

The **Watts-Strogatz model** was introduced to generate small-world networks, i.e. to have a high clustering coefficient and small shortest average path. It starts with a ring lattice, a special k-regular graph, which by construction has a high clustering coefficient. Some percentage of the edges are then randomly rewired to decrease the average shortest path. These graphs have a homogeneous degree distribution centered around k. More details can be found in their original paper [86] or in Section 5.4.1 of [23]. Figure 2.7 gives an example of a k-regular graph, a Watts-Strogatz graph generated by rewiring the first graph's edges with probability 0.1, and an entirely random graph generated by randomly rewiring all the edges from the first.

A graph's degree distribution is another property that has received attention when comparing models to real-world networks. In the Erdös-Rényi random graph the degree of a given vertex follows a binomial distribution as the existence of each edge is the outcome of a Bernoulli random variable. Recall that the Poisson distribution is the limiting distribution of the binomial distribution when the expected number of successes is fixed and the number of trials goes to infinity. Hence a Poisson with $\lambda = np$ can be used as an approximation for a binomial if n is sufficiently large and

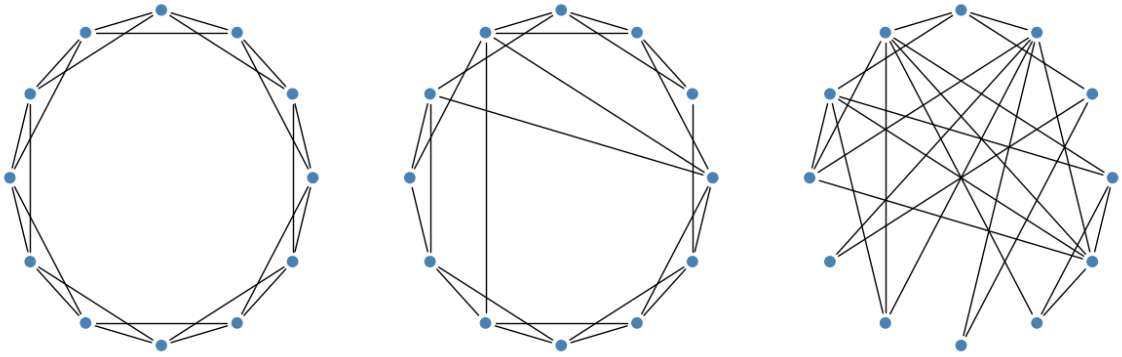

Figure 2.7: From left to right, a 4-regular ring lattice, a Watts-Strogatz graph with rewiring probability 0.1, and an Erdös-Rényi random graph of fixed edge count. All graphs were on 12 vertices and 24 edges. The randomness of the graphs increases from left to right.

p sufficiently small, with a good approximation for $n \geq 20$ and $p \leq 0.05$, and an excellent approximation for $n \geq 100$ and $np \leq 10$ [41]. Thus in that regime the degree distribution for the Erdös-Rényi random graph is approximately Poisson(np) for large n.

It has been argued that real-world networks follow a **power-law** distribution for their degree where $\mathbb{P}(\deg(v) = k) \sim k^{-\gamma}$ as $n \to \infty$ and where $2 < \gamma < 3$ [5]. Such graphs are said to be **scale-free**. Power-law distributions notably possess a long tail which manifests in the existence of well connected "hub" vertices that have high degree. However there has been skepticism about this result, and more recently it was shown that few real-world networks display convincing evidence for scale-freeness [14].

The **Lancichinetti-Fortunato-Radicchi (LFR)** family of artificial networks was designed to resemble real-world networks and is commonly used to benchmark clustering algorithms [51]. They notably have heterogeneous distributions for vertex degrees and community sizes. A comparative analysis of community detection algorithms using this benchmark was conducted in [89]. They were later extended to produce overlapping communities [49]. LFR has mixing parameter μ which controls the expected percentage of edges that connect vertices of differing communities. They were used for benchmarking [75] by the authors of ensemble clustering for graphs (introduced in Section 2.3.3) .

2.2.2 Stochastic Block Model (SBM)

The **stochastic block model (SBM)**, or **planted partition model**, is a generative model of a random graph originally proposed in [44]. It is significant in that it

produces definite communities, i.e. subgraphs with greater edge density within than without. The model assumes that the n vertices $\{1, 2, \ldots, n\}$ are partitioned into r disjoint subsets, i.e. communities, $C_1, \ldots, C_r$. This partition can be described by the membership vector $c \in \{1, \ldots, r\}^n$ where c_i indicates the community membership of vertex i. The symmetric $r \times r$ **preference matrix** P has entries p_{ij} that denote the probability of an edge forming between a vertex in community i and a vertex in community j. The simplest case is when matrix P has value $p \in [0, 1]$ along the diagonals and $q \in [0, 1]$ elsewhere. There p is the probability of edge formation within a community and q the probability of edge formation between communities.

Vertices then need to be assigned to a block. This could be the outcome of a random variable with vertices assigned to a block randomly. Alternatively, a vector of block sizes $n = [n_1, \ldots, n_r]$ satisfying $\sum_{i=1}^{r} n_i = |V|$ could be specified, where n_i vertices are assigned to block C_i.

A graph is then randomly generated according to this matrix P; each vertex in C_i and each vertex in C_j are connected with an edge with probability p_{ij}. Given a graph that has been generated this way, the problem then becomes how to recover the original partition, i.e. to assign each vertex to their correct communities.

Under this model all vertices within a given community are **exchangeable**, i.e. indistinguishable, and have the same expected degree. However as we noted earlier real-world networks are observed to have non-uniform power-law degree distributions, and thus this oversimplification makes the SBM poorer at fitting naturally occurring networks. This is argued in Section V.G of [92] where it is noted that real networks are not generated by the SBM. The **degree-corrected** stochastic block model [47] corrects for this by taking a degree sequence as an additional parameter. Nevertheless for the sake of simplicity we will use the standard SBM in our analysis instead of the more complex degree-corrected version.

The statistic of counting the number of edges within and between communities is sufficient for edge formation probabilities, i.e. no other statistic that can be calculated from the same sample provides any additional information as to the value of the parameters. This is due to the exchangeability of vertices under the stochastic block model. If A is the adjacency matrix of G, then the number of edges between communities i and j is given by

$$e_{ij} = \begin{cases} \frac{1}{2} \sum_{v,v' \in G} A_{vv'} \delta(c_v, i) \delta(c_{v'}, j) & i \neq j \\ \sum_{v,v' \in G} A_{vv'} \delta(c_v, i) \delta(c_{v'}, j) & i = j \end{cases}$$

The factor of $\frac{1}{2}$ is because for $i \neq j$ there are two ways $\{(i, j), (j, i)\}$ to count the same quantity as $e_{ij} = e_{ji}$, and so this overcounting needs to be accounted for. An $r \times r$-matrix E of all inter-community edge counts can then be formed from these e_{ij} and is equivalent to the parameter P.

In the first formulation involving P, individual edges were placed according to a Bernoulli distribution. Under the edge count formulation, communities can be

thought of as nodes in a multigraph that allows multiple edges, with the number of edges following a geometric distribution.

Given a **graphon**, defined as a symmetric measurable function $W : [0,1]^2 \to [0,1]$, you can construct a random graph of size n by sampling $U_i, U_j, U_{i,j} \sim Uniform[0,1]$ and then adding (i,j) to the edge set if and only if $U_{i,j} \leq W(U_i, U_j)$. It turns out that all exchangeable random graphs are of this form.

The Erdös-Rényi random graph $G(n,p)$ corresponds to a graphon $W(x,y) = p$ for all $x, y \in [0,1]$. The SBM corresponds to graphons of the form $W(x,y) = c_{a,b}\mathbf{1}(\ell_a \leq x \leq r_a, \ell_b \leq y \leq r_b)$, where $0 = \ell_0 < r_0 = \ell_1 < r_1 = \ell_2 \leq \ldots < r_k = 1$ define the block boundaries and $P(a,b) = c_{a,b}$ defines the edge formation probabilities between blocks a, b.

SBMs can be further generalized in another way. In the **latent space model** [42], each vertex is assumed to have a position in some underlying latent space. The probability that two vertices share an edge is then defined to be dependent on their distance and increase the closer they are to each other. In the **latent position cluster model** [40], vertices are assumed to be grouped in clusters within this space, resulting in more of a community structure.

2.2.3 Models with Overlapping Communities

The edge formation process that creates communities within the SBM can be thought of as separate Erdös-Rényi random process restricted to the vertices of each community. In this way the stochastic block model can be viewed as a mixture model of Erdös-Rényi processes restricted to *disjoint* subsets of the vertices. However the subsets defining the communities need not be disjoint, and allowing for unconstrained subsets gives rise to overlapping communities.

An "overlapping community" is obtained by combining two "single community" graphons W_1, W_2 into a single "overlap" graphon W. Natural pointwise operations for combining might be addition $W = \min(1, W_1 + W_2)$, maximum $W = \max(W_1, W_2)$, the "or" operation $W = W_1 + W_2 - W_1 W_2$, and so on.

Stochastic block models assign each vertex to only one community, whereas in reality a vertex could belong to multiple overlapping communities. Consider the following generative procedure that produces overlapping communities. Let $C^{(1)}, \ldots, C^{(k)}$ be k partitions of a vertex set V, let $P_1, \ldots, P_k$ be preference matrices corresponding to a respective partition, and $SBM(C^{(1)}, P_1), \ldots, SBM(C^{(k)}, P_k)$ be the SBMs generated from these parameters. Let $Simplify(G)$ take a multigraph G and return a simple graph with all multi-edges replaced with only a single edge. A **graph sum** of a set of graphs is the graph whose adjacency matrix is the sum of the adjacency matrices of the graphs being summed [87]. Then the graph sum

$$G = Simplify\left(SBM(C^{(1)}, P_1) + \cdots + SBM(C^{(k)}, P_k)\right) \tag{2.2.1}$$

is a mixture of SBMs. Thus there is a simple latent-parameter model comprised of the multiple independent generative processes that produce the edges. Note that this is in contrast to the standard definition of a mixture, i.e. where f is said to be a mixture distribution of k component distributions $f_1, \ldots, f_k$ if $f(x) = \sum_{i=1}^{k} \lambda_i f_i(x)$ with λ_i being the mixing weights, $\lambda_i > 0$, $\sum_i \lambda_i = 1$.

In practice by the homophily property the preference matrices P_i are likely to be non-independent, with some individuals belonging to several common groups of interest. This can be modelled. For simplicity in our experiments where synthetic graphs are generated the partitions are assumed to be independent.

A mixture of SBMs can still be captured by a larger, more complex SBM. To see how this is possible we must first introduce the idea of a partition refinement. Sets form an algebra, and the **refinement** is just the algebra generated by some collection of sets. In the case of partitions the collection is the parts of each partition as sets.

SBMs can still describe multiple memberships, albeit at the cost of a much more complex model. This larger model is introduced as the **overlapping stochastic block model (OSBM)** in Section 4 of [2]. Let there be n vertices, t communities, and a probability distribution p on $\{0,1\}^t$ i.e. over community membership profiles. Then for each vertex v a membership vector $C(v) \in \{0,1\}^t$ is drawn under p. Let there be a symmetric function $f : \{0,1\}^t \times \{0,1\}^t \to [0,1]$ that assigns edge probabilities to pairs of membership vectors. Then for each unique pair of vertices (v,u), an edge forms with probability $f(C(v), C(u))$. They note that the overlapping SBM can be represented as an SBM with $k = 2^t$ communities where each community represents a possible membership vector in $\{0,1\}^t$.

The OSBM is even more complex than the refined SBM we introduced above. This is because it does not take advantage of the fact that some communities will be disjoint if they come from the same SBM. Furthermore we might expect that many of the 2^t communities under the OSBM are empty, with no vertices belonging to that combination of the t communities. Under the refined partition these empty sets will not appear, and thus the OSBM would be sparser. Even knowledge of which parts come from the same partition and hence are disjoint does not reduce the size of the OSBM compared to the mixture of SBMs. Suppose there are R partitions each with r_i parts (i.e. communities), and hence $t = \sum_{i=1}^{R} r_i$. Then the OSBM lets every possible combination of parts from each of the R partitions become the blocks of the new larger model. Thus there are $\prod_{i=1}^{R} r_i$ such combinations of parts when one part is chosen from each partition, and $\frac{1}{2}\left(\prod_{i=1}^{R} r_i\right)^2$ parameters corresponding to each of the edge formation probabilities. This is significantly more parameters than in the mixture model we described in Equation 2.2.1 with its $\frac{1}{2}\sum_{i=1}^{R} (r_i)^2$ parameters.

A further generalization is the **mixed membership model (MMM)** which allows every vertex to partially belong to every community according to that vertex's individual community membership vector. Each individual's proportion represents

how much they reflect each community, such as how many people they know from school, from a club, from their neighbourhood, etc. More information can be found in David Blei's notes on MMM [8] or in another introduction to MMM coauthored by Blei [3]. MMM's membership expression for each community lies in the interval $[0, 1]$ whereas under the OSBM it is a boolean true/false value from $\{0, 1\}$.

Example 2.2.1. Let A and B be partitions of n, with A labelling vertices as either a or $\neg a$ and similarly B labelling them as either b or $\neg b$. If two vertices both have label a then they will share an edge with probability p_a, and similarly two vertices with label b share an edge with probability p_b. Otherwise if they share no labels then two vertices share an edge with probability 0. This constructs a graph as the mixture

$$G \sim SBM\left(A, \begin{bmatrix} p_a & 0 \\ 0 & 0 \end{bmatrix}\right) + SBM\left(B, \begin{bmatrix} p_b & 0 \\ 0 & 0 \end{bmatrix}\right)$$

Figures 2.8 and 2.9 give the contingency table for edges occuring under each of the component partitions. We expect partition A and B to each correspond to a separate mode. Figure 2.10 gives the contingency table for the larger model given by the refinement of A and B. Algorithms to recover the latent partitions prefer the smaller models over the larger ones. Further, they may prefer one mode over the other, even if the algorithm is stochastic and run multiple times. For instance if $p_a > p_b$, the mode corresponding to partition A would be favoured, and partition B may never be seen. This multi-modality is an artifact of the algorithm's preference for smaller models. Our objective is to see all significant partitions, and hence learn the larger model by learning the components of the mixture individually. This enumeration of the smaller models will allow us to use existing algorithms; to get around their affinity for the a particular mode we will modify the graph after each run to make that dominant mode less attractive.

	a	$\neg a$
a	$p_a + p_b \cdot (\mathbb{P}(b \mid a))^2$	$p_b \cdot \mathbb{P}(b \mid a) \cdot \mathbb{P}(b \mid \neg a)$
$\neg a$	$p_b \cdot \mathbb{P}(b \mid \neg a) \cdot \mathbb{P}(b \mid a)$	$p_b \cdot (\mathbb{P}(b \mid \neg a))^2$

Figure 2.8: SBM A

	b	$\neg b$
b	$p_b + p_a \cdot (\mathbb{P}(a \mid b))^2$	$p_a \cdot \mathbb{P}(a \mid b) \cdot \mathbb{P}(a \mid \neg b)$
$\neg b$	$p_a \cdot \mathbb{P}(a \mid \neg b) \cdot \mathbb{P}(a \mid b)$	$p_a \cdot (\mathbb{P}(a \mid \neg b))^2$

Figure 2.9: SBM B

	$a \cap b$	$a \cap \neg b$	$\neg a \cap b$	$\neg a \cap \neg b$
$a \cap b$	$p_a + p_b$	p_a	p_b	0
$a \cap \neg b$	p_a	p_a	0	0
$\neg a \cap b$	p_b	0	p_b	0
$\neg a \cap \neg b$	0	0	0	0

Figure 2.10: SBM of Refinement of A and B

2.3 Community Detection

"Birds of a feather flock together". It has been well established that this old proverb holds true; we are likely to resemble our friends, a property called **homophily** [57]. Thus in trying to understand individuals, it is useful to understand the communities they belong to. In epidemiology, infectious disease spreads between neighbours, computer viruses between connected computers, the popularity of items among friends, etc., and so understanding the underlying network that these stochastic processes are directed by is crucial to understanding the processes themselves. Further, when performing network based sampling such as **snowball** or **respondent-driven** sampling, bias resulting from homophily needs to be taken into account. This section presents methods to identify communities.

A graph **partition** is a grouping of its vertices into disjoint sets. A **community** (also referred to as a **block** or **part** of a partition) is a subset of the vertices that are densely connected internally and sparsely connected externally. A network has **community structure** if its vertices can be partitioned into sets of vertices that are each communities. A partition with dense edges within each community and sparse edges between communities is desirable. The definition of community is problematically vague, leading to a multitude of community detection algorithms that each try to optimize different objective functions derived from different definitions of community.

Community detection is an ill-defined problem [33] as there is no definition of what constitutes a good partition into communities. This has led to many different approaches to community detection. Divisive algorithms [32, 38, 61] detect weak links between communities and remove them. Agglomerative algorithms [73] recursively merge similar vertices and communities. This family includes the Clique Percolation Method (CPM) that merges together cliques (the best possible community structure) [63]. Optimization algorithms [18, 88, 59] maximize some community quality function. Our approach focuses on the family that appears most promising, namely the Louvain method that greedily optimizes a partition's modularity.

The following Sections present some different definitions of communities, algorithms that make use of them, and their strengths and weaknesses. Section 2.3.4 presents a definition that a community is defined by cliques. The Clique Percolation Method, an algorithm that detects overlapping communities but does not scale to

large graphs, is then briefly discussed. Section 2.3.1 introduces the modularity definition of a community, with Section 2.3.2 using this definition to develop the Louvain community detection algorithm. This is the method that is used in our algorithm as it has a fast run time for large graphs. Louvain is improved upon in the Ensemble Clustering for Graphs algorithm described in Section 2.3.3, which increases the quality and stability of partitions at the cost of running Louvain more than once. Section 2.3.6 Section 2.3.6 presents a Bayesian community detection method capable of finding overlapping communities, but which faces performance issues for large graphs as it is an MCMC method. Section 2.3.5 introduces the spectral clustering algorithm that produces a single partition. While not appropriate for large graphs due to its high computational costs, it will be used by us to cluster the partitions themselves that are returned from multiple calls to our method.

2.3.1 Modularity

The quality of a partition C can be measured by its **modularity** [12], $Q(C) \in [-\frac{1}{2}, 1]$, which measures the density of edges within a community against edges between communities. More specifically, it measures the number of edges within the communities compared to the expected number of edges within subgraphs on a random graph that have the same size and degree. Let $G = (V, E)$ be a weighted graph. Let $C = \{ C_1, \ldots, C_r \}$ be a partition of the vertices of G into r communities. It necessarily follows that $V = C_1 \sqcup C_2 \sqcup \cdots \sqcup C_r$ since C is a partition. For $v \in V$, let $C(v)$ denote the community $C_i \in C$ that v belongs to, i.e. $v \in C_i$.

The notation proposed in the original Louvain paper [11] can be made clearer. Let Σ_{ST} denote the sum of edge weights for all edges between sets S and T of vertices. If S is a singleton set we simply write its sole member i.e. Σ_{uv} instead of $\Sigma_{\{u\}\{v\}}$. Let V be the set of all vertices on the graph, $C_i \subseteq V$ a subset that we call a community, and $v \in V$ be some vertex. Then the following notations are equivalent.

Proposed	Louvain	Meaning
Σ_{VV}	$m = \frac{1}{2} \sum_{u,v \in V} A_{uv}$ or $\lvert E \rvert$	sum of all edge weights
Σ_{VC_i}	Σ_{tot} or d_C	sum of all edge weights that have at least one endpoint in C_i
$\Sigma_{C_iC_i}$	Σ_{in} or $\lvert E_C \rvert$	sum of all edge weights for edges inside C_i
Σ_{Vv}	k_v	sum of all edge weights that have v as an endpoint
Σ_{C_iv}	$k_{v,in} = \sum_{u \in V} A_{vu}$	sum of all edge weights between v and vertices in C_i
Σ_{uv}	A_{uv}	edge weight between vertices u and v

Table 2.1: Correspondence between the Louvain paper's and our proposed notation.

Three equivalent definitions of modularity will be introduced. Intuition for why

modularity is a useful value is provided by looking at the community-level formulation. The Louvain community detection algorithm uses the vertex-level formulation to iteratively construct communities. A matrix formulation will later be introduced as Equation (3.2.2) and will appear in this thesis's proposed method.

The community-level definition of the modularity of partition C is

$$Q(C) = \sum_{i=1}^{|C|} \left[\frac{\Sigma_{C_i C_i}}{\Sigma_{VV}} - \left(\frac{\Sigma_{V C_i}}{2\Sigma_{VV}} \right)^2 \right] \tag{2.3.1}$$

The term $\frac{\Sigma_{C_i C_i}}{\Sigma_{VV}}$ is the fraction of edges that lie within community C_i, while $\left(\frac{\Sigma_{V C_i}}{2\Sigma_{VV}}\right)^2$ is the expected number of edges within C_i if the graph were randomly generated.

Suppose that all edges are split in half and then randomly reconnected to another. This produces a random graph with the same degree sequence as the original, i.e. the number of vertices each vertex has is preserved. Note that this may not produce a simple graph; self-loops could form if two of a vertex's half-edges are connected together, or multiple-edges between the same two vertices. For sparser graphs this is less likely, and furthermore as it is the aggregate count of edges within and between communities. The total number of half-edges is $2|E|$. Considering a specific half-edge of vertex v, there are $2|E| - 1$ remaining half-edges that it could connect to. Then the probability it connects to any half-edge of vertex u is $\frac{\deg(u)}{2|E|-1}$. Thus the expected number of edges between vertices v and u is $\deg(v)\frac{\deg(u)}{2|E|-1}$. As an approximation $\frac{1}{2|E|}$ is used instead of $\frac{1}{2|E|-1}$ under the assumption that the total number of edges $|E|$ will be large. A further assumption is that there will be at most one edge between vertices and no self-loops, which is likely for large graphs. The probability of an edge between two nodes then becomes $\frac{\deg(v)\deg(u)}{2|E|-1}$. This then gives the vertex-level definition of modularity (2.3.2).

If this first term is larger, then there are more links within that community than one would expect, providing evidence that C_i is a community. The individual subgraph induced by C_i can be considered a community if

$$\frac{\Sigma_{C_i C_i}}{\Sigma_{VV}} - \left(\frac{\Sigma_{V C_i}}{2\Sigma_{VV}} \right)^2 > 0$$

The first term can be rewritten in terms of vertices. The sum over community degrees $\Sigma_{V C_i}$ and $\Sigma_{C_i C_i}$ can be split into a sum over the individual vertex contributions

$$\sum_{i=1}^{|C|} \frac{\Sigma_{C_i C_i}}{\Sigma_{VV}} = \sum_{i=1}^{|C|} \frac{1}{2\Sigma_{VV}} \sum_{u,v \in C_i} \Sigma_{uv}$$

Note $\sum_{u,v\in C_i}\Sigma_{uv}$ counts each vertex twice (i.e. the pair (u,v) and (v,u) appear in the sum) and so the factor of $\frac{1}{2}$ is introduced to not overcount. The sum over vertex pairs within a community can be replaced with a sum over all vertex pairs and a Kronecker delta function to indicate community agreement.

$$\sum_{i=1}^{|C|}\frac{1}{2\Sigma_{VV}}\sum_{u,v\in C_i}\Sigma_{uv}=\sum_{u,v\in V}\frac{\Sigma_{uv}}{2\Sigma_{VV}}\delta(C(u),C(v))$$

For the second term,

$$\sum_{i=1}^{|C|}\left(\frac{\Sigma_{VC_i}}{2\Sigma_{VV}}\right)^2=\sum_{i=1}^{|C|}\left(\frac{1}{2\Sigma_{VV}}\right)^2\sum_{u,v\in C_i}\Sigma_{Vu}\Sigma_{Vv}$$

The likelihood that a half-edge connects to a half-edge inside community C_i is $\frac{\Sigma_{VC_i}}{2\Sigma_{VV}}$. Summing over each of the Σ_{VC_i} half-edges in C_i gives the expected number of edges within C_i. This double counts the half-edges, requiring dividing by 2 as a correction. $(\Sigma_{VC_i})^2=\sum_{u,v\in C_i}\Sigma_{Vu}\Sigma_{Vv}$ As before, replacing the summation over vertices in community C_i with the Kronecker delta function gives a vertex level term.

$$\sum_{i=1}^{|C|}\left(\frac{1}{2\Sigma_{VV}}\right)^2\sum_{u,v\in C_i}\Sigma_{Vu}\Sigma_{Vv}=\sum_{u,v\in V}\frac{\Sigma_{Vu}\Sigma_{Vv}}{(2\Sigma_{VV})^2}\delta(C(u),C(v))$$

Modularity can equivalently be written at the vertex-level as

$$Q(C)=\frac{1}{\Sigma_{VV}}\sum_{u,v\in V}\left[\Sigma_{uv}-\frac{\Sigma_{Vv}\Sigma_{Vu}}{2\Sigma_{VV}}\right]\delta(C(u),C(v)) \tag{2.3.2}$$

where $\delta(C(u),C(v))$ is the Kronecker delta function

$$\delta(C(u),C(v))=\begin{cases}1 & C(u)=C(v)[\text{ i.e. } v\in C(u) \text{ and } u\in C(v)]\\ 0 & C(u)\neq C(v)[\text{ i.e. } v\notin C(u) \text{ and } u\notin C(v)]\end{cases}$$

Direct modularity maximization is one possible strategy for finding the best partition. However as the number of partitions is given by the Bell number, which grows super-exponentially, this quickly becomes intractable. It is in this context that a greedy algorithm such as Louvain is considered. Modularity will be central to the Louvain community detection algorithm.

2.3.2 Multilevel Louvain Method

The Louvain method [11] (also called the multi-level method) currently presents the best trade-off between speed and quality of communities [89]. It does this by greedily and iteratively maximizing each candidate partition's modularity. As the number of possible partitions grows exponentially in the number of vertices, direct maximization of modularity is not feasible and is known to be at least NP-complete [13], this greedy heuristic leads to low computational cost while beating the modularity of competing algorithms. It is available as a pure Python package [4], as `community_multilevel` in the Python `igraph` package that calls a faster C implementation, or in the R `igraph` [22] package as `cluster_louvain`.

Algorithm 2: Louvain Method of Community Detection

Input: graph $G = (V, E)$
Initialize the set of communities C by assigning each vertex to its own community
$C = \{ \{v\} \mid v \in V \}$
while modularity keeps improving **do**
  First phase: modularity optimization
  while modularity keeps improving **do**
    for $v \in C$ **do**
      $\Delta Q_{best} = 0$
      $v'_{best} = \emptyset$
      for $v' \in Neighbours(v)$ **do**
        $\Delta Q_{out} = \Delta Q\,(Community(v) \to v)$
        $\Delta Q_{in} = \Delta Q\,(v \to Community(v'))$
        $\Delta Q_{curr} = \Delta Q_{in} + \Delta Q_{out}$
        if $\Delta Q_{best} < \Delta Q_{curr}$ **then**
          $\Delta Q_{best} = \Delta Q_{curr}$
          $v'_{best} = v'$
      if $\Delta Q_{best} > 0$ **then**
        $Community(v) = Community(v'_{best})$
  Second phase: community aggregation
  for community $C_i \in C$ **do**
    Contract all internal edges (u, v) for all $u, v \in C_i$, leaving a single vertex
    Collapse multiple edges to a single edge by summing their weights
return C

The algorithm (Algorithm 2) starts with every vertex assigned to its own community (Figure 2.11) and repeats two phases until no improvement in modularity is made. The first phase considers each vertex one at a time and tries to maximize the modularity of the overall partition. For each vertex, its neighbours (i.e. vertices

Figure 2.11: The partition of the 3-barbell graph that Louvain is initialized to. This partition has a modularity -0.17.

connected to it by an edge) are considered. The difference in modularity that would result from moving the current vertex into its neighbour's community is computed. If changing the community of the current vertex does not lead to an improvement then its community is left unchanged. Otherwise it is moved to the community that gives the best improvement in modularity. This process repeats sequentially with the next vertex until no further improvement in modularity can be made. At this point the algorithm enters the second phase.

This computation can be sped up. Instead of computing the modularity of the entire partition when considering each vertex, only the difference in modularity resulting from the moving the current vertex needs to be computed. The difference in modularity for moving an isolated node v into community C is given by

$$\begin{aligned}\Delta Q(v \to C) = &\left[\frac{\Sigma_{CC} + \Sigma_{Cv}}{2\Sigma_{VV}} - \left(\frac{\Sigma_{VC} + \Sigma_{Vv}}{2\Sigma_{VV}}\right)^2\right] \\ &- \left[\frac{\Sigma_{CC}}{2\Sigma_{VV}} - \left(\frac{\Sigma_{VC}}{2\Sigma_{VV}}\right)^2 - \left(\frac{\Sigma_{Vv}}{2\Sigma_{VV}}\right)^2\right]\end{aligned} \tag{2.3.3}$$

This can be further simplified as in the Louvain paper [11]

$$\begin{aligned}&= \frac{\Sigma_{CC}}{2\Sigma_{VV}} + \frac{\Sigma_{Cv}}{2\Sigma_{VV}} - \frac{\Sigma_{VC}^2 + 2\Sigma_{VC}\Sigma_{Vv} + \Sigma_{Vv}^2}{4\Sigma_{VV}^2} - \frac{\Sigma_{CC}}{2\Sigma_{VV}} + \frac{\Sigma_{VC}^2}{4\Sigma_{VV}^2} + \frac{\Sigma_{Vv}^2}{4\Sigma_{VV}^2} \\ &= \frac{\Sigma_{Cv}}{2\Sigma_{VV}} + \frac{-\Sigma_{VC}^2 - 2\Sigma_{VC}\Sigma_{Vv} - \Sigma_{Vv}^2 + \Sigma_{VC}^2 + \Sigma_{Vv}^2}{4\Sigma_{VV}^2} \\ &= \frac{\Sigma_{Cv}}{2\Sigma_{VV}} - \frac{\Sigma_{VC}\Sigma_{Vv}}{2\Sigma_{VV}^2}\end{aligned} \tag{2.3.4}$$

To compute the change when v is removed from community C, the change when v is added to a copy of C that doesn't contain v can be computed and the negative of this value taken. I.e., $\Delta Q(C \to v) = -\Delta Q(v \to C)$. With both of these formulas, the effect of moving v from C to C' is $\Delta Q(v \to C') - \Delta Q(v \to C)$. (Note that the notation $Q(v \to C)$ is not standard; the literature forces the reader to infer what is changing).

Example 2.3.1. Consider vertex 5 in the yellow community in Figure 2.11, and suppose all edges are initialized with a weight of 1. Then there are two choices for where to move this vertex; into the green or teal communities. When moving it into the green community, the variables have values $\Sigma_{green,green} = 0$, $\Sigma_{V,green} = 2$, $\Sigma_{green,5} = 1$, $\Sigma_{V,5} = 2$, and $\Sigma_{V,V}$ is the number of edges in the entire graph which is 7. Then

$$\begin{aligned}\Delta Q(5 \to green) = &\left[\frac{0+1}{2(7)} - \left(\frac{2+2}{2(7)}\right)^2\right] \\ &- \left[\frac{0}{2(7)} - \left(\frac{2}{2(7)}\right)^2 - \left(\frac{2}{2(7)}\right)^2\right] \\ &= \frac{14}{196} - \frac{16}{196} + \frac{4}{196} + \frac{4}{196} = \frac{6}{196}\end{aligned}$$

To calculate the change of removing vertex 5 from the yellow community, consider the change of adding this vertex if the yellow community did not contain it. In this case the yellow community would be empty. Then the variables would have values $\Sigma_{yellow,yellow} = 0$, $\Sigma_{V,yellow} = 0$, $\Sigma_{yellow,5} = 0$, $\Sigma_{V,5} = 2$, and $\Sigma_{V,V} = 7$. Then

$$\begin{aligned}\Delta Q(5 \to yellow) = &\left[\frac{0+0}{2(7)} - \left(\frac{0+2}{2(7)}\right)^2\right] \\ &- \left[\frac{0}{2(7)} - \left(\frac{0}{2(7)}\right)^2 - \left(\frac{2}{2(7)}\right)^2\right] \\ &= -\frac{4}{196} + \frac{4}{196} = 0\end{aligned}$$

Hence $\Delta Q(yellow \to 5) = -\Delta Q(5 \to yellow) = 0$.

For the teal community, $\Sigma_{teal,teal} = 0$, $\Sigma_{V,teal} = 3$, $\Sigma_{teal,5} = 1$, $\Sigma_{V,5} = 2$, and $\Sigma_{V,V} = 7$.

$$\begin{aligned}\Delta Q(5 \to teal) = &\left[\frac{0+1}{2(7)} - \left(\frac{3+2}{2(7)}\right)^2\right] \\ &- \left[\frac{0}{2(7)} - \left(\frac{3}{2(7)}\right)^2 - \left(\frac{2}{2(7)}\right)^2\right] \\ &= \frac{14}{196} - \frac{25}{196} + \frac{4}{196} + \frac{9}{196} = \frac{2}{196}\end{aligned}$$

Thus to move vertex 5 to the green community gives a change of $\Delta Q(5 \to green) + \Delta Q(yellow \to 5) = \frac{6}{196} + 0 = \frac{6}{196}$, and to move it to the teal community gives a change of $\Delta Q(5 \to teal) + \Delta Q(yellow \to 5) = \frac{2}{196} + 0 = \frac{2}{196}$. Thus the algorithm

moves vertex 5 to the green community as this results in the greatest improvement in modularity. This continues for the rest of the vertices, finally producing the partition in Figure 2.12.

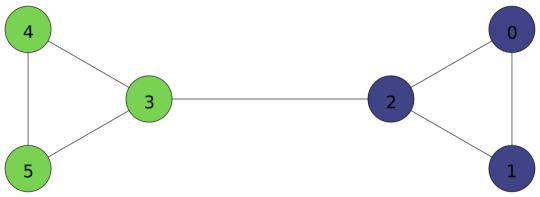

Figure 2.12: The partition returned by the first phase of the Louvain method. This partition has a modularity 0.36.

The second phase is to aggregate the resulting communities. That is, all vertices in the same community are collapsed into one vertex. All edges that stay inside the community become a self loop on the new vertex. The weights of the new edges between the new vertices are equal to the sum of all edges from one community to the other.

2.3.3 Ensemble Clustering for Graphs (ECG)

Consider the network in Figure 2.13 comprised of four cliques, with all edges added between each pair of cliques except for two. These four cliques on two vertices are themselves part of larger cliques on four vertices. The Louvain method does not select these subcliques, instead greedily selecting the larger cliques as in Figure 2.14. However the order that it considers vertices in determines which subcliques to pair in the same community, for example producing the different partition in Figure 2.15. Is there a way to recover the partition of Figure 2.13? Or at least to improve the stability of the algorithm by decreasing the variance in partitions?

This example also highlights a further problem in community detection, namely the **resolution limit** [31]. That is, modularity optimization fails to identify communities that are smaller than some value depending on the total number of edges in the network and the degree of interconnectedness between communities. The probability that the subcommunity structure of a community is undetected is greatest when that community has a number of internal edges less than or equal to $\sqrt{2|E|}$ where $|E|$ is the total number of edges [31]. In the example from Figure 2.13, $|E| = 8$ and so $\sqrt{2|E|} = \sqrt{16} = 4$. Thus the subcliques on two vertices contained within the cliques on four vertices are very likely to be undetected, as was the outcome of the Louvain method in Figure 2.14 and Figure 2.15. Thus for such communities obtained through modularity optimization it is impossible to tell whether it is a single community or agglomeration of smaller communities.

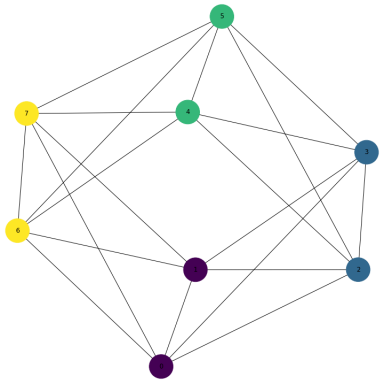

Figure 2.13: A network with its four cliques given their own communities. This has a modularity of -0.05 .

