Graph Clustering With Constraints

Matthew Werenski

Fourth Year Project Report

School of Informatics

University of Edinburgh

2019

Abstract

This thesis introduces a new, probabilistic, notion of constrained conductance and explores it in both theory and practice. The probabilistic notion works using soft partitions, which we call probability partitions, of the vertices, allowing for each vertex to partially belong to each cluster. We successfully demonstrate that when clustering a graph into two clusters that the constrained probabilistic conductance is precisely equal to the standard constrained conductance. We also provide the HardCluster algorithm which takes any 2-way probabilistic partition and converts it into a clustering without increasing the conductance. We also empirically demonstrate that this method is competitive in practice with existing methods. We include experiments in image segmentation and cluster recovery, the latter of which includes interesting results about using randomly generated constraints.

Acknowledgements

I would like to acknowledge my advisor Mihai Cucuringu for sharing with me experiments from his work and He Sun for the many useful suggestions, edits, and discussions over the past year.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Matthew Werenski*)

Table of Contents

Chapter 1

Introduction

Graph clustering is a useful tool in many fields. It has applications in image segmentation [Wu and Leahy, 1993], language processing [Biemann, 2006], and biology [Aittokallio and Schwikowski, 2006] to name a few. Traditionally, clustering is done on a single graph with edges indicate that vertices should be clustered together. In this thesis we look both at this and the constrained case, in which there is a second graph which explicitly encodes information about which vertices should not be kept together.

Standard clustering can often be adapted to a constrained clustering by incorporating expert knowledge or labeled data. Examples of its usage include GPS lane finding [Wagstaff et al., 2001], video object recognition [Yan et al., 2006], and data mining [Strehl and Ghosh, 2003]. As datasets become larger and more complex, it is reasonable to expect that labeled data will become more readily available and methods that can take advantage of it will flourish.

In this thesis we explore constrained spectral methods for cut based clustering. These methods have become popular in the unconstrained setting largely due to the ease of implementation and the power they possess. None the less, in the constrained setting spectral methods are still effective and applicable.

Our original work consists of introducing the notion of probabilistic conductance and analyzing its relation to the standard conductance. This enables the use of mixture models for clustering on a spectral embedding. To compliment this we give two algorithms for converting the soft clusterings to hard clusterings.

1.1 Basics in Graph Clustering

The basic object of this thesis is the undirected graph. Any graph can be thought of as a set of vertices, and a set of edges which connect two vertices. What makes a graph undirected is the notion that edges go both ways, that is to say if vertex v is connected to vertex u by an edge, then u is connected to v by that same edge. Graphs which allow for the connection to be in only one direction are known as directed graphs but we shall not be discussing directed graphs in this thesis, and as such whenever we say a "graph" we shall mean an undirected graph.

In the real world there are many examples of graphs. For example, the network of roads can be thought of as a graph with the intersections being vertices and the roads between them the edges. Or in social circles where people are the vertices and friendships act as the edges. Using graph theory we can mathematically answer questions like "What roads are most important to traffic?" and "Which people make up real friend groups?" For an in depth look at the applications of graph theory see [Gross and Yellen, 2005].

We denote a graph as $G = (V, E)$. It is common to introduce $n := |V|$ to mean the number of vertices in a graph. We shall write $\{u, v\} \in E$ to denote that vertices $u, v \in V$ share an edge in E, and because we are using undirected graphs $\{u, v\}$ refers to the same edge is $\{v, u\}$.

Graphs may also be weighted which means that they come with a weight function $w : V \times V \to \mathbb{R}$. A weighted graph $G = (V, E, w)$ has the properties that for $\{u, v\} \in E$, $w(u, v) > 0$ and for any pair $\{u', v'\} \notin E$, $w(u', v') = 0$. In general, an unweighted graph is a special case of a weighted graph with the condition that $w(u, v) = 1$ for every $\{u, v\} \in E$.

1.1.1 Properties of Graphs

There are a few basic concepts that are required throughout this thesis. For a graph $G = (V, E, w)$ the degree of a vertex $v \in V$ is defined $d(v) := \sum_{\{u,v\} \in E} w(u, v)$. This can be roughly thought of how strongly connected an individual vertex is to the rest of the graph. The volume of a subset of the vertices $S \subset V$ is defined $\mathrm{vol}(S) := \sum_{v \in S} d(v)$.

For the disjoint subsets $S, T \subset V$, the cut cost between them is defined

$$w(S, T) := \sum_{u \in S, v \in T} w(u, v).$$

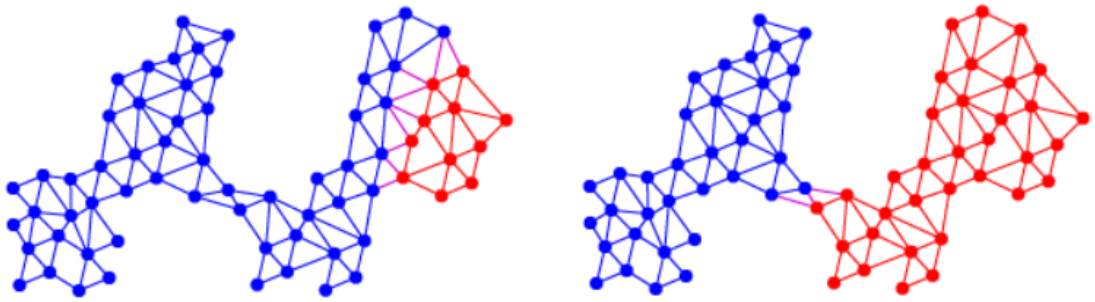


Figure 1.1: Two examples of cuts on the same graph. The left cut having a higher cost due to the greater number of edges broken compared to the right cut. Image from [Nahshon, 2018]

The cut cost between two vertex sets is an indicator of how strongly the two sets are connected to each other. In figure 1.1 two possible cuts on the same graph are provided. When clustering a graph, the cut cost is a commonly used metric of the quality of two clusters.

For convenience we write the compliment of a vertex set $S \subset V$ as $\bar{S} = V \backslash S$. Similarly we shall use ∂S to represent the set of edges with one endpoint in S and the other endpoint in $\bar{S}$.

The edge expansion problem is to find the cheapest cut of the graph that disconnects a relatively large number of vertices. Solving this problem is the objective of many graph clustering techniques [Brandes et al., 2003]. This problem equivalent to minimizing the conductance of the graph. The conductance of a subset of the vertices $S \subset V$ of a graph $G = (V, E, w)$ is defined

$$\phi_G(S) := \frac{w(S, \bar{S})}{\min\{\mathrm{vol}(S), \mathrm{vol}(\bar{S})\}} \tag{1.1}$$

and is a measure of how well a subset of the vertices is connected to a graph, relative to its volume. The conductance of the graph is then defined as

$$\phi_G := \min_{S \subset V} \phi_G(S). \tag{1.2}$$

The conductance of a graph tells us about how well connected the entirety of the graph is. For the conductance to be high means that there are no parts of the graph that can be easily removed without breaking a high number of edges. Conversely, a low conductance indicates that at least one part of the graph can be easily taken off. However, finding the subset that achieves the optimal value is known to be an NP-hard problem [Garey and Johnson, 1990].

It is important at this point to briefly formalize exactly what it means to "cluster a graph". To cluster a graph is really to form partitions of the vertices such that the resulting partition achieves some clustering criteria. A clustering is just another way of referring to the partition, and one portion of the partition is just called a cluster.

For us the criteria will be to have a low conductance, so ultimately what we aim to do when "clustering a graph" is to find a partition of the vertices that achieves a low conductance. As touched on above, the reason that a low conductance is the goal is because a clustering that achieves a low conductance will have clusters with two attractive properties. Firstly, in a cluster the sum of the edge weights between vertices within the cluster will be high. Secondly, the sum of the edge weights between vertices in the cluster and vertices not in the cluster will be low. Thus, a low conductance cluster will be one that satisfies two of the intuitive notions of a good clustering.

The conductance in the sense above is concerned with the ability to split a graph into two pieces. There is a k-way sense as well. The k-way conductance of a graph is given in a parallel way to the 2-way case. That is to say

$$\phi_G^k := \min_{S_1,\ldots,S_k} \max_{1\leq i\leq k} \phi_G(S_i) \tag{1.3}$$

where $S_1,\ldots,S_k$ is a valid partition of the vertices of the graph. One can observe that the definition given for the conductance of a graph when cutting it into two pieces is precisely a special case for $k = 2$.

1.1.2 Using Constraints

The main focus of the thesis is clustering with constraints which is in some ways actually a very similar task to standard clustering. This task involves working with two graphs that share a vertex set. Formally, we have two graphs $G = (V, E_G, w_G), H = (V, E_H, w_H)$, where for one graph, G, w_G indicates how strongly believed it is that two vertices should be kept together, and in the other graph, H, w_H indicates how strongly believed two vertices should be kept apart. In the literature these graphs are usually referred to as ML and CL graphs, ML standing for "Must Link" and CL standing for "Cannot Link" [Basu et al., 2008].

The conductance above is not a perfect one-to-one with constrained conductance but it does provide a good starting point. In this setting the constrained conductance, which will also be referred to as just the conductance, of a subset of the vertices $S \subset V$ is given by

$$\phi_{G,H}(S) := \frac{w_G(S,\bar{S})}{w_H(S,\bar{S})}. \tag{1.4}$$

In this equation we have borrowed w_G and w_H which define the cut costs on G and H respectively. A set S that achieves a low $\phi_{G,H}(S)$ will usually be one that breaks many of the edges in H while at the same time preserving as many edges as possible in G. In parallel to the conductance of a graph, the conductance of a pair of graphs is defined

$$\phi_{G,H} = \min_{S \subset V} \phi_{G,H}(S). \tag{1.5}$$

The main difference between constrained conductance and the standard conductance is that in the constrained case a clustering achieves a low conductance by breaking a lot of edges in the CL-graph, while in the standard case a clustering achieves a low conductance by having large clusters. The k-way constrained conductance also mirrors the standard version and is defined

$$\phi^k_{G,H} := \min_{S_1,\ldots,S_k} \max_{1 \le i \le k} \phi_{G,H}(S_i) \tag{1.6}$$

where $S_1, \ldots, S_k$ is a valid partition of the vertices of the graph.