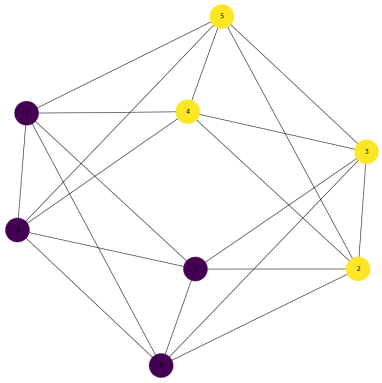

Figure 2.14: A Louvain partition of the graph from Figure 2.13 with a modularity of 0.1 .

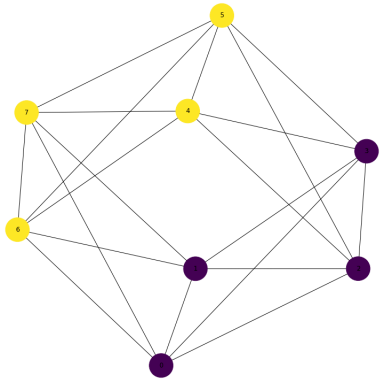

Figure 2.15: The Louvain method producing a different partition of the same graph from Figure 2.13 with a modularity of 0.1 . The different partition was caused by the vertices being considered in a different order.

$\Sigma_{\{C_i \cap C_j\}\{C_i \cap C_j\}} = \Sigma_{C_i C_i} + \Sigma_{C_j C_j} + \Sigma_{C_i C_j}$ since the set of edges internal to C_i is disjoint from the set of edges internal to C_j. $\Sigma_{V\{C_i \cap C_j\}} = \Sigma_{VC_i} + \Sigma_{VC_j}$

$$\begin{aligned}
\Delta Q(C_i \to C_j) &= [Q(\{C_i \cap C_j\})] - [Q(C_i) + Q(C_j)] \\
&= \left[\frac{\Sigma_{\{C_i \cap C_j\}\{C_i \cap C_j\}}}{\Sigma_{VV}} - \left(\frac{\Sigma_{V\{C_i \cap C_j\}}}{2\Sigma_{VV}}\right)^2\right] \\
&\quad - \left[\frac{\Sigma_{C_i C_i}}{\Sigma_{VV}} - \left(\frac{\Sigma_{VC_i}}{2\Sigma_{VV}}\right)^2 + \frac{\Sigma_{C_j C_j}}{\Sigma_{VV}} - \left(\frac{\Sigma_{VC_j}}{2\Sigma_{VV}}\right)^2\right] \\
&= \left[\frac{\Sigma_{C_i C_i} + \Sigma_{C_j C_j} + \Sigma_{C_i C_j}}{\Sigma_{VV}} - \left(\frac{\Sigma_{VC_i} + \Sigma_{VC_j}}{2\Sigma_{VV}}\right)^2\right] \\
&\quad - \left[\frac{\Sigma_{C_i C_i}}{\Sigma_{VV}} - \left(\frac{\Sigma_{VC_i}}{2\Sigma_{VV}}\right)^2 + \frac{\Sigma_{C_j C_j}}{\Sigma_{VV}} - \left(\frac{\Sigma_{VC_j}}{2\Sigma_{VV}}\right)^2\right] \\
&= \left[\frac{\Sigma_{C_i C_j}}{\Sigma_{VV}} - \frac{(\Sigma_{VC_i})^2 + 2\Sigma_{VC_i}\Sigma_{VC_j} + (\Sigma_{VC_j})^2}{4(\Sigma_{VV})^2}\right] \\
&\quad + \left[\left(\frac{\Sigma_{VC_i}}{2\Sigma_{VV}}\right)^2 + \left(\frac{\Sigma_{VC_j}}{2\Sigma_{VV}}\right)^2\right] \\
&= \frac{\Sigma_{C_i C_j}}{\Sigma_{VV}} - \frac{\Sigma_{VC_i}\Sigma_{VC_j}}{2(\Sigma_{VV})^2}
\end{aligned}$$

Merging communities C_i and C_j creates a change in modularity of

$$\Delta Q(C_i \to C_j) = \frac{\Sigma_{C_i C_j}}{|E|} - \frac{\Sigma_{VC_i}\Sigma_{VC_j}}{2|E|^2}$$

If $\frac{\Sigma_{VC_i}\Sigma_{VC_j}}{2|E|^2} < 1$ and there is at least one edge between C_i and C_j, i.e. $\Sigma_{C_i C_j} \geq 1$, then $\Delta Q(C_i \to C_j) > 0$ and so the two communities will be merged. If $\Sigma_{VC_i} \approx \Sigma_{VC_j}$, then $\Sigma_{VC_i} \leq \sqrt{2|E|}$ implies that merging C_i and C_j will improve modularity. This will be true even if C_i and C_j "should" be distinct communities. If Σ_{VC_i} and Σ_{VC_j} are less than $2|E|$ then the expected number of edges between them is less than 1, and so will be merged to maximize modularity. Communities of total degree less than $2|E|$ will be undetectable by direct modularity maximization. Since real networks contain many small communities, direct application of modularity is problematic when trying to detect fine structure. There is a plateau in modularity which makes it difficult to distinguish the maxima from "close" partitions.

The authors of the Louvain method argue that at intermediate steps the resolution limit is overcome, because before agglomeration the algorithm is only making

individual local decisions for each vertex (or later for each agglomeration of vertices). While it may improve over other algorithms, the Louvain method is clearly still faced with the resolution limit.

Algorithm 3: Ensemble Clustering for Graphs (ECG)

Input: graph $G = (V, E)$, minimum edge weight $\omega_{\min}$, ensemble size k.
Compute the 2-core of G.
Run the first modularity optimization stage of the Louvain method k times independently on G with different random seeds to produce k partitions of G.
Integrate the ensemble results by constructing a reweighted graph:
for edge $(u, v) \in E$ **do**
  Compute the number of partitions where u and v were assigned to the same community, i.e. $a(u, v) = \sum_{i=1}^{k} \delta(C_i(u), C_i(v))$ where $C_i(v)$ is the community of v in partition i and $\delta(\cdot, \cdot)$ is 1 if both arguments are the same, 0 otherwise.
  $$\omega((u, v)) = \begin{cases} \omega_{\min} & (u, v) \notin \text{ 2-core of } G \\ \omega_{\min} + (1 - \omega_{\min})\frac{a(u,v)}{k} & \textit{otherwise} \end{cases}$$
Run the Louvain method on this reweighted graph.

One approach to improve the stability of partitioning is **Ensemble Clustering for Graphs (ECG)** [74] which improves the stability of the Louvain algorithm by combining it with ensemble clustering. It was benchmarked favourably by its authors in [75]. Ensemble clustering combines multiple partitions of the graph into one. Consensus on a final partition is achieved by either looking for the co-occurrence of vertices in a cluster or finding a median partition that maximizes the similarity between all found partitions.

The first step of ECG is to run the first optimization phase of the Louvain method multiple times over the graph. This creates an ensemble of partitions. The partitions from this ensemble are then integrated together by reweighting the edges of the graph according to how often that edge appeared inside a community. That is, if two connected vertices appeared in the same community in every partition, the edge between them will be given a weight of 1. If they never appeared in the same community, the edge is given some minimum weight $\omega_{\min}$. All other edges are assigned a weight between $\omega_{\min}$ and 1 proportional to how many times the endpoints appear in the same community. The authors recommend an ensemble size of $k = 16$ and $\omega_{\min} = 0.05$. After this weighted graph is obtained, it is passed as input into the Louvain method to produce a final partition.

Vertices of degree one, i.e. that have only a single neighbour, are guaranteed to appear in the same community as their neighbour in any partition that maximizes modularity (this follows from Corollary 3.5 of [12]). Thus to simplify computation,

these vertices can initially be removed from the graph and afterwards added to their neighbour's community.

The authors go beyond this to improve the results of their algorithm. The non-2-core induced subgraph is a collection of trees, which they argue cannot have community structure. Further, if they were not assigned the minimum edge weight then they would be assigned very high weights. This is because trees are stable under the Louvain method, and so their vertices are likely to be assigned to the same communities for each partition in the ensemble.

ECG is not able to correctly partition the tiny graph from Figure 2.13. However it does perform better on another well-known example of the resolution limit; the **ring of cliques**. This graph is constructed by taking n complete graphs and adding a single edge between these graphs to connect them all in a ring (see Figure 2.17; note the ring has been twisted to fit on the page). Algorithms that try to directly optimize modularity will end up combining multiple cliques into the same community, as the Louvain method did in Figure 2.16. ECG was able to correctly recover the true partition (Figure 2.17).

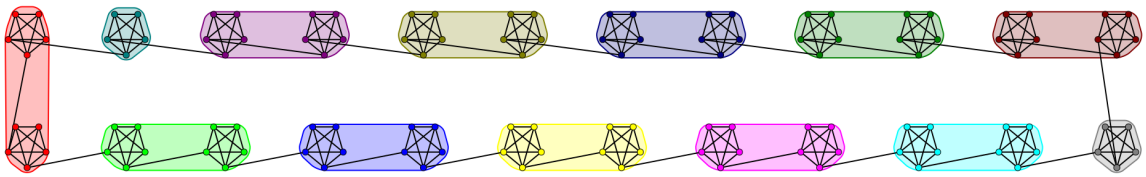

Figure 2.16: The partition produced by the Louvain method on the ring of 24 cliques with clique size of 5. This partition has a modularity 0.871 . Most communities incorrectly contains two cliques.

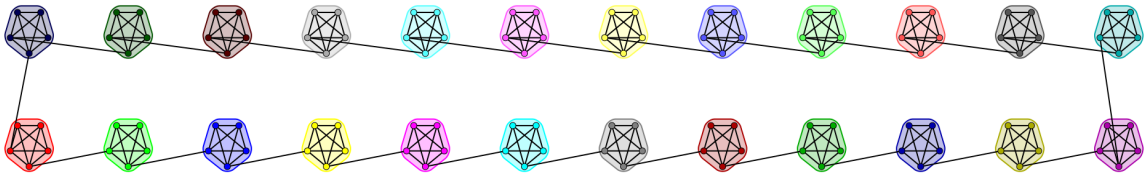

Figure 2.17: The partition produced by ECG on the same graph as Figure 2.16. This partition has a lower modularity 0.867, but correctly assigns each clique to its own community.

2.3.4 Clique Percolation Method (CPM)

What has been tried to detect overlapping communities? The **Clique Percolation Method (CPM)** [63, 52, 34] was motivated by the problem of detecting overlapping

communities. It is an agglomerative algorithm that recursively merges similar vertices and communities together. As the best possible community structure occurs when all vertices of a subgraph are connected, i.e. when they form a clique, the CPM uses the k-cliques present in a graph to determine communities. Two k-cliques are said to be adjacent if they differ by only one vertex, and a collection of adjacent k-cliques then forms a community. This allows communities to overlap as a vertex may be part of two separate k-cliques that are not adjacent, for example vertex "o" in Figure 2.18.

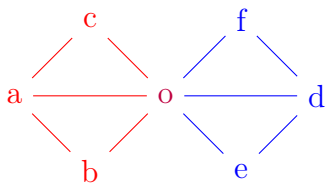

Figure 2.18: Overlapping communities returned by CPM. The red (o, a, b, c) and blue (o, d, e, f) communities are both composed of two adjacent 3-cliques. Vertex "o" is a member of both communities.

This definition of a community is a local property of a subgraph. This is in contrast to the global property of the next community definition we introduce, modularity, where the addition or removal of edges and vertices to a graph changes the perceived community structure of even far away unchanged regions. It also avoids a resolution limit where the number of vertices determines the size of the smallest community that can be detected.

Unfortunately CPM is not efficient as a graph can contain many cliques that all need to be considered. For example for any $k \leq K$, a K-clique contains $\binom{K}{k}$ distinct k-cliques. This presents one speedup of instead finding all maximal cliques, i.e. those that cannot be extended by adding an additional adjacent vertex. However even this problem is not easy as it requires finding the maximum clique. The **clique decision problem** of determining whether a graph contains a k-cliques is NP-complete [46], and the more difficult problem of finding the maximum clique in a graph is NP-hard. While faster algorithms exist for sparse graphs, without making any assumptions about the graph, we must conclude that CPM has potentially exponential run time. Thus we must continue to look for a community detection method that scales to large graphs.

2.3.5 Spectral Clustering

Another general clustering technique considers the **spectrum** of a matrix, i.e. the set of its eigenvalues. Spectral methods provide both a means of determining the optimal number of clusters as well as a natural dimension reduction that allows clustering in a lower dimensional space. [84] offers a good treatment of this subject. This method can

be applied to clustering vertices into communities, however as this involves computing at least part of the spectrum of a $|V| \times |V|$ matrix it is not computationally feasible for large graphs. Instead we will use this technique to cluster the clusterings of vertices themselves that our algorithm produces. A similarity measure between parts of partitions will be constructed in Section 3.3.2, and the resulting similarity matrix will be given to a spectral clusterer. We now go over how spectral clustering works.

Let G be a graph on n vertices with associated adjacency matrix A. The **degree matrix** of A is the diagonal matrix with diagonal entries $D_{ii} = \deg(v_i)$ and zeros everywhere else. If G is a weighted graph, then A_{ij} is the weight of the edge between vertices v_i and v_j, and the degree $\deg(v_i)$ is taken to be the sum of all edge weights of edges incident to v_i.

Example 2.3.2. Suppose A is the 5×5 block diagonal matrix with blocks of size 3 and 2. Then it has adjacency matrix and degree matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Properties of this graph are revealed by considering a matrix representation called the **graph Laplacian**, defined as $L := D - A$ where

$$L_{i,j} := \begin{cases} \deg(v_i) & i = j \\ -1 & i \neq j \text{ and } v_i \text{ is adjacent to } v_j \text{ i.e. } A_{i,j} \neq 0 \\ 0 & otherwise \end{cases}$$

The **symmetric normalized Laplacian** is defined as $L_{sym} := D^{-1/2}LD^{-1/2} = I - D^{-1/2}AD^{-1/2}$

$$[L_{sym}]_{i,j} := \begin{cases} 1 & i = j \\ -1 \Big/ \sqrt{\deg(v_i)\deg(v_j)} & i \neq j \text{ and } v_i \text{ is adjacent to } v_j \text{ i.e. } A_{i,j} \neq 0 \\ 0 & otherwise \end{cases}$$

The **random-walk normalized Laplacian** is defined as $L_{rw} := D^{-1}L = I - D^{-1}A$

$$[L_{rw}]_{i,j} := \begin{cases} 1 & i = j \\ -1 \Big/ \sqrt{\deg(v_i)} & i \neq j \text{ and } v_i \text{ is adjacent to } v_j \text{ i.e. } A_{i,j} \neq 0 \\ 0 & otherwise \end{cases}$$

Example 2.3.3. The Laplacian, symmetric normalized Laplacian, and random-walk Laplacian of Example 2.3.2 are then

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix},$$

$$L_{sym} = \begin{bmatrix} 1 & -1/2 & -1/2 & 0 & 0 \\ -1/2 & 1 & -1/2 & 0 & 0 \\ -1/2 & -1/2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, L_{rw} = \begin{bmatrix} 1 & -1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & 1 & -1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & -1/\sqrt{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Theorem 2.3.1. Let L be the Laplacian of some graph. Then L is block diagonal with as many blocks as there are components in the graph. Further, each block of L has an associated eigenvector x with eigenvalue 0 where x_i is one if i is in that block and zero otherwise.

This is Proposition 2 of [84].

Proof: Let G be some graph and A the associated weighted adjacency matrix. Suppose G has k components. Then A can be arranged into a block diagonal matrix with k blocks. As D is also block diagonal, this implies the sum $L = D - A$ will also be block diagonal. It is sufficient to show the theorem holds for a single block, as each of the eigenvectors are orthogonal and hence linearly independent. Let L_i be the submatrix associated with block i. Let $w(v, u)$ be the weight of the edge between vertices v and u. Then $w(v, u) = A_{u,v}$. Consider row $[L_m]_i$.

$$[L_m]_{i,i} = \deg(v_i) = \sum_{u \in L_m} w(v, u) = \sum_{u \in L_m} A_{i,u} = \sum_{u \in L_m, u \neq} -L_{i,u}$$

Without loss of generality we only need to consider one row of L_i.

$$L\mathbb{1}_{L_m} = L_m\mathbb{1} = [D_{i,i} - \sum A_{i,j}] = 0$$

∎

Example 2.3.4. By inspection of L and L_{sym} in Example 2.3.3 it is clear that the vectors $x_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \end{bmatrix}^T$ and $x_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T$ are eigenvectors with eigenvalue 0 as $Lx_1 = 0x_1$ and $Lx_2 = 0x_2$.

A stronger statement than Theorem 2.3.1 holds, namely that these are all the 0 eigenvalues.

Proof: Recall that the spectrum of a block diagonal matrix is equal to the union of the spectra of each block, with eigenvectors equal to each block's eigenvectors embedded in the larger space. The Laplacian of a graph with k components will be block diagonal with k blocks. Thus it suffices to show that for each block 0 is an eigenvalue with multiplicity 1.

Suppose that the graph is connected and hence $k = 1$. If x is an eigenvector with eigenvalue 0, then

$$0 = x^T Lx = \sum_{i,j} L_{i,j}(x_i - x_j)^2$$

∎

This theorem says that the multiplicity of the 0 eigenvector is equal to the number of clusters. However in reality the eigenvalues will be perturbed by noise and will not be exactly equal to 0. A more robust method involves looking at the **spectral gap** or **eigengap**, which is the difference between two consecutive elements in the sorted list of eigenvalues. This thesis assumes they are sorted in non-decreasing order. The position of the largest eigengap then demarcates the zero and non-zero eigenvalues, thus giving the number of components k. This eigengap heuristic is useful for finding the optimal number of clusters.

Characterizing the rest of the spectrum will make why there is a significant spectral gap clear. Again it suffices to consider the spectrum of a single block to describe them all. Initially consider the case of homogeneous degree distribution within a cluster where each vertex has the same degree. This would correspond to the expectation when using a SBM without degree correction. Suppose L_i is the $\ell_i \times \ell_i$ block of corresponding to a component of a Laplacian. Then $\mathbb{1}_{L_i}$. There will be a further $\ell_i - 1$ eigenvectors of the form $x_j = \begin{bmatrix} -1 & 0 & \cdots 0 & 1 & 0 \cdots 0 \end{bmatrix}$ i.e. -1 in the first position and 1 in the j-th position. All of these eigenvectors will have eigenvalue ℓ_i. Thus there will be a significant eigengap equal to the size of the smallest block.

Now consider the other extreme case where a cluster corresponds to the **star graph** S_n on n vertices comprised of a single vertex hub of degree $n - 1$ with the remaining $n - 1$ vertices connecting as spokes to this hub and each having degree 1. It has $n \times n$ Laplacian matrix

$$L = \begin{bmatrix} n-1 & -1 & -1 & \dots & -1 \\ -1 & 1 & 0 & & 0 \\ -1 & 0 & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ -1 & 0 & 0 & \dots & 1 \end{bmatrix}$$

Clearly $\mathbb{1}$ is still an eigenvector with eigenvalue 0. Vector $\begin{bmatrix}-(n-1) & 1 & \cdots 1\end{bmatrix}$ is a multiplicity 1 eigenvector with eigenvalue $(n-1)+1$. The remaining eigenvalues are 1 with multiplicity $n-2$ and eigenvectors of the form $\begin{bmatrix}0 & -1 & 0\cdots 0 & 1 & 0\cdots & 0\end{bmatrix}$ i.e. -1 in the 2nd position and 1 in the j-th position for $j > 2$. Thus the eigengap between the 0 eigenvalue and the next largest exists even in this extreme case. Such an imbalanced degree distribution causes problems for the eigengap heuristic; it will see that the maximum eigengap occurs between 1 and n, essentially saying that the $n-1$ leaf nodes each belong in their own community. One could argue that this is a better result than claiming a star graph is a single community. This example is fairly degenerate and so not expected.

Some authors sort eigenvalues in non-increasing order. For example the authors in [93] define $L := D^{-1/2}AD^{-1/2} = I - (I - D^{-1/2}AD^{-1/2}) = I - L_{sym}$. As noted in [62], replacing L with $I - L$ only changes the eigenvalues (from λ_i to $1 - \lambda_i$) and not the eigenvectors. Thus instead of the smallest k eigenvalues corresponding to the 0 eigenvalues with multiplicity k, the largest k eigenvalues correspond to the 1 eigenvalues with multiplicity k.

In Figure 2.19, the symmetric perturbation matrix H was generated such that $H_{i,j} = H_{j,i} \sim N(0, 1/4)$.

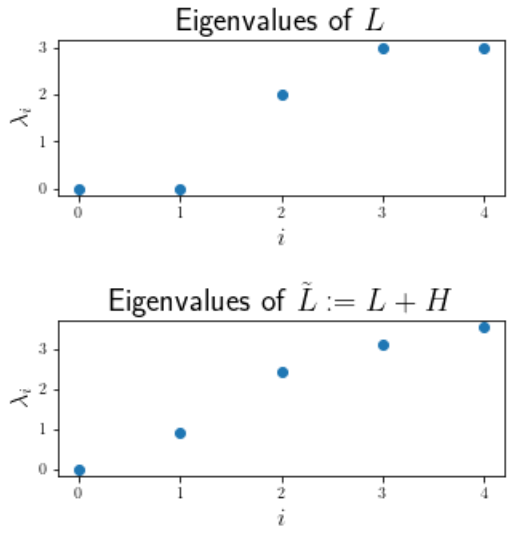

Figure 2.19: The spectrum of L from Example 2.3.3

Theorem 2.3.1 also gives a means of identifying the members of each cluster by looking for the all ones indicator eigenvectors. In practice numerical eigensolvers may not return the cluster indicator vectors as a basis of the 0 eigenspace, instead returning some other orthonormal basis. Let $\mathbb{1}_i$ for $i \in \{1, \ldots, k\}$ denote the i-th cluster indicator vector. Then each eigenvector is some linear combination $x_j = \sum_{i=1}^{k} a_i \mathbb{1}_i$

for some coefficient a_i.

In practice algorithms to recover eigenvectors are unlikely to find this indicator vector, instead finding some other basis of the block's eigenspace. Instead the eigenspace associated with each block may be rotated. A basis is rotated if a linear combination This thesis's proposed method will not use the indicator eigenvector method to identify clusters.

Once the optimal number of clusters k has been determined through the eigengap heuristic, it remains to reduce the dimension and employ a clustering method over this smaller space. The first k eigenvectors $\{x_1, \ldots, x_k\}$ (i.e. those associated with the k smallest eigenvalues) are computed. For unnormalized spectral clustering, these will be eigenvectors of L. Alternatively, the eigenvectors of L_{rw} or L_{sym} can be used with some modifications as detailed in Section 4 *Spectral Clustering Algorithms* of [84]. These eigenvectors then become columns in the $n \times k$ matrix $[x_1|\cdots|x_k]$. Each row is taken to be the k-dimensional embedding of its corresponding vertex in the original graph. Finally these rows are clustered using a technique such as k-means clustering.

Example 2.3.5. The normalized indicator vectors of Example 2.3.3 are

$$\left\{x_1 = [1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}, 0, 0]^T, x_2 = [0, 0, 0, 1/\sqrt{2}, 1/\sqrt{2}]^T\right\}$$

Suppose an eigensolver returned the orthonormal basis

$$\begin{Bmatrix} \bar{x}_1 = & [1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}]^T, \\ \bar{x}_2 = & [-\sqrt{2/15}, -\sqrt{2/15}, -\sqrt{2/15}, \sqrt{3/10}, \sqrt{3/10}]^T \end{Bmatrix}$$

Then $\bar{x}_1 = \frac{\sqrt{3}}{\sqrt{5}}x_1 + \frac{\sqrt{2}}{\sqrt{5}}x_2$ and $\bar{x}_2 = -\sqrt{\frac{2}{15}}x_1 + \sqrt{\frac{2}{15}}x_2$. The basis $\{\bar{x}_1, \bar{x}_2\}$ produces the embedding

$$[\bar{x}_1|\bar{x}_2] = \begin{bmatrix} 1/\sqrt{5} & -\sqrt{2/15} \\ 1/\sqrt{5} & -\sqrt{2/15} \\ 1/\sqrt{5} & -\sqrt{2/15} \\ 1/\sqrt{5} & \sqrt{3/10} \\ 1/\sqrt{5} & \sqrt{3/10} \end{bmatrix}$$

A clustering method like k-means would then cluster the points $[1/\sqrt{5}, -\sqrt{2/15}]$ and $[1/\sqrt{5}, \sqrt{3/10}]$ into separate clusters, thus correctly separating them.

In reality we do not see the true block diagonal adjacency matrix associated with a graph whose vertices we are trying to cluster. Instead we observe this matrix with some added noise. Let H be a random Hermitian matrix, say $H_{i,j} \sim N(0, \epsilon^2)$ for some "small" $\epsilon \in \mathbb{R}$. Suppose $A + H$ is what is observed. Then one can ask if the above techniques apply here. Thankfully the Davis-Kahan theorem states that

the eigenspace of this perturbed matrix is close to the eigenspace of the unperturbed matrix. There have been more recent improvements [27] on the Davis-Kahan bound.

Computing the spectrum of a large matrix is expensive, making spectral clustering infeasible for clustering large graphs. Our proposed method uses spectral clustering to cluster the parts of partitions that are returned by our partition sampling method. The core idea of our sampling method is derived from Wang-Landau sampling.

One final consideration is that the Laplacian is not well-defined for vertices without neighbours as this involves an empty sum. This situation is likely to occur in our use case as we are aiming to only observe each unique part once in order to improve sampling efficiency. This requires identifying and removing disimilar parts, i.e. isolated vertices in the associated graph, before proceeding with spectral clustering.

2.3.6 Bayesian Inference on Graphs

Given an observed network G generated from a SBM, it remains to find the likeliest block partition C that generated it (i.e. to fit a model). The procedure is described in [71] which serves as documentation for the Python `graph-tool` package [67] for performing SBM inference. Further details can be found in [68]. Blei also introduced a Bayesian approach for detecting overlapping communities [39].

To find the likeliest partition, the Bayesian posterior probability $\mathbb{P}(C|G)$ must first be computed or approximated

$$\mathbb{P}(C|G) = \frac{\sum_P \mathbb{P}(G|P,C)\mathbb{P}(P,C)}{\mathbb{P}(G)}$$

where $\mathbb{P}(P,C)$ is the prior probability of the model parameters and

$$\mathbb{P}(G) = \sum_{P,C} \mathbb{P}(G|P,C)\mathbb{P}(P,C)$$

is the evidence, which is the total probability of the observation summed over all possible model parameters. In practice models where C determines P can be used, since the likeliest edge formation probabilities are those actually computed from the empirical edge counts within the communities of a fixed partition C and graph G. Thus the above simplifies to

$$\mathbb{P}(C|G) = \frac{\mathbb{P}(G|P,C)\mathbb{P}(P,C)}{\mathbb{P}(G)}$$

where P is the only inter-community edge probability matrix compatible with C and G.

To perform inference requires either finding a partition C that maximizes $\mathbb{P}(C|G)$ or sampling different partitions according to the posterior probability. As is standard

in Bayesian inference, a Markov chain Monte Carlo (MCMC) method to find the maximum or sample from the posterior distribution has been proposed in [65] and implemented in [67]. With these samples, a weighted average over the different model fits and with weights based on posterior probability can be constructed. At each step of the MCMC method, the community membership of each vertex is randomly modified with the modification being accepted or rejected with a probability that is a function of the difference in entropy that this change creates. After sufficient mixing this should produce partitions C with probability proportional to $\mathbb{P}(G|C)$.

SBMs suffer as well from the resolution limit, where they can find at most $\mathcal{O}(\sqrt{|V|})$ communities [64]. Hierarchical SBMs attempt to avoid the resolution limit by describing the network hierarchy on multiple scales [66].

Label Swapping Problem

Methods like the above MCMC sampling that consider many partitions face the practical concern of label swapping or group identification. This is caused by not having a canonical representation for partitions that lends itself to efficient computation. For computational reasons set partitions are represented by a membership vector in $\mathbb{Z}_n^n$ where n is the number of elements, where the i-th entry of the membership vector gives the label of the community that element i is a member of. This requires only constant $\mathcal{O}(1)$ time to look up the membership of any element, versus potentially $\mathcal{O}(n)$ time if partitions were represented as a set of sets of elements. This space of membership vectors can also be thought of as the space of all colourings of n objects. The restriction that there be no more than n choices for each element comes from the fact there need not be more colours than objects to colour, as the most colours needed are when each of the n elements is given its own.

The problem is that for $n > 2$ the set of all membership vectors is not isomorphic to the set of all partitions, and in fact there will be many membership vectors that represent the same partition. For instance the trivial partition $\{1, 2, 3\}$ has corresponding membership vectors $[0, 0, 0], [1, 1, 1], [2, 2, 2]$ that are all equivalent.

[70] discusses the group identification problem in community detection, noting that the numeric values chosen for the group labels have no meaning. They further argue that MCMC algorithms based on moving a single node at a time get stuck with one choice of labelling, as a label swap is very unlikely to occur in an algorithm that moves single nodes at a time as it requires passing through low probability states. The posterior probability of a partition should be invariant under permutation of the labels, which won't be the case if only one choice of labelling was seen while sampling.

This will cause problems later in Example 2.4.6 when an MCMC partition sampler is constructed. This thesis's sampling algorithm will not suffer from this problem as it does not perform random walks or work with posterior probabilities, instead using a greedy Louvain method to propose likely partitions. It will however be a problem

when we reconcile the ensemble of partitions produced by our sampling algorithm and will be addressed in Section 3.3.

First, some means of comparing partitions that does not depend on the choice of labelling is needed. We introduce two metrics that can disambiguate community labels in an ensemble of partitions. The **maximum overlap distance** [70] finds a bijection (i.e. permutation) between the labels of two partitions that maximizes overlap. The **split/join metric** [25] (in places called the Dongen metric) just finds any map from one partition's labels to the other that maximizes overlap, dropping the injectivity condition. Other graph partition measures are discussed in [76] in the context of evaluating clustering algorithms.

Definition 2.3.1. Let $A, B \in \mathbb{Z}_n^n$ be membership vectors representing partitions of n elements, and n_A and n_B be the number of parts of A and B respectively. Without loss of generality we can permute the labels of A so that they are consecutive integers from 0 to $n_A - 1$, i.e. $A \in \mathbb{Z}_{n_A}^n$, and similarly $B \in \mathbb{Z}_{n_B}^n$.

Let $\mathfrak{S}_n$ denote the symmetric group on n elements. Then $\mathfrak{S}_n$ acts on $\mathbb{Z}_n^n$ elementwise, i.e. if $A = [A_1, \ldots, A_n]$ then for $\sigma \in \mathfrak{S}_n$ we have $\sigma(A) = [\sigma(A_1), \ldots, \sigma(A_n)]$ The maximum overlap distance $d_{MO}(A, B)$ is given by

$$d_{MO}(A, B) = n - \max_{\sigma \in \mathfrak{S}_n} \sum_{i=1}^{n} \delta(\sigma(A_i), B_i)$$

Equivalently,

$$d_{MO}(A, B) = \min_{\sigma \in \mathfrak{S}_n} \{d_{Hamming}(\sigma(A), B)\}$$

In fact we do not need to consider all of $\mathfrak{S}_n$, only the subgroup that acts on the labels seen in A and B, i.e. $\mathfrak{S}_{\max(n_A, n_B)}$.

This can be interpreted as the minimum classification error, i.e. elements with incorrect community label, when one partition is assumed to be the true one. Appendix A and B of [70] prove that this distance is a metric.

Suppose we are trying to find a permutation of the labels in the first partition that maximizes the overlap with the labels of the second partition (or equivalently that we are trying to minimize the Hamming distance between two vectors). Then this problem of finding the maximum overlap distance between two partitions is an instance of the assignment problem, also called the two-dimensional assignment problem, the linear sum assignment problem, and the bipartite matching problem.

The assignment problem involves matching "workers" to "tasks" where entries $C_{i,j}$ in a cost matrix C correspond to the cost of assigning worker i to task j. The problem is then to minimize the cost (or in our case to maximize the amount of overlap). The algorithm can then return a permutation matrix that sends workers to tasks. More compactly, it can return only the positions of the ones in the permutation matrix, which corresponds to the pairs of assignments.

First, each pair of one label from the first with one label from the second is considered. The number of positions that would agree under such a permutation is computed. Let $M_{i,j}$ store the number of positions that would agree if label i were permuted to label j. This matrix can be constructed in linear time i.e. with one pass of the partition by incrementing M_{a_k,b_k} for $k \in \{0,\ldots,|a|-1=|b|-1\}$. This matrix then becomes the cost matrix passed to an algorithm that solves the assignment problem.

There exists the polynomial time Kuhn-Munkres algorithm (sometimes called the Hungarian algorithm) for solving the assignment problem [21]. There also exists an approximation algorithm [26] that for $\epsilon > 0$ computes a $(1-\epsilon)$-approximate maximum weight matching in $\mathcal{O}(m\epsilon^{-1}\log(\epsilon^{-1}))$.

Alternatively we could consider a greedy algorithm that for each column (i.e. label from the first partition) picks the previously unpicked row with the largest agreement. This would have $\mathcal{O}(n)$ run time and would likely perform well, especially if rows were ordered by their max value which would bound the worst case error. Example 2.3.6

Example 2.3.6. Consider the partitions $A = \{\{1,2,3\},\{4,5,6,7\}\}$ and $B = \{\{1\},\{2,3,4,5,6,7\}\}$ with membership vectors $v_A = [0,0,0,1,1,1,1]$ and $v_B = [0,1,1,1,1,1,1]$ respectively. The associated cost matrix for the assignment problem is

		B	
		0	1
A	0	1	2
	1	0	4

Figure 2.20: Overlap for sending each label to the other.

Suppose we are greedily maximizing the overlap between the labels of v_A and v_B, and that we start by considering what the label of part $\{1,2,3\}$ from A should be. If we give it the same label as part $\{1\}$ of B it will only overlap in one location, whereas with the same label as part $\{2,3,4,5,6,7\}$ of B it will overlap in two. Thus at this step we would greedily choose to give $\{1,2,3\}$ from A and $\{2,3,4,5,6,7\}$ of B the same label. However this would be a mistake globally as in the next step we are forced to give $\{4,5,6,7\}$ of A and $\{1\}$ of B the same label even though they share no overlap, whereas we could have achieved an overlap of four by aligning the labels of $\{4,5,6,7\}$ of A with $\{2,3,4,5,6,7\}$ of B. This later choice would have lead to the global maxima for number of overlaps. Sorting the rows of 2.20 by max value would have meant row 2 would be considered first and a better distance achieved.

Alternatively, the injectivity condition for relabellings could be relaxed and any map between the labels considered, instead of the bijective permutations that the maximum overlap distance is restricted to. This could be computed with a single pass

over the labels by greedily choosing the relabelling that maximizes overlap, without caring if multiple labels are sent to the same target label. Implementing this idea we define the split-join distance.

Definition 2.3.2. Consider two partitions of n elements $A = A_1 \sqcup \cdots \sqcup A_{n_A}$ and $B = B_1 \sqcup \cdots \sqcup B_{n_B}$ where A_i is a subset of the elements.

The **partition number** of B with respect to A is given by

$$p_A(B) = \sum_{i=1}^{n_A} \max_{j=1}^{n_B} \{|A_i \cap B_j|\}$$

The split/join distance $d_{SJ}(A, B)$ is then

$$d_{SJ}(A, B) = d_{SJ}(A, A \cap B) + d_{SJ}(B, A \cap B) = n - p_A(B) + n - p_B(A) = 2n - p_A(B) - p_B(A)$$

In Theorem 4 of [25] it was shown that this distance is a metric.

A small projection number $p_A(B)$ indicates that A is close to being a subpartition, i.e. refinement, of B. This distance comes closer to capturing the number of single element moves required to move from one partition to another, which is of interest in algorithms that make such moves.