1.2 Prior Work

The field of graph clustering has been deeply studied and there exists in the literature many interesting theoretical and empirical results in both the constrained and unconstrained setting.

In practice, two of the most common spectral algorithms for graph clustering are those in [Shi and Malik, 2000] and [Ng et al., 2001], the former uses a recursive clustering algorithm to divide a graph in two and then to divide the subgraphs, the latter proposes using the spectral embedding and then to cluster the points using the *k-means* algorithm. In [Peng et al., 2017] the algorithm which first embeds then clusters with *k-means* is analyzed.

In [Lee et al., 2014] some of the strongest theorems relating k-way conductance to the eigenvalues are demonstrated including that

$$\frac{\lambda_k}{2} \le \phi^k_G \le O(k^3)\sqrt{\lambda_k}$$

Studies involving the use of constraints are also bountiful [Basu et al., 2008]. One of the earliest works in this domain is [Wagstaff et al., 2001]. In this paper they introduce the notion of CL and ML graphs and use them to design a modified version

of k-means which incorporates the constraints. Another paper which is of particular interest to us is [Cucuringu et al., 2016]. in which they prove the following Cheeger inequality for a pair of graphs.

Theorem 1.1 ([Cucuringu et al., 2016])**.** Let G and H be any two weighted graphs and d be the vector containing the degrees of the vertices in G. For any vector x such that $x^\intercal d = 0$, we have

$$\frac{x^\intercal L_G x}{x^\intercal L_H x} \geq \phi_{G,D_G}\phi_{G,H}/4$$

where D_G is the demand graph of G whose adjacency matrix is defined $A_{ij} = d_i d_j / vol(V)$. A cut meeting the guarantee of the inequality can be obtained via a Cheeger sweep on x.

In the same paper they discuss using an embedding which is constructed from the solutions to the generalized eigenvalue problems $L_G x = \lambda L_H x$. This is an idea that we will use later in our experiments chapter.

Another work in constrained clustering that is of interest is [Lu and Carreira-Perpinan, 2008] in which affinity propagation is used as a way of incorporating constraints into the data matrix before a non-constrained spectral clustering algorithm is applied. There are many papers in which similar ideas are used to incorporate constraints before the application of an algorithm in place of designing an algorithm to use the constraints (See for example [Kamvar et al., 2003]).

There is also related work in clustering signed graphs [Harary, 1953]. These works explore graphs in which edge weights may be either positive or negative instead of just positive. The objective of a clustering being to have mostly positive edges within a cluster while simultaneously having mostly negative edges across a cluster. There are clear parallels between signed graphs and working with ML and CL graphs. In [Kunegis et al., 2010] it is demonstrated that spectral methods can be applied to signed graphs by using a signed Laplacian. Further analysis of the ability to cluster signed graphs and an application of generalized eigenproblems can be found in [Cucuringu et al., 2019].

In our experimental chapter we look at two uses, stochastic block models and image segmentation. In our work on stochastic block models we use the planted partition model from [Condon and Karp, 2001] to generate random graphs and partition them. The primary objective we are concerned with is cluster recovery. Spectral methods for this task on random graphs of this type are explored in [Rohe et al., 2011]. We perform

further experiments which utilize pairs of graphs in multiple different ways. For an in depth look at the basics of stochastic block models as well as the current state we refer the reader to [Abbe, 2017].

Image segmentation with graph based methods is another rich area of study. A particularly popular method can be found in [Shi and Malik, 2000]. Other works in the area include [Felzenszwalb and Huttenlocher, 2004] and [Hung and Ho, 1999]. In general, most of these methods involve characterizing the pixels of an image as vertices in the graph and weight the edges between vertices using the pixel similarity of neighboring pixels. Using the generated graph cut based methods are applied and segments are formed. The rough idea being that the distinct regions of an image will correspond to weakly connected parts of a graph, and a good cut of the graph will create a good segmentation. It would be remiss not to mention LOBPCG [Knyazev, 2001] which is what enables the calculation of the Laplacians eigenvectors of extremely large and sparse graphs. In our work on image segmentation we largely follow in the footsteps of [Cucuringu et al., 2016]. For yet another survey of the field we direct the reader to [Peng et al., 2013].

1.3 Organization of the Thesis

The structure of this work is as follows. First we provide this introduction which includes a refresher on the background knowledge required by the reader as well as a look at related works in the field. After this we proceed into a look at the essentials in spectral clustering which will be of use to those not well versed in the practice, as well as throughout the paper. In the third chapter we look deeply at the concept of probabilistic conductance; we set up the necessary definitions, followed by a series of lemmas which build up to one theorem which demonstrates the equality of probabilistic conductance and the standard conductance. Using the details of the proof we present the HardCluster algorithm. To conclude the chapter we provide a counter example to the generalization of the theorem to k-way conductances and propose the MultiHardCluster algorithm which minimizes aims to convert a k-way probability into a hard clustering.

In the chapter four we look at two applications of constrained clustering, the first being cluster recovery with stochastic block models and the second being image segmentation. The experiments in cluster recovery yield interesting experimental results on their own. In the image segmentation setting we demonstrate that the work we

propose is competitive with existing methods. After the experiments we conclude the thesis with a discussion of the work done as well as provide questions which future work could address.

Chapter 2

Basics in Spectral Clustering

2.1 Graph Matrices

There are many ways of representing a graph. Typically each vertex has an identifier, which in this thesis will be given by a subscript index from 1 to n, for example v_{10} would be the tenth vertex. The indices are arbitrary but useful for keeping track of vertices especially when iterating over them. With the indices of the vertices established, it remains to represent the edges. Typical representations include edge lists, adjacency lists, and adjacency matrices which will be of interest to us.

The adjacency matrix $A \in \mathbb{R}^{n\times n}$ of a graph $G = (V,E,w)$ is defined $A_{ij} = w(v_i, v_j)$. Recall that for an edge $\{v_i, v_j\} \notin E$ we define $w(v_i, v_j) = 0$. Intuitively, the ijth entry to A is just the weight of the edge connecting v_i to v_j. The degree matrix $D \in \mathbb{R}^{n\times n}$ of a graph is defined $D_{ii} = d(v_i)$ with the off-diagonal elements being zero.

The Laplacian L is defined $L := D - A$. This is usually referred to as the unnormalized Laplacian. There are also a few normalized versions which are commonly studied (see [Chung, 1997]). These include the symmetric Laplacian and the random walk Laplacian [von Luxburg, 2007], both of which have properties similar to the unnormalized Laplacian. Throughout this thesis when we say just "Laplacian" it shall refer to the unnormalized Laplacian.

Graph Laplacians are well understood and thoroughly researched objects. For an in depth look at them see [Belkin and Niyogi, 2002] and [Nascimento and de Carvalho, 2011]. It is worth quickly observing a few basic properties that they have.

Lemma 2.1. (Properties of the Laplacian) The Laplacian of a graph $G = (V,E,w)$ has the following properties:

1. for every vector $x \in \mathbb{R}^n$ we have

$$x^\intercal Lx = \sum_{\{v_i,v_j\}\in E} w(v_i,v_j)(x_i - x_j)^2.$$

2. L is symmetric and positive semi-definite.

3. the smallest eigenvalue of L is 0 and the corresponding eigenvector is the constant vector.

4. L has n non-negative real eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$.

The proof that L has these properties is well known so we shall not re-create it here, instead we refer the reader to [von Luxburg, 2007]. Since the Laplacian is a real symmetric matrix the following theorem about all real symmetric matrices applies to it.

Theorem 2.2 (Spectral Theorem [Dunford and Schwartz, 1957])**.** Suppose $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix. Then

1. Every eigenvalue of A is real and the eigenvectors of A are also real.

2. The eigenvectors of distinct eigenvalues are all orthogonal.

3. There exists a diagonal matrix $D \in R^{n\times n}$ and orthogonal matrix $U \in \mathbb{R}^{n\times n}$ such that $A = UDU^\intercal$. The entries of D are the eigenvalues of A and the columns of U are the corresponding eigenvectors of A.

This is a classic result (originally shown by Cauchy) and as such we shall not go into its proof. The primary takes are that the eigenvalues are real, there are n distinct eigenvectors (this is a corollary of the third property), and that they are orthogonal.

By the properties of the Laplacian and an application of the Spectral Theorem we can show tht the number of connected components is the same as the number of zero eigenvalues; which is formalized below.

Lemma 2.3. Let L be the Laplacian of $G = (V,E,w)$. Let $0 = \lambda_1 \leq \lambda_1 \leq ... \leq \lambda_n$ be the eigenvalues of L. Then $\lambda_k = 0$ if and only if G has at least k connected components.

When we define the Laplacian of a graph we need to give each vertex of the graph an index so that way we know which entries in the Laplacian correspond to which

vertices. The indices that we assign are completely arbitrary so the Laplacian is only unique up to a permutation of the indices. Therefore, we can permute the vertices as we see fit and still have an equivalent Laplacian.

In the case where the graph has two or more connected components, it is possible to create a perfectly blocked Laplacian. In fact, when there are two connected components C_1 and C_2 the Laplacian L_G of the graph G can be written as

$$L_G = \begin{bmatrix} L_{C_1} & 0 \\ 0 & L_{C_2} \end{bmatrix},$$

where L_{C_1} and L_{C_2} denote the Laplacians of C_1 and C_2 respectively and the 0's indicate that all the other entries are just zero.

The fact that this is possible is from the definition of a connected component. Since the vertices were sorted by their components, and there are no edges that cross between components we immediately get that the entries outside the component Laplacians must be zero. Similarly, justifying that we can use the two components' Laplacians is easy to do from the definition of the Laplacian and the fact that there are no edges between them.

If the graph is connected it is still interesting to see what happens when a of low conductance cut is found and the vertices are permuted so that all the vertices on one side of the cut are the first entries, followed by all the vertices on the other side of the cut.

From figure 2.1 we can see that when there is a low conductance cut, the Laplacian permuted to conform to that cut achieves a very clear two block structure, like an imperfect version of the two connected component case. This is to be expected since any entry that lies outside of the two blocks is an edge that crosses the cut between them, and a low conductance cut will minimize these.

There is a fundamental difference between cutting a disconnected graph compared to cutting a connected graph. In the disconnected case, if S is a connected component then $w(S,\bar{S}) = 0$ and therefore regardless of $\mathrm{vol}(S)$ and $\mathrm{vol}(\bar{S})$ the conductance will be 0. If the graph is connected then inevitably there will be at least one edge between S and $\bar{S}$ which will be broken so $w(S,\bar{S}) > 0$. In this case, the volumes do matter towards the conductance. In terms of trying to create two blocks, this adds some weight to having both sides of the cut have larger volume at the cost of having a less "blocky" Laplacian. That is to say, there may be cuts which have a lower conductance than cuts which come closer to a perfect block matrix. When thinking of cuts as re-arranging

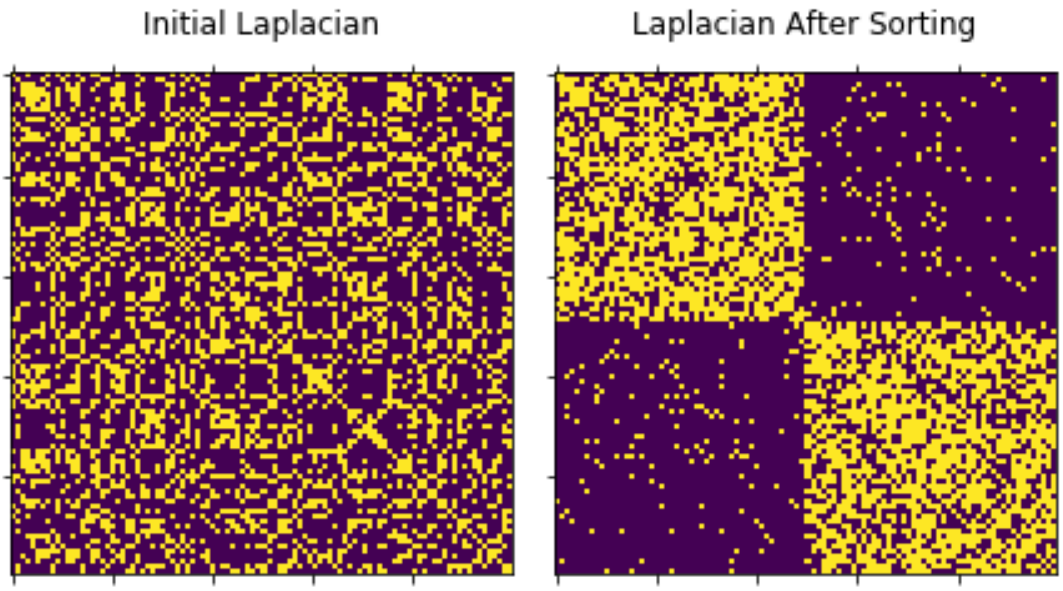


Figure 2.1: Two Laplacians for the same graph. On the left is the initial Laplacian who's vertices have been randomly indexed. On the right is the Laplacian which comes from computing a low conductance cut and ordering the vertices based on the side of the cut.

the Laplacian to create a block matrix, minimizing the conductance yields a cut which has a clear block structure and the two blocks are similar in size.

2.2 Relating Conductance and Laplacians

Throughout this section we will be assuming that we have a graph $G = (V, E, w)$ and that it has Laplacian L. From Lemma 2.1 we have that for any vector $x \in \mathbb{R}^n$ that $x^\intercal Lx = \sum_{\{v_i,v_j\}\in E} w(v_i, v_j)(x_i - x_j)^2$. Now consider the case that x is an indicator vector of $S \subset V$, defined by

$$x_i = \begin{cases} 1 & v_i \in S, \\ 0 & v_i \notin S. \end{cases}$$

There are a few nice properties that such a vector has. Firstly $(x_i - x_j)^2$ will be 0 if v_i and v_j are either both in S or both not in S. The expression will be 1 if v_i is in S and v_j is not or vice versa. This has the effect that when summing over the edges any edge that is not broken by the cut creating S is not counted, and the resulting sum is just the total weight of the edges that are broken. Therefore we get that $w(S, \bar{S}) = x^\intercal Lx$.

Continuing with standard conductance, if D is the degree matrix of G then the volume of S can be expressed as $x^\intercal Dx$. This nicely ties together the conductance with

the Laplacian with the following expression

$$\phi_G(S) = \frac{x^\intercal Lx}{x^\intercal Dx}$$

still assuming that x is the indicator vector for S.

It is common to see the normalized Laplacian $\mathcal{L} = \mathbb{I} - D^{-1/2}AD^{-1/2}$ being used in the literature because it has the property that for $y = D^{1/2}x$ the conductance can be written as

$$\phi_G(S) = \frac{y^\intercal \mathcal{L} y}{y^\intercal y}$$

which drops the D in the denominator. This is known as the Rayleigh quotient of $\mathcal{L}$. The Rayleigh quotient is used in many eigenvalue characterizations. In particular, it is used in the Rayleigh quotient theorem and the Courant-Fisher theorem which are both included in the following theorem.

Theorem 2.4. Let A be an $n \times n$ real symmetric matrix with eigenvalues $\lambda_1 \leq ... \leq \lambda_n$ and corresponding eigenvectors $v_1, ..., v_n$. Then

$$\lambda_i = \min_{S:\dim S=i} \max_{x \in S\setminus\{0\}} \frac{x^\intercal Ax}{x^\intercal x} = \max_{S:\dim S=n-i+1} \min_{x \in S\setminus\{0\}} \frac{x^\intercal Ax}{x^\intercal x}$$

and

$$\lambda_i = \min_{\substack{x \in \mathbb{R}^n\setminus\{0\} \\ x \perp v_1,...,v_{i-1}}} \frac{x^\intercal Ax}{x^\intercal x} = \max_{\substack{x \in \mathbb{R}^n\setminus\{0\} \\ x \perp v_{i+1},...,v_n}} \frac{x^\intercal Ax}{x^\intercal x}.$$

In the latter equation the minimizer (maximizer) is achieved with v_i.

This theorem is used in [Lee et al., 2014] to show the left-hand side of the higher-order Cheeger inequality. It also tells us that the k linearly independent vectors that form a basis of the minimizing subspace are the bottom k eigenvectors. Since the Laplacian is a symmetric matrix the theorem applies to it and it informs us that looking for a partition with indicator vectors that approximate the bottom eigenvectors will yield a low conductance.

Also in [Lee et al., 2014] they show that the eigenvectors can be used to create an isotropic mapping. This is achieved by the embedding function $F : V \to \mathbb{R}^k$ defined $F(v) = (x_1(v), x_2(v), ..., x_k(v))$ where $x_1, ..., x_k$ are the k bottom eigenvectors and $x_i(v)$ is the component of x_i that corresponds to v. From this embedding reasonable clusters can be achieved using k-means [Peng et al., 2017].

To sum up, the bottom eigenvectors of the Laplacian can be used to embed the vertices into $\mathbb{R}^k$. Using this embedding the clusters can be created using any geometric clustering algorithm, classically k-means is used.

A famous technique for finding a subset that approximates the 2-way conductance is known as the Cheeger sweep (TODO cite) and it comes with the following theorem.

Theorem 2.5 (Cheeger's Inequality [Chung, 1997])**.** Let $\mathcal{L}$ be the normalized Laplacain of an undirected weighted graph $G = (V, E, w)$. Let λ_2 be the second smallest eigenvalue of $\mathcal{L}$. It holds that

$$\frac{\lambda_2}{2} \leq \phi_G \leq \sqrt{2\lambda_2}.$$

The proof also tells us how to find a subset that achieves a conductance less than $\sqrt{2\lambda_2}$. Roughly speaking, all that one needs to do is to calculate the second eigenvector of the Laplacian, sort the entries of it, and then perform a "sweep" over them. This idea will be further explored in the next chapter.

2.3 Constrained Setting

In the prior section we touch on how the Laplacian of a graph can connect a subset of the vertices to the conductance. In the constrained setting it is not so different. Assume that we have two graphs $G = (V, E_G, w_G)$ and $H = (V, E_H, w_H)$ with Laplacians L_G and L_H respectively. the constrained conductance for a subset $S \subset V$ with indicator vector x is given by

$$\phi_{G,H}(S) = \frac{x^\intercal L_G x}{x^\intercal L_H x}$$

due to the same argument about cut cost as the previous section.

We are looking to find k disjoint subsets of vertices $S_1, ..., S_k \subset V$ which partition V while simultaneously achieving a low conductance. We can rephrase this to looking for k disjoint vectors $x_1, ..., x_k \in \{0,1\}^n$ such that $\sum_{i=1}^k x_i = \mathbb{1}$ and $\frac{x_i^\intercal L_G x_i}{x_i^\intercal L_H x_i}$ is low for all i. The spectral relaxation is to remove the constraint that $\sum_{i=1}^k x_i = \mathbb{1}$ in place of imposing that the k vectors are linearly independent.

One idea to achieve this relaxed version comes from generalized eigenproblems. The generalized eigenproblem is to solve equation of the form

$$Ax = \lambda Bx$$

where $A, B \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$. The observation that can be made is that if x is a generalized eigenvector corresponding to λ then

$$\frac{x^\intercal Ax}{x^\intercal Bx} = \frac{x^\intercal(\lambda Bx)}{x^\intercal Bx} = \lambda$$

using that $Ax = \lambda Bx$ since it is a generalized eigenvector. Applying this fact with L_G and L_H replacing A and B respectively gives the intuitive idea of using the eigenvectors that correspond to the low eigenvalues of $L_G x = \lambda L_H x$. In fact, it is well known that the bottom k eigenvectors in this setting also provide a basis that minimizes the constrained version of the Rayleigh quotient.

Chapter 3

Probabilistic Conductance

In this chapter we introduce a new probabilistic version of the conductance of a pair of graphs. We begin first by providing the necessary definitions and then demonstrate that in the 2-way case it is equal to the standard definition of conductance. After that we discuss the k-way case and present two algorithms for converting probabilistic partitions to clusterings.

3.1 Generalizing Conductance

A standard two way partition of a set assigns explicitly each element of the set to one side of a partition. A probabilistic partition on the other hand assigns a probability for each element to be on each side of the partition, with the condition that for each element the sum of the probabilities of its assignment is exactly one. For example, a 10%-90% probability assignment is valid because they sum to 1, but a 30%-50% probability assignment is invalid because their sum is not 1.