Example 2.3.7. Consider the partitions $\{\{1, 2, 3\}, \{4, 5, 6\}\}$ and $\{\{1\}, \{2, 3, 4, 5\}, \{6\}\}$ with membership vector $A = [0, 0, 0, 1, 1, 1]$ and $B = [0, 1, 1, 1, 1, 2]$ respectively. The maximum overlap distance would be achieved by without loss of generality the identity permutation, giving a distance of $6 - 1 - 2 = 3$. However if both labels of A were sent to 1 then the overlap would be 4, producing a distance of 2. The projection number $p_A(B)$ captures this with a value of $6 - 2 - 2 = 2$. The split/join distance is $(6 - 2 - 2) + (6 - 1 - 2 - 1) = 2 + 2 = 4$.

2.3.7 When Recovery is Feasible

When is recovery easy or even possible? There are two important thresholds regarding recovering the defining partition of a stochastic block model [1]. Consider the two block SBM with equally sized clusters and edge formation probabilities p within a block and q between blocks. Define **detection** or **partial recovery** to be correctly recovering $1/2 + \epsilon$ vertices for some $\epsilon > 0$, i.e. doing better than randomly guessing. Detection was first shown possible when $(a - b)^2 > 2\log(a + b)(a + b)$ [19]. The stronger threshold $(a - b)^2 > 2(a + b)$ was more recently shown [54, 58].

Define **exact recovery** to be when the partition can be recovered correctly with probability tending to one as the number of vertices goes to infinity. A necessary condition for exact recovery is that each block is connected, otherwise it is impossible to know which disconnected components of a block belong together. Consider a

single block, which internally is a $G(n,p)$ Erdös-Rényi random graph. Thinking of p as a function of n, we can consider the asymptotic properties of $G(n,p)$ as $n \to \infty$. We have $\mathbb{P}(G(n,p) \text{ is connected}) \sim 1 - n(1-p)^n$ asymptotically as $n \to \infty$, and an explicit formula also provided [37]. Letting $\epsilon > 0$, if $p > (1+\epsilon)\ln(n)/n$ then $G(n,p)$ will be connected asymptotically almost surely, and if $p < (1-\epsilon)\ln(n)/n$ then $G(n,p)$ will be disconnected asymptotically almost surely [29]. Thus $\ln(n)/n$ is a sharp threshold for the connectedness of $G(n,p)$.

Now suppose there are n vertices in total in the graph. If $p = a/n$ and $q = b/n$, asymptotically the graph will be disconnected with high probability and so exact recovery will not be possible. With $p = \alpha \log(n)/n$ and $q = \beta \log(n)/n$, exact recovery is possible when $(\alpha+\beta)/2 > 1 + \sqrt{\alpha\beta}$ and impossible otherwise [1]. Note that $(\alpha+\beta)/2 > 1$ is the connectivity threshold which is a necessary condition.

[2] introduces exact recovery conditions for SBMs with more than two blocks. They define the Chernoff-Hellinger (CH) divergence

$$D_+(u,v) := \max_{t\in[0,1]} \sum_{i\in[k]} \left(tu_i + (1-t)v_i - u_i^t v_i^{1-t}\right) \qquad \text{(Definition 11 of [2])}$$

where u, v are two nonnegative vectors and k the number of communities. Let $p \in (0,1)^k$ with $|p| = 1$ denote the prior probability of membership in each community, and $P = \text{diag}(p)$. Finally let $Q \in (0,\infty)^{k\times k}$ be a symmetric matrix with no two rows equal, with the edge probability between communities i and j taken to be $\min(1, Q_{ij}\log(n)/n)$. Then by their Theorem 6, exact recovery is possible if and only if for all i and j in different subsets of the partition

$$D_+((PQ)_i, (PQ)_j) \geq 1 \qquad \text{(Equation 29 of [2])}$$

where $(PQ)_i$ denotes the i-th row of the matrix PQ.

Corollary 3 of [2] extends their result about exact recovery to the overlapping stochastic block model (OSBM) that they introduce in their Section 4. Recall from our discussion of OSBM in Section 2.2.3 that every possible combination of the latent communities is assigned a new community under this model. Redefining the communities this way and considering this much larger model eliminates the problem of a vertex belong to multiple communities. They then apply their results on exact recovery of the SBM to the OSBM.

We end by noting that there is a gap between the accuracy promised by information theoretic results and the accuracy that current algorithms can actually achieve [77]. They specifically looked at the belief propagation algorithm [48] for community detection, calculating the limit of its accuracy as its number of steps was allowed to go to infinity. With respect to the signal-to-noise ratio (SNR) of the probability of edges within blocks (signal) to edges between blocks (noise), they observed an **undetectable** phase for low SNR where the algorithm had no accuracy but the theoretical

accuracy was nonzero, followed by an **easy** phase where the algorithmic accuracy matched the theory. For 2, 3, or 4 communities, the optimal and algorithmic accuracies were in agreement, however for 5 or more communities a gap appeared. This is captured in their Figure 1.

Our goal in this thesis is not specifically to shrink the gap between the theoretical and current best algorithmic accuracies in the hard low-SNR case. Instead we are focusing on a new approach to the overlapping community detection problem, and in reducing the required computational complexity.

2.4 Markov Chain Monte Carlo (MCMC)

When studying Markov chains we are interested in computing their limiting distribution. However for some chains it can take a long time for the empirical distribution to converge. This section develops this problem and some means of quantifying it. Finally we provide an overview of a solution to increase the speed of convergence, the Wang-Landau sampling method. Wang-Landau inspired this thesis's sampling method for graph partitions, detailed in Section 3.2.

2.4.1 Ergodicity

A sequence of random variables $(X_0, X_1, \ldots)$ is defined as a **Markov chain** with state space Ω and transition matrix P if for all $x_0, \ldots, x_{t+1} \in \Omega$, all $t \geq 1$, and all well defined events satisfying

$$\mathbb{P}\left(X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0\right) > 0$$

we have

$$\mathbb{P}\left(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0\right) = \mathbb{P}\left(X_{t+1} = x_{t+1} | X_t = x_t\right) =: P(x, y)$$

(Equation 1.1 of [53])

This condition is called the **Markov property**. A simple example is the **random walk** on a graph $G = (V, E)$, which is defined as the Markov chain with $\Omega = V$ where at current state $v \in V$ the next state is chosen from amongst the neighbours of v according to $P(v, \cdot)$. A random walk is defined as **simple** if the next vertex is chosen uniformly at random from amongst the current vertex's neighbours.

A Markov chain is defined as **finite** if its state space Ω is finite. Consider a finite Markov chain with transition matrix P and state space Ω. A chain is **reversible** if it satisfies the **detailed balance equations** defined as

$$\pi(x)P(x, y) = \pi(y)P(y, x) \quad \text{for all } x, y \in \Omega$$

Define the chain P as **irreducible** if for all states $x, y \in \Omega$ there exists some $t > 0$ such that $P^t(x, y) > 0$. In other words, it is possible to get from any state to any other

using only transitions of positive probability. Let $\mathcal{T}(x) := \{t \geq 1 \mid P^t(x,x) > 0\}$. The **period** of x is defined as the greatest common divisor of $\mathcal{T}(x)$. If P is irreducible then $\gcd(\mathcal{T}(x)) = \gcd(\mathcal{T}(y))$ for all $x, y \in \Omega$ (Lemma 1.6 of [53]). Define the chain as **aperiodic** if all states have period 1.

A distribution π on Ω is defined as **stationary** if $\pi = \pi P$, i.e. if π is an eigenvector of P with eigenvalue 1. Proposition 1.14 and Remark 1.15 of [53] state that every Markov chain has a stationary distribution. A Markov chain P is defined as **ergodic** if it converges to its stationary distribution π, i.e. $|P^t(x,y) - \pi_y| \to 0$ as $t \to \infty$, and is **geometrically ergodic** if it converges at a geometric rate, i.e. there exists some $\beta < 1$ such that $|P^t(x,y) - \pi_y| \leq M_{x,y}\beta^t$ as $t \to \infty$ and where M is some constant matrix [45]. Theorem 4.9 of [53] shows that irreducible aperiodic Markov chains converge to their stationary distributions. We are especially interested in how quickly a Markov chain converges to its stationary distribution as this determines the computational cost of sampling. In particular if the cost is too great then a problem can be computationally intractable.

At state x the next state is chosen according to probability distribution $P(x,\cdot)$ (which is the row of the transition matrix P corresponding to x). Similarly the distribution at time t given initial state x_0 is denoted $P^t(x_0,\cdot)$, We then want to bound the maximal distance between distribution $P^t(x_0,\cdot)$ and the stationary distribution π. To do this we define the distance

$$d(t) := \max_{x\in\Omega} \|P^t(x,\cdot) - \pi\|_{TV} \qquad \text{(Equation 4.22 of [53])}$$

where $\|\cdot\|_{TV}$ is the **total variation distance**. A Markov chain's ergodicity is explicitly quantified by its **mixing time**, denoted $t_{mix}(\epsilon)$, which measures the time required before the distance to stationarity is small.

$$t_{mix}(\epsilon) := \min\{t : d(t) \leq \epsilon\}$$

Define the **hitting time** of a Markov chain to be the number of steps before it enters a set for the first time. The authors of [72] show that "if the Markov chain has not hit a big set, then it cannot have mixed" and so relate hitting and mixing times. Let $d(t) := \max_{x\in\Omega}\|P^t(x,\cdot) - \pi\|_{TV}$. We must consider the lazy version of the chain to avoid issues of periodicity. Let P_L^t be the transition probability in t steps of the lazy chain, i.e. the chain with transition matrix $(P+I)/2$. Let $d_L(t) := \max_{x\in\Omega}\|P_L^t(x,\cdot) - \pi\|_{TV}$. The lazy mixing time is then defined as $t_L(\epsilon) := \min\{t \geq 0 \mid d_L(t) \leq \epsilon\}$. Let $\alpha < 1/2$ and define the maximum hitting time of "big" sets as

$$t_H(\alpha) = \max_{x,A:\pi(A)\geq\alpha} \mathbb{E}_x[\tau_A] \qquad (2.4.1)$$

where τ_A is the first hitting time of the set A by the Markov chain with transition matrix P. [72] shows that for every $\alpha > 0$ there exists a positive constant c'_α such

that

$$t_L(1/4) \geq c'_\alpha t_H(\alpha)$$

They show that when the chain is reversible the converse is true, stating that for $\alpha = 1/2$, there exists positive constants c'_α and c_α so that for every reversible chain

$$c'_\alpha t_H(\alpha) \leq t_L(1/4) \leq c_\alpha t_H(\alpha) \qquad \text{(Theorem 1.1 of [72])}$$

Define the average over two successive times as

$$t_{ave}(\epsilon) = \min\left\{t \geq 0 \mid \max_{x\in\Omega}\left\|\frac{P^t(x,\cdot) + P^{t+1}(x,\cdot)}{2} - \pi\right| \leq \epsilon\right\}$$

t_{ave} suffices as an alternative to considering the lazy chain. That is $t_L \asymp t_{ave}(1/4)$, i.e. there exist universal positive constants c and c' such that for every reversible Markov chain

$$ct_L \leq t_{ave}(1/4) \leq c't_L \qquad \text{(Theorem 1.4 of [72])}$$

We now relate this back to the problem of interest to this thesis. Let Ω be the set of all partitions of a graph. We can construct a graph from this set of partitions by creating an edge between two partitions if they differ by the membership of exactly one vertex. Louvain can be viewed as a random walk over this graph as it changes the community of one vertex at each step. The Markov chain Louvain induces on Ω contains many transient states that are only visited once. These are the lower modularity partitions along the path to a local modularity maxima as Louvain eventually settles on a stable partition from which it does not move. Since it has transient states it cannot be reversible.

In practice most implementations of Louvain are not true Markov chains as they break the Markov memoryless property in the following way. In its first phase Louvain considers each vertex sequentially when deciding whether to or into which community to move a vertex. Let the vertices be numbered $[1, \ldots, n]$ and considered in that order. Then if at step t vertex i was moved this means that at step $t+1$ vertex $i+1 \mod n$ is most likely to be moved, followed by vertex $i+2 \mod n$, etc. This is more efficient than randomly selecting a vertex to try moving. However when only the current state is known, the last vertex to change membership cannot be inferred, and hence the chain does not have the Markov property.

Example 2.4.1 (Lack of ergodicity in a real-world graph)**.** The karate club graph [91] is a real-world social network capturing friendships within a university's karate club. Figure 2.21 labels each vertex with the resulting change in modularity if it were removed from the single-part partition and placed in its own community. As can be seen, each move results in a decrease in modularity and so would not be chosen by Louvain. This is because the new part with only one vertex will not have any internal edges itself, and so will not have a positive modularity, and since removing

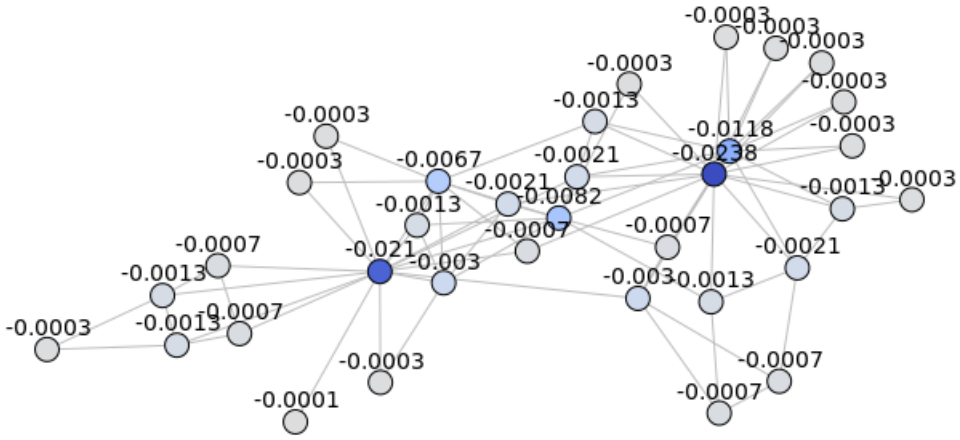

Figure 2.21: The karate club graph. Each vertex is labelled with the change in modularity if it were moved from the single-part partition to its own.

the vertex removes at least one internal edge from the single-part community, which is penalized. The only scenario where this penalty would not be incurred is if a disconnected singleton vertex were placed in its own community, resulting in no change in modularity. Thus the single-part partition is an absorbing state that cannot be left by Louvain. Louvain has many such states.

Louvain strictly increases the modularity at every step. Since modularity is strictly increasing and the state space is finite, eventually Louvain will reach a local maximum. Hence the Markov chain induced by Louvain is an absorbing Markov chain, i.e. one where every state can reach an absorbing state, and the absorbing states are exactly the local maxima. Thus Louvain is guaranteed to terminate. As far as we know there are some unknowns in the theory behind Louvain. With respect to the Markov chain induced by Louvain, the hitting time of the set of absorbing states, i.e. the time until the algorithm terminates, appears unknown. Likewise a general bound on the computational complexity of the algorithm appears unknown. The authors of [50] claim without proof that Louvain is "essentially linear in the number of links of the graph".

This problem of a lack of ergodicity was identified by Peixoto in [69]. That paper proposes the split-merge algorithm for sampling partitions. In contrast to Louvain, the Bayesian split-merge sampling algorithm is constructed to be a MCMC method. These single-vertex communities will have a very small acceptance probability, and so the single-part partition is again an absorbing state. To overcome this Peixoto introduces "split" moves that allow more than one vertex to be moved at once into a new community. This greatly increases the number of states obtainable at each step from at most n if every vertex belongs to its own community, to something on the

order of B_n.

The number of partitions of n vertices is the n-th Bell number B_n. In Louvain the number of moves for a given vertex is bounded by the number of neighbours, and is realized in the worst case when every neighbour belongs to a different community. In split-merge there are in theory n possible moves for a single vertex. This is much higher than the average degrees of most families of random graphs we have looked at. Thus split-merge allows many more moves.

2.4.2 Bottlenecks

Another way to rephrase the problem of ergodicity is as a problem of the graph having bottlenecks between regions in the state space. More formally, a graph can be said to contain a bottleneck if there exists some subgraph with more vertices than neighbours. Having $|U| > |\partial U|$ is the same as saying that the ratio $\frac{|\partial U|}{|U|} < 1$. This ratio will be developed below into a more formal measure of bottleneckedness.

The graph-theoretic **Cheeger constant**, also called the **conductance**, is a measure of whether a graph has a bottleneck, i.e. if there exists a partition of the graph into two sets with few edges between them. Let $G = (V, E)$ be a graph and $U \subseteq V$ a subset of the vertices. Then the **edge boundary** of U is the set of edges between vertices in U and the remaining edges in V, i.e.

$$\partial U = \{(u, v) \in E \mid |u \in U, v \in V \setminus U\}$$

Selecting U induces a 2-part partition on V. Define the **Cheeger ratio** or **bottleneck ratio** to be the number of edges leaving a subgraph, normalized by the number of vertices in that subgraph. Then define the Cheeger constant $h(G)$ to be the minimum of the Cheeger ratio over all possible subgraphs

$$h(G) := \min_{\substack{\emptyset \neq U \subseteq V \\ 0 < |U| \leq \frac{1}{2}|V|}} \left\{ \frac{|\partial U|}{|U|} \right\}$$

The constraint that the size of U be not more than half the number of vertices in G ensures that the Cheeger ratio is well-defined with respect to the proposed 2-part partition. Otherwise both U and its complement $V \setminus U$ represent the same 2-part partition and share an edge boundary, but will have different Cheeger ratios if their sizes do not agree. In particular the larger set will have a smaller Cheeger ratio. By only considering the smaller of this pair a unique value is assigned to each 2-part partition.

The Cheeger constant can be extended to Markov chains. Intuitively we expect an ergodic Markov chain to Let $G = (V, E)$ be the underlying graph of an ergodic reversible Markov chain with stationary distribution π. Let U be a non-empty set of

states with non-empty complement $\bar{U}$. Define the **capacity** of U to be

$$C_U := \sum_{u \in U} \pi_u$$

and the **ergodic flow** of U to be

$$F_U := \sum_{u \in U} \sum_{v \in \bar{U}} p_{uv} \pi_u$$

Note that $0 < F_U \leq C_U < 1$. Then the Cheeger ratio is $\Phi_U = F_U / C_U$, which is the conditional probability that the stationary process leaves U in a single step given that it starts in U. The Cheeger constant is then defined as

$$\Phi = \min_{\substack{U \subseteq V \\ C_U \leq 1/2}} \Phi_U$$

We can relate the bottleneck ratio directly to ergodicity. Consider a Markov chain with transition matrix P and state space Ω. Suppose it is irreducible and aperiodic with stationary distribution π. The bottleneck ratio relates back to ergodicity by providing the following lower bound on the mixing time

$$t_{mix}(1/4) \geq \frac{1}{4h(G)} \qquad \text{(Theorem 7.3 of [53])}$$

We can also get an upper bound on the mixing time from the bottleneck ratio. Let λ_2 be the second largest eigenvalue of a reversible transition matrix P, and define the **spectral gap** $\gamma = 1 - \lambda_2$. Then

$$\frac{h(G)^2}{2} \leq \gamma \leq 2h(G) \qquad \text{(Theorem 13.14 of [53])}$$

It follows that

$$2h(G)^{-2} \geq \gamma^{-1}$$

Let $\pi_{\min} := \min_{x \in \Omega} \pi(x)$ and define the **absolute spectral gap** $\gamma_* = 1 - \max_{j \geq 1} |\lambda_j|$. Then

$$t_{mix}(\epsilon) \leq -\log(\epsilon \pi_{\min}) \gamma_*^{-1} \qquad \text{(Theorem 12.3 of [53])}$$

If our chain is **lazy**, that is it remains in its current state with probability at least $1/2$, then $\lambda = \lambda_*$ (Exercise 12.3 of [53]). Thus we get the upper bound

$$t_{mix}(\epsilon) \leq -\log(\epsilon \pi_{\min}) 2h(G)^{-2}$$

Example 2.4.2 (Cheeger constant of complete graph)**.** The complete graph on n vertices obtains the maximum Cheeger constant as each vertex is connected to every other, thus having no bottlenecks. By symmetry and the indistinguishability of the vertices, it suffices to consider each possible size of subset to determine which is the minimizer of the Cheeger ratio. Note that each subset of k vertices will form a k-clique.

Let U be some k-clique of K_n. The allowed values of k are $\left[1, \ldots, \left\lfloor \frac{1}{2}|V| \right\rfloor\right]$. U has $|\partial U| = k(n-k)$ edges in the boundary as each of the k vertices is connected to the $n-k$ vertices outside U. The Cheeger constant is the minimum of $\frac{k(n-k)}{k} = n-k$ which is decreasing in k. Thus $k^* = \left\lfloor \frac{1}{2}|V| \right\rfloor$ is the largest allowable subset, producing the Cheeger constant of

$$h(K_n) = n - \left\lfloor \frac{1}{2} n \right\rfloor = \begin{cases} \frac{n}{2} & n \text{ even} \\ \frac{n-1}{2} & n \text{ odd} \end{cases}$$

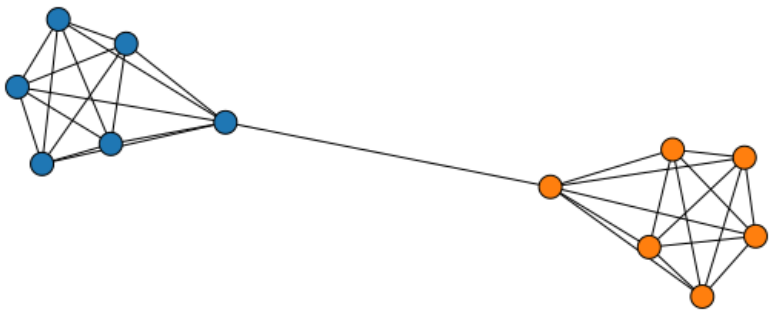

Figure 2.22: The 6-barbell graph obtained by connecting two copies of K_6 by a bridge.

Example 2.4.3 (Cheeger constant of barbell graph)**.** Consider the **barbell graph** composed of two copies of the complete graph on n vertices (i.e. K_n) and a single edge connecting these two components. We refer to the edge connecting these two as a bridge.

For the barbell graph, the Cheeger constant will achieve its minimum when partitioning the graph into its two n-cliques. The edge boundary will have cardinality $|\partial U| = 1$ as it only includes the bridge edge, giving $h(n - \text{barbell graph}) = \frac{2}{n}$. Note that each copy of K_n could be replaced with any connected graph on n vertices and the Cheeger constant would remain the same.

Consider a random walk on the barbell graph where at each vertex a neighbour is chosen uniformly at random as the next state. Then it is likely the walk would remain trapped in the clique containing as the probability of choosing the bridge edge to the other clique is small. The expected waiting time before leaving the starting clique follows a geometric distribution. At time step t, there is a $\frac{1}{n}$ chance if the

current state is the endpoint of the bridge, or otherwise a $\frac{1}{n-1}\frac{1}{n}$ probability to first go to the bridge endpoint and then leave. A simpler upper bound is given by picking an edge and direction uniformly at random from the edges incident to the starting clique. This follows a Bernoulli distribution with a $\frac{1}{(n-1)n+1}$ probability of picking the bridge. The expected waiting time for the first success follows a geometric distribution and is $(n-1)n+1$ time steps. Now compare this to the complete graph on the same number of vertices, namely K_{2n}, where the vertices have similarly been partitioned into two disjoint cliques of equal size. There the probability of leaving the starting clique is $\frac{n}{2n-1}$ at each step, and so the expected waiting time to leave is $\frac{2n-1}{n} < 2$ time steps. Thus the required number of samples before observing a member of the other clique is quadratic in n for the barbell graph but constant for the complete graph.

The barbell and complete graphs give lower and upper bounds respectively on the Cheeger constant. Thus $\frac{2}{n} \leq h(G) \leq \frac{n}{2}$ for any connected graph G on n vertices, and $h(G) = 0$ if G is disconnected.

2.4.3 Wang-Landau Algorithm

Often we want to get past bottlenecks faster than standard Markov chain Monte Carlo sampling techniques allow. This motivated the development of sampling techniques that modify the density of states over the course of the algorithm. **Importance sampling** is a family of techniques that estimate properties of a distribution from samples of a different but related distribution. **Umbrella sampling** [83] is a related method that seeks to overcome bottlenecks. In Example 2.4.4 we will see that the mean-field Ising model has exponential mixing time, whereas the Wang-Landau process converges to a Markov chain with mixing time in $\mathcal{O}(n \log n)$. This is a significant potential improvement. This section introduces the Wang-Landau sampling method that inspired this thesis's proposed sampling method.

The Wang-Landau algorithm [85] estimates the density of states in a Markov chain with respect to some cost function. It does this faster than the standard random walk by performing a specific non-Markovian, i.e. non-memoryless, random walk. As states are visited the walk becomes biased away from states with costs that it has already seen. This process eventually converges to a Markov chain with a uniform distribution over the possible costs, and hence has fast mixing. While modifying the original Markov chain to produce faster mixing, the true densities of the cost function are calculated. We now describe the algorithm more carefully.

Let Ω be the state space. Under the Ising model described in Example 2.4.4, $\Omega = \{-1, +1\}^{|V|}$ is all possible labellings of the vertices of a graph $G = (V, E)$ with either positive or negative one. Equivalently, it is the space of all 2-part partitions of the graph. In community detection, Ω would be all possible partitions of the graph's vertices. In general we are often interested in limiting the choice of valid partitions on

Algorithm 4: Wang-Landau Density Estimation Algorithm

Input: graph $G = (V, E)$,
space of allowed partitions $\Omega \subseteq P_{|V|}$ where $P_{|V|}$ is the space of all partitions of G
random proposal function $\sigma : \Omega \to \Omega$
energy function $E : \Omega \to [E_{\min}, E_{\max}]$,
number of bins $N \in \mathbb{N}$,
density tolerance δ used as a termination condition
Initialize: Let $\Delta_N = \frac{E_{\max} - E_{\min}}{N}$ and
$I_N = \{[E_{\min} + (i-1)\Delta_N, E_{\min} + i\Delta_N]\}_{i=1}^{N}$
Set energy level histogram $H(I_i) = 0$ and density $g(I_i) = 1$ for each $I_i \in I_N$
Let $D : [E_{\min}, E_{\max}] \to I_N$ be the map that identifies the interval containing the argument
Set convergence factor $f(t) = \exp(1/t)$
Set $t = 0$
Generate a valid starting partition
repeat
  Select random $C_0 \in \Omega$
until $E(C_0) \in [E_{\min}, E_{\max}]$;
Random Walk:
while $|1 - f(t)| < \delta$ **do**
  Generate a valid proposal
  repeat
    Select random move proposal $C' = \sigma(C_t) \in \Omega$
  until $E(C') \in [E_{\min}, E_{\max}]$;
  Compute the acceptance probability
  $A(C') = \min\left\{1, \frac{g(E(C_t))}{g(E(C'))}\right\}$
  Sample $U \sim Uniform(0, 1)$
  if $U \leq A(C')$ **then**
    Accept the proposal $C_{t+1} = C'$
  else
    Reject the proposal and keep the current state $C_{t+1} = C_t$
  Update energy level histogram $H(D(E(C_{t+1})) = H(D(E(C_{t+1})) + 1$
  Update energy level density by convergence factor
  $g(D(E(C_{t+1})) = g(D(E(C_{t+1})) \cdot f(t)$
  Set $t = t + 1$ to increment time
$g_{norm}(I_i) = \frac{\ln(g(I_i))}{\sum_{j=1}^{N} \ln(g(I_j))}$ for $i \in [1, \ldots, N]$
return g_{norm}

a graph to the subset that generate connected subgraphs for each part, thus reducing the state space. This can be achieved by defining Ω to be this set.

Let $E : \Omega \to [E_{\min}, E_{\max}]$ be a cost function. In statistical physics the cost function is often the energy of a partition. Later in Example 2.4.6 the cost function will be the negative of the modularity of a partition. For $N \in \mathbb{N}$, let $\Delta_N = \frac{E_{\max} - E_{\min}}{N}$ and let $I_N = \{[E_{\min} + (i-1)\Delta_N, E_{\min} + i\Delta_N]\}_{i=1}^{N}$ be the usual partition of the interval $[E_{\min}, E_{\max}]$ into N parts.

Let $g : I_N \to [0, \infty)$ be a density we assign to each interval in I_N, and let $H : I_N \to \{0\} \cup \mathbb{N}$ record the number of times the algorithm has visited each interval in I_N. The values of these functions will be updated at each step of the algorithm's random walk. H is initialized identically to zero as no states have yet been visited and so no values for the cost function yet observed. g is initialized identically to the uninformed prior of one as a cost value from any of the intervals is a priori equally likely to be seen.

A random initial partition $C \in \Omega$ is chosen and the algorithm begins its random walk by proposing new states and either accepting or rejecting each proposal. Under the Ising model that Wang and Landau originally worked with this involved randomly selecting a vertex and flipping its label's sign. In the community detection setting it would again involve randomly selecting a vertex but then changing its community assignment. Suppose $C' \in \Omega$ is proposed as a move. The transition probability for moving from cost $E(C)$ to $E(C')$ is given by

$$\mathbb{P}(E(C) \to E(C')) = \min\left\{1, \frac{g(E(C))}{g(E(C'))}\right\}$$

This probability is also taken to be the probability of accepting state C'. In the traditional Metropolis algorithm, a function $f(x)$ that is proportional to the desired probability distribution $\mathbb{P}(x)$ is used to construct the acceptance ratio $f(x')/f(x) = \mathbb{P}(x')/\mathbb{P}(x)$ for moving from state x to x'. In the Wang-Landau algorithm the acceptance ratio is the inverse of the Metropolis construction.

Let $D : [E_{\min}, E_{\max}] \to I_N$ be a function that discretizes an energy value by sending it to the interval in I_N that it is contained by. After each proposal the histogram H is updated by incrementing the count for the interval $D(E)$ containing the new energy level E, i.e. $H(D(E(C'))) = H(D(E(C'))) + 1$ if the proposal is accepted or $H(D(E(C))) = H(D(E(C))) + 1$ if the proposal is rejected. Additionally the corresponding density g is updated by multiplying by a convergence factor f, with $g(D(E(C'))) = g(D(E(C'))) \times f$ or similarly if C was chosen. The authors suggested $f = e$. Changing the acceptance probabilities at each step is the central idea of the Wang-Landau algorithm as this update makes it less likely that states of already seen energy levels will be visited in the future. This speeds up traversal of the image of the cost function. Updating this probability makes the algorithm non-Markovian,

as the stochastic process now depends on the previous states visited and so is not memoryless.

Proposals are repeatedly made until the histogram $H(I_N)$ becomes sufficiently close to the uniform distribution. The authors define this as the **flatness condition** and require the histogram value $H(I_i)$ for each $I_i \in I_N$ not be less than 80% of the average of all the histogram values. Alternatively the condition could be that a fixed number of steps at the current stage have been taken. At this point the convergence factor is decreased, usually by $f_{k+1} = \sqrt{f_k}$ where f_k is the convergence factor at stage k. The histogram H is then reset to zero identically everywhere. These stages continue until f becomes sufficiently close to 1, for example the authors suggest $f < \exp(10^{-8}) \simeq 1.00000001$ as a termination condition. This can be written more compactly by specifying a density tolerance δ and terminating when $|1 - f| < \delta$, i.e. $\delta = 10^{-8}$ for their suggestion.

After terminating we are left with $g(E(I_i))$, which is the exponential of the relative density of each interval I_i. To get the absolute density the normalization

$$g_{norm}(E_i) = \frac{\ln(g(E(I_i)))}{\sum_{j=1}^{N} \ln(g(E(I_j)))}$$

is performed for each interval I_i in I_N.

It was later shown in [7] that decreasing f faster than $1/t$, where t is the Monte Carlo time, causes the error to converge to some nonzero value and hence causes the density to not converge to the true value. Instead the $1/t$ **algorithm** was proposed to avoid this problem. It sets the convergence factor to $f(t) = \exp(1/t)$ and is updated at each Monte Carlo time step. While this may make the estimate more accurate than the original algorithm, the longer run time makes it less efficient. In Example 2.4.6 it will be shown that the $1/t$ convergence factor leads to convergence that is far too slow to be practically useful. The authors that proposed the $1/t$ algorithm later suggested a hybrid version in [6] that uses $f = e$ for the first 1000 steps and then switches to $f(t) = 1/t$. This provides faster convergence while also some theoretical guarantee of convergence to the correct value.

Example 2.4.4 (Ising model)**.** The **Ising model** on a graph $G = (V, E)$ at inverse temperature $\beta \geq 0$ is a probability measure on labels $C : V \to \{-1, 1\}$. The **energy** of a partition C is defined to be

$$H(C) = -\sum_{\substack{v,w \in V \\ (v,w) \in E}} C(v)C(w) \qquad \text{(Equation 3.7 of [53])}$$

The Ising model has likelihood proportional to $\mu(C) = \exp(-\beta H(C))$ (see Equation 3.8 of [53]). At each step a vertex is chosen uniformly at random and its label updated according to the distribution μ conditioned on the labels of the other vertices. The inverse temperature β determines the importance of the energy function.

At infinite temperature ($\beta = 0$) the energy H plays no role and μ is the uniform distribution. For larger β, μ becomes biased towards low-energy partitions. In the low-energy setting the most likely partitions are those with all -1 or all $+1$.

Define the **mean-field** Ising model to be when $G = K_n$ is the complete graph. For $\beta = \gamma n^{-1}$, if $\gamma < 1$ then $t_{mix} = \mathcal{O}(n \log n)$, and if $\gamma > 1$ then there exists some positive function $r(\gamma)$ such that $t_{mix} \geq \mathcal{O}(\exp[r(\gamma)n])$ (Theorem 15.3 of [53]). Thus at low temperatures ($\gamma > 1$) the mean-field Ising model has slow mixing.

The Wang-Landau process on the mean-field Ising model converges to the simple random walk on the n-dimensional hypercube, where at each step a vertex is chosen uniformly at random and its label flipped. Its mixing time has the upper bound

$$t_{mix}(\epsilon) \leq n \log n - \log(\epsilon) n \qquad \text{(Equation 6.15 of [53])}$$

This is much smaller than the exponential mixing time of the mean-field Ising model. It is for this improvement to mixing time that the Wang-Landau algorithm is used.

Above we identified a phase transition in the dynamics of the mean-field Ising model; for inverse temperature β less than n^{-1} there was fast mixing, and for β greater than n^{-1} there was slow mixing. These phase transitions are of especial interest in statistical physics. Matter can be given a simple representation by taking G to be a lattice graph, which allows only neighbouring particles to interact. These phase transitions then occur when a more ordered and a more disordered form of the same material are in thermal equilibrium with each other. One example is the changes between states of matter. At the melting point a highly structured crystalline solid transforms into a liquid with weaker interactions between atoms/molecules. At the boiling point even these weak interactions disappear and individual particles begin moving independently. Another example is the magnetic phase transition between ferro- and para-magnetic states. Magnetic interaction energy comes from the magnetic spins of neighbouring particles being aligned. The ± 1 labels of the Ising model then correspond to up and down orientations of the individual magnetic particles. Ferromagnets have greater magnetic interaction energy and lesser thermal energy, while paramagnets have greater temperature and randomly aligned spins. Other examples of phase transitions include super versus normal conducting states, crystallographic (structural) changes within solid phases, etc. Much of the motivation (and language) for the method presented in this section comes from statistical physics.

Example 2.4.5 (Wang-Landau on the barbell graph)**.** Recall the barbell graph from Example 2.4.3 that consists of two copies of K_n connected by a single edge for a total of $2n$ vertices. A random walk is defined on this graph by choosing a neighbouring vertex uniformly at random. For simplicity we will consider a chain merely inspired by the barbell. To construct it we "collapse" the associated Markov chain into four states; the exchangeable vertices within the left and right bells become their own states and the unique left and right vertices that form the bar remain as is. Figure 2.23

depicts this Markov chain and its transition probabilities. We'll see that this chain has hitting/mixing time on the order of n^2, but that the limiting Wang-Landau chain has hitting/mixing time on the order of n.

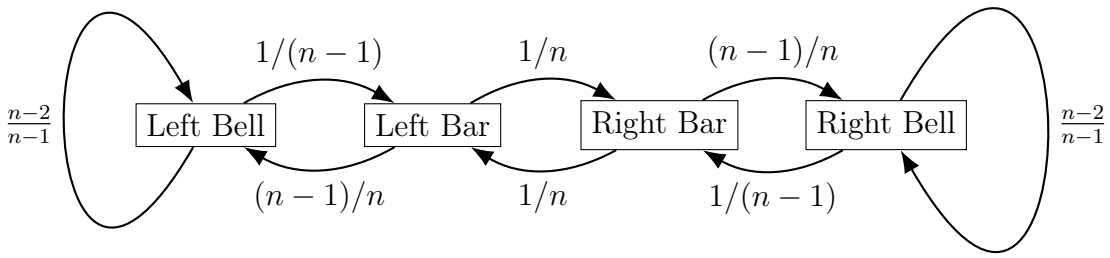

Figure 2.23: "Collapsed" Markov chain of the barbell graph.

This has transition matrix

$$P = \begin{bmatrix} (n-2)/(n-1) & 1/(n-1) & 0 & 0 \\ (n-1)/n & 0 & 1/n & 0 \\ 0 & 1/n & 0 & (n-1)/n \\ 0 & 0 & 1/(n-1) & (n-2)/(n-1) \end{bmatrix}$$

and stationary distribution π satisfying $\pi P = \pi$ where

$$\pi = \frac{1}{2 + 2[n/(n-1)^2]} \begin{pmatrix} 1 & n/(n-1)^2 & n/(n-1)^2 & 1 \end{pmatrix}.$$

It has eigenvalues

$$\left\{ \begin{matrix} 1, \\ -\frac{1}{(n-1)n}, \end{matrix} \quad \begin{matrix} \frac{n^2+\sqrt{n^4+2n^3-9n^2+6n+1}-3n+1}{2(n-1)n}, \\ \frac{n^2-\sqrt{n^4+2n^3-9n^2+6n+1}-3n+1}{2(n-1)n} \end{matrix} \right\}$$

and hence spectral gap

$$\gamma(n) = 1 - \frac{n^2 + \sqrt{n^4 + 2n^3 - 9n^2 + 6n + 1} - 3n + 1}{2(n-1)n}$$

$$= \frac{1}{2}\left[-\frac{\sqrt{n^4+2n^3-9n^2+6n+1}}{(n-1)n} - \frac{n}{(n-1)} + \frac{3}{(n-1)} - \frac{1}{(n-1)n} + 2\right]$$

which converges to zero as n goes to infinity. For $n > 2$ we have $\gamma(n) < \frac{2}{n^2}$, and hence $\gamma(n) = \mathcal{O}\left(\frac{1}{n^2}\right)$. Thus this chain will have very slow mixing as the spectral gap converges to zero like $\frac{1}{n^2}$ i.e. very quickly.

We now compute the Cheeger constant of this Markov chain. There are only finitely many cases, and it can be shown that the minimizer of the Cheeger ratio is the subset containing a bell and bar of the same half. Without loss of generality we consider the left half. Let U be this subset containing the left bell and left bar. Its capacity is $C_U = \frac{1}{2}$ and its ergodic flow is $F_U = \frac{1}{n}\frac{n}{(n-1)^2} = \frac{1}{(n-1)^2}$. Then its Cheeger ratio is $\Phi_U = F_U/C_U = \frac{2}{(n-1)^2}$. Thus for $n \geq 3$, $\Phi = \frac{2}{(n-1)^2}$. This gives us the lower bound on the mixing time of

$$t_{mix}(1/4) \geq \frac{1}{4\Phi} = \frac{(n-1)^2}{8}$$

which again shows this chain will have very slow mixing on the order of at least $\mathcal{O}(n^2)$.

The collapsed barbell graph is an instance of a **birth-and-death chain** as defined in Section 2.5 of [53]. Note that every birth-and-death chain is reversible (Proposition 2.8 of [53]). For $a < b$ we have

$$\mathbb{E}_a[\tau_b] = \sum_{\ell=a+1}^{b} \mathbb{E}_{\ell-1}[\tau_\ell]$$

where

$$\mathbb{E}_{\ell-1}[\tau_\ell] = \frac{1}{q_\ell w_\ell}\sum_{j=0}^{\ell-1} w_j \qquad \text{(Equation 2.16 of [53])}$$

Let the states $\{\text{left bell}, \text{left bar}, \text{right bar}, \text{right bell}\}$ correspond to the labels $\{0,1,2,3\}$. The hitting time of the right half starting from the left bell is then

$$\begin{aligned}
\mathbb{E}_0[\tau_2] &= \sum_{\ell=1}^{2} \mathbb{E}_{\ell-1}[\tau_\ell] \\
&= \mathbb{E}_0[\tau_1] + \mathbb{E}_1[\tau_2] \\
&= \frac{1}{q_1 w_1} w_0 + \frac{1}{q_2 w_2}\left[w_0 + w_1\right] \\
&= \frac{1}{\frac{n-1}{n}\frac{n}{(n-1)^2}} + \frac{1}{\frac{1}{n}\frac{n}{(n-1)^2}}\left[1 + \frac{n}{(n-1)^2}\right] \\
&= (n-1) + (n-1)^2 + n \\
&= n^2
\end{aligned}$$

In Equation 2.4.1 we defined the maximum hitting time of "big" sets where, for $\alpha < 1/2$, we have

$$t_H(\alpha) = \max_{x,A:\pi(A)\geq\alpha} \mathbb{E}_x[\tau_A]$$

Let $\alpha = (n-1)/(2n) < 1/2$ and A be the right half. By symmetry $\pi(A) = 1/2$ which is greater than α. By inspection the maximizer of $t_H(\alpha)$ is A as above and x equal to the left bell. Hence $t_H(\alpha) = \mathbb{E}_0[\tau_2] = n^2$. Then by Theorems 1.1 and 1.4 of [72] there exists positive constants c and c'_α such that

$$\frac{c'_\alpha}{c} t_H(\alpha) \leq t_{ave}(1/4)$$

This providing a lower bound on the average mixing time.

$$\frac{c'_\alpha}{c} n^2 \leq t_{ave}(1/4)$$

We would like to proceed similarly by computing the hitting time for the Wang-Landau algorithm, and then using the result from [72] to get an upper bound on the average mixing time. With an upper bound on the Wang-Landau algorithm's hitting time and lower bound on that of the original chain, the improvement in rate of convergence should be apparent. Unfortunately the real Wang-Landau algorithm isn't a Markov chain on the original state space, and so does not have a mixing time (or spectral gap). It does have worst-case hitting times. There are two further problems when trying to apply this theory to a non-Markovian chain like the Wang-Landau algorithm. The first is that the chain must be reversible in order to get the upper bound from Theorem 1.1 of [72] of $t_L \leq c_\alpha t_H(\alpha)$ for some constant c_α. The second is that π is the stationary distribution of the original Markov chain and this is not equal to the limiting distribution of the Wang-Landau process, which is the uniform distribution.

Instead we will sketch how Wang-Landau improves mixing. Let $\Omega = V$ and cost function $E : V \to \mathbb{Z}_4$ simply identify which state of the collapsed barbell we are in. At each step of the random walk Wang-Landau will increment the density of the current state, biasing future steps away from it. Without loss of generality suppose the walk starts in the left bell. We collapse into one state the two states of the right bell, i.e. states 2 and 3.

The acceptance ratio for moving from states 1 to 2 will be 1 until we first hit state 2. Moving between states 0 and 1 will be affected by the Wang-Landau process. Let $g_t(x)$ denote the density of state x at time t. Note that $g_t(x)$ is a statistic of the previous random states the walk visited. When computing the hitting time we are

interested in the truncated chain with the following transition matrix

$$P = \begin{bmatrix} \frac{n-2}{n-1}\left(1 - \min\left\{1, \frac{g_t(1)}{g_t(0)}\right\}\right) & \frac{1}{n-1} + \frac{n-2}{n-1}\min\left\{1, \frac{g_t(1)}{g_t(0)}\right\} & 0 \\ \frac{n-1}{n}\min\left\{1, \frac{g_t(0)}{g_t(1)}\right\} & \frac{n-1}{n}\left(1 - \min\left\{1, \frac{g_t(0)}{g_t(1)}\right\}\right) & \frac{1}{n} \\ 0 & \frac{1}{n} & \frac{n-1}{n} \end{bmatrix}$$

This converges to

$$P = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{n-1}{n}\frac{1}{2} & \frac{n-1}{n}\frac{1}{2} & \frac{1}{n} \\ 0 & \frac{1}{n} & \frac{n-1}{n} \end{bmatrix}$$

This has stationary distribution

$$\pi = \left(1 + 2\frac{n}{n-1}\right)\begin{pmatrix} 1 & \frac{n}{n-1} & \frac{n}{n-1} \end{pmatrix}.$$

The expected hitting time of this limiting truncated chain is then

$$\begin{aligned} \mathbb{E}_0[\tau_2] &= \sum_{\ell=1}^{2} \mathbb{E}_{\ell-1}[\tau_\ell] \\ &= \mathbb{E}_0[\tau_1] + \mathbb{E}_1[\tau_2] \\ &= \frac{1}{q_1 w_1} w_0 + \frac{1}{q_2 w_2}\left[w_0 + w_1\right] \\ &= \frac{1}{\frac{n-1}{n}\frac{1}{2}\frac{n}{n-1}} + \frac{1}{\frac{1}{n}\frac{n}{n-1}}\left[1 + \frac{n}{n-1}\right] \\ &= 2 + (n-1) + n \\ &= 2n + 1 \end{aligned}$$

As a linear hitting time this is significantly faster than the quadratic hitting time before. As it takes time for the Wang-Landau process to converge to this chain, this is only a lower bound on the hitting time.

Example 2.4.6 (Wang-Landau estimation of partition density at each modularity level)**.** This example gives an estimate of the density of partitions at each modularity level using the Wang-Landau method. The original Wang-Landau paper considered the Ising model with G equal to the 50×50 square lattice graph This small size was chosen as the exact energy distribution could be calculated. Here a similarly small example is chosen in the community detection setting where every partition can be enumerated and its modularity computed to yield the exact modularity distribution. This is just a toy example and as far as we know not a problem that is actually studied.

To generate a proposal the Wang-Landau algorithm picks a vertex at random and flips the sign of its label. For this two community case there is a bijection between $\leq$ 2-part partitions and 2-colourings and so sampling by randomly picking 2-colourings is equivalent. In the case where there are more than two communities a vertex could belong to this will not work as the number of colourings grows faster than the number of partitions. For instance the number of partitions of 8 vertices is $B_8 = 4140$ and the number of 8-colourings of 8 vertices is 8^8, so the partitions only represent approximately 2.4×10^{-4} of all colouring states. As this is far from injective, sampling from the colourings is insufficient and so a different proposal generator must be used. Instead a vertex can be selected at random and a label randomly chosen from among the communities of the other vertices. Additionally if the selected vertex is not the sole member of a community, then a new label is also proposed as a choice. If it did belong to a singleton community, then suggesting a new community would be simply proposing the existing state. Thus sampling is equivalent to a random walk on the poset of partitions ordered by whether they are one move away from each other.

Consider the graph on 8 vertices introduced in Figure 2.13 and comprised of four 2-cliques with all edges added between each pair of cliques except for two. In this configuration there are actually two disjoint pairs of 2 4-cliques; one vertical and one horizontal pair. This graph will be referred to as a $[2, 2]$-block lattice graph; with the cliques forming the corners of a lattice, there are two ways to partition them into 4-cliques. In total there are $B_8 = 4140$ partitions of 8 vertices so it is feasible to enumerate every possible partition, calculate the associated modularity, and update the histogram to get the true density. In fact, due to the symmetry of the blocks and indistinguishability of the vertices within each block, less partitions need to be considered, but for the simplicity of computation this was not done. In general it will not be possible to calculate the exact entropy, however studying this small example allows the accuracy of Wang-Landau to be tested.

The termination condition $\exp(1/t) < 1 + 10^{-6}$ is achieved when $t = 10^6$ so this number of time steps will be used as an equivalent termination condition.

Figure 2.29 gives the true density of partitions at each energy level, computed by enumerating each possible partition and computing its modularity. The energy spectrum was discretized into 41 energy levels from -0.3 to 0.1. Figure 2.30 gives the estimated density computed after taking 100,000 steps of the modified $1/t$ Wang-Landau method. Figure 2.28 gives the difference in the densities, capturing the error in this case for the Wang-Landau method. In this example we had $100,000/4140 \approx 24$ times the number of samples as the total number of partitions and still had a sizable error between the true and estimated distributions. However because this example was so small it is not surprising that Wang-Landau did not perform well, as MCMC and related methods like Wang-Landau work very poorly for small problems. For "sufficiently large" problems we would expect MCMC to outperform enumeration. Tuning the parameters of Wang-Landau may also have improve performance.

The energy landscape can be visualized through an embedding of the partitions into $\mathbb{R}^2$ as in Figure 2.24. This was created via multi-dimensional scaling (MDS) using the split/join metric introduced in Section 2.3.6. Just the partitions with positive modularity can be embedded as in Figure 2.26. The two maxima are the two modes.

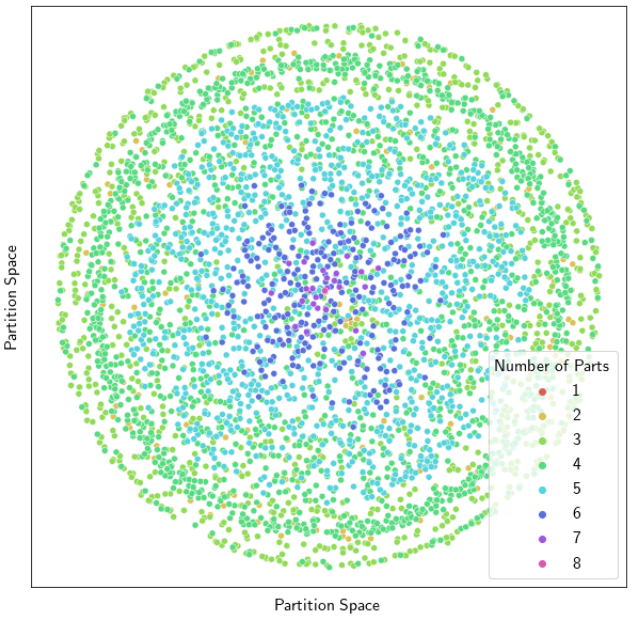

Figure 2.24: MDS embedding of all partitions of 8 elements using the split/join metric. Coloured by the number of parts in each partition.

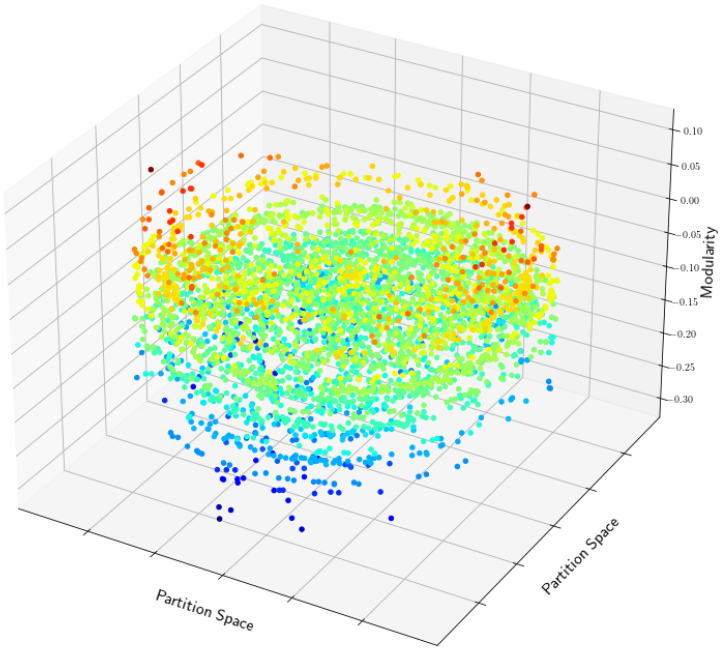

Figure 2.25: Energy landscape of the 8-vertex [2,2]-block lattice graph. The embedding from Figure 2.24 was used along with the modularity of each partition. Coloured by the modularity of each partition.

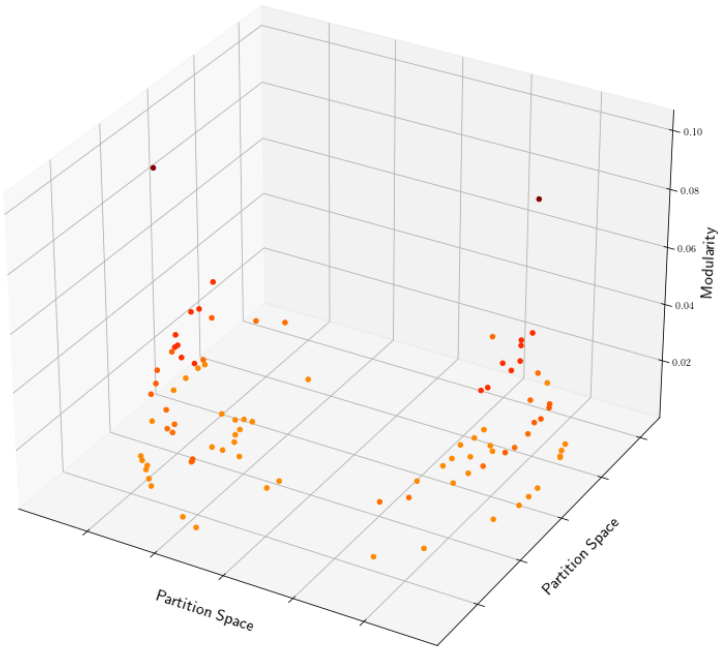

Figure 2.26: Energy landscape of the 8-vertex [2,2]-block lattice graph restricted to partitions with positive modularity. The two maxima are the two modes.

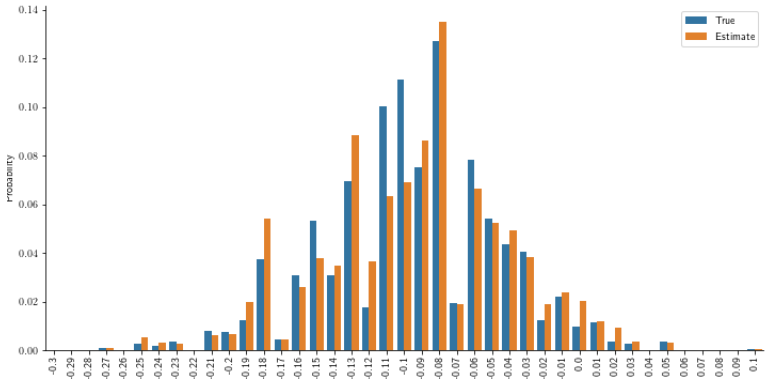

Figure 2.27: True and estimated modularity distribution for the 8-vertex [2,2]-block lattice graph. Estimate produced by a 100,000 step run of the Wang-Landau $1/t$ algorithm.

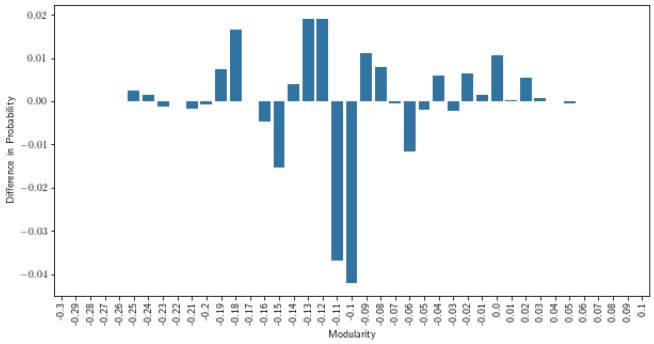

Figure 2.28: Error in the estimate from Figure 2.27 calculated as the difference between the Wang-Landau $1/t$ estimate and the true modularity distribution for the 8-vertex [2,2]-block lattice graph.

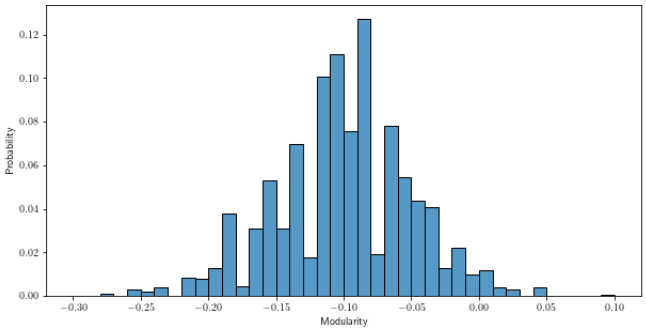

Figure 2.29: True modularity distribution for the 8-vertex [2,2]-block lattice graph.

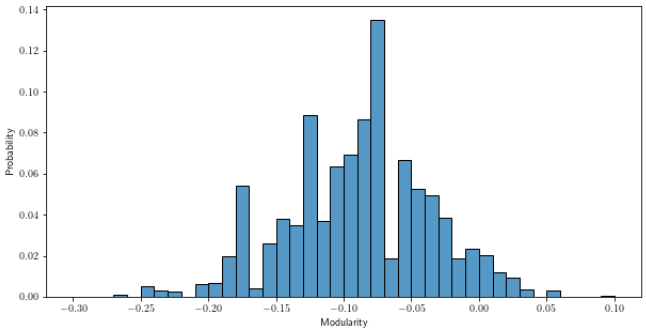

Figure 2.30: Estimated modularity distribution for the 8-vertex [2,2]-block lattice graph from a 100,000 step run of the Wang-Landau $1/t$ algorithm.

Chapter 3

Results

3.1 Algorithm Description

This section first presents the algorithm that we settled on and then discusses the choices and tradeoffs that were made. We now provide more details about how our algorithm works, with Algorithm 5 describing more specifics than were given earlier in Algorithm 1.

Our algorithm should have a computational complexity close to multiple runs of Louvain. At each step Louvain is run on the updated graph. Then the the downweight is calculated, which can be done by sampling a small number of vertices. In practice the total number of rounds can be limited to some fixed value. The final component is the spectral clustering, with the number of parts it is given to cluster bounded by the total number of rounds and Louvain's resolution limit.

Algorithm 5: Overlapping Community Detection (Detailed)

Input: graph $G = (V, E)$
If G is unweighted, assign each edge a weight of 1
Set $t = -1$, $G_0 = G$, and $R^0 =$ the trivial single-part partition
repeat
  $t = t + 1$
  Sample a partition $C^t = Louvain(G_t)$
  Find the part C^t_{best} of C^t with the greatest modularity i.e.
    $C^t_{best} = \arg\max_{C^t_i \in C^t} \{Q(C^t_i)\}$
  for $v \in C^t_{best}$ **do**
    Calculate the downweight $\alpha_{t,v}$ required to move vertex v from C^t_{best} (Equation 3.2.1)
  Calculate the required downweight α_t to make C^t_{best} unattractive to Louvain i.e.
    $\alpha_t = \min \{\{\alpha_{t,v} | v \in C^t_{best}\}\}$
  (For large graphs taking only a random sample of $v \in C^t_{best}$ may be feasible)
  $G_{t+1} = G_t$
  Downweight the edges of G_{t+1} that are internal to C^t_{best} i.e.
  for each edge $e = (u, v) \in E_{t+1}$ where $u, v \in C^t_{best}$ **do**
    Downweight edge e by $weight(e) = \alpha_t \cdot weight(e)$
  $R^{t+1} =$ the refinement of R^t and C^t
until C^t is not near the set algebra formed by R^t;
Compute the similarity matrix from the parts of $C^1, \ldots, C^{t-1}$
Identify and remove the singleton parts from this matrix
Spectral cluster the parts of partitions $C^1, \ldots, C^t$, only keeping the best representative from each cluster
return the representative and removed singleton parts from $C^1, \ldots, C^t$

3.2 Choosing the Right Downweighting Approach

The following is a discussion of the decisions that were made when designing our Algorithm 5 for sampling graph partitions. To inform which approach to use in our algorithm we ran our algorithm with the different parameters on a synthetic SBM and the real-world karate club graph. The SBM had two modes, which we call vertical and horizontal with respect to the layout we provide. The karate club graph is a very small social network and is expected to have only a single mode and no overlapping community structure. To see how the algorithms degrade after visiting all modes, no stopping criteria was considered so as to take more samples than would be recommended. Being able to identify such degradation is important in constructing a stopping criteria that only returns useful partitions.

Cold or warm start?

The usual picture for a "greedy" descent algorithm is as follows. We have a state space (possible partitions), a "height" or loss function on the state space (the modularity function), and a notion of which points are next to each other (partitions that differ by swapping one vertex's label). A greedy descent algorithm will then start at one point and basically roll downhill until it comes to a local minimum in the height function. Since a height function can have many minima, different initializations will result in different outputs. We call the set of initializations leading to a given output P the "domain of attraction" of P, since balls started in P's domain of attraction will be attracted to P.

When we discuss downweighting, we are trying to ensure that our Louvain "ball" rolls to a new local minimum P' and thus gives us a new partition. While there is a way to force this to happen, doing this well turns out to be subtle and difficult for reasons we explain here.

The simplest approach is to smash the height function completely flat in the neighbourhood of the previous best partition P, ensuring that any initialization of Louvain cannot possibly stop near P. We can do this by downweighting the edges that are internal to the old partition, decreasing the modularity of any partition that is similar to the old one. This works, but requires a very large downweight to the loss function.

If our state space were 1-dimensional, this would be the only possible strategy - you would need to smash flat all local minima to make sure they weren't traps. But in high dimensions, another approach is possible. Smaller downweights might leave P a local minimum while radically decreasing the size of its basin of attraction. This might seem counter-intuitive, but it can happen for essentially all nontrivial examples - consider e.g. gradient descent for simple functions like $(x-1)^2 + C(x+1)^2$ as C changes.

This leads us to consider the problem of "warm" or "cold" starts - do we want to start a new run at P or somewhere else? An algorithm has a **coldstart** if subsequent runs start from a random or uninformed state. In our case, Louvain would be rerun each time from the partition of singletons that Louvain normally starts from. An alternative is a **warmstart** where the algorithm resumes from a previous state or a state informed by previous runs. Each iteration of our algorithm would then look like running Louvain until it reaches a stable partition, downweighting that partition until it is no longer stable, resuming Louvain from this partition and following as it moves on to a new stable partition.

Starting from a stable partition requires more energy to move away, thus requiring a greater downweight than if starting from the initial unstable position. In fact our analytical results overestimate the amount of downweighting necessary to force mode swap. This is due to a nucleation phenomenon described in more detail in Section 3.4.2.

Hence in a warmstart when starting from the previous partition, i.e. one that is stable and likely to have the best modularity, a greater downweight will be required. This required downweight is easy to calculate as the change in modularity needed to move vertices out of their current community is given by an explicit formula.

Otherwise in a coldstart when starting from a random partition or the initialization partition of singletons, a lesser downweight is required to make alternative modes more attractive. In this case the minimal required downweighting is harder to calculate due to the ergodicity of Louvain. The problem of moving between modes is further discussed in Section 3.2.4. Thinking about our heuristics, it is clear that we should expect to be able to get away with a smaller downweight if we have a cold start - this allows us to escape from our old optimum P even if it remains a local optimum.

For the SBM with two modes, we can see in Figure 3.1 that the coldstart algorithm recovers these two modes in its first two samples.

The warmstart option of Figure 3.2 did not perform as well. After the first step the best cluster from the first sample, coloured blue, was downweighted. In the second step when $t = 1$, the vertices from this best cluster were absorbed by the two unchanged vertical communities from the first mode. At the third step, an agglomeration of the horizontal communities is found. Later steps return partitions that generally lie in the algebra formed from the parts of the refined partition. Thus the warmstart option did not identify the two modes like the coldstart method did, first getting stuck near the dominant mode and then visiting low resolution agglomerations of parts of the second mode.

For the karate club graph we expect there to be only one mode. The warmstart option of Figure 3.4 became progressively lower resolution. The first four steps were simply a back and forth agglomeration of the best mode after downweighting. The final 3 steps reach a stable partition with only a few vertices changing membership

at each step. The coldstart option of Figure 3.3 It appeared to explore more of the partition space and had a higher resolution i.e. greater number of parts for most partitions it proposed.

Based on these experiments the coldstart option was chosen. It appears better at exploring the partition space, and in particular does not get trapped near the dominant mode. Additionally it appears to produce higher resolution partitions.

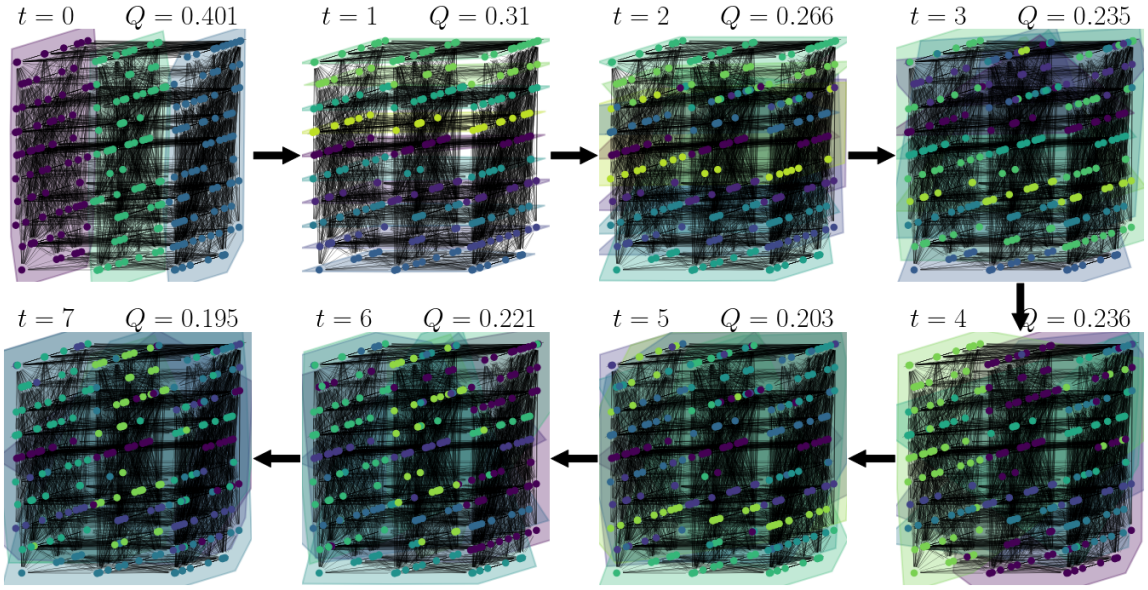

Figure 3.1: Coldstart multi-level Wang-Landau sampling on a SBM.

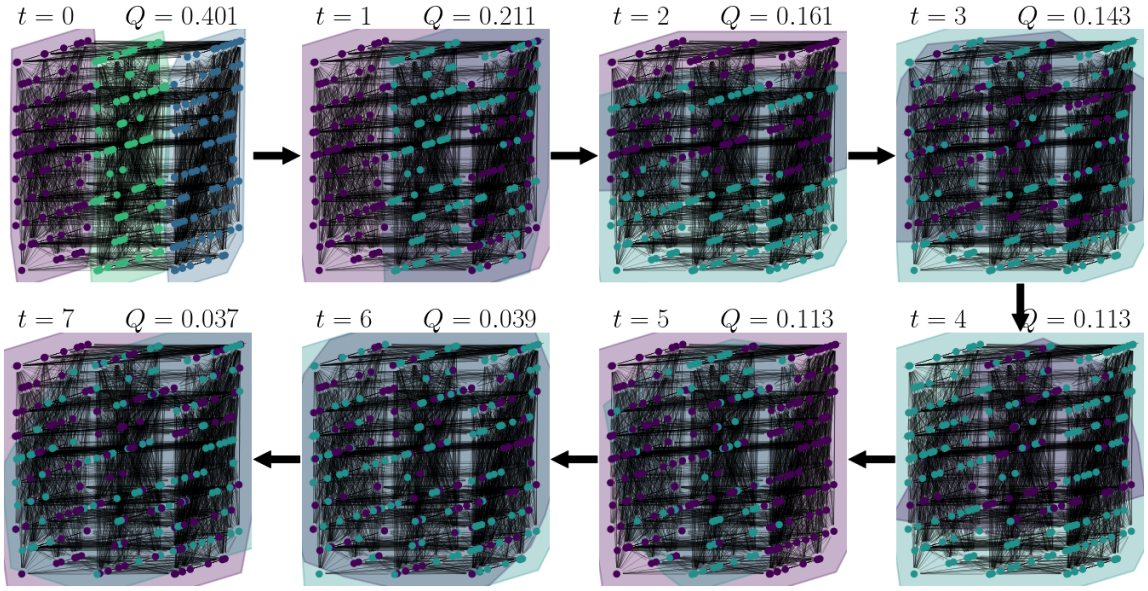

Figure 3.2: Warmstart multi-level Wang-Landau sampling on a SBM.

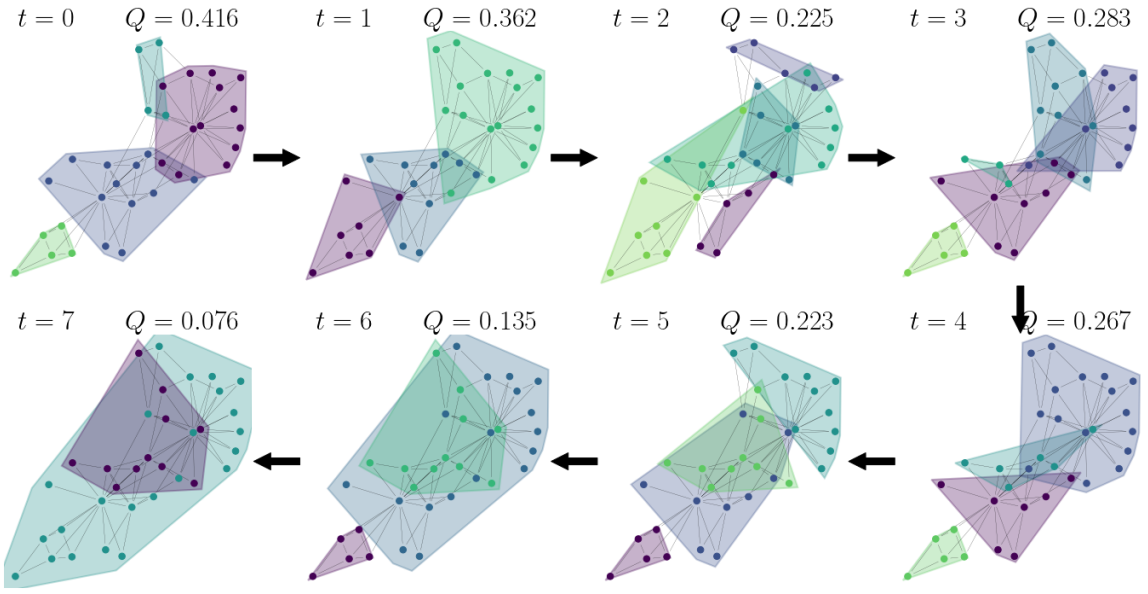

Figure 3.3: Coldstart multi-level Wang-Landau sampling on the karate-club graph.

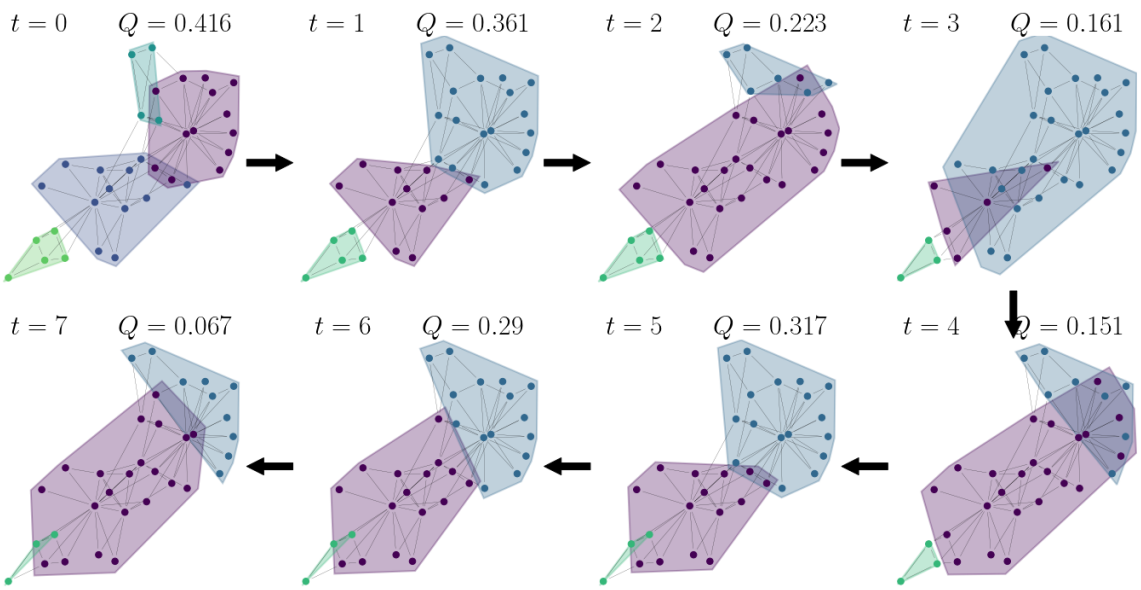

Figure 3.4: Warmstart multi-level Wang-Landau sampling on the karate-club graph.