This notion can be seen as a generalization of partitioning a set since a standard partition is just a probabilistic partition which assigns 100% of the probability to one side and 0% to the other. We formally define it as follows.

Definition 3.1.1. A k-way probabilistic partition of a set A is a set of k probability assignments $P = \{P_i : 1 \leq i \leq k\}$ such that $0 \leq P_i(a) \leq 1$ and

$$\sum_{i=1}^{k} P_j(a) = 1 \tag{3.1}$$

for all $a \in A$. We shall write $\bar{P}_i$ to be the complement of P_i. That is $\bar{P}_i(a) = 1 - P_i(a)$ for every $a \in A$.

Using a probabilistic partition we can take the cut cost of a graph and generalize it to a probabilistic cut.

Definition 3.1.2. Let $G = (V, E, w)$ and P be a probabilistic partition of V. Let $P_i \neq P_j \in P$. The probabilistic cut cost is given by

$$w(P_i, P_j) := \sum_{\{u,v\} \in E} w(u,v)(P_i(u)P_j(v) + P_i(v)P_j(u)). \tag{3.2}$$

For convenience we introduce the notation

$$P_{ij}(u,v) := P_i(u)P_j(v) + P_i(v)P_j(u). \tag{3.3}$$

The definition $P_{ij}(u,v)$ can be interpreted as the probability that the edge connecting u and v is broken by a clustering made by randomly assigning vertices to clusters using the distributions from P. From this it is clear to see that $P_{ij}(u,v) = P_{ji}(u,v)$ and that $P_{ij}(u,v) = P_{ij}(v,u)$. These are all a direct consequence of the symmetry of $P_{ij}(u,v)$. To make use of this notation we will often write $P = \{P_1, P_2\}$ and directly assume that $P_2 = \bar{P}_1$ and $P_1 = \bar{P}_2$. We now move on to a few useful lemmas for probabilistic partitions.

Lemma 3.1. Let $G = (V, E, w)$ and P be a probabilistic partition of V. For any $P_i \neq P_j \in P$

$$w(P_i, P_j) = w(P_j, P_i).$$

Proof. This is shown by a direct calculation.

$$w(P_i, P_j) = \sum_{\{u,v\} \in E} w(u,v)P_{ij}(u,v) = \sum_{\{u,v\} \in E} w(u,v)P_{ji}(u,v) = w(P_i, P_j)$$

where we have used that $P_{ij}(u,v) = P_{ji}(u,v)$ to go from the second to third expression. The final equality is just the definition of a probabilistic cut. □

Given the definition of a probabilistic cut, it is natural to extend the definition of conductance. We will only discuss this in the paired graph setting.

Definition 3.1.3. For a pair of graphs, $G = (V, E_G, w_G)$ and $H = (V, E_H, w_H)$, let $\{P, \bar{P}\}$ be a 2-way probabilistic partition of V. The probabilistic conductance of P is defined

$$\rho_{G,H}(P) := \frac{w_G(P, \bar{P})}{w_H(P, \bar{P})}. \tag{3.4}$$

Notice that in the definition we do not worry about $\rho_{G,H}(\bar{P})$. The reason that we do not specify it is because of the following lemma.

Lemma 3.2. Let $G = (V, E_G, w_G)$ and $H = (V, E_H, w_H)$ be two graphs over the same vertex set. Let $\{P, \bar{P}\}$ be a 2-way probabilistic partition of V. It holds that

$$\rho_{G,H}(P) = \rho_{G,H}(\bar{P}).$$

Proof. We can manipulate $\rho_{G,H}(P)$ as follows.

$$\rho_{G,H}(P) = \frac{w_G(P,\bar{P})}{w_H(P,\bar{P})} = \frac{w_G(\bar{P},P)}{w_H(\bar{P},P)} = \rho_{G,H}(\bar{P}).$$

Where we have used that the probabilistic cut cost is symmetric to go from the second to third expression. □

Now that we have the definition for the probabilistic conductance for a given partition, it is natural to extend the definition of the conductance of a graph. Similarly to above, the probabilistic conductance of a graph is defined

$$\rho_{G,H} := \min_{\{P_1,P_2\} \text{is a prob. part.}} \rho_{G,H}(P_1). \tag{3.5}$$

Using probabilistic partitions of the vertices allows for even more flexibility than a standard partition of the vertices. This raises the question, how does this flexibility impact the relation between $\rho_{G,H}$ and $\phi_{G,H}$ is?. We answer this question with the following theorem.

Theorem 3.3. Let $G = (V, E_G, w_G)$ and $H = (V, E_H, w_H)$ be two graphs over the same vertex set. It holds that

$$\rho_{G,H} = \phi_{G,H}. \tag{3.6}$$

Proof. By trichotomy, it is equivalent to show that neither $\phi_{G,H} > \rho_{G,H}$ nor $\phi_{G,H} < \rho_{G,H}$.

It is easy to show that it cannot be that $\rho_{G,H} > \phi_{G,H}$. Let $S, \bar{S}$ be the optimal partition of V. Let $\{P_1, P_2\}$ be the probabilistic partition of V defined as follows.

$$P_1(v) := \begin{cases} 1 & v \in S \\ 0 & v \in \bar{S} \end{cases}$$

and let $P_2 = \bar{P_1}$ be the compliment of P_1. Using these we have

$$\begin{aligned}
\phi_{G,H} &= \phi_{G,H}(S) \\
&= \frac{w_G(S,\bar{S})}{w_H(S,\bar{S})} \\
&= \frac{\sum_{\{u,v\}\in\partial S} w_G(u,v)}{\sum_{\{u,v\}\in\partial S} w_H(u,v)} \\
&= \frac{\sum_{\{u,v\}\in\partial S} w_G(u,v)(1\cdot 1+0\cdot 0)}{\sum_{\{u,v\}\in\partial S} w_H(u,v)(1\cdot 1+0\cdot 0)} \\
&= \frac{\sum_{\{u,v\}\in\partial S} w_G(u,v)(P_1(u)P_2(v)+P_1(v)P_2(u))}{\sum_{\{u,v\}\in\partial S} w_H(u,v)(P_1(u)P_2(v)+P_1(v)P_2(u))} \\
&= \frac{\sum_{\{u,v\}\in E_G} w_G(u,v)P_{12}(u,v)}{\sum_{\{u,v\}\in E_H} w_H(u,v)P_{12}(u,v)} \\
&= \frac{w_G(P_1,P_2)}{w_H(P_1,P_2)} = \rho_{G,H}(P_1) \geq \rho_{G,H}.
\end{aligned}$$

where in the sixth line we used that for any edge $\{u,v\}$ with both endpoints in either S or $\bar{S}$, the quantity $P_{12}(u,v) = 0$. This allows the sum to be unchanged when moving from summing over just broken edges to summing over all edges. Therefore, it cannot be that $\rho_{G,H} > \phi_{G,H}$ since we have constructed a probabilistic partition which achieves a probabilistic conduction exactly equal to the standard conductance.

We now proceed to the more difficult part, showing that it is not the case that $\rho_{G,H} < \phi_{G,H}$. Assume for contradiction that $\rho_{G,H} < \phi_{G,H}$, and let $P = \{P_1, P_2\}$ be the probabilistic partition that achieves the optimal $\rho_{G,H}$, that is $\rho_{G,H} = \rho_{G,H}(P_1)$. If there are no vertices such that $P_1(v) \in (0,1)$ then we can define the sets $S := \{v : P_1(v) = 1\}$ and $\bar{S} := \{v : P_2(v) = 1\}$. Using these sets we have

$$\begin{aligned}
\rho_{G,H} &= \rho_{G,H}(P_1) \\
&= \frac{w_G(P_1,P_2)}{w_H(P_1,P_2)} \\
&= \frac{\sum_{\{u,v\}\in E_G} w_G(u,v)P_{12}(u,v)}{\sum_{\{u,v\}\in E_H} w_H(u,v)P_{12}(u,v)} \\
&= \frac{\sum_{\{u,v\}\in\partial S} w_G(u,v)P_{12}(u,v)}{\sum_{\{u,v\}\in\partial S} w_H(u,v)P_{12}(u,v)} \\
&= \frac{\sum_{\{u,v\}\in\partial S} w_G(u,v)}{\sum_{\{u,v\}\in\partial S} w_H(u,v)} \\
&= \frac{w_G(S,\bar{S})}{w_H(S,\bar{S})} = \phi_{G,H}(S) \geq \phi_{G,H}
\end{aligned}$$

where we have used that $P_{12}(u,v) = 0$ if u,v are either both in S or both in $\bar{S}$ to go from the third line to the fourth. Similarly we use that $P_{12}(u,v) = 1$ for any edge $\{u,v\}$ with one edge in S and the other in $\bar{S}$. From this calculation we see that if for all the vertices $P_1(u) \in 0,1$ then we can construct a clustering of vertices which contradicts the assumption $\rho_{G,H} < \phi_{G,H}$

Therefore there must be at least one point $v' \in V$ in any optimal solution such that $P_1(v'), P_2(v') \in (0,1)$. If there are multiple unique optimal solutions then we can choose one without loss of generality as the following argument can be applied to any of them. Using that we have a vertex v' such that $P_1(v'), P_2(v') \in (0,1)$, we can expand out $\rho_{G,H}$ as follows

$$
\begin{aligned}
&\rho_{G,H} \\
&= \rho_{G,H}(P_1) \\
&= \frac{w_G(P_1,P_2)}{w_H(P_1,P_2)} \\
&= \frac{\sum_{\{u,v\}\in E_G} w_G(u,v)P_{12}(u,v)}{\sum_{\{u,v\}\in E_H} w_H(u,v)P_{12}(u,v)} \\
&= \frac{\sum_{\{u,v\}\in E_G, v\neq v'} w_G(u,v)P_{12}(u,v) + \sum_{\{u,v'\}\in E_G} w_G(u,v')(P_1(u)P_2(v') + P_1(v')P_2(u))}{\sum_{\{u,v\}\in E_H, v\neq v'} w_H(u,v)P_{12}(u,v) + \sum_{\{u,v'\}\in E_H} w_H(u,v')(P_1(u)P_2(v') + P_1(v')P_2(u))}.
\end{aligned}
$$

For convenience, we shall introduce the following variables.

$$A := \sum_{\{u,v\}\in E_G, v\neq v'} w_G(u,v)P_{12}(u,v) \tag{3.7}$$

$$B := \sum_{\{u,v\}\in E_H, v\neq v'} w_H(u,v)P_{12}(u,v) \tag{3.8}$$

Observe that $A,B \geq 0$ since they are both finite sums over non-negative terms. Introducing these we have

$$
\begin{aligned}
\rho_{G,H} &= \frac{A + \sum_{\{u,v'\}\in E_G} w_G(u,v')(P_1(u)P_2(v') + P_1(v')P_2(u))}{B + \sum_{\{u,v'\}\in E_H} w_H(u,v')(P_1(u)P_2(v') + P_1(v')P_2(u))} \\
&= \frac{A + P_2(v')\sum_{\{u,v'\}\in E_G} w_G(u,v')P_1(u) + P_1(v')\sum_{\{u,v'\}\in E_G} w_G(u,v')P_2(u))}{B + P_2(v')\sum_{\{u,v'\}\in E_H} w_H(u,v')P_1(u) + P_1(v')\sum_{\{u,v'\}\in E_H} w_H(u,v')P_2(u))}
\end{aligned}
$$

Again for convenience, we introduce the variables

$$c_1 := \sum_{\{u,v'\}\in E_G} w_G(u,v')P_2(u), \tag{3.9}$$

$$c_2 := \sum_{\{u,v'\}\in E_G} w_G(u,v')P_1(u), \tag{3.10}$$

$$d_1 := \sum_{\{u,v'\}\in E_H} w_H(u,v')P_2(u), \tag{3.11}$$

$$d_2 := \sum_{\{u,v'\}\in E_H} w_H(u,v')P_1(u). \tag{3.12}$$

Observe that again $c_1, c_2, d_1, d_2 \geq 0$ since they are finite sums over non-negative terms. Introducing these we have

$$\rho_{G,H} = \frac{A + c_1P_1(v') + c_2P_2(v')}{B + d_1P_1(v') + d_2P_2(v')}. \tag{3.13}$$

By construction we have $P_2(v') = 1 - P_1(v')$. Finally this gives

$$\rho_{G,H} = \frac{A + c_1P_1(v') + c_2(1 - P_1(v'))}{B + d_1P_1(v') + d_2(1 - P_1(v'))}. \tag{3.14}$$

We now look at what happens to this function as we alter $P_1(v')$.

$$\frac{\mathrm{d}\rho_{G,H}}{\mathrm{d}P_1(v')} = \frac{B(c_1 - c_2) + A(d_2 - d_1) + c_1d_2 - c_2d_1}{(B + d_1P_1(v') + d_2(1 - P_1(v')))^2} \tag{3.15}$$

Observe first that the numerator is a constant since there is no $P_1(v')$ term present in it. Secondly notice

$$(B + d_1P_1(v') + d_2(1 - P_1(v')))^2 \geq 0$$

and that since $P_1(v') \in (0,1)$, we have $(B + d_1P_1(v') + d_2(1 - P_1(v')))^2 = 0 \iff B = d_1 = d_2 = 0$. It is only possible for d_1 and d_2 to be 0 when v' is not part of any edges in H. B is only ever zero when there are no edges in H that have any probabilistic of being broken. Therefore we can ignore this case because for a problem to have an optimal solution with this property would be ill-defined.

Since the numerator is a constant we now only need to consider what happens when if it is positive, negative, or zero.

- If the numerator is positive, then decreasing $P_(v')$ will strictly decrease $\rho_{G,H}$. That is, if we set $P(v') = 0$ we will achieve an even lower $\rho_{G,H}$.
- If the numerator is negative, then increasing $P_1(v')$ will strictly decrease $\rho_{G,H}$. That is, if we set $P_1(v') = 1$ we will achieve an even lower $\rho_{G,H}$.

- Finally if the numerator is 0, then any value of $P_1(v')$ will not change $\rho_{G,H}$. This means that we can arbitrarily set $P_1(v') = 0$ and $\phi^p_{G,H}$ will be the same.

Using these three facts we have that for at least one of $P(v') = 0$ or $P(v') = 1$ the value of $\rho_{G,H}$ will not increase. However, we assumed that P achieved the optimal probabilistic partition and showed that $\rho_{G,H} < \phi_{G,H}$ could only be done if $P(v') \in (0,1)$. Therefore we have reached a contradiction and can conclude that $\phi^p_{G,H} = \phi_{G,H}$. □

The proof of the theorem is constructive in the sense that it gives a simple algorithmic way of taking a probabilistic partition and moving to a clustering without increasing the probabilistic conductance. We summarize this with the algorithm HardCluster.

Algorithm 1 HardCluster(G,H,P_1,P_2) Converts a 2-way probabilistic partition into a clustering.

```
for v ∈ V do
    Compute A, B, c1, c2, d1, d2 as defined in (3.7) through (3.12) for v;
    if B(c1 − c2) + A(d2 − d1) + c1d2 − c2d1 > 0 then
                    ▷ Checking the sign of the derivative and updating accordingly.
        P1(v) ← 0;
        P2(v) ← 1;
    else
        P1(v) ← 1;
        P2(v) ← 0;
return {P1, P2};
```

The calculations involved are one-to-one with the proof that the probabilistic conductance must be exactly equal to the standard conductance.

3.2 Discussion on the Multiway Probabilistic Conductance

Up to this point, we have only been dealing with 2-way probabilistic cuts. Like the definition of the standard conductance, we can generalize it to the k-way case by

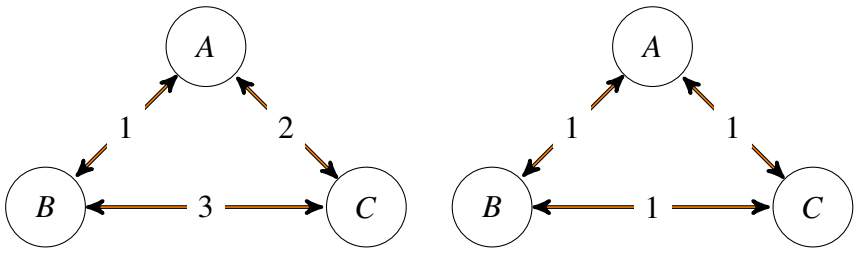


Figure 3.1: An example pair of graphs for which the 3-way conductance is greater than the 3-way probabilistic conductance. On the left is the ML-Graph and on the right is the CL-Graph.

$$\rho^k_{G,H} := \min_{P_1,\ldots,P_k \text{prob. part.} V} \max_{1\le i\le k} \rho_{G,H}(P_i). \tag{3.16}$$

which is a mirror of the standard k-way conductance with the partition of the vertices replaced by a probabilistic partition.

The idea motivating this definition as well as the standard k-way case is to answer how well we can split a graph into k pieces (in either the hard or probabilistic sense) such that each piece on its own achieves a low conductance. This raises the immediate question though, does the theorem from the previous section generalize? That is, is $\rho^k_{G,H} = \phi^k_{G,H}$?

It turns out that there is a very small example that demonstrates that this is not the case. Consider the graph pair G,H with adjacency matrices

$$A_G := \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix} A_H := \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

These graphs can be seen in Figure 3.1. The 3-way clustering of the vertices that achieves the optimal cut cost is the one takes one vertex in each cluster. The 3-way conductance in this case is exactly 5/2. The 3-way probabilistic partition

$$P := \begin{bmatrix} 0.9 & 0 & 0.1 \\ 0 & 1 & 0 \\ 0.1 & 0 & 0.9 \end{bmatrix}$$

where $P_i(v_j) = P_{ji}$ achieves a probabilistic conductance of $2.439 < 5/2$. This is a clear counter-example to the claim that for all G,H that $\rho^k_{G,H} = \phi^k_{G,H}$. It is also notable that in trivial examples where $G = H$ that the equality always holds for any valid cut so the claim that for all G,H that $\rho^k_{G,H} < \phi^k_{G,H}$ is also false for $k > 2$.

With it being established that the probabilistic conductance for a graph can be less than the constrained conductance we have as a direct consequence that there are no algorithms that can take a k-way probabilistic partition and return a k-way clustering that is guaranteed to achieve a lower, or even equal, conductance.

3.3 A Multiway HardCluster

While there can never be a perfect multiway version of the HardCluster algorithm, there is still a useful insight that can be taken from it. That is for a probabilistic partition $\{P_i\}$ and a vertex v, if there is a P_i that benefits from decreasing $P_i(v)$ and another P_j that benefits from increasing $P_j(v)$ then the multiway conductance will not increase if $P_j(v)$ is updated to be $P_j(v) \leftarrow P_j(v) + P_i(v)$ and $P_i(v) \leftarrow 0$.

The reason for this is that derivatives of $\phi^p_{G,H}(P_i)$ with respect to $P_i(v)$ are monotonic for all P_i and v. So if $d\phi^p_{G,H}(P_i)/dP_i(v) > 0$ and $d\phi_{G,H}(P_j)/dP_j(v) < 0$ then increasing $P_j(v)$ decreases $\phi^p_{G,H}(P_j)$ and decreasing $P_i(v)$ also decreases $\phi^p_{G,H}(P_i)$. Therefore the update rules given above decreases both $\phi^p_{G,H}(P_j)$ and $\phi^p_{G,H}(P_i)$ while moving towards a hard clustering. Updates like this will be called "free updates" and a method for updating a probabilistic partition to take advantage of these is given in the FreeUpdates algorithm.

The only remaining detail to consider is how to implement the free updates when there are multiple partitions that can receive it. One way is to split it evenly, another is to calculate which would lower the maximum more, or finally it could be just randomly assigned. All are valid since they will cause a decrease but the one that we choose is to perform an even split as this is the fastest and doesn't involve any randomness.

Once there are no more "free updates" the partition will be stuck at a point where the probability of each vertex is split among partitions that either all want to acquire the point or want to give away their probability. It is at this point that the probabilistic conductances will inevitably go up since a hard clustering will have to go against some derivatives. The easiest way to deal with any leftover probabilities like these is to minimize the maximum conductance among the remaining candidates. An implementation for this process is given in the CostlyUpdates algorithm.