Multi- or one-level Louvain?

The authors of ECG decide to use only the one level version of Louvain, saying "It has been shown that using weak clustering algorithms in the generation step can produce high quality results. Running a single level of the ML algorithm is a good example a weak learner, where vertices are grouped into many small clusters" [74]. Using only one level may avoid the resolution limit as the original Louvain authors note, saying "the intermediate solutions found by our algorithm may also be meaningful and that the uncovered hierarchical structure may allow the end-user to zoom in the network and to observe its structure with the desired resolution" [11].

As our algorithm does not have as explicit an integration step, using the weaker one-level option to get higher resolution partitions does not seem as beneficial. Our post-processing step does attempt some level of integration, detailed in Section 3.3. Downweighting itself appears to solve some of this problem. If two communities are agglomerated and the internal edges of this agglomeration downweighted, at the next step it is likely they would be either agglomerated with different communities or end up alone. If again agglomerated, considering the refinement (i.e. disjoint intersections) of the two partitions reveals the individual communities. For example in Figure 3.5 the individual communities are mostly recovered by the third step. Post-processing should resolve the remaining agglomerations into the higher resolution refined parts. In the example the three 5-cliques that were agglomerated in the third step appeared in two unique agglomerations in the preceding two steps.

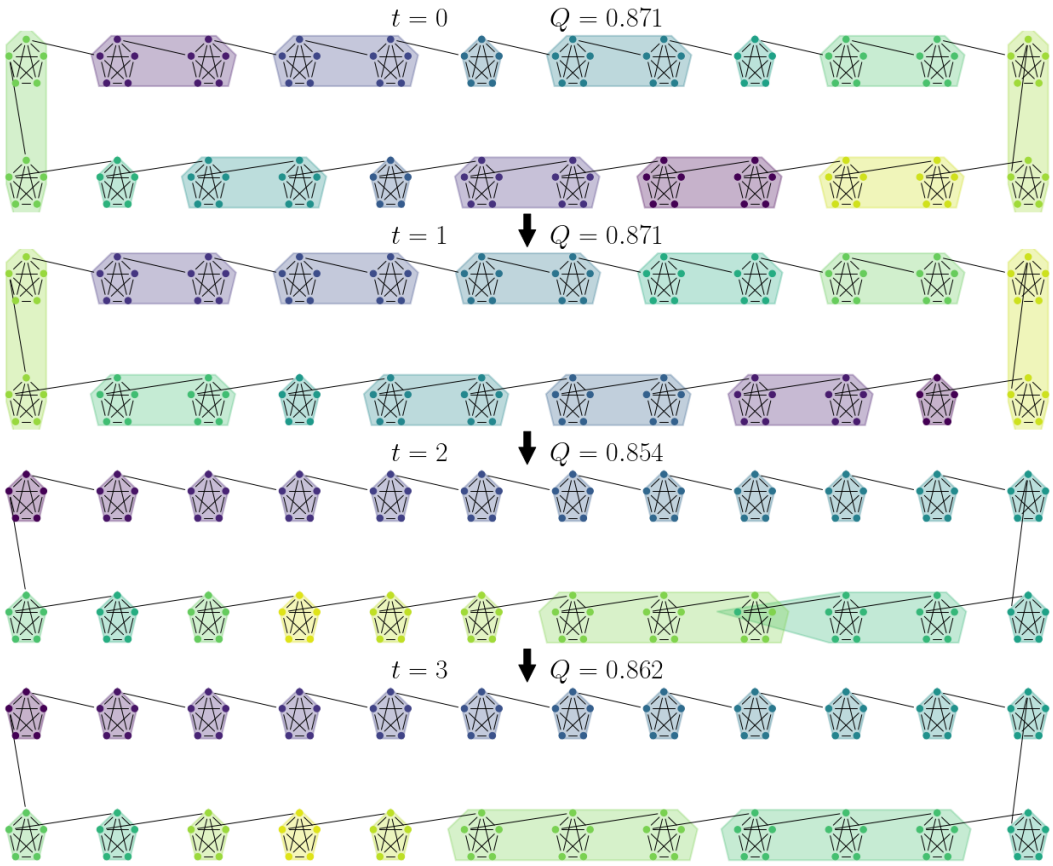

Figure 3.5: Coldstart multi-level Wang-Landau sampling on the ring of 24 cliques with clique size of 5. The first two steps suffer from the resolution limit where small communities are undesirably agglomerated. This is largely resolved by the third step.

With respect to the SBM, the warmstart multi-level (Figure 3.2) and one-level (Figure 3.7) produced the identical sequence of partitions. This is likely because the community structure was strong enough in the SBM to be found at the first level of Louvain. As the first two samples corresponded with the two modes of the SBM, this was the best possible result.

For the karate-club graph, the initial sample of the multi-level option (Figure 3.3) does not have the very small communities found in its one-level counterpart (Figure 3.8). Thus the multi-level option was deemed preferable.

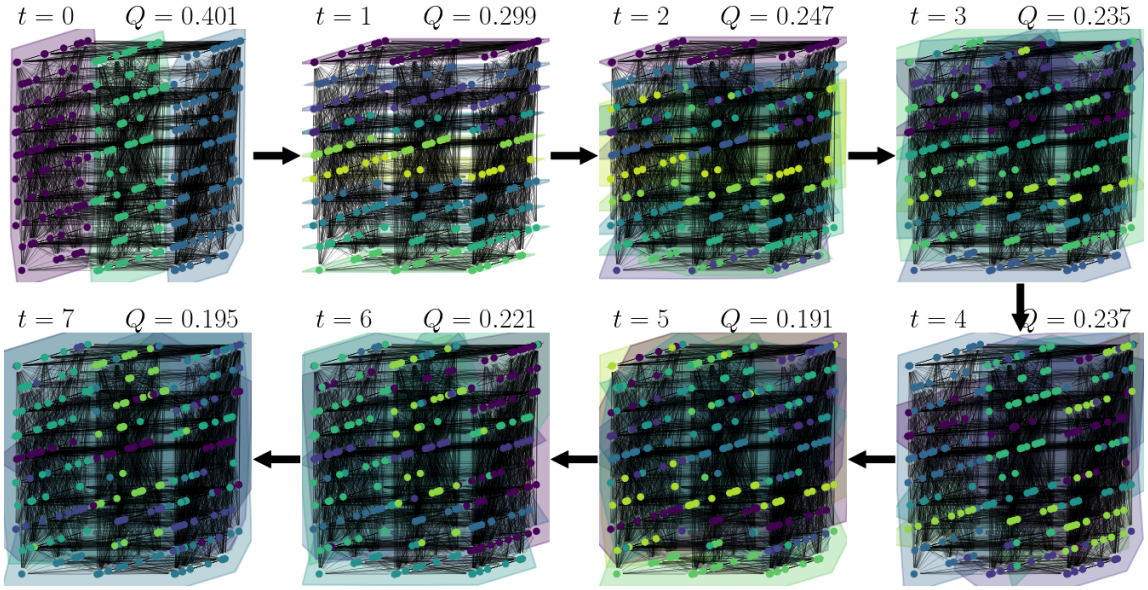

Figure 3.6: Coldstart one-level Wang-Landau sampling on a SBM.

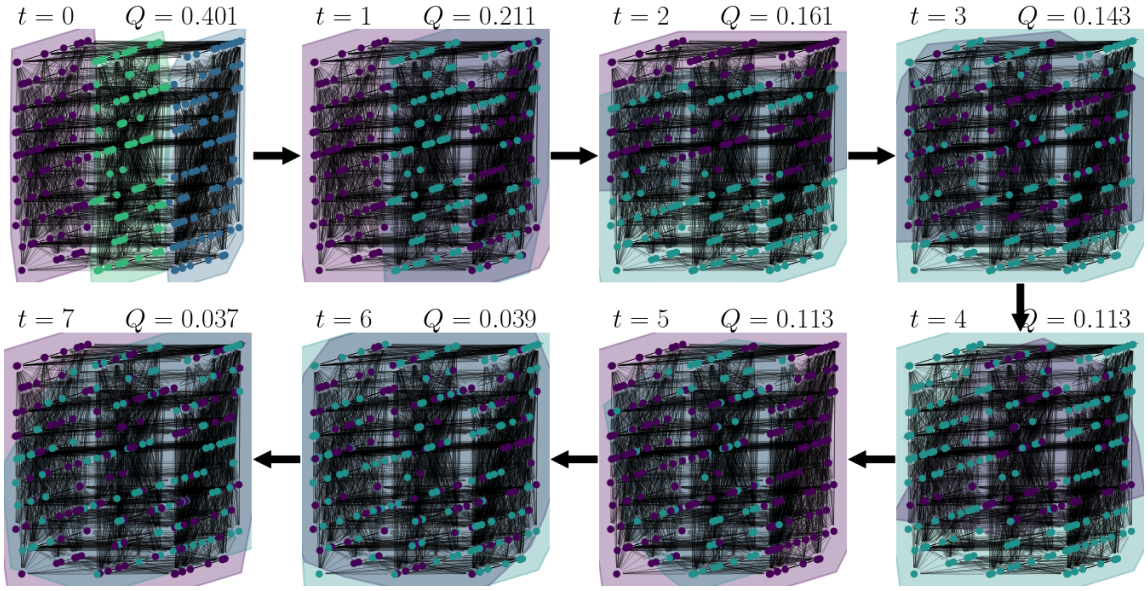

Figure 3.7: Warmstart one-level Wang-Landau sampling on a SBM.

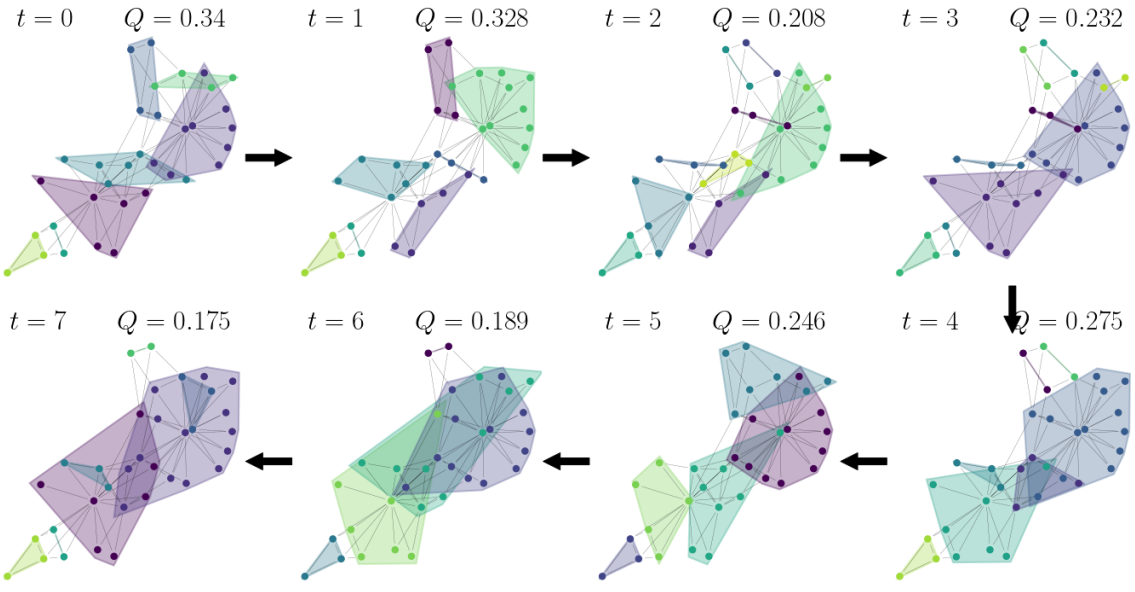

Figure 3.8: Coldstart one-level Wang-Landau sampling on the karate-club graph.

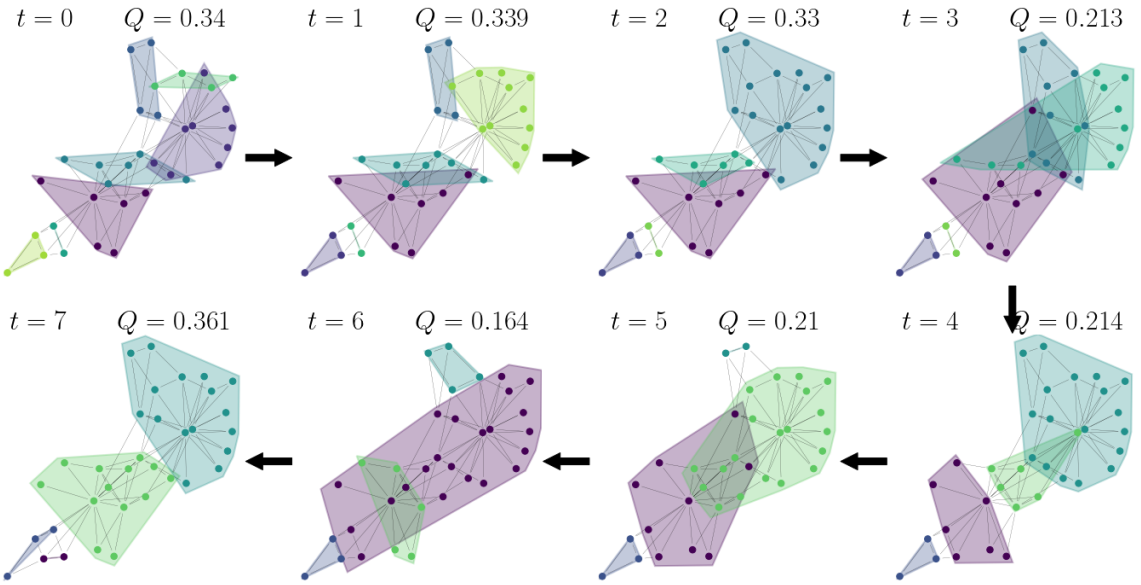

Figure 3.9: Warmstart one-level Wang-Landau sampling on the karate-club graph.

Downweight only one cluster, everything, some weighted mixture?

How far should you go between the smallest downweight that causes a vertex to switch and obliterating the cluster?

We take the approach of downweighting only the best community (i.e. the one with the greatest contribution to the overall modularity) at each iteration. This approach felt more surgical and accommodating of a heterogeneous energy landscape with varied edge densities. While it is conceivable that it may take longer to force a mode swap as each part would need to be downweighted individually, for the example SBM of Figure 3.1 it was enough to force a mode swap in the second step. This may be because eliminating a single part is enough to make the alternative mode more attractive.

Experimental or analytical approach?

Without any knowledge about how edge weights affect the partitions Louvain produces, the natural way of downweighting would be to make many small downweights until a change is produced. However this has the downside of being slow. Instead the downweight that breaks up the best community is used, and is analytically determined.

An alternative would be to use the gradient of the modularity in the energy

landscape to determine the downweight. Around each local modularity maxima there exists a basin of attraction, i.e. a neighbourhood of partitions upon which Louvain is drawn to this local maxima. The size of this basin and the gradient of the modularity around the maxima could be estimated. These values could then be fed into a downweight calculation. With this the internal edges within the maximizing partition could downweighted so that this maxima and basin no longer exist, forcing partitions in the basin to be attracted to other maxima. This would be faster than making incremental, uninformed downweights until a change is produced. However calculating the size of the basis and gradient is non trivial and would be more computationally intensive than our downweight.

3.2.1 Analytic Warmstart Downweight

In this section we determine the downweight required to make Louvain take at least one step away from its current stable partition, i.e. move v to another community by switching its label. We call such a downweight and resumption of Louvain a warmstart for our algorithm. It will take exactly one step away if v had the weakest attraction to its current community out of all vertices. (Recall that Σ_{AB} denotes the sum of edges between the set A of edges and B. Refer to Table 2.1 for a detailed explanation of the notation.) Recall Equation 2.3.4 that says the change in modularity for moving a vertex v into community C_j is given by

$$\Delta Q(v \to C_j) = \frac{\Sigma_{C_j}}{2\Sigma_{VV}} - \frac{\Sigma_{VC_j}\Sigma_{Vv}}{2\Sigma_{VV}^2}$$

The change when v is removed from community C_i is the negative of the change if it were added, i.e. $\Delta Q(C_i \to v) = -\Delta Q(v \to C_i)$. With both of these formulas, the effect of moving v from C_i to C_j is $\Delta Q(v \to C_j) - \Delta Q(v \to C_i)$.

Suppose we want to find the largest $\alpha \in [0, 1]$ such that downweighting the internal edge weights of C_i by α, i.e. taking αw_e to be the new edge weight for each edge e internal to C_i, causes vertex v to change communities.

After downweighting the internal edges of C_i by α the components of the modularity formula become

$$\begin{aligned}
\Sigma'_{VV} &= \sum_{e\in E} w'_e = \sum_{e\in C_i} \alpha w_e + \sum_{e\in E\setminus C_i} w_e = \alpha\Sigma_{C_iC_i} + \left(\Sigma_{VV} - \Sigma_{C_iC_i}\right), \\
\Sigma'_{VC_i} &= \alpha\Sigma_{C_iC_i} + \left(\Sigma_{VC_i} - \Sigma_{C_iC_i}\right), \\
\Sigma'_{C_iv} &= \alpha\Sigma_{C_iv}, \\
\Sigma'_{Vv} &= \alpha\Sigma_{C_iv} + \left(\Sigma_{Vv} - \Sigma_{C_iv}\right)
\end{aligned}$$

With the internal edges of C_i downweighted by α, the change $\Delta Q(C_i \to v)$ for

removing v from C_i is then

$$\begin{aligned}\Delta Q(C_i \to v) &= -\Delta Q(v \to C_i) \\ &= -\frac{\Sigma'_{C_iv}}{2\Sigma'_{VV}} + \frac{\Sigma'_{VC_i}\Sigma'_{Vv}}{2\Sigma'^2_{VV}} \\ &= -\frac{\alpha\Sigma_{C_iv}}{2(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i}))} \\ &\quad + \frac{(\alpha\Sigma_{C_iC_i} + (\Sigma_{VC_i} - \Sigma_{C_iC_i}))(\alpha\Sigma_{C_iv} + (\Sigma_{Vv} - \Sigma_{C_iv}))}{2(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i}))^2}\end{aligned}$$

With respect to community C_j, the only terms that are changed by downweighting C_i are the total edge weight sum Σ_{VV} and the degree Σ_{Vv} of v as these are the ones that involve internal edges of C_i. The change $\Delta Q(v \to C_j)$ for adding v to C_j is then

$$\begin{aligned}\Delta Q(v \to C_j) &= \frac{\Sigma_{C_jv}}{2\Sigma'_{VV}} - \frac{\Sigma_{VC_j}\Sigma'_{Vv}}{2\Sigma'^2_{VV}} \\ &= \frac{\Sigma_{C_jv}}{2\left(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i})\right)} - \frac{\Sigma_{VC_j}\left(\alpha\Sigma_{C_iv} + (\Sigma_{Vv} - \Sigma_{C_iv})\right)}{2\left(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i})\right)^2}\end{aligned}$$

We can then calculate the value of α at which there is a positive change in modularity for moving vertex v, and hence v would be moved by Louvain.

$$\begin{aligned}0 &= \Delta Q(v \to C_j) - \Delta Q(v \to C_i) \\ &= \left[\frac{\Sigma_{C_jv}}{2\left(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i})\right)} - \frac{\Sigma_{VC_j}\left(\alpha\Sigma_{C_iv} + (\Sigma_{Vv} - \Sigma_{C_iv})\right)}{2\left(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i})\right)^2}\right] \\ &\quad + \left[-\frac{\alpha\Sigma_{C_iv}}{2(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i}))} + \frac{(\alpha\Sigma_{C_iC_i} + (\Sigma_{VC_i} - \Sigma_{C_iC_i}))(\alpha\Sigma_{C_iv} + (\Sigma_{Vv} - \Sigma_{C_iv}))}{2(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i}))^2}\right] \\ &= \Sigma_{C_jv}\left(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i})\right) - \Sigma_{VC_j}\left(\alpha\Sigma_{C_iv} + (\Sigma_{Vv} - \Sigma_{C_iv})\right) \\ &\quad - \alpha\Sigma_{C_iv}\left(\alpha\Sigma_{C_iC_i} + (\Sigma_{VV} - \Sigma_{C_iC_i})\right) + (\alpha\Sigma_{C_iC_i} + (\Sigma_{VC_i} - \Sigma_{C_iC_i}))(\alpha\Sigma_{C_iv} + (\Sigma_{Vv} - \Sigma_{C_iv})) \\ &= \alpha\Sigma_{C_iC_i}\Sigma_{C_jv} + \Sigma_{VV}\Sigma_{C_jv} - \Sigma_{C_iC_i}\Sigma_{C_jv} - \alpha\Sigma_{C_iv}\Sigma_{VC_j} + \Sigma_{Vv}\Sigma_{VC_j} - \Sigma_{C_iv}\Sigma_{VC_j} \\ &\quad - \alpha^2\Sigma_{C_iC_i}\Sigma_{C_iv} + \alpha\Sigma_{VV}\Sigma_{C_iv} - \alpha\Sigma_{C_iC_i}\Sigma_{C_iv} + \alpha^2\Sigma_{C_iC_i}\Sigma_{C_iv} + \alpha\Sigma_{C_iC_i}\Sigma_{Vv} - \alpha\Sigma_{C_iC_i}\Sigma_{C_iv} \\ &\quad + \alpha\Sigma_{VC_i}\Sigma_{C_iv} + \Sigma_{VC_i}\Sigma_{Vv} - \Sigma_{VC_i}\Sigma_{C_iv} - \alpha\Sigma_{C_iC_i}\Sigma_{C_iv} - \Sigma_{C_iC_i}\Sigma_{Vv} + \Sigma_{C_iC_i}\Sigma_{C_iv} \\ &= \alpha^2\left(\Sigma_{C_iC_i}\Sigma_{C_iv} - \Sigma_{C_iC_i}\Sigma_{C_iv}\right) + \alpha\left(\Sigma_{C_iC_i}\Sigma_{C_jv} - \Sigma_{C_iv}\Sigma_{VC_j} + \Sigma_{VV}\Sigma_{C_iv} - \Sigma_{C_iC_i}\Sigma_{C_iv}\right. \\ &\quad \left.+\Sigma_{C_iC_i}\Sigma_{Vv} - \Sigma_{C_iC_i}\Sigma_{C_iv} + \Sigma_{VC_i}\Sigma_{C_iv} - \Sigma_{C_iC_i}\Sigma_{C_iv}\right) + \left(\Sigma_{VV}\Sigma_{C_jv} - \Sigma_{C_iC_i}\Sigma_{C_jv} + \Sigma_{Vv}\Sigma_{VC_j}\right. \\ &\quad \left.-\Sigma_{C_iv}\Sigma_{VC_j} + \Sigma_{VC_i}\Sigma_{Vv} - \Sigma_{VC_i}\Sigma_{C_iv} - \Sigma_{C_iC_i}\Sigma_{Vv} + \Sigma_{C_iC_i}\Sigma_{C_iv}\right) \\ &= \alpha\left(\Sigma_{C_iC_i}\Sigma_{C_jv} - \Sigma_{C_iv}\Sigma_{VC_j} + \Sigma_{VV}\Sigma_{C_iv} + \Sigma_{C_iC_i}\Sigma_{Vv} + \Sigma_{VC_i}\Sigma_{C_iv} - 3\Sigma_{C_iC_i}\Sigma_{C_iv}\right) \\ &\quad + \left(\Sigma_{VV}\Sigma_{C_jv} - \Sigma_{C_iC_i}\Sigma_{C_jv} + \Sigma_{Vv}\Sigma_{VC_j} - \Sigma_{C_iv}\Sigma_{VC_j} + \Sigma_{VC_i}\Sigma_{Vv} - \Sigma_{VC_i}\Sigma_{C_iv} - \Sigma_{C_iC_i}\Sigma_{Vv}\right. \\ &\quad \left.+\Sigma_{C_iC_i}\Sigma_{C_iv}\right)\end{aligned} \tag{3.2.1}$$

The quadratic formula then gives the desired downweight α (if one exists).

3.2.2 Analytic Coldstart Downweight

To develop some understanding of how downweighting affects modularity we will consider a situation where we know all modes a priori and need to determine the required downweight to move between them. Note that in practice we would not have this information and so must rely on the analytic warmstart downweight. Let $G = (E, V)$ be a weighted undirected graph and let C_D and C_I be two partitions (i.e. modes) of the vertices of G such that C_D dominates the inferior mode C_I (i.e. C_D has higher modularity than C_I). Suppose we want to find a downweight $\alpha \in [0, 1]$ such that when the internal edges of C_D are downweighted by α the modularity of C_D is equal to that of C_I (both modularities are computed with respect to this new downweighted graph). Note that the greatest downweight corresponds to the smallest change in the edge weights of G. This section will give a formula for the required downweight α. In practice if Louvain first returned C_D we would not know about C_I a priori, and so would need to make incremental trial downweights until Louvain arrived at this inferior mode. Nevertheless knowing for what downweight Louvain has a phase transition between the modes it proposes is important for informing what the incremental downweights should be, and for our understanding of the behaviour of Louvain.

In order to find such a downweight it is easier to consider the matrix formulation of modularity as given in Equation 2 of [60]. We will switch to the original notation that, while less clear, is more compact. The **modularity matrix** is defined as $B = A - \frac{\mathbf{k}\mathbf{k}^T}{2m}$ where $\mathbf{k}$ is the $n \times 1$ degree vector for the vertices in G, i.e. $\mathbf{k}_i$ is the degree of vertex i. For our weight graph, degree is the sum of the edge weights adjacent to the vertex. The degree vector can be rewritten as $\mathbf{k} = A\mathbf{1}$ and $\mathbf{k}_i = \deg(i) = \sum_j A_{ij} = e_i A\mathbf{1}$ where $\mathbf{1}$ is the $n \times 1$ ones vector and e_i is the i-th standard basis vector. Thus $\mathbf{k}\mathbf{k}^T = A\mathbf{1}[A\mathbf{1}]^T = A\mathbf{1}\mathbf{1}^T A^T = A\mathbf{1}\mathbf{1}^T A$ since A is symmetric.

Let C be a partition of V into r communities. Let S be the $n \times r$ matrix where S_{vi} is 1 if v is in community i and 0 otherwise. Thus the columns of S are the membership vectors for each of the respective communities. Note each row has exactly one 1 as each vertex can be assigned to only one community.

Then the matrix definition of the modularity of C is given by

$$Q(C) = \frac{1}{4m}\mathrm{Tr}(S^T B S) \tag{3.2.2}$$

In order to downweight edges internal to the communities of C, we need to decompose A into its internal and inter-community edges. Here we will be downweighting all communities of C, however the following results generalize to downweighting e.g. only the community with the best community. Let $\tilde{A}$ be the matrix of edges internal to communities of C, i.e.

$$\tilde{A}_{ij} = \begin{cases} A_{ij} & (i,j) \text{ is an internal edge of a community in } C \\ 0 & \text{otherwise} \end{cases}$$

Equivalently, $\tilde{A}_i$ can be viewed as A_i "masked" by the membership vector of the community that vertex i belongs to. We similarly set $\bar{\tilde{A}}$ to be the "complement" of $\tilde{A}$, i.e. the matrix of non-internal edges. Hence

$$\bar{\tilde{A}}_{ij} = \begin{cases} 0 & \tilde{A}_{ij} \neq 0 \\ A_{ij} & \text{otherwise} \end{cases}$$

Thus $A = \tilde{A} + \bar{\tilde{A}}$. Then downweighting the internal edges of C by α produces the matrix $\alpha\tilde{A} + \bar{\tilde{A}}$. Note that this downweighting and the construction of $\tilde{A}$ and $\bar{\tilde{A}}$ are with respect to a specific partition.

The sum of all edge weights in the graph (counted twice) is given by

$$2m = \sum_i \mathbf{k}_i = \sum_{ij} A_{ij} = \mathbf{1}^T A \mathbf{1}$$

After downweighting this becomes

$$2m = \mathbf{1}^T \left[\alpha\tilde{A} + \bar{\tilde{A}}\right] \mathbf{1} = \alpha \mathbf{1}^T \tilde{A} \mathbf{1} + \mathbf{1}^T \bar{\tilde{A}} \mathbf{1}$$

The normalized degree vector $\frac{\mathbf{k}_i}{m}$ gives the proportion of all edges that have i as an endpoint. Under our generative process, we "split" these edges into two and assume vertex i is responsible for the creation of half of these, giving us $\frac{\mathbf{k}_i}{2m}$. Multiplying by k_j gives the expected weight of edges between vertices i and j that were generated by i.

After downweighting the $n \times n$ matrix $\mathbf{k}\mathbf{k}^T$ becomes

$$\begin{aligned}
\mathbf{k}\mathbf{k}^T &= A\mathbf{1}\mathbf{1}^T A \\
&= \left[\alpha\tilde{A} + \bar{\tilde{A}}\right] \mathbf{1}\mathbf{1}^T \left[\alpha\tilde{A} + \bar{\tilde{A}}\right] \\
&= \left[\alpha\tilde{A}\mathbf{1}\mathbf{1}^T + \bar{\tilde{A}}\mathbf{1}\mathbf{1}^T\right] \left[\alpha\tilde{A} + \bar{\tilde{A}}\right] \\
&= \alpha\tilde{A}\mathbf{1}\mathbf{1}^T\alpha\tilde{A} + \bar{\tilde{A}}\mathbf{1}\mathbf{1}^T\alpha\tilde{A} + \alpha\tilde{A}\mathbf{1}\mathbf{1}^T\bar{\tilde{A}} + \bar{\tilde{A}}\mathbf{1}\mathbf{1}^T\bar{\tilde{A}} \\
&= \alpha^2\tilde{A}\mathbf{1}\mathbf{1}^T\tilde{A} + \alpha\left[\bar{\tilde{A}}\mathbf{1}\mathbf{1}^T\tilde{A} + \tilde{A}\mathbf{1}\mathbf{1}^T\bar{\tilde{A}}\right] + \bar{\tilde{A}}\mathbf{1}\mathbf{1}^T\bar{\tilde{A}}
\end{aligned}$$

We develop an equivalent statement for when the modularities of the two partitions are equal.

$$\begin{aligned}
& Q(I) = Q(D) \\
\Leftrightarrow \quad & \frac{1}{4m}\mathrm{Tr}(S_I^T B S_I) = \frac{1}{4m}\mathrm{Tr}(S_D^T B S_D) \\
\Leftrightarrow \quad & \mathrm{Tr}(S_I^T B S_I) = \mathrm{Tr}(S_D^T B S_D)
\end{aligned}$$

$$\Leftrightarrow \qquad \mathrm{Tr}(BS_IS_I^T) = \mathrm{Tr}(BS_DS_D^T)$$

by trace's invariance under cyclic permutation

$$\begin{aligned}\Leftrightarrow \qquad 0 &= \mathrm{Tr}(BS_DS_D^T) - \mathrm{Tr}(BS_IS_I^T)\\ &= \mathrm{Tr}(BS_DS_D^T - BS_IS_I^T)\\ &= \mathrm{Tr}(B\left[S_DS_D^T - S_IS_I^T\right])\end{aligned}$$

setting $\Delta := \left[S_DS_D^T - S_IS_I^T\right]$

$$= \mathrm{Tr}(B\Delta)$$

The product S_C^TA selects the vertices that are in each community. Here the i-th column of $S_CS_C^T$ gives the membership vector for the community that vertex i belongs to, and is repeated for each vertex. Thus the product $A\left[S_CS_C^T\right]$ selects the neighbours of vertex i that belong to the same community and sums their edge weights. The columns of $\Delta \in \{0, \pm 1\}^{n\times n}$ capture the difference in neighbours between the modes C_D and C_I.

$$\Delta_{ij} := \left[S_DS_D^T - S_IS_I^T\right]_{ij} = \begin{cases} 0 & i,j \text{ are not neighbours in a common community in either mode}\\ 0 & i,j \text{ are neighbours in a common community in both modes}\\ 1 & i,j \text{ are neighbours in a common community in } D \text{ but not } I\\ -1 & i,j \text{ are neighbours in a common community in } I \text{ but not } D\end{cases}$$

We can now isolate for α

$$\begin{aligned}0 &= \mathrm{Tr}(B\Delta)\\ &= \mathrm{Tr}(\left[\alpha\tilde{A} + \bar{\tilde{A}} - \frac{\mathbf{k}\mathbf{k}^T}{2m}\right]\Delta)\\ &= 2m\mathrm{Tr}(\left[\alpha\tilde{A} + \bar{\tilde{A}} - \frac{\mathbf{k}\mathbf{k}^T}{2m}\right]\Delta)\end{aligned}$$

by the linearity of trace

$$\begin{aligned}&= \mathrm{Tr}(\left[2m\left[\alpha\tilde{A} + \bar{\tilde{A}}\right] - \mathbf{k}\mathbf{k}^T\right]\Delta)\\ &= \mathrm{Tr}(\left[\left[\alpha\mathbf{1}^T\tilde{A}\mathbf{1} + \mathbf{1}^T\bar{\tilde{A}}\mathbf{1}\right]\left[\alpha\tilde{A} + \bar{\tilde{A}}\right] - \mathbf{k}\mathbf{k}^T\right]\Delta)\\ &= \mathrm{Tr}(\left[\alpha\mathbf{1}^T\tilde{A}\mathbf{1}\alpha\tilde{A} + \mathbf{1}^T\bar{\tilde{A}}\mathbf{1}\alpha\tilde{A} + \alpha\mathbf{1}^T\tilde{A}\mathbf{1}\bar{\tilde{A}} + \mathbf{1}^T\bar{\tilde{A}}\mathbf{1}\bar{\tilde{A}} - \mathbf{k}\mathbf{k}^T\right]\Delta)\\ &= \mathrm{Tr}(\left[\alpha^2\mathbf{1}^T\tilde{A}\mathbf{1}\tilde{A} + \alpha\left[\mathbf{1}^T\bar{\tilde{A}}\mathbf{1}\tilde{A} + \mathbf{1}^T\tilde{A}\mathbf{1}\bar{\tilde{A}}\right] + \mathbf{1}^T\bar{\tilde{A}}\mathbf{1}\bar{\tilde{A}} - \mathbf{k}\mathbf{k}^T\right]\Delta)\end{aligned}$$

$$
\begin{aligned}
&= \mathrm{Tr}\left(\left[\alpha^2 \mathbf{1}^T\tilde{A}\mathbf{1}\tilde{A} + \alpha\left[\mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\tilde{A} + \mathbf{1}^T\tilde{A}\mathbf{1}\tilde{\bar{A}}\right] + \mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\tilde{\bar{A}}\right.\right.\\
&\qquad \left.\left. -\alpha^2\tilde{A}\mathbf{1}\mathbf{1}^T\tilde{A} - \alpha\left[\tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{A} + \tilde{A}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\right] - \tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\right]\Delta\right)\\
&= \mathrm{Tr}\left(\left[\alpha^2\left[\mathbf{1}^T\tilde{A}\mathbf{1}\tilde{A} - \tilde{A}\mathbf{1}\mathbf{1}^T\tilde{A}\right]\right.\right.\\
&\qquad + \alpha\left[\mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\tilde{A} + \mathbf{1}^T\tilde{A}\mathbf{1}\tilde{\bar{A}} - \tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{A} - \tilde{A}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\right]\\
&\qquad \left.\left. + \left[\mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\tilde{\bar{A}} - \tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\right]\right]\Delta\right)\\
&= \mathrm{Tr}\left(\alpha^2\left[\mathbf{1}^T\tilde{A}\mathbf{1}\tilde{A}\Delta - \tilde{A}\mathbf{1}\mathbf{1}^T\tilde{A}\Delta\right]\right.\\
&\qquad + \alpha\left[\mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\tilde{A}\Delta + \mathbf{1}^T\tilde{A}\mathbf{1}\tilde{\bar{A}}\Delta - \tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{A}\Delta - \tilde{A}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\Delta\right]\\
&\qquad \left. + \left[\mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\tilde{\bar{A}}\Delta - \tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\Delta\right]\right)\\
&= \alpha^2\left[\mathbf{1}^T\tilde{A}\mathbf{1}\mathrm{Tr}(\tilde{A}\Delta) - \mathrm{Tr}(\tilde{A}\mathbf{1}\mathbf{1}^T\tilde{A}\Delta)\right]\\
&\qquad + \alpha\left[\mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\mathrm{Tr}(\tilde{A}\Delta) + \mathbf{1}^T\tilde{A}\mathbf{1}\mathrm{Tr}(\tilde{\bar{A}}\Delta) - \mathrm{Tr}(\tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{A}\Delta) - \mathrm{Tr}(\tilde{A}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\Delta)\right]\\
&\qquad + \left[\mathbf{1}^T\tilde{\bar{A}}\mathbf{1}\mathrm{Tr}(\tilde{\bar{A}}\Delta) - \mathrm{Tr}(\tilde{\bar{A}}\mathbf{1}\mathbf{1}^T\tilde{\bar{A}}\Delta)\right]
\end{aligned}
$$

The quadratic formula then gives the desired downweight α (if one exists).

3.2.3 Monte Carlo Downweight

Figure 3.10 plots the modularity of the dominant and inferior modes as the best community of the dominant mode is downweighted. The two analytic downweights developed above, the analytic coldstart downweight from Section 3.2.2 and analytic warmstart downweight from Section 3.2.1, were added to the plot as vertical lines. The min, median, and max values of the analytic warmstart downweight were plotted.

To see how closely the theory matches the reality of Louvain's behaviour, Monte Carlo experiments were conducted with varying downweights counting the number of times Louvain returned each mode. A partition was marked as belonging to "neither" mode if it did not perfectly agree with either the dominant or inferior mode. A 250 vertex [3, 10]-block SBM with edge formation probabilities [0.3, 0.6] and random assignment of vertices amongst the blocks was used for these experiments.

Figures 3.11 and 3.12 present two separate Monte Carlo experiments after downweighting the best community of the dominant mode, the former considering downweights in the interval [0, 0.05] and the later downweights in [0, 1]. Figure 3.13 presents a Monte Carlo experiment where all the communities of the dominant mode were downweighted. Of interest is a mode swap visible in Figure 3.12 that occurs before the analytic coldstart downweight, which is when we might have expected it to occur. This is likely explained by an emergent "nucleation" behaviour in Louvain,

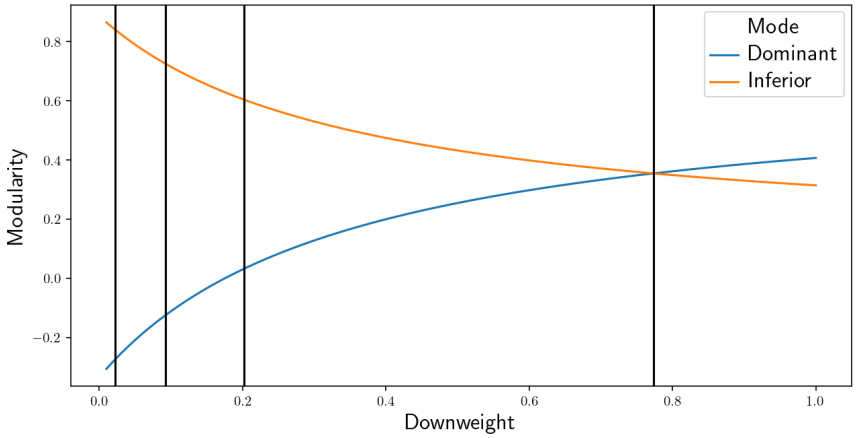

Figure 3.10: Modularity of the dominant and inferior partition as the best community of the dominant mode is downweighted. The vertical lines at the left of the plot are the min, median, and max downweights needed to move individual vertices out of the best community, with values [0.02, 0.09, 0.2]. The vertical line at the right is the analytic coldstart downweight of 0.77 to make the two modes have the same modularity.

explained further in Section 3.4.2. From these plots we can conclude that the chosen downweight strategy will with high probability produce a mode swap, which is what we desire when trying to improve the efficiency of partition sampling.

Figure 3.14 shows the each mode's empirical probability under warmstart Louvain, where the algorithm is initialized at the dominant mode. The dominant mode is stable under Louvain until just before the downweight that would move a single vertex from the best community of the dominant mode. At this point it abruptly begins returning neither mode instead of the dominant one.

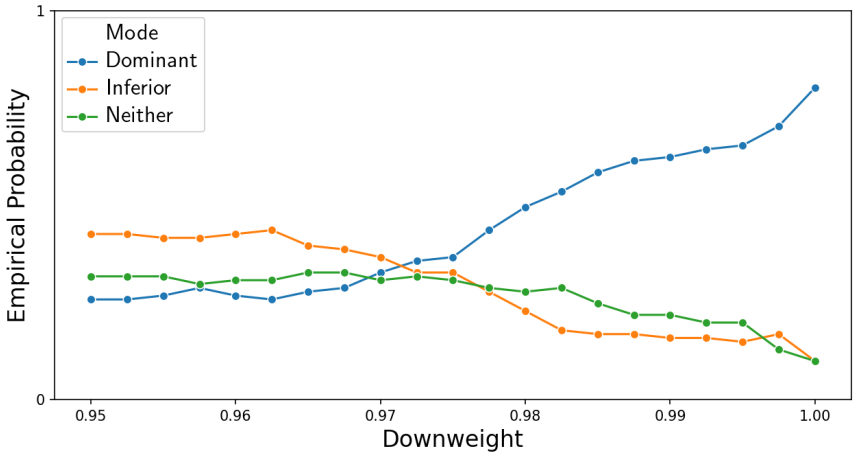

Figure 3.11: Empirical probability of each mode over 101 replicates of cold-start Louvain downweighting best community of dominant mode.

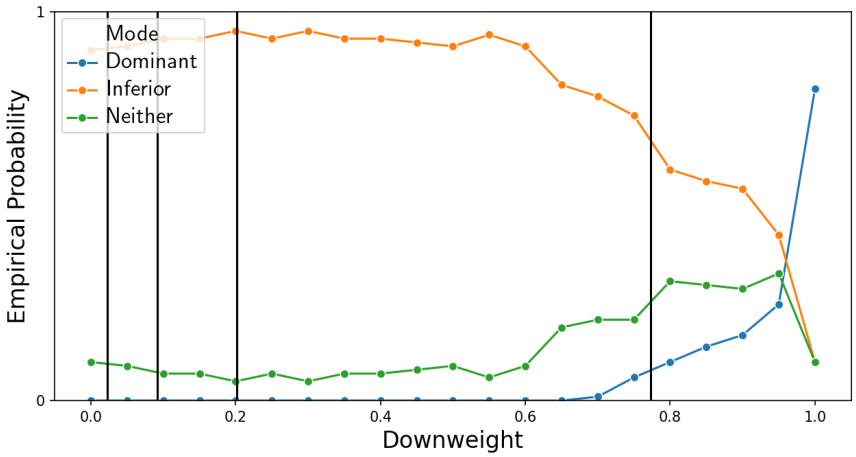

Figure 3.12: Empirical probability of each mode over 101 replicates of cold-start Louvain downweighting best community of dominant mode. The vertical lines at the left of the plot are the min, median, and max downweights with values [0.02, 0.09, 0.2] needed to move individual vertices out of the best community. The vertical line at the right is the analytic coldstart downweight of 0.77 to make the two modes have the same modularity.

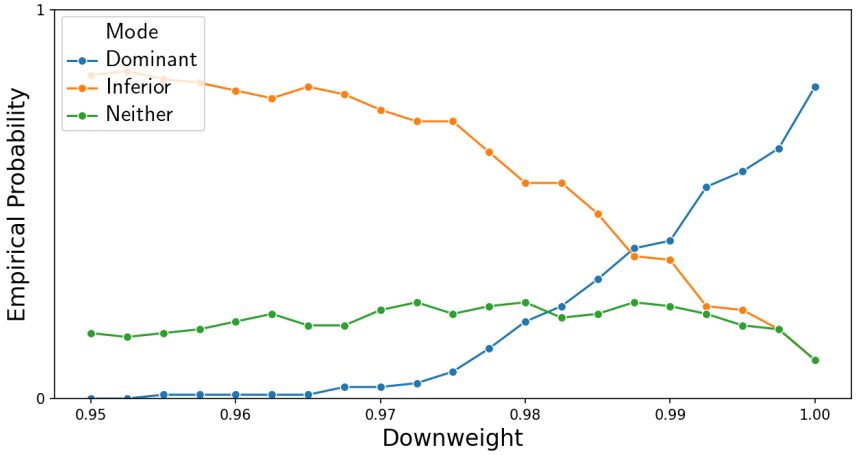

Figure 3.13: Empirical probability of each mode over 101 replicates of cold-start Louvain downweighting all communities of dominant mode.

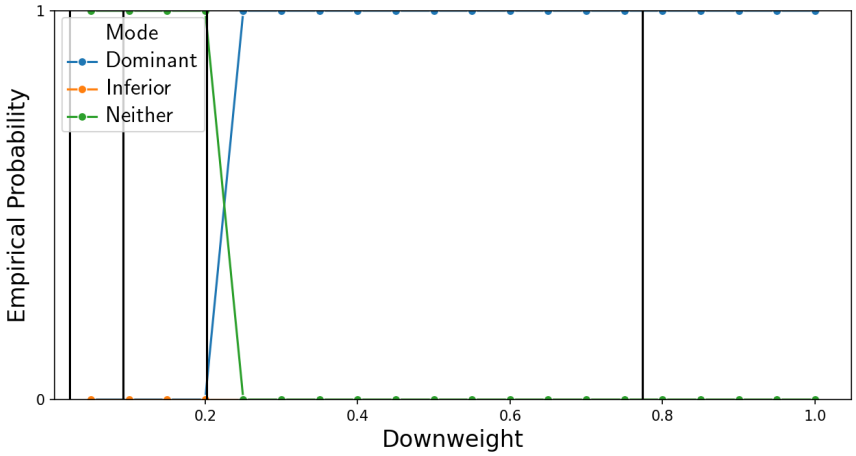

Figure 3.14: Empirical probability of each mode over 101 replicates of warm-start Louvain downweighting best community of dominant mode.

3.2.4 Moving Between Modes

Can we downweight until Louvain finds the tunnel between modes? What does this tunnel look like? How likely is Louvain to find it?

To appreciate the difficulty that single-move MCMC methods have in moving between modes it is instructive to look at the paths (i.e. tunnels) they must take between modes. Figure 3.15 visualizes the shortest paths between the vertical and horizontal modes of the 8-vertex [2,2] block lattice graph with edge formation probabilities of [1.0, 1.0]. These shortest paths are the best tunnels between the modes. From this Figure it is clear that in order to move between modes we must pass through states that have significantly lower modularity, thus making such swaps unattractive. If the modularity of one of the modes was reduced to zero and Louvain started from there, then Louvain would move to the other mode.

Even amongst these shortest paths there is variability in the cost of taking them, with some paths having a greater modularity cost. In this example two partitions that appear on the shortest paths (coloured yellow) have negative modularity, and are $[0, 1, 0, 1, 1, 1, 1, 1]$ and $[0, 0, 0, 0, 0, 1, 0, 1]$.

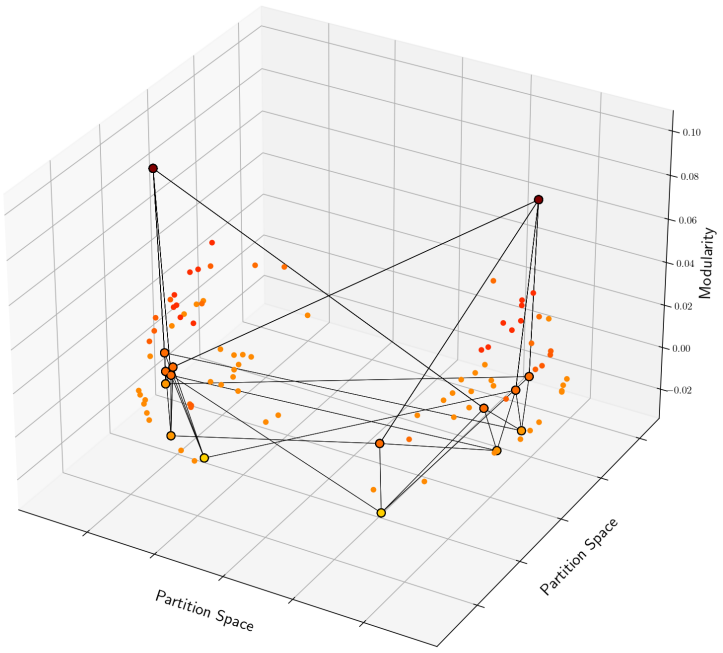

Figure 3.15: The 4! = 24 shortest paths between the horizontal and vertical modes of the 8-vertex [2,2] lattice graph. All partitions with positive modularity were also plotted.

We can ask what the optimal tunnel through the energy landscape looks like, if forced to make only individual Louvain moves i.e. one vertex at a time. We present two algorithms to do this, both of which give a characterization of the tunnels between modes. They both work with the blocks of the refined partition,

They both work with the blocks of the refined partition, which we can think of as working over the algebra induced by both partitions. We should move each block at a time to maximize the internal edge counts and hence the overall modularity. Two natural choices for algorithms to move between two modes present themselves; a split-then-merge approach and a merge-then-split approach.

In the split-then-merge approach, for each horizontal cluster the corresponding block in each vertical cluster is split off one vertex at a time and then merged. An example is given in Figure 3.17.

In the merge-then-split approach, for each horizontal cluster, split off the corresponding block from each vertical cluster not in the first vertical cluster. Merge each of these blocks into the first vertical cluster. Once they are all merged, split off the remaining vertical nodes that have not yet been assigned to a horizontal cluster. An example is given in Figure 3.18.

Figure 3.16 compares the modularity of the partitions visited by each approach. The merge-then-split passes through states of lower modularity, and so is not the optimal tunnel between the modes. Some local maxima are observed whenever all vertices within the same block of the refined partition have the same label.

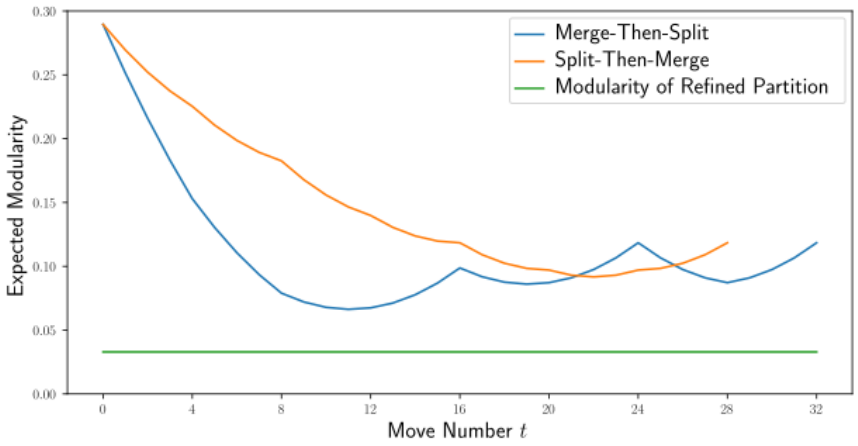

Figure 3.16: Change in modularity when moving between modes. The split-then-merge has the lower cost. Neither path visits a state with modularity as low as the refined partition.

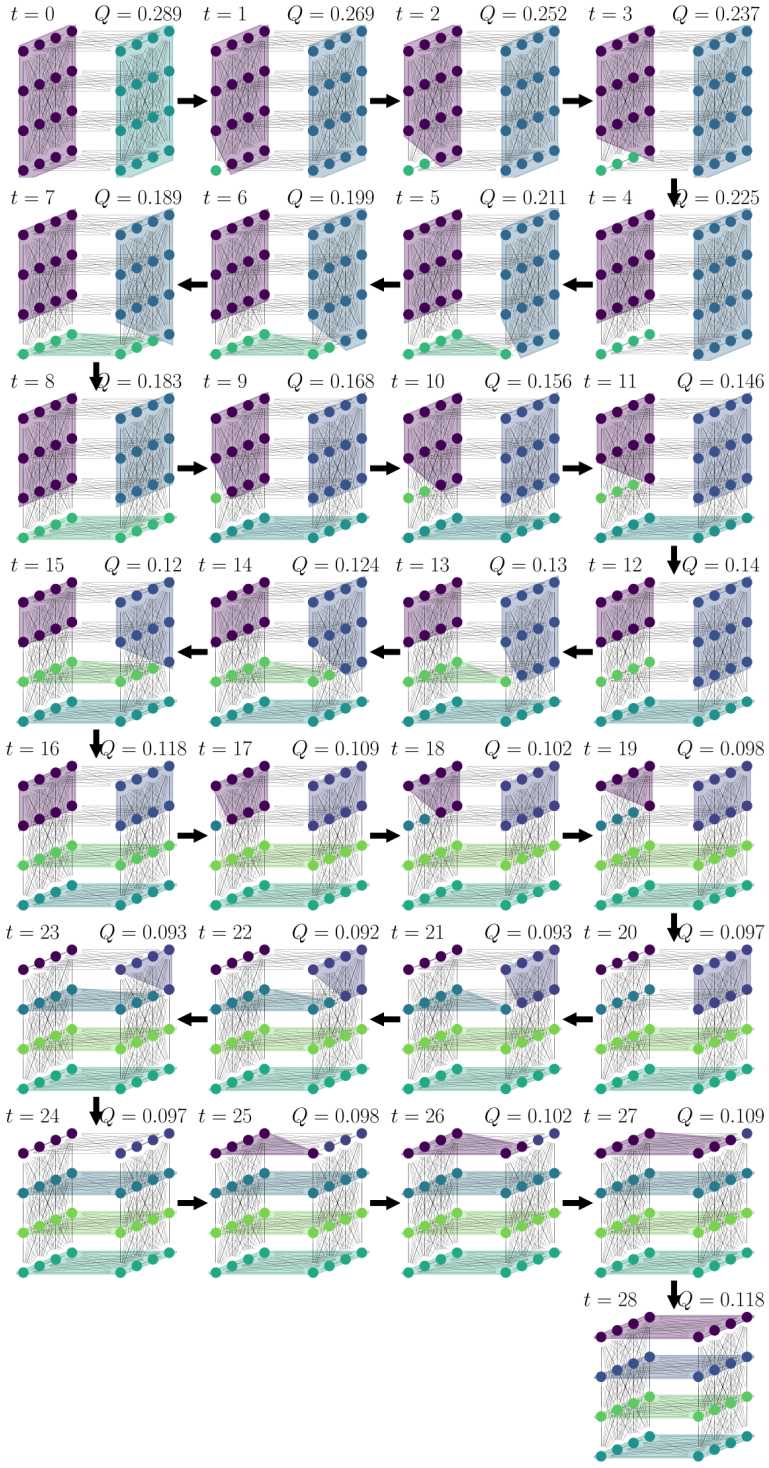

Figure 3.17: The split-then-merge approach to moving between modes. The Figure moves between the vertical and horizontal modes of the expectation graph of the [2,4]-block SBM on 32 vertices with edge formation probabilities of [0.3, 0.6].

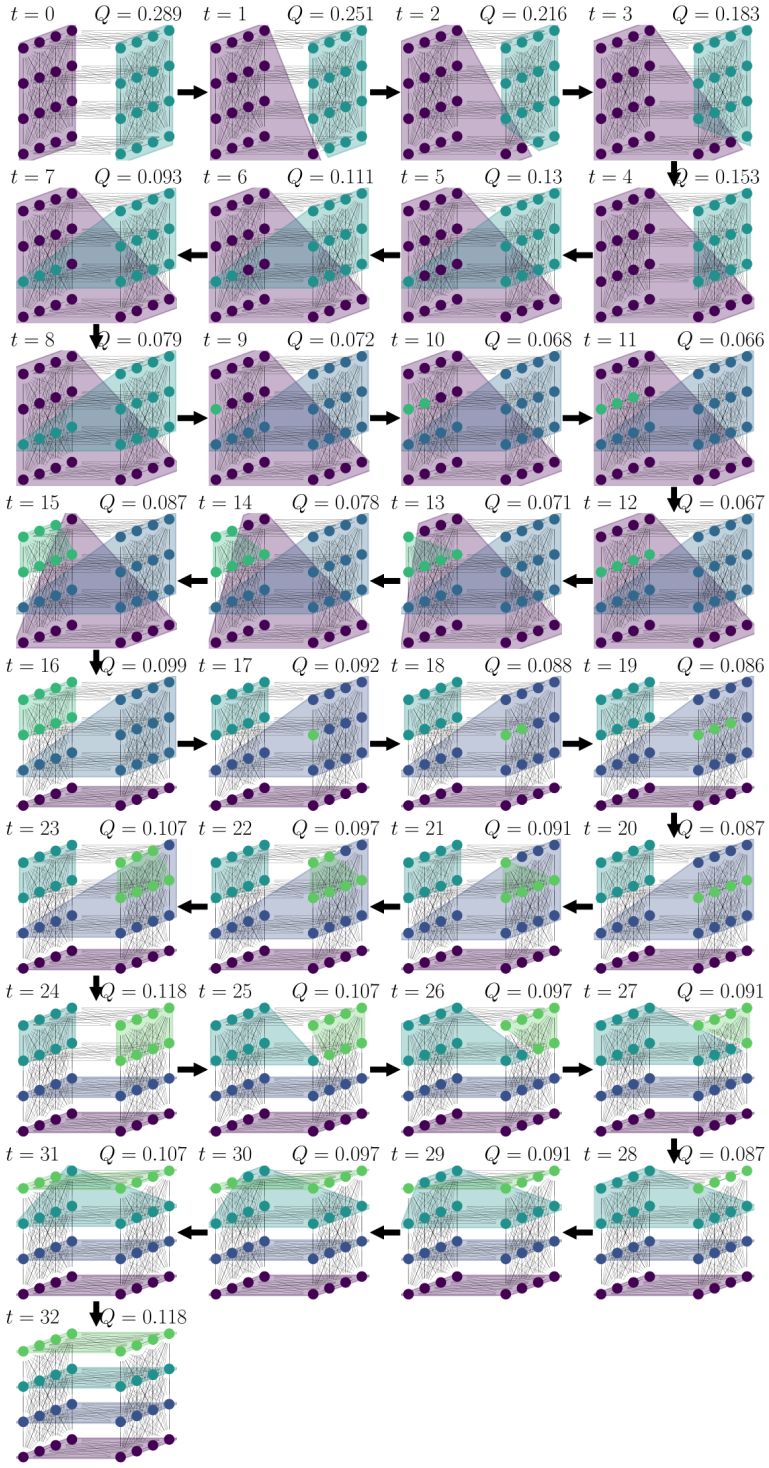

Figure 3.18: The merge-then-split approach to moving between modes. The Figure moves between the vertical and horizontal modes of the expectation graph of the [2,4]-block SBM on 32 vertices with edge formation probabilities of [0.3, 0.6].

3.3 Post-Processing

Iterative downweighting and calls to Louvain give us a procedure for producing a sequence of partitions. However at some point these partitions will deteriorate in quality by suggesting random partitions or parts that were already seen. Thus we need some heuristic for deciding when to stop sampling partitions.

One heuristic for deciding to stop sampling is if the current step returns a partition that is similar to those previously seen. At step $i+1$ Wang-Landau will downweight the best community from step i. If the other communities in the partition proposed by step $i+1$ only absorb the orphaned vertices from this best community, and do not materially change themselves, one could conclude that Wang-Landau has not proposed a new mode and therefore should stop after step i. This is evidence that Louvain is staying near a local extrema (i.e. mode) and not moving to another.

3.3.1 Stopping Criteria

We need some criteria for deciding when to stop sampling partitions. To inform this decision, after each step of sampling we can consider the partition refinement of all previous samples. Given a new partition, we can then ask if it is different enough from those already seen that we want to keep it and continue sampling.

The first heuristic we can apply is to check if partition C_{t+1} has stayed near the mode described by partition C_t. As our algorithm downweights the best part of C_t, if the only difference between C_{t+1} and C_t is that the vertices of the best community have been redistributed amongst the others, then the sampler has stayed close to the previous mode.

An **algebra of sets** is a pair $\langle V, C\rangle$ of a set V and family C of subsets of V. It contains the empty set as an element and is closed under taking complements in V, finite unions, and finite intersections. The refined partition gives us a set of disjoint parts that can be used to form such an algebra. Thus we can construct an algebra of sets from all previously seen parts, and use this to determine how similar the parts of a new partition are based on how closely they belong to this algebra.

Suppose $\{C_1, \ldots, C_t\}$ are the first t partitions generated from our sampler, and that $R(\{C_1, \ldots, C_t\})$ is the refinement of these partitions. Note that R itself is a partition, and so the subsets it contains are all disjoint. We now consider the algebra over V given by $\langle V, R(\{C_1, \ldots, C_t\})\rangle$ where V is the vertex set and the family of subsets is the refinement of all previous partitions.

If partition C_{t+1} lies near this set algebra, then it is similar to what has already been seen and should be rejected. At that point sampling could be stopped, however the sampler may simply be stuck near a modularity maximum and require more downweighting to move away. This would especially be a concern if only the community with the greatest modularity were being downweighted at each step.

Alternatively this algebra of sets can be thought of as a subspace of $\mathbb{Z}_2^{|V|}$ with basis consisting of the indicator vectors of each part of the partition refinement. The distance from the indicator vector of each part of C_{t+1} to this subspace can then be computed.

Once we have decided to stop we essentially have all the information we need and just need to decide on the resolution we want. This will determine to what extent intersections of observed parts are accepted into the final parts returned. We also need to define a distance to compare parts to the algebra.

3.3.2 Clustering Parts

Once we have a sequence of partitions from our algorithm some final post-processing is required to remove redundant parts that appeared in multiple samples. To do this we will construct a similarity measure between pairs of parts that will then be used to cluster the parts from these partitions.

Given two parts from two partitions, we want to know the likelihood they describe the same SBM. Let A and B be two partitions of n vertices. Suppose graphs G and G' were generated from the SBMs of A or B. Can we tell if they were both generated from the same SBM? If so, how many vertices do we expect to differ between the two? Roughly speaking, Louvain guarantees that v will be assigned to the community it has the greatest edge density within. What is the probability that v ends up in the wrong community? How many vertices do we expect to see mislabelled? What is the probability of seeing more than a given number of vertices mislabelled? Suppose we have a classifier that places a vertex in the part that it has the greatest edge density within. This is a simplification of what late stage Louvain does but will serve as a heuristic. What is the rate of misclassification?

Suppose we generate a graph G from an SBM defined by partition C and a matrix of edge probabilities P. Let $\hat{C}$ be the partition inferred from G. Suppose we have aligned the labels of the estimated part $\hat{C}_i$ and true part C_i, for instance by permuting the labels in order to maximize the maximum overlap distance. What is probability of observing a diference greater than or equal to Δ between $\hat{C}_i$ and C_i, i.e. what is $\mathbb{P}\left(\Delta \leq |C_i \setminus \hat{C}_i|\right)$? To answer this we first need to know the probability of a vertex being misclassifed. That is, if v has true membership C_i, what is the probability it is not classified as belonging to C_i? To make this problem tractable, we assume our classifier has a simple criteria for making decisions; it will assign v to whichever community its edges are densest in. This is a rough approximation of how Louvain's decision based on modularity works. We further assume that all previous vertices were correctly classified. This is going to make our bound optimistic.

Recall that P_{ij} is the probability of an edge between communities i and j. The number of edges between v and vertices from its true community i is given by a $X_i \sim \text{Binomial}(|C_i|, P_{ii})$ random variable, while the number of edges between v and

an alternative community is a $X_j \sim \text{Binomial}(|C_j|, P_{ij})$ for all $j \neq i$. Then the probability v is misclassified as community C_j is

$$\begin{aligned}
\mathbb{P}(v \text{ is misclassified}) &= \mathbb{P}\left(\frac{X_i}{|C_i|} \leq \frac{X_j}{|C_j|}\right) \\
&= \sum_{x_i=0}^{|C_i|} \sum_{x_j=0}^{|C_j|} \mathbb{P}(X_i = x_i)\mathbb{P}(X_j = x_j)\mathbb{1}_{x_i/|C_i| \leq x_j/|C_j|} \\
&= \sum_{x_i=0}^{|C_i|} \sum_{x_j=\lceil (x_i|C_j|)/|C_i| \rceil}^{|C_j|} \mathbb{P}(X_i = x_i)\mathbb{P}(X_j = x_j) \\
&= \sum_{x_i=0}^{|C_i|} \sum_{x_j=\lceil (x_i|C_j|)/|C_i| \rceil}^{|C_j|} \left(\binom{|C_i|}{x_i} P_{ii}^{x_i}(1-P_{ii})^{|C_i|-x_i}\right)\left(\binom{|C_j|}{x_j} P_{ij}^{x_j}(1-P_{ij})^{|C_j|-x_j}\right)
\end{aligned}$$

Vertex v will be misclassified if it has a greater edge density within any community other than its true one C_i. Thus the probability of misclassification is

$$\mathbb{P}\left(\frac{X_i}{|C_i|} < \max_{j \neq i}\left\{\frac{X_j}{|C_j|}\right\}\right)$$

which can be computed as

$$\mathbb{P}\left(\frac{X_i}{|C_i|} < \max_{j \neq i}\left\{\frac{X_j}{|C_j|}\right\}\right) = 1 - \mathbb{P}\left(\frac{X_i}{|C_i|} \geq \max_{j \neq i}\left\{\frac{X_j}{|C_j|}\right\}\right) = 1 - \prod_{j \neq i} \mathbb{P}\left(\frac{X_i}{|C_i|} \geq \frac{X_j}{|C_j|}\right)$$

As the true value of probability matrix P is unknown, the empirical probabilites from $\hat{C}$ can be used as an estimate.

Let $\Delta := |C_i \setminus \hat{C}_i|$. Then $\Delta \sim \text{Binomial}(|C_i|, \mathbb{P}(v \text{ is misclassified}))$ and we get a one-sided right-tail p-value that for another sample C' at least $|C_i' \setminus \hat{C}_i'|$ vertices will be misclassified from

$$\mathbb{P}\left(\Delta \geq |C_i' \setminus \hat{C}_i'|\right)$$

We take this as a similarity between parts.

To evaluate this p-value we compare it to a Monte Carlo p-value. Consider a 2-block SBM on 100 vertices evenly divided between blocks. Figure 3.19 shows that this anayltic formula is a good approximation. For the sake of comparison, a block size of 50-1 was used for v's block, i.e. v was taken to be the 50-th vertex of the block of interest in the analytic calculations. The 50-th vertex was found to have an analytic misclassification rate of 0.139816 and a Monte Carlo misclassification rate of 0.139986.

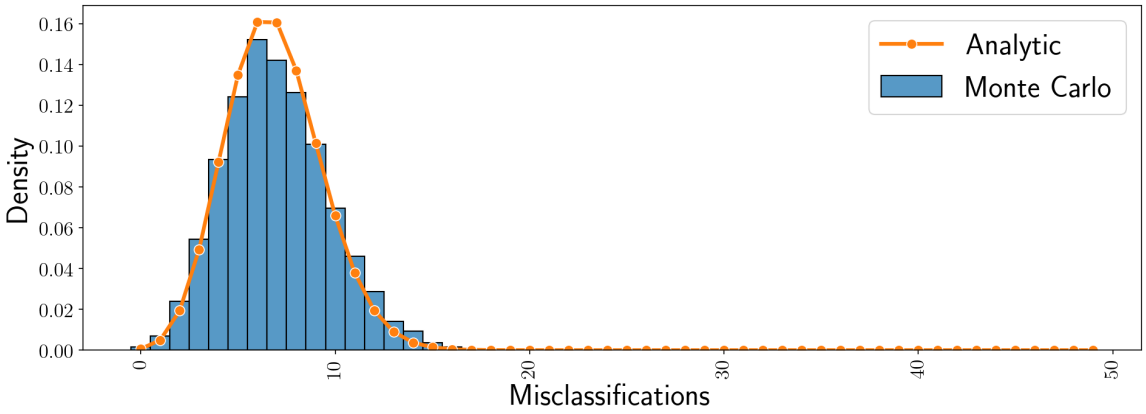

Figure 3.19: Monte Carlo and analytic probabilities of a given number of vertices being misclassified. 10,000 replicates were generated of a 2-block SBM with 50 vertices in each block, edge probablities of 0.5 within blocks and 0.4 between blocks. This was compared to the analytic value.

Does the above heuristic of comparing edge density to determine misclassification actually coincide with misclassification by Louvain? To answer this another Monte Carlo experiment was conducted with the same parameters as above. The maximum overlap distance was used to align Louvain's estimate of the membership with the true membership. This is the most charitable alignment possible and did not attempt to maximize alignment with the second part. From Figure 3.20 it is clear that the heuristic is much more optimistic than the reality of Louvain, with many more misclassifications in Louvain. The signal-to-noise ratio for this example is too high for Louvain to succeed, i.e. it is not in the feasible region. A Monte Carlo probability of a given difference between the heuristic and Louvain misclassification is given in Figure 3.21. The empirical probability of misclassification was 0.144 for the edge density heuristic and 0.329 for Louvain.

In practice we do not know the true parameters, however we can obtain empirical estimates for the edge probabilities $\hat{p}_{11}, \hat{p}_{22}, \hat{p}_{12}$ from the number of edges within and between the parts of $\hat{C}$ if we assume $\hat{C}$ is correct. These probabilities are not going to be true probabilities since we observe the data twice, once to place vertices in parts and again to obtain edge probality estimates, but they will still allow us to separate the parts.

Note that $\mathbb{P}(Y \geq |C_i \backslash \hat{C}_i|)$ is not symmetric i.e. $\mathbb{P}(Y \geq |C_i \backslash \hat{C}_i|) \neq P(Y \geq |\hat{C}_i \backslash C_i|)$. To symmetricize this in order to handle containment relationships, we take the larger of the two probabilities.

This probability gives us a similarity between parts, and taking one minus this probability gives a distance. The pairwise distance matrix can be computed and given to a spectral clusterer, about which more details can be found in Section 2.3.5.

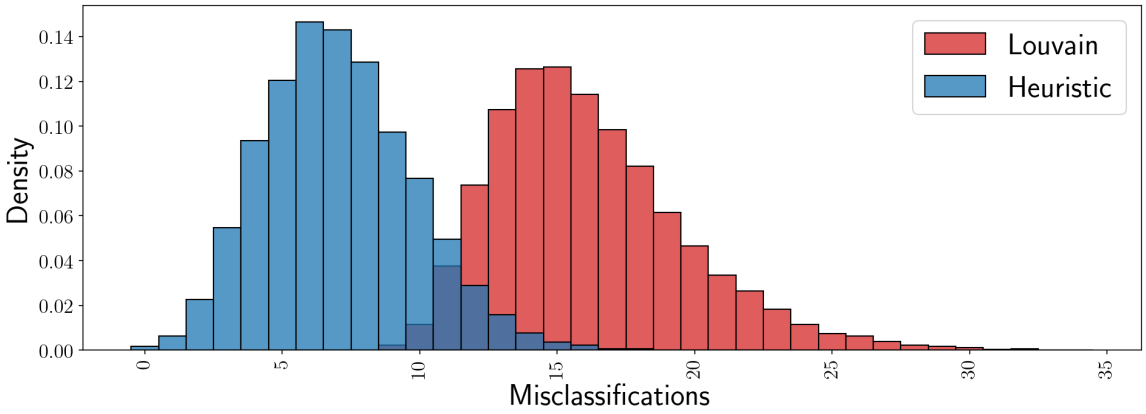

Figure 3.20: Monte Carlo probability distributions for the number of misclassifications according to the edge density heuristic and Louvain. 10,000 replicates were generated of a 2-block SBM with 50 vertices in each block and edge probablities of 0.5 within blocks and 0.4 between blocks.