Finally, to go from a probabilistic partition to a hard clustering, all that is left is to apply the free updates followed by the costly updates. This shifts all of the probability to a single cluster for each vertex, yielding a hard clustering. The algorithm MultiHardCluster which takes in a probabilistic partition and converts it to a hard clustering,

Algorithm 2 FreeUpdates$(G,H,\{P_1,...,P_k\})$ Applies the free updates in the probabilistic partition.

for $v \in V$ **do**
  $positives \leftarrow \{P_i : P_i(v) > 0$ & the derivative of $\rho_{G,H}(P_i)$ w.r.t. $P_i(v) \geq 0\}$;
  $negatives \leftarrow \{P_i : P_i(v) > 0$ & the derivative of $\rho_{G,H}(P_i)$ w.r.t. $P_i(v) < 0\}$;
  if $|positives| = 0$ or $|negatives| = 0$ **then**
    ▷ There must be at least one group that can receive and one to give.
    Continue to the next vertex;
  $total \leftarrow \sum_{P^+ \in positives} P^+(v)$;
  For every P^+ in $positives$ set $P^+(v) \leftarrow 0$;
  For every P^- in $negatives$ set $P^-(v) \leftarrow P^-(v) + \frac{total}{|negatives|}$
return $\{P_1,...,P_k\}$;

Algorithm 3 CostlyUpdates$(G,H,\{P_1,...,P_k\})$ Applies the updates which can only be done for a cost.

for $v \in V$ **do**
  if $P_i(v) = 1$ for any P_i **then**
    Continue to the next vertex;
  $involved \leftarrow \{P_i : P_i(v) > 0\}$;
  $minmax \leftarrow \infty$;
  $index \leftarrow 0$;
  for i from 1 to $|involved|$ **do**
    $adj \leftarrow \{P'_j : P_j \in involved\}$ where

$$P'_j(u) = \begin{cases} P_j(u) & u \neq v \\ \delta_{ij} & u = v \end{cases}$$

    $maximum \leftarrow \max_{P'_j \in adj} \rho_{G,H}(P'_j)$;
    if $maximum < minmax$ **then**
      $minmax \leftarrow maximum$;
      $index \leftarrow i$;
  $P_{index} \leftarrow 1$;
  $P_j \leftarrow 0$ for $j \neq index$;
return $\{P_1,...,P_k\}$;

implements this exact process.

Algorithm 4 MultiHardCluster$(G,H,\{P_1,...,P_k\})$ Takes a probabilistic partition and returns a hard clustering.

1: $free \leftarrow$ FreeUpdates$(G,H,\{P_1,...,P_k\})$;
2: $hard \leftarrow$ CostlyUpdates$(G,H,free)$;
3: **return** *hard*;

On a technical level, the algorithms here are far from the most efficient implementations possible. In them the vast majority of the calculations are redundant, primarily due to the fact that during each iteration at most k values in the partition are updated for a k-way partition. If one wished to greatly boost the performance they could surely use several tricks to avoid redundant calculation. We have chosen to present the algorithms without these optimizations as they would greatly obscure what exactly is happening in each step.

One observation that can be quickly spotted is that the FreeUpdates algorithm can be repeatedly applied until a convergence is reached. While this is guaranteed to converge due to the fact that it strictly decreases the cut cost we provide no guarantees on the convergence rates. A second observation to be made is what happens when the MultiHardCluster algorithm is applied to a 2-way partition. It turns out that in this case all the updates are free and by the time that the CostlyUpdates algorithm is ran the vertices will already be in a hard clustering. The reason for this is that the FreeUpdates algorithm when $k = 2$ becomes identical to HardCluster.

3.4 Practical Computations

In general, many of the calculations involving probabilistic partitions can be significantly sped up through the use of matrix and vector operations. Many software packages come bundle with routines that are able to vastly improve the calculation speed using matrices and vectors when compared to loops and summations. Examples of libraries of these routines include BLAS [Lawson et al., 1979] and LAPACK [Anderson et al., 1999]. It is at this point that we take a moment to demonstrate places that these can often be applied.

Lemma 3.4. Given two graphs G,H with no self-loops, on the same vertex set, with adjacency matrices A_G,A_H respectively and a probabilistic partition $\{P_1,P_2\}$, the probabilistic conductance can be computed as

$$\rho_{G,H}(P_1) = \frac{P_1^\intercal A_G(\mathbb{1} - P_1)}{P_1^\intercal A_H(\mathbb{1} - P_1)} \tag{3.17}$$

where P_1 is a column vector of probabilities.

Proof. The proof is just a direct calculation.

$$\begin{aligned}
\rho_{G,H}(P_1) &= \frac{w_G(P_1, P_2)}{w_H(P_1, \bar{P_2})} \\
&= \frac{\sum_{\{u,v\}\in E_G} w_G(u,v)(P_1(u)P_2(v) + P_1(v)P_2(u))}{\sum_{\{u,v\}\in E_H} w_H(u,v)(P_1(u)P_2(v) + P_1(v)P_2(u))} \\
&= \frac{\sum_{i=1}^n \sum_{j=1}^n A_{G_{ij}} P_1(v_i)P_2(v_j)}{\sum_{i=1}^n \sum_{j=1}^n A_{H_{ij}} P_1(v_i)P_2(v_j)} \\
&= \frac{\sum_{i=1}^n \sum_{j=1}^n A_{G_{ij}} P_1(v_i)(1 - P_1(v_j))}{\sum_{i=1}^n \sum_{j=1}^n A_{H_{ij}} P_1(v_i)(1 - P_1(v_j))} \\
&= \frac{P_1^\intercal A_G(\mathbb{1} - P_1)}{P_1^\intercal A_H(\mathbb{1} - P_1)}
\end{aligned}$$

□

Further more, the derivatives required in the *HardCluster* algorithm can be efficiently computed. This is because of the following.

Lemma 3.5. Given two graphs G, H with no self-loops, on the same vertex set, with adjacency matrices A_G, A_H respectively and a probabilistic partition $\{P_1, P_2\}$ the following calculations hold for a given $v \in V$.

1. $A = P_1^{0\intercal} A_G(\mathbb{1} - P_1^1)$ where P_1^0 is defined $P_1^0(u) = P_1(u), u \neq v$ and $P_1^0(v) = 0$, and P_1^1 is defined $P_1^1(u) = P_1(u), u \neq v$ and $P_1^1(v) = 1$.
2. $B = P_1^{0\intercal} A_H(\mathbb{1} - P_1^1)$.
3. $c_1 = A_G[v,:](\mathbb{1} - P_1)$.
4. $c_2 = A_G[v,:]P_1$.
5. $d_1 = A_H[v,:](\mathbb{1} - P_1)$.
6. $d_2 = A_H[v,:]P_1$.

where A, B, c_1, c_2, d_1, d_2 are defined in equations (3.7) through (3.12), P_1 is a column vector of probabilities, and $A_G[v,:]$ denotes the row of A_G indexed by v.

Proof. Statements 1 and 2 can be seen as a modification of the preceding lemma such that whenever there would be a weight multiplied by the probability for v it is flipped off.

Statements 3,4,5, and 6 are all using the dot product of the probabilities and the row of edge weights of v which is equivalent to iterating over the edges and multiplying by the required probability. This is because any non-existent edge is just 0 in the adjacency matrix and as such doesn't have any impact on the resulting dot product. □

These two lemmas are incredibly useful in practice allowing implementations to run significantly faster than would otherwise be possible, especially the procedures that are defined in the next section.

Chapter 4

Experiments

Now that we have established a method of taking a probabilistic partition and converting it into a hard clustering, the question of when and why we would do this is raised. Well it turns out that while k-means is popular in practice and directly produces a hard clustering, there are also many other algorithms which produce probabilistic partitions. These include the EM algorithm [Dempster et al., 1977], as well as more sophisticated methods which use the constraints such as in [Lu and Leen, 2008]. Both algorithms yield not just cluster assignments, but the probability that a point belongs to each cluster. It is from the outputs of these algorithms that we can apply MultiHardCluster in order to recover a true hard clustering. Figure 4.1 demonstrates a standard flow for graph clustering. The first step may be skipped if G and H are given directly instead of a data graph, ML graph, and CL graph.

In our we use the EM algorithm on a Gaussian Mixture Model which is constrained to use a shared diagonal covariance matrix. In our tests we found that this consistently produced the best results. The posterior probabilities of each vertex are used as the probabilistic partition and fed into the MultiHardCluster algorithm which then outputs a hard clustering.

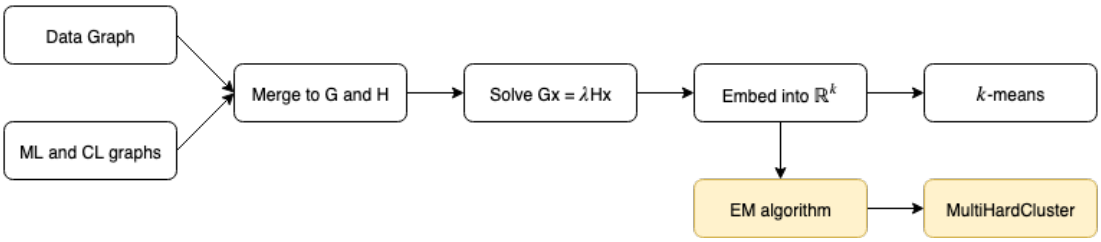


Figure 4.1: The flow for partitioning a graph given a data graph and sets of constraints. The method of [Cucuringu et al., 2016] is in white. Highlighted is our variant which uses the EM algorithm followed by MultiHardCluster

For our experiments we consider applications in two settings. The first being image segmentation and the latter being cluster recovery in stochastic block models. For each will first give a brief review of the task and then delve into our methods and experiments.

4.1 Image Segmentation

Spectral methods for image segmentation involve converting an image into a graph and then clustering that graph, and from the clusters segments are made. The steps involved are easy to describe but in practice are not so straightforward to implement. We will briefly go over, step by step, the process that we have used and conclude with experimental results.

4.1.1 Converting to a Graph

It is not entirely obvious how one should go about taking an image and converting it to a graph. The most common way of doing this is by creating a grid graph [Felzenszwalb and Huttenlocher, 2004]. For an image, the vertices of a grid graph correspond to the pixels. Two vertices are connected by an edge if the pixels they correspond to are adjacent or very near by. Typical methods for doing this are to attach a pixel to either its four side sharing pixels or to the eight pixels that it shares a corner with. Alternatively a radius can be given and a pixel is connected to the pixel within the given radius.