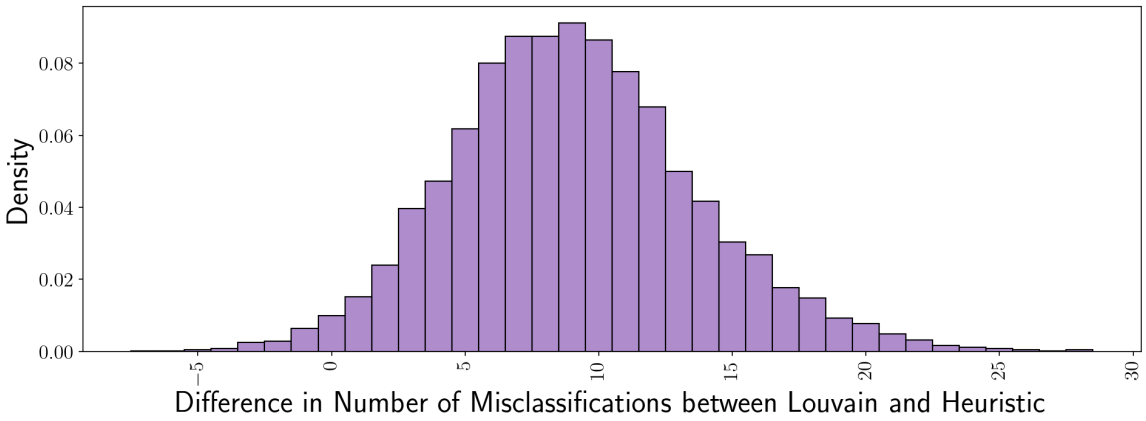

Figure 3.21: Difference between the number of misclassifications made by Louvain and predicted by the edge density heuristic. 10,000 replicates were generated of a 2-block SBM with 50 vertices in each block and edge probablities of 0.5 within blocks and 0.4 between blocks.

From each cluster only a single representative belonging to the earliest step is returned. The returned parts are ordered by which step of the algorithm they came from as an ordering of quality. Alternatively they could be ordered by their respective modularities.

Example 3.3.1. Consider the graph with two modes detailed in Figure 3.22. The first three samples from our algorithm are displayed in Figure 3.23, and the similarity matrix between all parts is visualized in Figure 3.24. The first two samples capture the two modes of this graph, with the third sample devolving into a perturbation of the previously seen parts. This is captured in the similarity matrix where the parts of the third partition have high similarity to the preceding parts. The Erdös-Rényi component is fairly stable throughout all the samples, and is identified in the (1,4) and (4,1) entries of the similarity matrix comparing the first two partitions, and again in (4,11),(11,4), and (1,11) entries. Note that the (11,1) pair of parts was not deemed similar, a consequence of the asymetric similarity used. However they will still be reported as similar after clustering because of the chain of similarity between them through other parts.

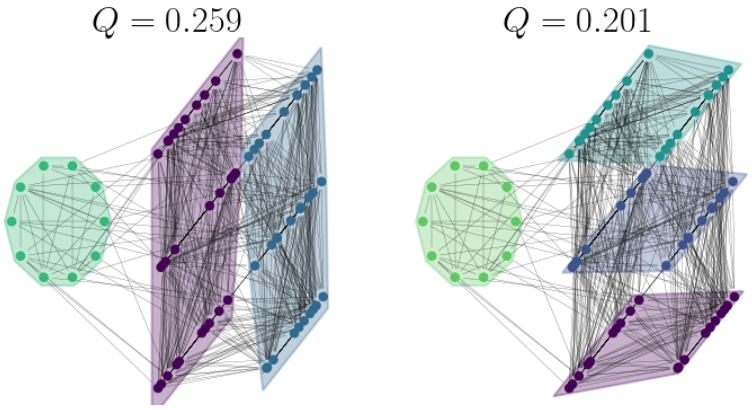

Figure 3.22: The disjoint union of an Erdös-Rényi random graph ($n = 10, p = 0.4$) and a lattice ($n = 60$, $p = [0.4, 0.4]$, and number of parts $r = [2, 3]$). Edges were then added as noise between all vertices from these two disconnected components with $p = 0.05$.

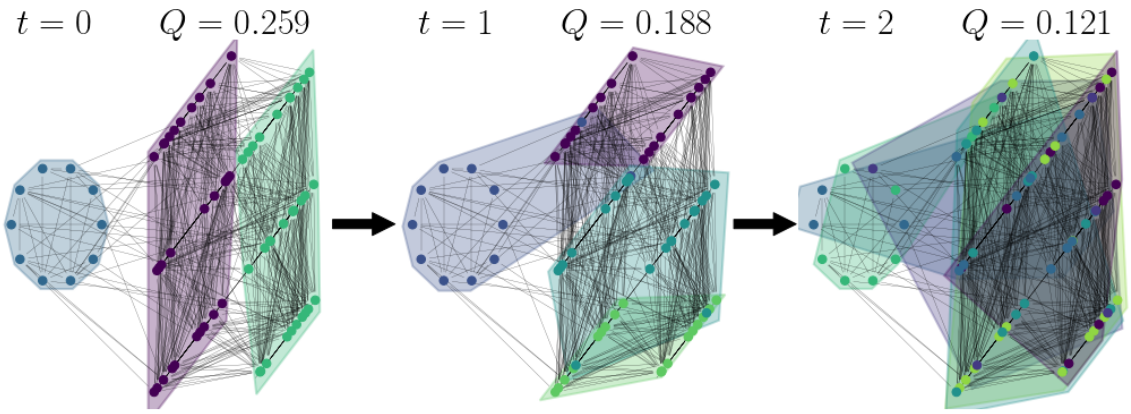

Figure 3.23: A sequence of partitions returned by our algorithm.

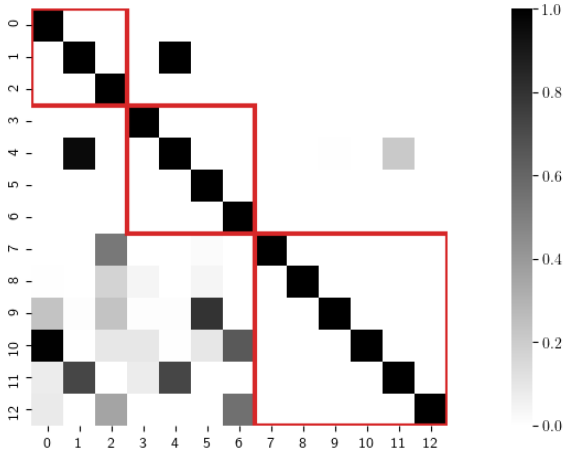

Figure 3.24: The similarity matrix of parts of the partitions from Figure 3.23 visualized as a heatmap. The red squares delimit the internal parts of each partition.

3.4 Some Properties of Louvain

3.4.1 Louvain Never Proposes the Refined Partition

Theorem 3.4.1. Let $A = A_1 \sqcup \cdots \sqcup A_{m_A}$ and $B = B_1 \sqcup \cdots \sqcup B_{m_B}$ be partitions of a set of vertices V, i.e. $V = \bigsqcup_{1 \le i \le m_A} A_i = \bigsqcup_{1 \le j \le m_B} B_j$. Let $G = (V, E)$ be a graph with vertex set V and edge set E. An edge forms within part A_i with probability p_A and within B_j with probability p_B. With high probability this graph is then multimodal as it is a mixture of the SBMs defined by A and B. Let $R = \bigsqcup_{1 \le i \le m_A, 1 \le j \le m_B} A_i \cap B_j$ be the refined partition with respect to A and B. Then Louvain will never choose the refined partition. Furthermore, during Louvain's first phase this refined partition will not be stable. The refined partition will always have modularity smaller than one of the modes A or B, i.e. $Q(A) > Q(R)$ or $Q(B) > Q(R)$.

We can see an intuitive argument for why this is true. Suppose Louvain was initialized to, or in the first phase of the first level chose, the refined partition. The second phase then sees the aggregation of each community's vertices into one vertex, i.e. each intersection $A_i \cap B_j$ becomes one vertex. Since it is multimodal, these aggregated vertices will be well connected and hence this induced graph will still be very modular. Consider the refinement of each part A_i of A with respect to B, i.e. $A_i = (A_i \cap B_1) \sqcup \cdots \sqcup (A_i \cap B_m)$, and similarly for part B_j of B $B_j = (B_j \cap A_1) \sqcup \cdots \sqcup (B_j \cap A_m)$. Then there will be many edges between the new vertices $(A_i \cap B_1), \ldots, (A_i \cap B_m)$ for each A_i of A and similarly amongst $(B_j \cap A_1), \ldots, (B_j \cap A_m)$ for each part B_j of B. Thus Louvain will further group these individual refined parts, leading to one of the modes being chosen. Note this is different than the resolution limit which assigns higher modularity to agglomerations of communities. We will explicitly show that $Q(A) > Q(R)$ or $Q(B) > Q(R)$.

Lemma 3.4.1. $\max\{Q(A), Q(B)\} > Q(R)$

Proof: Let graph G and partitions A, B, R be as above. For simplicity assume all blocks have the same size, i.e. $n := |A_i \cap B_j| \ \forall i, j$, and that each partition has the same number of parts, i.e. $m := m_A = m_B$. Then $|A_i| = |B_j| \ \forall i, j$, $|A_i| = |B_j| = m \cdot n$, and $|V| = |A| = |B| = m \cdot |A_i| = m \cdot |B_j| = m^2 \cdot n$.

Let $p_A, p_B \in [0, 1]$ be the probabilities of an edge forming between two vertices in a part A_i of A or B_j of B respectively. These parameters control the density of edges within G. Following the notation we introduced in Section 2.3.1, let $\Sigma_{C_i C_j}$ be the sum of weights of all edges between the sets of vertices C_i and C_j. Let $\mathbb{E}_A[\Sigma_{C_i C_j}]$ be the expected number of these edges generated by the SBM defined by partition A

and probability p_A. Then

$$\begin{aligned}\mathbb{E}[\Sigma_{A_iA_i}] &= \mathbb{E}_A[\Sigma_{A_iA_i}] + \mathbb{E}_B[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] \\ &= \frac{mn(mn-1)}{2}p_A + \sum_{j=1}^{m}\frac{n(n-1)}{2}p_B \\ &= \frac{mn(mn-1)}{2}p_A + m\frac{n(n-1)}{2}p_B\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\Sigma_{AA}] &= \sum_{i=1}^{m}\mathbb{E}[\Sigma_{A_iA_i}] \\ &= \sum_{i=1}^{m}\left(\mathbb{E}_A[\Sigma_{A_iA_i}] + \mathbb{E}_B[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}]\right) \\ &= \sum_{i=1}^{m}\left[\frac{mn(mn-1)}{2}p_A + \sum_{j=1}^{m}\frac{n(n-1)}{2}p_B\right] \\ &= m\frac{mn(mn-1)}{2}p_A + m^2\frac{n(n-1)}{2}p_B\end{aligned}$$

which is the sum of a Binomial$(m\frac{mn(mn-1)}{2}, p_A)$ and a Binomial$(m^2\frac{n(n-1)}{2}, p_B)$ random variable. Similarly $\mathbb{E}[\Sigma_{BB}] = m\frac{n(n-1)}{2}p_B$.

Since edge formation is independent by assumption, the expected sum of all edge weights is

$$\begin{aligned}\mathbb{E}[\Sigma_{VV}] &= \mathbb{E}_A[\Sigma_{AA}] + \mathbb{E}_B[\Sigma_{BB}] \\ &= \sum_{i=1}^{m}\mathbb{E}_A[\Sigma_{A_iA_i}] + \sum_{j=1}^{m}\mathbb{E}_B[\Sigma_{B_jB_j}] \\ &= m\frac{mn(mn-1)}{2}p_A + m\frac{mn(mn-1)}{2}p_B \\ &= \frac{m^2n(mn-1)}{2}(p_A+p_B)\end{aligned}$$

The sum of the expected degrees of each vertex in A_i, i.e. total expected degree of A_i, is given by

$$\begin{aligned}\mathbb{E}[\Sigma_{VA_i}] &= 2\cdot\mathbb{E}_A[\Sigma_{A_iA_i}] + \sum_{j=1}^{m}\mathbb{E}_B[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] + \sum_{j=1}^{m}\mathbb{E}_B[\Sigma_{A_iB_j}] \\ &= [mn(mn-1)]p_A + m[n(n-1)]p_B + m[n(mn-n)]p_B\end{aligned}$$

For the refined SBM where each $A_i \cap B_j$ is given its own community, the expected number of internal edges is

$$\begin{aligned}\mathbb{E}[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] &= \mathbb{E}_A[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] + \mathbb{E}_B[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] \\ &= \frac{n(n-1)}{2}p_A + \frac{n(n-1)}{2}p_B \\ &= \frac{n(n-1)}{2}(p_A + p_B)\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\Sigma_{RR}] &= \sum_{i=1}^{m}\sum_{j=1}^{m}\mathbb{E}[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m}\left[\mathbb{E}_A[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] + \mathbb{E}_B[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}]\right] \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m}\left[\frac{n(n-1)}{2}p_A + \frac{n(n-1)}{2}p_B\right] \\ &= m^2\frac{n(n-1)}{2}(p_A + p_B)\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\Sigma_{V\{A_i\cap B_j\}}] &= \mathbb{E}[\Sigma_{\{A_i\cap B_j\}\{A_i\cap B_j\}}] + \mathbb{E}_A[\Sigma_{\{A\setminus A_i\}\{A_i\cap B_j\}}] + \mathbb{E}_B[\Sigma_{\{B\setminus B_i\}\{A_i\cap B_j\}}] \\ &= [n(n-1)](p_A + p_B) + [n(mn-n)](p_A + p_B)\end{aligned}$$

$$
\begin{aligned}
Q(A) &= \sum_{i=1}^{m}\left[\frac{\mathbb{E}[\Sigma_{A_iA_i}]}{\mathbb{E}[\Sigma_{VV}]} - \left(\frac{\mathbb{E}[\Sigma_{VA_i}]}{2\mathbb{E}[\Sigma_{VV}]}\right)^2\right] \\
&= \sum_{i=1}^{m}\left[\frac{\frac{mn(mn-1)}{2}p_A + \frac{mn(n-1)}{2}p_B}{\frac{m^2n(mn-1)}{2}(p_A+p_B)}\right. \\
&\qquad \left. - \left(\frac{[mn(mn-1)]p_A + m[n(n-1)]p_B + m[n(mn-n)]p_B}{2\left[m\frac{mn(mn-1)}{2}p_A + m\frac{mn(mn-1)}{2}p_B\right]}\right)^2\right] \\
&= m\left[\frac{1}{m}\frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \left(\frac{\frac{1}{m}p_A + \frac{(n-1)}{m(mn-1)}p_B + \frac{(mn-n)}{m(mn-1)}p_B}{p_A+p_B}\right)^2\right] \\
&= m\left[\frac{1}{m}\frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \left(\frac{\frac{1}{m}p_A + \frac{(n-1)+(mn-n)}{m(mn-1)}p_B}{p_A+p_B}\right)^2\right] \\
&= m\left[\frac{1}{m}\frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \left(\frac{\frac{1}{m}p_A + \frac{(mn-1)}{m(mn-1)}p_B}{p_A+p_B}\right)^2\right] \\
&= m\left[\frac{1}{m}\frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \left(\frac{\frac{1}{m}p_A + \frac{1}{m}p_B}{p_A+p_B}\right)^2\right] \\
&= m\left[\frac{1}{m}\frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \left(\frac{1}{m}\frac{p_A+p_B}{p_A+p_B}\right)^2\right] \\
&= m\left[\frac{1}{m}\frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \frac{1}{m^2}\right] \\
&= \frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \frac{1}{m}
\end{aligned}
$$

$$
\begin{aligned}
Q(R) &= \sum_{1\le i,j\le m}\left[\frac{\mathbb{E}[|E_{A_i\cap B_j}|]}{\mathbb{E}[|E|]} - \left(\frac{\mathbb{E}[d_{A_i\cap B_j}]}{2\mathbb{E}[|E|]}\right)^2\right] \\
&= m^2\left[\frac{\frac{n(n-1)}{2}(p_A+p_B)}{m\frac{mn(mn-1)}{2}p_A + m\frac{mn(mn-1)}{2}p_B}\right. \\
&\qquad \left. - \left(\frac{[n(n-1)](p_A+p_B)+[n(mn-n)](p_A+p_B)}{2\left(m\frac{mn(mn-1)}{2}p_A + m\frac{mn(mn-1)}{2}p_B\right)}\right)^2\right] \\
&= m^2\left[\frac{(n-1)(p_A+p_B)}{m^2(mn-1)(p_A+p_B)} - \left(\frac{[n^2-n+mn^2-n^2](p_A+p_B)}{m^2n(mn-1)(p_A+p_B)}\right)^2\right] \\
&= m^2\left[\frac{(n-1)}{m^2(mn-1)} - \left(\frac{-n+mn^2}{m^2n(mn-1)}\right)^2\right] \\
&= \frac{(n-1)}{(mn-1)} - m^2\left(\frac{n(mn-1)}{m^2n(mn-1)}\right)^2 \\
&= \frac{(n-1)}{(mn-1)} - m^2\left(\frac{1}{m^2}\right)^2 \\
&= \frac{(n-1)}{(mn-1)} - \frac{1}{m^2}
\end{aligned}
$$

$$
\begin{aligned}
& Q(R) < Q(A) \\
\Leftrightarrow \quad & \frac{(n-1)}{(mn-1)} - \frac{1}{m^2} < \frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \frac{1}{m} \\
\Leftrightarrow \quad & 0 < \frac{p_A + \frac{(n-1)}{(mn-1)}p_B}{p_A+p_B} - \frac{1}{m} - \frac{(n-1)}{(mn-1)} + \frac{1}{m^2} \\
\Leftrightarrow \quad & 0 < \frac{[p_A + \frac{(n-1)}{(mn-1)}p_B](mn-1)m^2}{(p_A+p_B)(mn-1)m^2} - \frac{(p_A+p_B)(mn-1)m}{(p_A+p_B)(mn-1)m^2} \\
& \qquad - \frac{(p_A+p_B)(n-1)m^2}{(p_A+p_B)(mn-1)m^2} + \frac{(p_A+p_B)(mn-1)}{(p_A+p_B)(mn-1)m^2} \\
\Leftrightarrow \quad & 0 < [p_A + \frac{(n-1)}{(mn-1)}p_B](mn-1)m^2 - (p_A+p_B)(mn-1)m \\
& \qquad - (p_A+p_B)(n-1)m^2 + (p_A+p_B)(mn-1) \\
\Leftrightarrow \quad & 0 < p_A(mn-1)m^2 + \frac{(n-1)}{(mn-1)}p_B(mn-1)m^2 - p_A(mn-1)m
\end{aligned}
$$

$$
\begin{aligned}
& & & - p_B(mn-1)m - p_A(n-1)m^2 - p_B(n-1)m^2 + p_A(mn-1) \\
& & & + p_B(mn-1) \\
\Leftrightarrow & & 0 &< p_A[(mn-1)m^2 - (mn-1)m - (n-1)m^2 + (mn-1)] \\
& & & + p_B[(n-1)m^2 - (mn-1)m - (n-1)m^2 + (mn-1)] \\
\Leftrightarrow & & 0 &< p_A[m^3n - m^2 - m^2n + m - m^2n + m^2 + mn - 1] \\
& & & + p_B[m^2n - m^2 - m^2n + m - m^2n + m^2 + mn - 1] \\
\Leftrightarrow & & 0 &< p_A[m^3n - 2m^2n + mn + m - 1] + p_B[-m^2n + mn + m - 1] \\
\Leftrightarrow & & 0 &< p_Am^3n - (2p_A + p_B)m^2n + (p_A + p_B)mn + (p_A + p_B)m - (p_A + p_B) \\
\Leftrightarrow & & 0 &< np_Am^3 - n(2p_A + p_B)m^2 + (n+1)(p_A + p_B)m - (p_A + p_B) \\
\Leftrightarrow & & 0 &< [p_Am - 2p_A - p_B]m^2n + (p_A + p_B)[m(n+1) - 1] \\
\Leftrightarrow & & 0 &< [p_A(m-2) - p_B]m^2n + (p_A + p_B)[m(n+1) - 1]
\end{aligned}
$$

Since $p_A, p_B \in [0,1]$, $(p_A + p_B)[m(n+1) - 1]$ will always be positive. $[p_A(m-2) - p_B]m^2n$ could be negative if $p_A(m-2) < p_B$. However without loss of generality we can assume $p_A > p_B$. Thus by symmetry $\max(Q(A), Q(B)) > Q(R)$.

$$0 < p_A(m-2) - p_B \Leftrightarrow -p_A(m-2) < -p_B \Leftrightarrow p_A(m-2) > p_B \Leftrightarrow \frac{p_A}{p_B} > \frac{1}{(m-2)}$$

Thus when $\frac{p_A}{p_B} < \frac{1}{m-2}$ partition A will have smaller modularity than the refined partition R. This is the only case when both of the partitions will not have modularity greater than the refined partition.

∎

Figure 3.25 gives an example of this relationship for a small graph on 100 vertices. The number of parts of A and B range from a singleton partition to each vertex being in its own. The regime where A has modularity less than R is restricted to the region of the top left corner in blue, when there is a small number of parts and $p_A << p_B$.

The black vertical line corresponds to when $p_A = p_B$ and hence $\log_{10}(1) = 0$. For a fixed ratio p_A/p_B and fixed m, the difference in modularity $Q(B) - Q(R)$ will be somewhere on the other side of the vertical black line i.e. at the corresponding p_B/p_A value.

Theorem 3.4.2. Suppose $p := p_A = p_B$. Then $Q(A)$ does not depend on p.

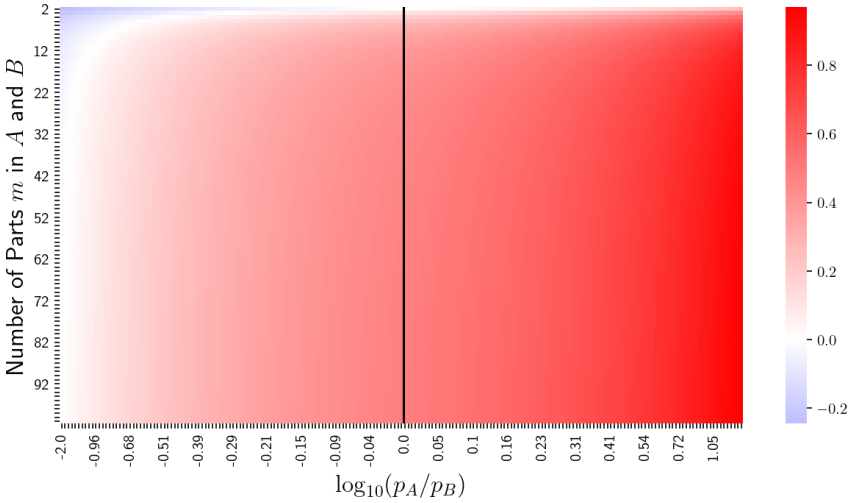

Figure 3.25: Difference between expected modularity of partition A and refined partition R as the ratio p_A/p_B of probabilities and number of parts m is varied. The number of vertices is fixed at $|V| = 100$. The colour is the difference $Q(A) - Q(R)$ between the modularity of partition A and the refined partition.

Proof:

$$
\begin{aligned}
Q(A) &= m\left[\frac{p+\frac{(n-1)}{(mn-1)}p}{p+p} - \left(\frac{\frac{1}{m}p+\frac{(n-1)}{m(mn-1)}p+\frac{(mn-n)}{m(mn-1)}p}{p+p}\right)^2\right] \\
&= m\left[\frac{p\left(1+\frac{(n-1)}{(mn-1)}\right)}{2p} - \left(\frac{p\left(\frac{1}{m}+\frac{(n-1)}{m(mn-1)}+\frac{(mn-n)}{m(mn-1)}\right)}{2p}\right)^2\right] \\
&= m\left[\frac{\left(1+\frac{(n-1)}{(mn-1)}\right)}{2} - \left(\frac{\left(\frac{1}{m}+\frac{(n-1)}{m(mn-1)}+\frac{(mn-n)}{m(mn-1)}\right)}{2}\right)^2\right]
\end{aligned}
$$

∎

3.4.2 Nucleation

This section describes an emergent phenomenon in the early decisions of Louvain and other greedy algorithms that explains why the downweight required to switch modes

is significantly smaller than expected. Suppose that we have a lattice graph where the edges generated by one mode have weight 1 and those from the other have weight $1-\epsilon$. Recall that at the beginning of Louvain each vertex starts in its own community, and that the algorithm at each steps decides whether to move a vertex into one of its neighbour's communities. At the beginning a vertex that has at least one neighbour with shared edge weight of one will be merged together and it will not merged with any neighbours it shares an edge weight of $1-\epsilon$ with. Thus Louvain will choose these 2-cliques with edge weights of 1 as these will have superior modularity compared to a 2-clique with weight $1-\epsilon$. This will carry on as 3-cliques, etc. are formed. Then after downweighting, during the initial stages it is more likely that these cliques will form within one mode, i.e. there will be a "nucleation" within that mode. This will bias the algorithm into choosing that mode over the other.

Suppose we have a SBM with a 2-mode lattice community structure. Let $C = C_1 \sqcup C_2 \sqcup \cdots \sqcup C_r$ and $C' = C'_1 \sqcup C'_2 \sqcup \cdots \sqcup C'_{r'}$ be the two partitions that were used to generate the internal edges of each mode. Let C be the dominant mode, i.e. $Q(C) > Q(C')$. This means that the edge formation probability within the parts of C is greater than the probability within parts of C'. Suppose each edge that exists is initially given a weight of 1. Now suppose that the internal edges of the dominant mode C have been downweighted by $1-\epsilon$. Initially Louvain will not select these edges, instead choosing the internal edges of the inferior mode C' that have weight 1. Thus initially cliques within the inferior mode's blocks will be chosen.

For each community C'_j, $j \in [1, \ldots, r']$, of the inferior mode, the edges inside each refined part $C_i \cap C'_j$ will be downweighted as they are internal to C_i. Thus these edges will not be chosen during the nucleation phase. Instead edges between the refined parts $C_1 \cap C'_j, \ldots, C_r \cap C'_j$ will be chosen. Hence there will be r' independent r-partite graphs that are formed during nucleation. That is, C induces a multipartite graph with respect to each community C'_j. The expected number of k-cliques in a multipartite graph will tell us the state that nucleation will leave Louvain in when it starts choosing edges with weight $1-\epsilon$.

3.5 Ensemble Topic Modelling

Community detection algorithms have previously been applied to natural language processing, specifically topic modelling of text data [36]. Let the vocabulary V be an ordered set of words, and corpus of text documents $\{D_1, \ldots, D_n\}$ be row vectors in $\mathbb{N}^{|V|}$ where the i-th entry of D_j gives the number of occurrences of the i-th word of V in document D_j for $i \in \{1, \ldots, |V|\}$ and $j \in \{1, \ldots, n\}$. For simplicity assume each document contains the same number of words, i.e. for all $j \in \{1, \ldots, n\}$ we have

$\|D_j\|_1 = N$ for some $N \in \mathbb{N}$. The matrix

$$D = \begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix}$$

is defined as the **document-term matrix**. Define **topics** $\beta_1, \ldots, \beta_k$ to be probability distributions over the words in vocabulary V. A **generative probabilistic topic model** is defined as a generative process by which the documents were created and is specified in Algorithm 6 from [9]. Each document D_i is given a document-topic vector $\theta_i \in [0,1]^k$ where $\|\theta_i\|_1 = 1$. This document-topic vector determines which topics the document's words are drawn from. Thus this is a type of mixed-membership model as each document can express multiple topics.

Algorithm 6: Document Generative Process

```
for topic β_j where j ∈ {1, ..., k} do
    Draw a topic distribution β_j over words in V
for document D_i where i ∈ {1, ..., n} do
    Draw a topic proportion θ_i
    for word-token W_m where m ∈ {1, ..., N} do
        Draw a topic Z_m with probability θ_i
        Draw a word W_m ∈ {1, ..., |V|} with probability β_{Z_m}
        Record word W_m as having appeared in the document by setting
          D_{i,W_m} = D_{i,W_m} + 1
return corpus of documents D_1, ..., D_n
```

Given a corpus of documents $\{D_1, \ldots, D_n\}$ generated in this way, the task is then to determine the latent parameters, i.e. the topic-word distributions $\beta_1, \ldots, \beta_k$ and document-topic distributions $\theta_1, \ldots, \theta_n$. **Probabilistic latent semantic analysis (PLSA)** or **indexing (PLSI)** [43] makes no assumptions about the distribution of the unknown parameters and uses an **expectation-maximization (EM)** algorithm [24] to find point estimates of them. **Latent Dirichlet allocation (LDA)** [10] is a Bayesian method based on PLSA that is commonly used. It specifies the topic-word and document-topic distributions as Dirichlet-multinomial distributions and uses techniques from Bayesian inference such as MCMC, Gibbs sampling, variational Bayes, etc. to learn these parameters. For both of these algorithms, the number of topics must be chosen beforehand. As the number of topics is generally unknown a priori, many models with differing numbers of topics are trained and the one with the best fit is chosen. However it may not be clear which to choose.

We now introduce our **ensemble PLSA (EPLSA)** algorithm 7 which extends ensemble clustering for graphs (ECG) (described in Section 2.3.3) or other consensus

clustering techniques on graphs to PLSA. Let D be the $n \times |V|$ document-term matrix of a corpus and $\{\beta_1^{(1)}, \ldots, \beta_k^{(1)}\}$ the k topics first returned by PLSA. We then append these topic vectors to the corpus to get a new corpus

$$D^{(1)} = \begin{bmatrix} D \\ \beta_1^{(1)} \\ \vdots \\ \beta_k^{(1)} \end{bmatrix}$$

This procedure of calling PLSA and appending the outputted topics is repeated, with the matrix at time t given by

$$D^{(t)} = \begin{bmatrix} D_{t-1} \\ \beta_1^{(t-1)} \\ \vdots \\ \beta_k^{(t-1)} \end{bmatrix}$$

Adding previously found topics to the corpus makes future partitions more like what was already seen. To speed up this rate of convergence, $\omega(t)$ copies of each of these topic vectors can be added, for some $\omega(t) > 0$. Equivalently a weight of $\omega(t)$ can be specified for each of these vectors within the PLSA algorithm.

Algorithm 7: Ensemble PLSA

Input: document-term matrix D, number of iterations $T \in \mathbb{N}$, number of topics $k \in \mathbb{N}$.

for $t \in \{1, \ldots, T\}$ **do**

$\{\beta_1^{(t)}, \ldots, \beta_k^{(t)}\} = PLSA(D^{(t)}, k)$

$$D^{(t+1)} = \begin{bmatrix} D^{(t)} \\ \beta_1^{(t)} \\ \vdots \\ \beta_k^{(t)} \end{bmatrix}$$

return topics $\{\beta_1^{(T)}, \ldots, \beta_k^{(T)}\}$

$D^{(t-1)}$ can be adjusted with the following transformation to row i.

$$D_i^{(t)} = D_i^{(t-1)} + \omega(t) \sum_{j=1}^{k} \left(\theta_i^{(t-1)}\right)_j \beta_j^{(t-1)}.$$

That is, the document at time t is equal to a mixture of the document at time $(t-1)$ and the estimate of the document obtained by the sparse methods. Note that β_j is a

distribution on words so the dimensions are correct. Now the size of D is static and the update instead makes D closer to the estimated low-rank matrix L where

$$L^{(t)} = \begin{bmatrix} \theta_1^{(t)} \\ \vdots \\ \theta_k^{(t)} \end{bmatrix} \begin{bmatrix} \beta_1^{(t)} \\ \vdots \\ \beta_k^{(t)} \end{bmatrix} =: \Theta^{(t)} B^{(t)}.$$

After rescaling this is equivalent to

$$D^{(t)} = (1-\epsilon)D^{(t-1)} + \epsilon L^{(t-1)}.$$

This is what ensemble clustering does, replacing the adjacency matrix $A^{(t-1)}$ by $A^{(t)} = (1-\epsilon)A^{(t-1)} + \epsilon L^{(t-1)}$, where now $L^{(t-1)}$ is the matrix given by the partition estimated at time $(t-1)$.

When $\omega(t)$ grows sufficiently quickly Algorithm 7 should converge to some stable set of topics. When $m(t)$ is very large, the difference $\|D^t - L^{t-1}\|$ is very small. In that situation, the Davis-Kahan theorem [90] tells us that the top eigenvectors and eigenvalues of D^t are very close to the top eigendata of L^{t-1}. The top eigendata then "contract" with t, implying convergence.

We can restate the problem of topic modelling as one of partitioning a graph. Let $\tilde{D}$ be the document-term matrix D with normalized rows, i.e. $D_j = D_j/\|D_j\|_1$. Using $\tilde{D}$ we can construct the following adjacency matrix of a bipartite graph

$$A = \begin{bmatrix} 0^{n\times n} & \tilde{D} \\ \tilde{D}^T & 0^{|V|\times|V|} \end{bmatrix}$$

This bipartite graph has documents as one part and words the other, with an edge between a given document and word if that word appears in that document. These edges are then weighted with the frequency that word occurs within that document. With respect to the graph statement of the problem, EPLSA (Algorithm 7) adds new vertices to the associated bipartite graph. However instead of adding new vertices, there should exist some equivalent reweighting of edges that produces the same change in PLSA.

Alternatively a community detection algorithm could be run on this graph.Words can have different meanings in different contexts and so could belong to multiple topics and documents can discuss multiple topics. Hence topic modelling is an instance where overlapping communities are likely to occur. Thus our overlapping community detection algorithm 5 could be applied here. It alternately downweights communities to recover overlapping communities and upweights them to improve stability. Our algorithm possesses the additional advantage of not needing to be told the number of topics beforehand as the communities emerge organically during the detection process.

Chapter 4

Application to Datasets

Heuristically, we expect to get very different clusterings of the same graph in the following conditions: (i) every vertex has two characteristics (e.g. city, university), (ii) edges are more likely if you share a value in a characteristic (e.g. people in the same city are more likely to become friends), and (iii) the two characteristics are somewhat independent (at the least, they should not have a very strongly positive correlation). For example considering relationships that formed from city and university affiliations, we might not expect (iii) to hold while people live within the same city they are going to university, and so the set of university students is a subset of the city's residents. For university graduates we might expect these two to be less correlated, as many people move for work after graduating. We applied our algorithm to two datasets. Unfortunately after doing so, neither appeared to have the overlapping community structure that would make applying our algorithm interesting.

First our algorithm was run over a dataset of email metadata [16]. It contains the `sender`, `to`, `cc`, `bcc`, and `time` fields for all emails sent or received by employees of the City of Seattle. The computer used to process this dataset was RAM i.e. memory limited; as such only the `sender` and `to` fields were kept. from 2017-01-01 to 2017-04-04. It was provided to Matt Chapman after he filed a Freedom of Information Act (FOIA) request; the story of how he obtained this data can be found at his blog [17]. Matt confirmed over email that this data can be used for research and education.

This dataset was augmented with the employee directory obtained from [80]. In August 2021 this directory was taken offline and a snapshot made available [81]. Thus the department labels of some employees were known and could be used to evaluate the proposed clusters.

The multilevel Louvain algorithm was run on this graph, followed by down-weighting and another run. The employees whose department was known were used as landmarks of a true clustering, under the assumption that employees within a department were more likely to email each other. The dispersion of the employees of each department across the proposed communities was visualized with the heatmaps

in Figure 4.1 and Figure 4.2. Each row of the heatmap sums to 1.

Downweighting caused the communities to aggregate together. This sort of hierarchical, lower resolution clustering was not desired. We didn't find evidence for other structure, and so believe that this dataset may not satisfy condition (i) of our heuristic for overlapping community structure, namely that vertices have more than one characteristic. In this case it appears an employee's department dictates which departments they correspond with, and that there is no secondary function within departments that would produce a significantly different alternative clustering.

We next looked at Reddit, a social media site where users organize in communities called subreddits to comment on submissions other users have made. Subreddits corresponding to major Canadian cities and universities were chosen. Comments on the 100 top submissions were downloaded, and the commenters on each submission noted. To construct the graph, the commenters comprised the vertices and an edge was formed if two commenters had commented on the same submission. Figures 4.3 and 4.4 show the conditional probability of a commenter in community X belonging to subreddit Y for the communities proposed before and after downweighting. Note the columns do not sum to 1 as users can comment in multiple subreddits, i.e. do not belong to a separated partition. There is little difference except for the emergence of a third community corresponding to the Vancouver community.

The communities suffered from a class imbalance that we believe affected the results, resulting in a case that was beyond what was information theoretically possible to recover. In particular having a true community greater than the others causes a resolution limit. The Vancouver community had significantly more comments by top commenters (2804 versus the next most of 723) as well as top commenters (239 versus the next most of 89). The number of comments for each community was {Vancouver: 2804, Calgary: 723, Toronto: 708, Ottawa: 600, UBC: 338, Montreal: 309, uCalgary: 225, UofT: 214, GeeGees: 172, Carletonu: 165, McGill: 152} and commenters was {Vancouver: 239, Calgary: 89, Toronto: 86, Ottawa: 74, UBC: 41, Montreal: 39, uCalgary: 24, UofT: 23, McGill: 21, Carletonu: 19, GeeGees: 16}. The city communities also contained significantly more comments and commenters than the university communities, creating a further imbalance. In an attempt to correct for the first imbalance only a subset of similarly sized city and university communities was considered. Figures 4.5 and 4.6 visualize the quality of the proposed clusters. Again they did not change significantly after downweighting.

From this it appears that condition (iii) of our heuristic, that the characteristics not be positively correlated, was not satisfied. Figures 4.7 and 4.8 show the conditional probability of commenting in true Y given a user has commented in true community X. It is clear that given a user has commented in a university community, they are likely to comment in the city community where that university is situated. Thus the city and university characteristics are not independent. This produced the hierarchical clustering with clusters in Figures 4.5 and 4.6 where there were clusters

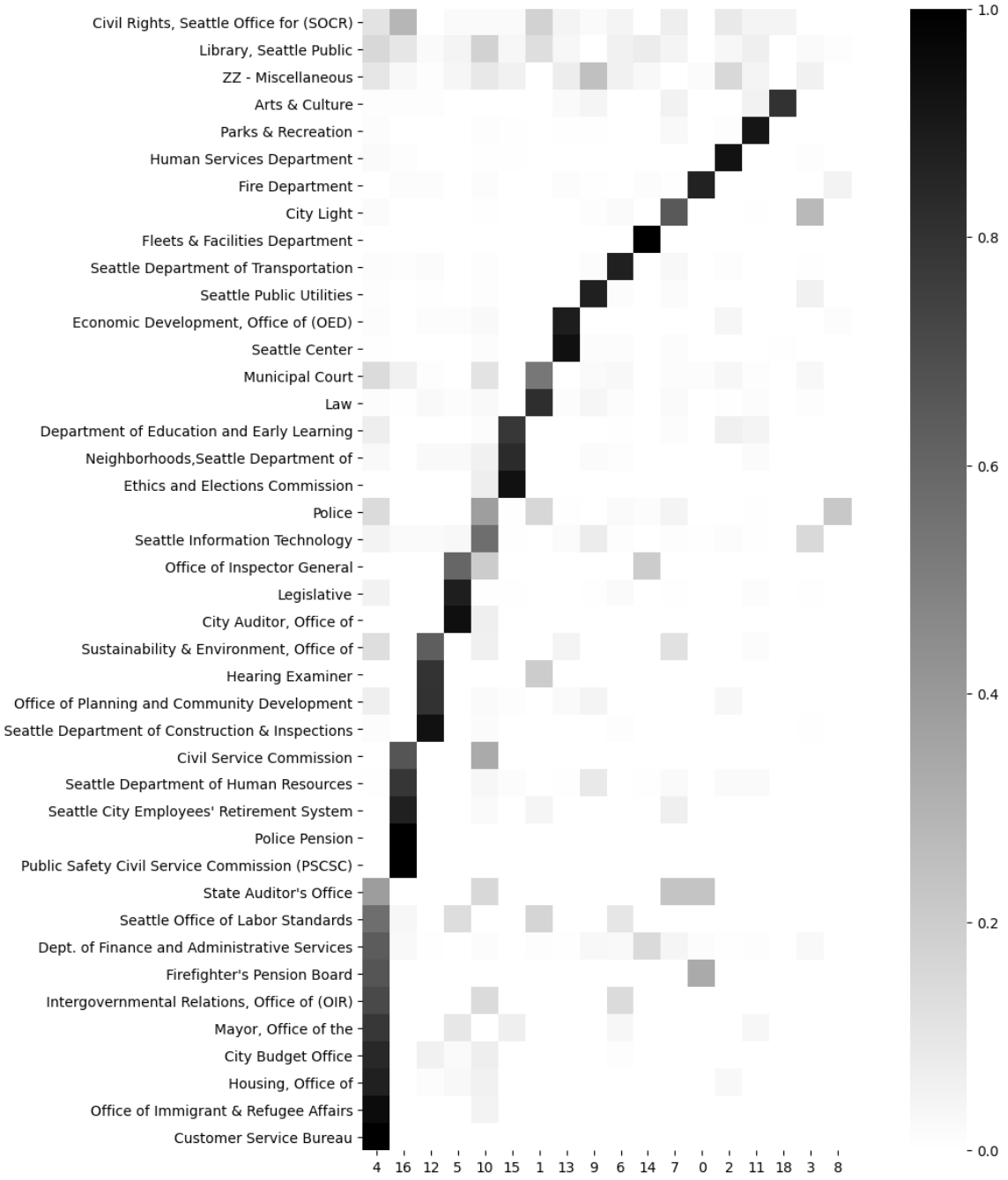

Figure 4.1: Dispersion of departments across the communities proposed by multilevel Louvain.

Figure 4.2: Dispersion of departments across the communities proposed after downweighting.

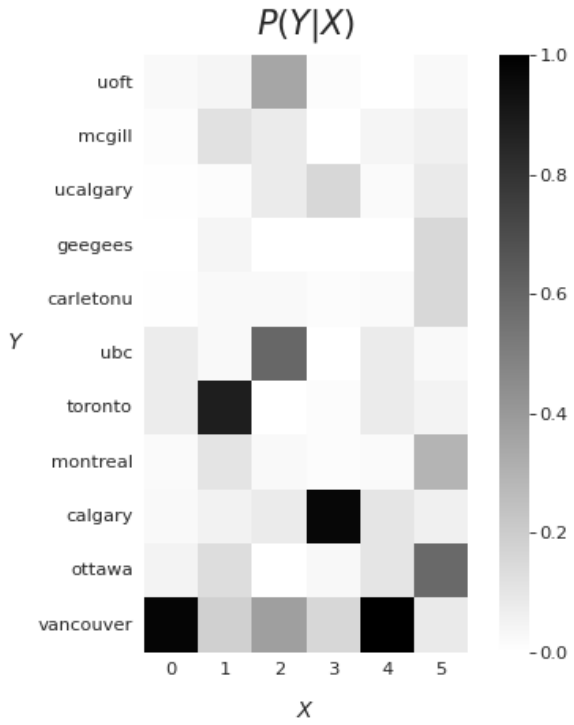

Figure 4.3: The conditional probability of posting in subreddit Y given a commenter was placed in community X before downweighting.

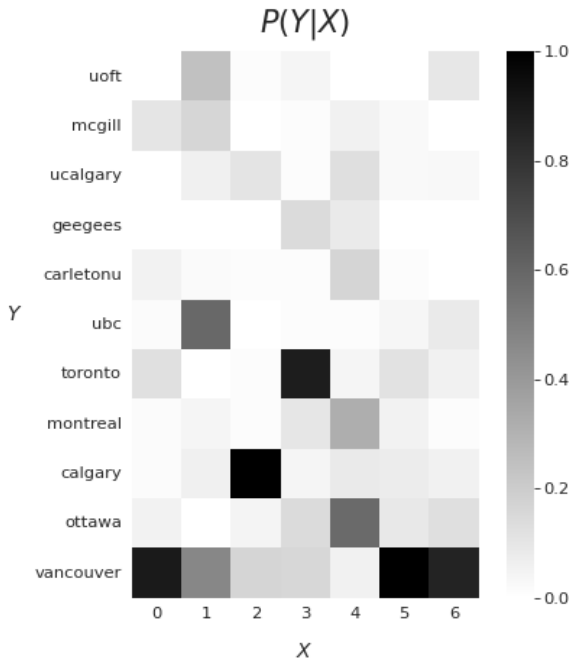

Figure 4.4: The conditional probability of posting in subreddit Y given a commenter was placed in community X after downweighting.

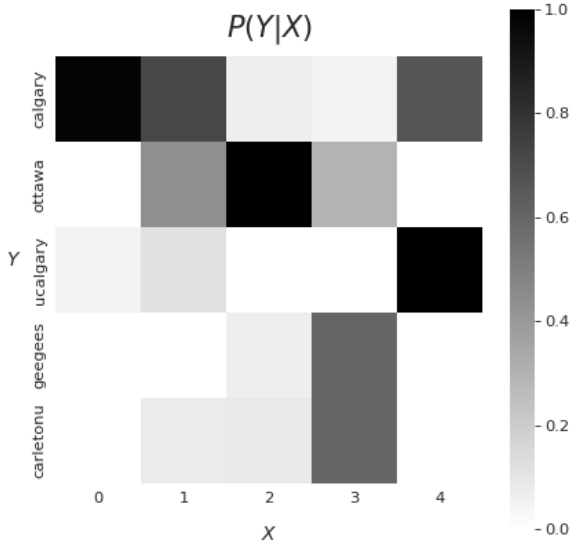

Figure 4.5: The conditional probability of posting in subreddit Y given a commenter was placed in community X before downweighting.

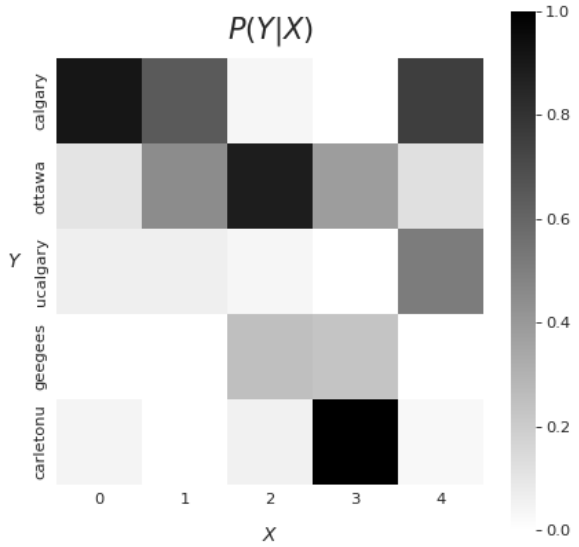

Figure 4.6: The conditional probability of posting in subreddit Y given a commenter was placed in community X after downweighting.

for the city and the city plus its university.

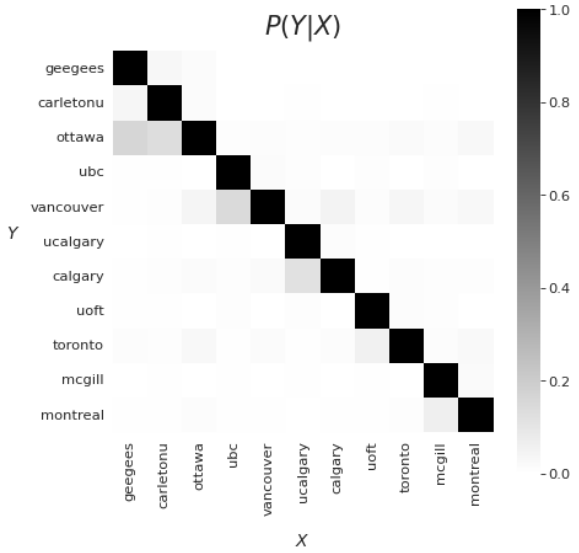

Figure 4.7: The conditional probability of posting in subreddit Y given a commenter has posted in subreddit X.

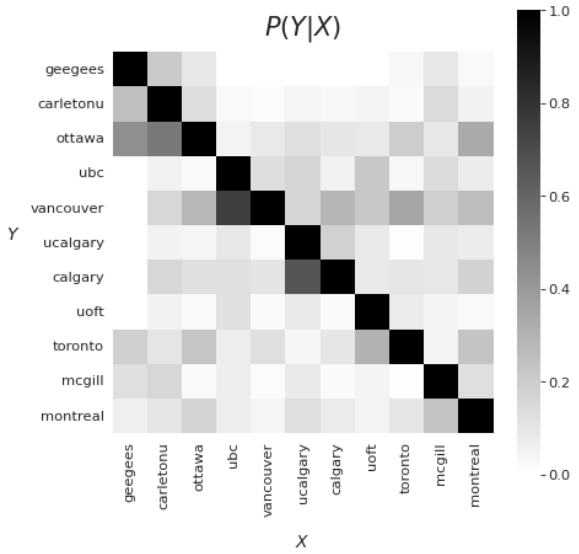

Figure 4.8: The conditional probability of posting in subreddit Y given a commenter has posted in subreddit X for the top 1% of commenters (i.e. those that made 9 or more than comments.

Chapter 5

Conclusion

This thesis presented an algorithm for detecting overlapping communities. We endeavoured throughout to construct an algorithm that scales efficiently to large graphs. In many places there was not an obvious best set of choices for the algorithm and so we provided variable options and some suggested tools for analysis. In particular this includes the decisions made at the beginning of Section 3.2 about how many and which clusters to downweight, and by how much, as well as the stopping criteria in Section 3.3.1 when to stop downweighting.

Unfortunately the two datasets we examined did not exhibit the overlapping structure that would necessitate applying our algorithm. Finding a real-world dataset with these properties, and then running our algorithm against it, is warranted to guage its effectiveness. Further to this, it would be nice to benchmark our algorithm against others on synthetic data, for instance on the artificial LFR benchmark of increasing noise and overlapping communities [49]. Our algorithm should be compared with other algorithms such as ego-splitting [28] and Bayesian inference [67]. We can also employ measures to evaluate the quality of the returned overlapping communities given the ground truth, such as the Normalized Mutual Information (NMI) measure for sets of overlapping clusters [56, 55] and the Omega measure [20, 79]. Applying the algorithm to topic model text data as discussed in Section 3.5 would also be interesting.

There is also further theory that could be developed, for instance around nucleation or Louvain's dynamics. The downweight could alternatively be informed by the gradient of the modularity around a stable partition. In Section 3.3.2 on clustering parts we constructed a similarity measure between two parts which required estimating the probability a vertex was misclassified. This estimate could be further improved by assuming there was misclassification before the final vertex v was added.

In Section 2.3.7 we discussed some results from Abbe about when recovery is feasible. New exact recovery and identifiability theorems for the overlapping case could be proven. Under certain conditions, you can get recovery for the overlapping

SBMs by using the Abbe result. This is not in the Abbe paper but follows as a consequence. Further, we believe that overlapping SBMs are heuristically different than how they are treated in that paper as they are much smaller than the associated full SBM, and as evidenced by Abbe's result, smaller objects are easier to recover.

References

[1] Emmanuel Abbe, Afonso S. Bandeira, and Georgina Hall. "Exact Recovery in the Stochastic Block Model". In: *CoRR* (2014). arXiv: 1405.3267 [cs.SI].

[2] Emmanuel Abbe and Colin Sandon. "Community detection in general stochastic block models: fundamental limits and efficient recovery algorithms". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 670–688. arXiv: 1503.00609 [math.PR].

[3] Edoardo M. Airoldi et al. "Introduction to Mixed Membership Models and Methods". In: *Handbook of Mixed Membership Models and Their Applications*. CRC press, 2014. Chap. 1, pp. 3–14. ISBN: 9780367330842. DOI: 10.1201/b17520. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.672.6529&rep=rep1&type=pdf.

[4] Thomas Aynaud. *python-louvain 0.15: Louvain algorithm for community detection*. 2020. URL: https://github.com/taynaud/python-louvain.

[5] Albert-László Barabási and Réka Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (Oct. 1999), pp. 509–512. ISSN: 1095-9203. DOI: 10.1126/science.286.5439.509. arXiv: cond-mat/9910332 [cond-mat.dis-nn].

[6] R. E. Belardinelli, S. Manzi, and V. D. Pereyra. "Analysis of the convergence of the $1/t$ and Wang-Landau algorithms in the calculation of multidimensional integrals". In: *Physical Review E* 78.6 (Dec. 2008). ISSN: 1550-2376. DOI: 10.1103/physreve.78.067701. arXiv: 0806.0268 [cond-mat.stat-mech].

[7] R. E. Belardinelli and V. D. Pereyra. "Wang-Landau algorithm: A theoretical analysis of the saturation of the error". In: *The Journal of Chemical Physics* 127.18 (Nov. 2007). ISSN: 1089-7690. DOI: 10.1063/1.2803061. arXiv: cond-mat/0702414 [cond-mat.stat-mech].

[8] David M. Blei. *Mixed-membership Models (and an introduction to variational inference)*. Course notes for Foundations of Graphical Models. Nov. 2015. URL: https://web.archive.org/web/20151129024315/https://www.cs.columbia.edu/~blei/fogm/2015F/notes/mixed-membership.pdf.

[9] David M. Blei and John D. Lafferty. "Topic Models". In: *Text Mining*. Chapman and Hall/CRC, 2009, pp. 101–124. ISBN: 9780429191985. DOI: 10.1201/9781420059458.

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022. URL: https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf.

[11] Vincent D. Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/p10008. arXiv: 0803.0476 [physics.soc-ph].

[12] U. Brandes et al. "On Modularity Clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 20.2 (2008), pp. 172–188. URL: https://kops.uni-konstanz.de/bitstream/handle/123456789/5853/modularity.pdf.

[13] Ulrik Brandes et al. "Maximizing Modularity is hard". In: *arXiv preprint* (2006). arXiv: physics/0608255 [physics.data-an].

[14] Anna D. Broido and Aaron Clauset. "Scale-free networks are rare". In: *Nature Communications* 10.1 (Mar. 2019). ISSN: 2041-1723. DOI: 10.1038/s41467-019-08746-5. arXiv: 1801.03400 [physics.soc-ph].

[15] Ramon Ferrer I. Cancho and Richard V. Solé. "The small world of human language". In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 268.1482 (2001), pp. 2261–2265. DOI: 10.1098/rspb.2001.1800.

[16] Matt Chapman. *01/17-03/17 City of Seattle Email Metadata*. Version 1. Kaggle, Oct. 2018. URL: https://www.kaggle.com/foiachap/01170317-city-of-seattle-email-metadata.

[17] Matt Chapman. *That Time the City of Seattle Accidentally Gave Me 32m Emails for 40 Dollars*. Oct. 2018. URL: https://web.archive.org/web/20210501233045/https://mchap.io/that-time-the-city-of-seattle-accidentally-gave-me-32m-emails-for-40-dollars4997.html.

[18] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. "Finding community structure in very large networks". In: *Physical Review E* 70.6 (Dec. 2004). ISSN: 1550-2376. DOI: 10.1103/physreve.70.066111. arXiv: cond-mat/0408187 [cond-mat.stat-mech].

[19] Amin Coja-Oghlan. "Graph partitioning via adaptive spectral techniques". In: *Combinatorics, Probability & Computing* 19.2 (2010), p. 227. DOI: 10.1017/S0963548309990514.

[20] Linda M. Collins and Clyde W. Dent. "Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions". In: *Multivariate Behavioral Research* 23.2 (1988), pp. 231–242. DOI: 10.1207/s15327906mbr2302_6.

[21] David F. Crouse. "On implementing 2D rectangular assignment algorithms". In: *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696. DOI: 10.1109/TAES.2016.140952.

[22] Gabor Csardi and Tamas Nepusz. *The igraph software package for complex network research*. 2006. URL: https://igraph.org.

[23] Gábor Csárdi and Eric D. Kolaczyk. *Statistical Analysis of Network Data with R*. New York: Springer, 2014. ISBN: 1493909827. URL: https://link.springer.com/book/10.1007%2F978-3-030-44129-6.

[24] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

[25] Stijn Van Dongen. *Performance Criteria for Graph Clustering and Markov Cluster Experiments*. Tech. rep. CWI (Centre for Mathematics and Computer Science), May 2000. URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.9783.

[26] Ran Duan and Seth Pettie. "Linear-time approximation for maximum weight matching". In: *Journal of the ACM (JACM)* 61.1 (Jan. 2014), pp. 1–23. ISSN: 0004-5411. DOI: 10.1145/2529989.

[27] Justin Eldridge, Mikhail Belkin, and Yusu Wang. "Unperturbed: spectral analysis beyond Davis-Kahan". In: *Algorithmic Learning Theory*. PMLR. 2018, pp. 321–358. arXiv: 1706.06516 [stat.ML].

[28] Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. "Ego-splitting framework: From non-overlapping to overlapping clusters". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 145–154. DOI: 10.1145/3097983.3098054. URL: https://dl.acm.org/doi/pdf/10.1145/3097983.3098054.

[29] Paul Erdös and Alfréd Rényi. "On the evolution of random graphs". In: *The structure and dynamics of networks*. Princeton University Press, 2011, pp. 38–82. URL: https://www.renyi.hu/~p_erdos/1960-10.pdf.

[30] Katherine Faust. "Using Correspondence Analysis for Joint Displays of Affiliation Networks". In: *Models and Methods in Social Network Analysis*. Structural Analysis in the Social Sciences. Cambridge University Press, 2005. Chap. 7, pp. 117–147. DOI: 10.1017/CBO9780511811395.007.

[31] S. Fortunato and M. Barthelemy. "Resolution limit in community detection". In: *Proceedings of the National Academy of Sciences* 104.1 (Dec. 2006), pp. 36–41. ISSN: 1091-6490. DOI: 10.1073/pnas.0605965104. arXiv: physics/0607100 [physics.soc-ph].

[32] Santo Fortunato and Claudio Castellano. "Community Structure in Graphs". In: *arXiv preprint* (2007). arXiv: 0712.2716 [physics.soc-ph].

[33] Santo Fortunato and Darko Hric. "Community detection in networks: A user guide". In: *Physics Reports* 659 (Nov. 2016), pp. 1–44. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2016.09.002. arXiv: 1608.00163 [physics.soc-ph].

[34] Eiko Fried. *R tutorial: clique percolation to detect communities in networks*. Nov. 2019. URL: https://psych-networks.com/r-tutorial-clique-percolation-to-detect-communities-in-networks/.

[35] Anne-Claude Gavin et al. "Functional organization of the yeast proteome by systematic analysis of protein complexes". In: *Nature* 415.6868 (2002), pp. 141–147. DOI: 10.1038/415141a.

[36] Martin Gerlach, Tiago P. Peixoto, and Eduardo G. Altmann. "A network approach to topic models". In: *Science Advances* 4.7 (July 2018), eaaq1360. ISSN: 2375-2548. DOI: 10.1126/sciadv.aaq1360. arXiv: 1708.01677 [stat.ML].

[37] E. N. Gilbert. "Random Graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144. DOI: 10.1214/aoms/1177706098.

[38] M. Girvan and M. E. J. Newman. "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12 (June 2002), pp. 7821–7826. ISSN: 1091-6490. DOI: 10.1073/pnas.122653799. arXiv: cond-mat/0112110 [cond-mat.stat-mech].

[39] Prem K Gopalan and David M Blei. "Efficient discovery of overlapping communities in massive networks". In: *Proceedings of the National Academy of Sciences* 110.36 (2013), pp. 14534–14539. DOI: 10.1073/pnas.1221839110.

[40] Mark S. Handcock, Adrian E. Raftery, and Jeremy M. Tantrum. "Model-based clustering for social networks". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 170.2 (2007), pp. 301–354. DOI: 10.1111/j.1467-985X.2007.00471.x.

[41] Barry Hembree. "Production Process Characterization". In: *NIST/SEMATECH e-Handbook of Statistical Methods*. 2002. Chap. 3. DOI: 10.18434/M32189. URL: https://www.itl.nist.gov/div898/handbook/pmc/section3/pmc331.htm.

[42] Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. "Latent space approaches to social network analysis". In: *Journal of the american Statistical association* 97.460 (2002), pp. 1090–1098. DOI: 10.1198/016214502388618906.

[43] Thomas Hofmann. "Probabilistic Latent Semantic Indexing". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 1999, pp. 50–57. DOI: 10.1145/312624.312649.

[44] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic blockmodels: First steps". In: *Social networks* 5.2 (1983), pp. 109–137. DOI: 10.1016/0378-8733(83)90021-7.

[45] Dean Isaacson. "A characterization of geometric ergodicity". eng. In: *Zeitschrift f r Wahrscheinlichkeitstheorie und Verwandte Gebiete* 49.3 (1979), pp. 267–273. ISSN: 0044-3719. DOI: 10.1007/BF00535499.

[46] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9.

[47] Brian Karrer and M. E. J. Newman. "Stochastic blockmodels and community structure in networks". In: *Physical Review E* 83.1 (Jan. 2011). ISSN: 1550-2376. DOI: 10.1103/physreve.83.016107. arXiv: 1008.3926 [physics.soc-ph].

[48] Frank R. Kschischang, Brendan J. Frey, and H.-A. Loeliger. "Factor graphs and the sum-product algorithm". In: *IEEE Transactions on information theory* 47.2 (2001), pp. 498–519. DOI: 10.1109/18.910572.

[49] Andrea Lancichinetti and Santo Fortunato. "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities". In: *Phys. Rev. E* 80 (1 July 2009), p. 016118. DOI: 10.1103/PhysRevE.80.016118. arXiv: 0904.3940 [physics.soc-ph].

[50] Andrea Lancichinetti and Santo Fortunato. "Community detection algorithms: A comparative analysis". In: *Physical Review E* 80.5 (Nov. 2009). ISSN: 1550-2376. DOI: 10.1103/physreve.80.056117. arXiv: 0908.1062 [physics.soc-ph].

[51] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. "Benchmark graphs for testing community detection algorithms". In: *Phys. Rev. E* 78 (4 Oct. 2008), p. 046110. DOI: 10.1103/PhysRevE.78.046110. arXiv: 0805.4770 [physics.soc-ph].

[52] Jens Lange. *An Introduction to the Clique Percolation Community Detection Algorithm*. Oct. 2019. URL: https://cran.r-project.org/web/packages/CliquePercolation/vignettes/CliquePercolation.html.

[53] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009. 371 pp. ISBN: 978-1-4704-1204-3. URL: https://bookstore.ams.org/mbk-58/.

[54] Laurent Massoulie. "Community detection thresholds and the weak Ramanujan property". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 2013, pp. 694–703. arXiv: 1311.3085 [cs.SI].

[55] Aaron F. McDaid. *An implementation of a Normalized Mutual Information (NMI) measure for sets of overlapping clusters*. 2016. URL: https://github.com/aaronmcdaid/Overlapping-NMI.

[56] Aaron F. McDaid, Derek Greene, and Neil Hurley. "Normalized Mutual Information to evaluate overlapping community finding algorithms". In: (2013). arXiv: 1110.2515 [physics.soc-ph].

[57] Miller Mcpherson, Lynn Smith-Lovin, and James Cook. "Birds of a Feather: Homophily in Social Networks". In: *Annual Review of Sociology* 27 (Jan. 2001), pp. 415–444. DOI: 10.3410/f.725356294.793504070. URL: http://www.leonidzhukov.net/hse/2017/networkscience/papers/McPherson_HomophilyInSocialNe pdf.

[58] Elchanan Mossel, Joe Neeman, and Allan Sly. *A Proof Of The Block Model Threshold Conjecture*. 2018. arXiv: 1311.4115 [math.PR].

[59] M. E. J. Newman. "Finding community structure in networks using the eigenvectors of matrices". In: *Physical Review E* 74.3 (Oct. 2006). ISSN: 1550-2376. DOI: 10.1103/physreve.74.036104. arXiv: physics/0605087 [physics.data-an].

[60] M. E. J. Newman. "Modularity and Community Structure in Networks". In: *Proceedings of the National Academy of Sciences* 103.23 (2006), pp. 8577–8582. DOI: 10.1073/pnas.0601602103.

[61] M. E. J. Newman and M. Girvan. "Finding and evaluating community structure in networks". In: *Physical Review E* 69.2 (Feb. 2004). ISSN: 1550-2376. DOI: 10.1103/physreve.69.026113. arXiv: cond-mat/0308217 [cond-mat.stat-mech].

[62] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 2 (2002), pp. 849–856. URL: https://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.

[63] Gergely Palla et al. "Uncovering the overlapping community structure of complex networks in nature and society". In: *Nature* 435.7043 (June 2005), pp. 814–818. ISSN: 1476-4687. DOI: 10.1038/nature03607.

[64] Tiago P. Peixoto. "Parsimonious Module Inference in Large Networks". In: *Physical Review Letters* 110.14 (Apr. 2013). ISSN: 1079-7114. DOI: 10.1103/physrevlett.110.148701. arXiv: 1212.4794 [physics.data-an].

[65] Tiago P. Peixoto. "Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models". In: *Physical Review E* 89.1 (Jan. 2014). ISSN: 1550-2376. DOI: 10.1103/physreve.89.012804. arXiv: 1310.4378 [physics.data-an].

[66] Tiago P. Peixoto. "Hierarchical Block Structures and High-Resolution Model Selection in Large Networks". In: *Physical Review X* 4.1 (Mar. 2014). ISSN: 2160-3308. DOI: 10.1103/physrevx.4.011047. arXiv: 1310.4377 [physics.data-an].

[67] Tiago P. Peixoto. "The graph-tool python library". In: *figshare* (2014). DOI: 10.6084/m9.figshare.1164194. URL: http://figshare.com/articles/graph_tool/1164194 (visited on 09/10/2014).

[68] Tiago P. Peixoto. "Bayesian Stochastic Blockmodeling". In: *Advances in Network Clustering and Blockmodeling* (Nov. 2019), pp. 289–332. DOI: 10.1002/9781119483298.ch11. arXiv: 1705.10225 [stat.ML].

[69] Tiago P. Peixoto. "Merge-split Markov chain Monte Carlo for community detection". In: *Physical Review E* 102.1 (July 2020). ISSN: 2470-0053. DOI: 10.1103/physreve.102.012305. arXiv: 2003.07070 [physics.soc-ph].

[70] Tiago P. Peixoto. "Revealing consensus and dissensus between network partitions". In: *Physical Review X* 11.2 (Apr. 2021). ISSN: 2160-3308. DOI: 10.1103/physrevx.11.021003. arXiv: 2005.13977 [physics.soc-ph].

[71] Tiago P. Peixoto. *Inferring modular network structure*. URL: https://graph-tool.skewed.de/static/doc/demos/inference/inference.html.

[72] Yuval Peres and Perla Sousi. "Mixing times are hitting times of large sets". In: *Journal of Theoretical Probability* 28.2 (2015), pp. 488–519. arXiv: 1108.0133 [math.PR].

[73] Pascal Pons and Matthieu Latapy. "Computing Communities in Large Networks Using Random Walks". In: *Computer and Information Sciences - ISCIS 2005*. Springer Berlin Heidelberg, 2005, pp. 284–293. ISBN: 978-3-540-32085-2. arXiv: physics/0512106 [physics.soc-ph].

[74] Valérie Poulin and François Théberge. "Ensemble Clustering for Graphs". In: *Complex Networks and Their Applications VII* (Dec. 2018), pp. 231–243. ISSN: 1860-9503. DOI: 10.1007/978-3-030-05411-3_19. arXiv: 1809.05578 [cs.LG].

[75] Valérie Poulin and François Théberge. "Ensemble Clustering for Graphs: Comparisons and Applications". In: *Applied Network Science* 4.1 (July 2019). ISSN: 2364-8228. DOI: 10.1007/s41109-019-0162-z. arXiv: 1903.08012 [cs.LG].

[76] Valérie Poulin and François Théberge. "Comparing Graph Clusterings: Set partition measures vs. Graph-aware measures". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020). ISSN: 1939-3539. DOI: 10.1109/tpami.2020.3009862. arXiv: 1806.11494 [cs.LG].

[77] Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. "Typology of phase transitions in Bayesian inference problems". In: *Physical Review E* 99.4 (Apr. 2019). ISSN: 2470-0053. DOI: 10.1103/physreve.99.042109. arXiv: 1806.11013 [cond-mat.dis-nn].

[78] Garry Robins and Martina Morris. "Advances in exponential random graph (p*) models". In: *Social Networks* 29.2 (2007), pp. 169–172. ISSN: 0378-8733. DOI: 10.1016/j.socnet.2006.08.004.

[79] Ilias Sarantopoulos and Giulio Rossetti. *Omega Index for evaluation of overlapping community structure*. 2021. URL: https://github.com/isaranto/omega_index.

[80] City of Seattle. *City of Seattle Staff Directory*. URL: https://www.seattle.gov/directory (visited on 07/23/2020).

[81] Seattle Information Technology. *July 2021 City of Seattle Employee Directory Snapshot*. Aug. 2021. URL: https://www.seattle.gov/Documents/Departments/EmployeeDirectory2021-07.pdf.

[82] Qawi K. Telesford et al. "The ubiquity of small-world networks". In: *Brain Connectivity* 1.5 (2011), pp. 367–375. DOI: 10.1089/brain.2011.0038.

[83] G. M. Torrie and J. P. Valleau. "Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling". In: *Journal of Computational Physics* 23.2 (1977), pp. 187–199. ISSN: 0021-9991. DOI: 10.1016/0021-9991(77)90121-8.

[84] Ulrike Von Luxburg. "A tutorial on spectral clustering". In: *Statistics and computing* 17.4 (2007), pp. 395–416. arXiv: 0711.0189 [cs.DS].

[85] Fugao Wang and D. P. Landau. "Efficient, Multiple-Range Random Walk Algorithm to Calculate the Density of States". In: *Physical Review Letters* 86.10 (Mar. 2001), pp. 2050–2053. ISSN: 1079-7114. DOI: 10.1103/PhysRevLett.86.2050. arXiv: cond-mat/0011174 [cond-mat.stat-mech].

[86] Duncan J. Watts and Steven H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (1998), pp. 440–442. DOI: 10.1038/30918.

[87] Eric W. Weisstein. *Graph Sum*. URL: https://mathworld.wolfram.com/GraphSum.html.

[88] Wu, F. and Huberman, B. A. "Finding communities in linear time: a physics approach". In: *The European Physical Journal B* 38.2 (2004), pp. 331–338. DOI: 10.1140/epjb/e2004-00125-x.

[89] Zhao Yang, René Algesheimer, and Claudio J. Tessone. "A Comparative Analysis of Community Detection Algorithms on Artificial Networks". In: *Scientific Reports* 6.1 (Aug. 2016), pp. 1–18. ISSN: 2045-2322. DOI: 10.1038/srep30750. arXiv: 1608.00763 [physics.soc-ph].

[90] Yi Yu, Tengyao Wang, and Richard J. Samworth. "A useful variant of the Davis–Kahan theorem for statisticians". In: 2014. arXiv: 1405.0680 [math.ST].

[91] Wayne W. Zachary. "An information flow model for conflict and fission in small groups". In: *Journal of anthropological research* 33.4 (1977), pp. 452–473. URL: http://www1.ind.ku.dk/complexLearning/zachary1977.pdf.

[92] Lenka Zdeborová and Florent Krzakala. "Statistical physics of inference: thresholds and algorithms". In: *Advances in Physics* 65.5 (Aug. 2016), pp. 453–552. ISSN: 1460-6976. DOI: 10.1080/00018732.2016.1211393. arXiv: 1511.02476 [cond-mat.stat-mech].

[93] Lihi Zelnik-Manor and Pietro Perona. "Self-Tuning Spectral Clustering". In: *Advances in Neural Information Processing Systems* 17 (Jan. 2004), pp. 1601–1608. URL: https://papers.nips.cc/paper/2619-self-tuning-spectral-clustering.