The edge weights are determined by pixel color similarity with similar pixels given high weights and dissimilar pixels given low weight. In the radius approach weights may also be modified by the distance between pixels with nearby pixels benefiting while the farther ones are penalized.

In our experiments we have used a grid graph with each pixel connected to its eight adjacent vertices. The weights are taken using the radial basis function applied to the L_2 distance between pixels in the XYZ color format [Smith and Guild, 1931]. We will refer to this as the data graph in accordance with the literature and refer to it as $G_D = (V, E, w_{G_D})$.

4.1.2 Introducing Constraints

One could take G_D and immediately perform spectral clustering on it to get a segmented image. This task has been explored in many papers including famously in [Shi

and Malik, 2000]. While the results are impressive both visually and quantitatively we modify the problem to utilize constraints.

Now the main issue is that we require a pair of graphs while we only have a single graph from the image. Furthermore it is not obvious if there even is a meaningful way of generating a CL graph directly from an image. Instead we hand label a few points in the image and get an additional set of ML and CL constraints from the equivalence classes on the labeled points. That is, we add an edge in the ML graph if two points are labeled to be in the same cluster and an edge is added to the CL graph if two points are labeled to be in different clusters. However this leaves the question of how to incorporate this information into the graph generated from the image.

To merge the constraints we default to [Cucuringu et al., 2016]. The technique they propose ,which we have adopted, is to first create two graphs $\hat{G}_{ML}$ and $\hat{G}_{CL}$. $\hat{G}_{ML}$ is defined by iterating over the edges in the ML graph. For every edge (v_i, v_j) in ML the weight of that edge in $\hat{G}_{ML}$ is $d_i d_j/(d_{min} d_{max})$ where d_i and d_j are the degrees of v_i and v_j in the grid graph respectively d_{min} is the smallest degree any vertex has, and d_{max} is the largest degree any vertex has. $\hat{G}_{CL}$ is defined in an identical way except going over the edges in the CL graph.

Once we have those two graphs we then create the final graphs that we will cluster as $G = G_D + \hat{G}_{ML}$ and $H = K/n + \hat{G}_{CL}$ where n is the number of vertices in G_D, K is the demand graph of CL whose adjacency matrix is defined $A_{ij} = d_i d_j / vol(V)$, d_i is the degree of v_i in the CL graph, $vol(V)$ is the volume of the CL graph. Here we have abused the notation slightly in that arithmetic operations are not strictly defined on graphs but we mean to be performing them on their adjacency matrices.

4.1.3 Solving the Generalized Eigenproblem and Embedding

As discussed in the second chapter, most of the work is done by solving the eigenproblem $L_G x = \lambda L_H x$. However this cannot be done using standard eigenvector routines due to the scale of the matrices involved. Each graph has xy vertices for an $x \times y$ image. Even for a medium sized image of 300×600 that's still 180000 vertices. This makes the Laplacian a 180000×180000 which is beyond the scale that most routines can handle. However, the matrix is incredibly sparse given that each vertex has at most 8 edges. This fact lends itself to using routines on sparse matrices. Specifically we use LOBPCG [Knyazev, 2001] which is able to efficiently compute the eigenvectors of large, sparse matrices as well as solve generalized eigenproblems on pairs of

large sparse matrices. This is what enables us in practice to get past the bottleneck of computing eigenvectors.

Once the eigenvectors have been computed they are still not quite ready to be clustered on. The resulting eigenvectors first have $c\mathbb{1}$ added to them such that the resulting eigenvectors are orthogonal to d, the vector of degrees of vertices in G. After that they are normalized so that $x^T Dx = 1$ where x is an eigenvector and D is the degree matrix of G. For an in depth discussion behind why this is done see [Cucuringu et al., 2016]. One can see that the result of the eigenvectors being adjusted are still eigenvectors with the same eigenvalues. This is because for any eigenvector x with eigenvalue λ

$$L_G(x + c\mathbb{1}) = \lambda L_H x + 0c\mathbb{1} = \lambda L_H(x + c\mathbb{1}),$$

since the constant vector is in the null space of both L_G and L_H. The second operation is just scalar multiplication which clearly preserves eigenvectors and eigenvalues. What these two steps are doing is picking exactly which versions of the eigenvectors to use.

Finally, once the eigenvectors have been adjusted they are put into a $n \times k$ matrix and the rows of that matrix are normalized to have unit length. The justification for this is given in [Lee et al., 2014]. The resulting rows are the points in $\mathbb{R}^k$ which will be clustered.

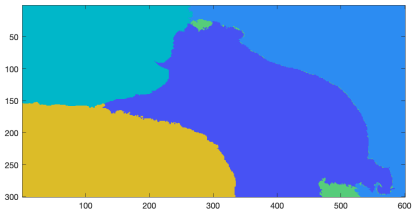
4.2 Results for Image Segmentation

The experiments that we have composed for image segmentation were made using Matlab. Our work uses functions for generating image data graphs by Leo Grady, an implementation of LOBPCG by Andrew Knyazev, and functions for merging constraints with the data graph and embedding by Mihai Cucuringu. The flow chart in figure 4.1 demonstrates how these have been used together.

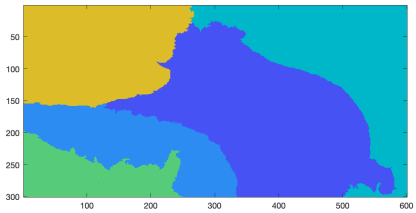
Once the points embedded we can experiment with clusterings using the EM algorithm followed by an application of the MultiHardCluster. Specifically we apply the EM algorithm to Gaussian Mixture Models [Bilmes et al., 1998] which are restricted to share a diagonal covariance. What follows in this section are two image segmentation examples as well as comparisons of their clusterings.

We begin with a picture of a bear which is 300 pixels tall and 600 pixels wide. We have segmented it using identical grid graphs G_D as well as identical ML and CL constraints, meaning that the output only differs from the clustering on the embedding. The results are shown in Figures 4.2b and 4.2c. The primary difference between these

(a) The original image with labels.



(b) Clustering using k-means



(c) Clustering using EM and MultiHardCluster

Figure 4.2: This and that

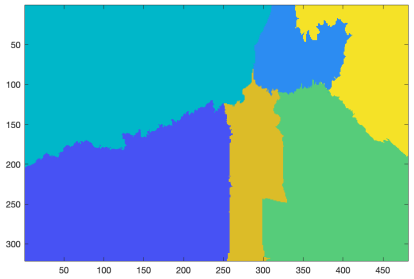
two images is that k-means does not effectively recover the water in the bottom left hand corner as its own cluster and instead merges it with the rock, leaving an awkward extra cluster. Using EM the bottom left water and the rock are effectively segmented and as such the five clusters are all of high quality.

For the next experiment we look at a photograph of a rooftop cast again a mountain and sky. The image is 321 pixels tall and 481 pixels wide. As in the first example we use an identical data graph and ML and CL graphs for both techniques. Also included is an additional image with one constraint missing.

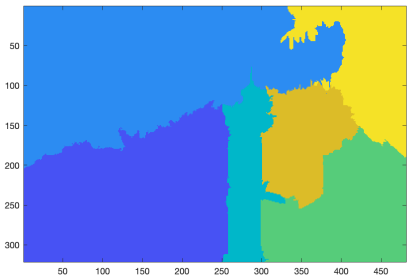
In the segmentation of this image the quality is high when using both k-means and EM. The only significant difference between the two is that k-means places a small cluster in the sky near the tree and EM places it over the second tower.

In the image in the bottom left corner we have omitted a single labeled point, the one in the window of the leftmost tower. What we see happening without this point is the entire right side of the tower not being segmented with the left side. In contrast, including the single label pulls the entire right side into the the tower. This experiment helps to demonstrate the impact that the constraints have in the segmentation.

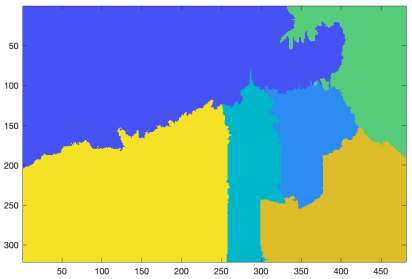
(a) The original image with labels.



(b) Clustering using k-means



(c) Clustering with one missing constraint



(d) Clustering using EM and MHC

Figure 4.3: (a) The original image with the labelled points. The diamond point is the constraint ignored in (c). (b) A clustering done with k-means. (c) A clustering done with the EM algorithm and MultiHardCluster but with the diamond point not included in the constraints. (d) A clustering done with the EM algorithm and MultiHardCluster with all the constraints.

4.2.1 Run Speed Comparison

In practice, the speed at which these algorithms run may be a serious concern. Since our method as well as that of [Cucuringu et al., 2016] are identical up until the embedding is completed will shall only consider differences in speed after the embedding is completed. We shall use the eigenvectors computed during the segmentation of the picture of the bear to do this.

In terms of the running time of k-means compared to EM on the 180,000 points we have a noticeable difference in speed. k-means averages 0.0211 seconds to form a clustering while EM averages 0.5473 seconds. This is to be expected though due to the computation complexity of EM relative of the of k-means.

Our Matlab implementation of MultiHardCluster is also quite slow. In our experiments, it took up to an entire hour to run. However, we have implemented it in a far from optimal way. While we have taken advantage of the computations described in lemma 3.5, we have done little else to increase speed. In our experiments we found that the vast majority of time was spent during the FreeUpdate routine. We expect that a carefully optimized version of this routine would have a running time of $O(kn)$ and could avoid the vector-matrix-vector multiplication required to calculate the A and B values, saving a considerable amount of redundant calculation.

4.3 Results in the Stochastic Block Model

The stochastic block model is a model used for generating random graphs. In general, a stochastic block model is given the number of vertices and the probability that a pair of vertices share an edge. When any pair of edges have an equal chance of being connected this is often referred to as one version of the Erdős-Rényi model (although it was originally proposed by Gilbert in [Gilbert, 1959]) and is denoted $G(n, p)$. While this model is interesting in its own right it does not make for a very good method of generating graphs for the purpose of clustering. This is largely due to the lack of cluster structure within most of these graphs.

We use a variant of the stochastic block model, the planted partition model, which takes four parameters, k the number of clusters, m the cluster size, and two probabilities p and q. The graph generated has k clusters each with m vertices in them. p is the probability that any two vertices within the same cluster have an edge between them. q is the probability that any two vertices from different clusters have an edge between

them. The Erdős-Rényi model can be seen as the special case where $p = q$. Using these parameters we define the model as $G(k,m,p,q)$ and shall sample graphs from this model in our experiments.

When trying to recover the clusters in a graph generated by the planted partition model with spectral methods, the standard technique is to compute the Laplacian, compute the embedding, and then generate the clusters. This is largely similar to what we will be doing. In this task there are really two types of edges, the first being true edges which connect vertices which should be connected, and the second being false edges which connect vertices which should not be connected. If the ratio of p to q favors p enough then the true edges outweigh the false ones and the clusters can be recovered easily with spectral clustering.

What we seek to investigate is the case where there is a ML graph and a CL graph. The standard setting can be seen as working with only an ML graph. In the constrained setting there are actually four types of edges that must be considered. They are

1. The true edges in the ML graph. Edges which correctly indicate that two vertices should be in the same cluster.

2. The false edges in the ML graph. Edges which indicate that two vertices should be in the same cluster when they really should not be.

3. The true edges in the CL graph. Edges which correctly indicate that two vertices should not be in the same cluster.

4. The false edges in the CL graph. Edges which indicate that two vertices should not be in the same cluster when they actually should be.

4.3.1 Experimental Set Up

The experiments that we composed for cluster recovery were made in the Python programming language. Our work leverages three libraries, NetworkX (`https://networkx.github.io/`), Numpy (`http://www.numpy.org/`), and Scipy (`https://www.scipy.org/`). NetworkX is a library for the creation, manipulation, and analysis of graphs. Numpy and Scipy are two libraries for performing fast linear algebra routines.

Throughout our experiments we use a standard setting with $k = 5$ and $m = 50$, yielding graphs with 250 vertices. For evaluating the quality of the clustering we use

the Adjusted Rand Index [Hubert and Arabie, 1985]. This is a measure of how well the predicted clusters agree with the true underlying clusters and is adjusted for chance. What this metric does it look at how pairs of vertices have been clustered. A clustering which produces many pairs of vertices which are in the same cluster when they are supposed to together and in different clusters when they are supposed to be apart will achieve a high score. A clustering that does the opposite of this will achieve a low score. The Adjusted Rand Index is also adjusted for chance such that if a clustering is randomly done it is expected to receive a score of zero.

4.3.2 Searching Over p and q

One thing that makes clustering easier is increasing the ratio of true edges to false edges in the graph. Decreasing this ratio on the other hand will have the opposite effect. One interesting question that we look at is the case where we have two graphs generated. The ML graph taken from the model $G(k,m,p,q)$ and the CL graph taken from $G(k,m,q,p)$. One should observe that in the ML graph the third argument is the one which determines the probability of a true edge while in the CL graph it is the fourth argument which determines the probability of a true edge. Therefore by swapping the ordering of p and q we preserve the probability of a true edge being in the graph as well as the probability of a false edge.

The task now becomes interesting due to the inclusion of the two new types of edges which may either improve or worsen the ability to recover the clusters. As our first experiment we search over the entire space of p and q on both a single graph as well as a pair. The ML and CL graphs are generated from $G(k,m,p,q)$ and $G(k,m,q,p)$ respectively. The ML graph is clustered both on its own and with the CL graph. Due to the randomness of the graphs, we repeat the experiment with 10 different ML and CL graphs for every parameter setting.

We also will sometimes use $at = p$ and $bt = q$ where $t = \frac{\ln n}{n}$ and $a,b \geq 1$. The reason we do this is from [Erdős and Rényi, 1961] which tells us that the if each edge is in the graph with probability at least $\ln n/n$ then there is a high probability the graph is connected.

It is interesting to observe the difference plot in Figure 4.4. The reason behind shape and location of this streak is not entirely obvious. Clearly in the constrained setting there is more information, but there is also more misinformation. Further, the chance of any true edge being present is the same in both the constrained and uncon-

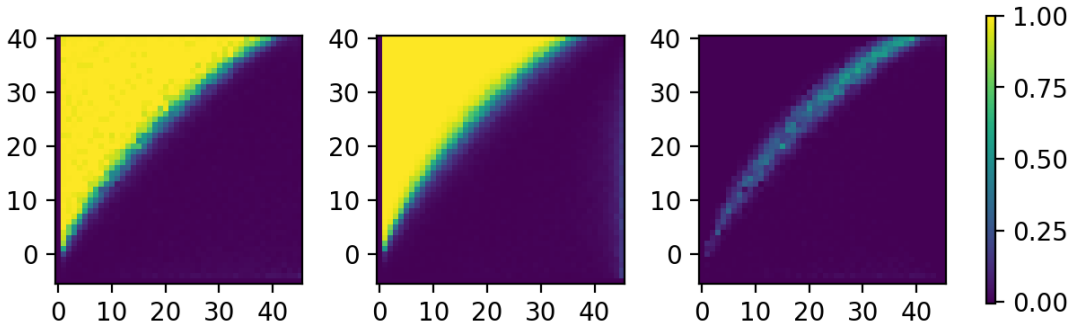
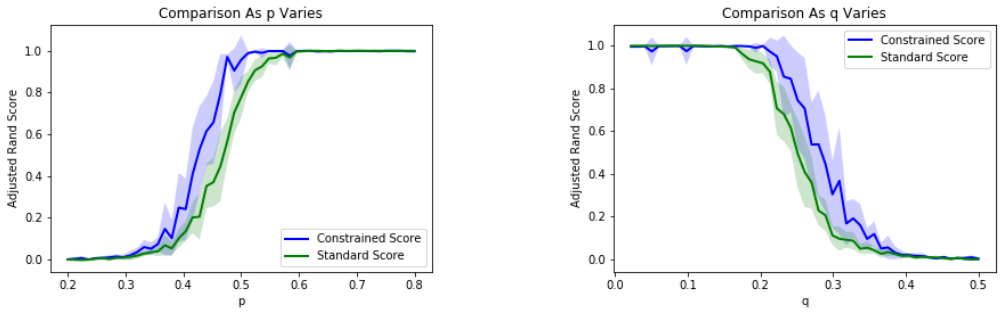


Figure 4.4: Plots of the adjusted rand score across all p and q combinations. On the left we have constrained setting which uses two graphs. In the middle we have the unconstrained setting which uses one graph. On the right we have the difference between the two, constrained - unconstrained. The a values are on the y axis and b values are on the x axis.



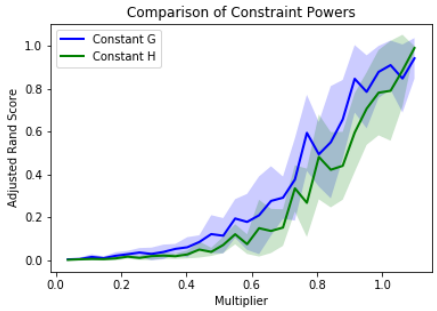
(a) Varying p with a fixed $q = 0.22$

(b) Varying q with a fixed $p = 0.5$

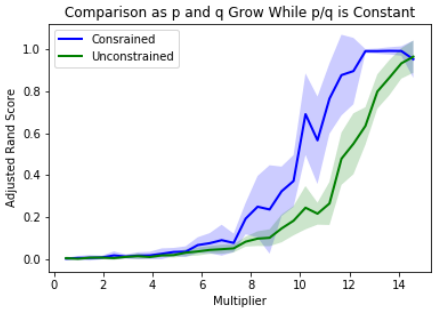
Figure 4.5: Plots of what happens while either p or q is kept constant and the other variable is varied. The standard deviation of the score is also shown.

strained case, the same with false edges. It is our conjecture that the reason there is a streak is because of the ratio of expected true edges to expected false edges. The reason that we propose this is because the CL graph should have a higher ratio of true edges to false edges than in the ML graph, since there are many more edges between clusters than within them. Further work would need to be done to verify this theoretically.

In Figure 4.5 we show what happens as the one of p or q varies and while the other is held constant. This is in some ways a cross section of Figure 4.4. The notable part of these plots is that the two settings do not transition drop or rise at exactly the same time. In the constrained case, it takes a lower p value to achieve a strong performance, and it takes higher q value for performance to deteriorate, when compared to the unconstrained case. This backs up the result in figure 4.4 in which the drops in performance happen at different times.



(a) Fixing one of G or H while scaling p and q at a ratio of 4/3.



(b) Fixing the ratio $p/q = 3/2$ while scaling p and q in both the constrained and unconstrained settings.

Figure 4.6: (a) A plot demonstrating what happens when both G and H are generated from the same ratio of p and q but one is held at fixed p and q and the other scales. (b) A plot demonstrating the impact scaling p and q has on both constrained and unconstrained cluster recovery.

4.3.3 Power of Constraints

In our next experiments we look to quantify how useful the constraints are to the recovery as well how much the scale of p and q matters. When we say "scale" we mean how large p and q are. In these experiments we have fixed the ratio of p/q and look to see what scaling the values does to the recovery.

The plots in figure 4.6 shed some light on the role that constraints play. In the plot on the left, we see that the scale of p and q is of roughly equal importance for both the ML and CL graph. It is interesting that the constant G graph requires a slightly lower scale H graph when compared to a constant H graph with a varying G graph. The reason for this is not obvious and further investigation into it could prove very interesting.

In the plot on the right we see what impact the scale has when comparing the constrained to unconstrained setting. In the plot we see that cluster recovery can be done at a lower scale when working with constraints and still achieve equal quality recoveries compared to the unconstrained case. This is actually quite a nice result to observe as it shows that the pair of graphs do not need to be as dense as a single graph does in order to recover clusters at a similar level.

Chapter 5

Conclusion

5.1 Summary of Work

In this thesis we have introduce a new notion of conductance as well as laid the ground work definitions that it requires. We have demonstrated an interesting theoretical result in theorem 3.3. The HardCluster algorithm encapsulates the work done in the proof and is readily implementable using fast linear algebra subroutines. Furthermore, with the MultiHardCluster algorithm we have given a heuristic solution to the problem of converting a k-way probability partition to a clustering.

In our experiments chapter we have demonstrated that the technique proposed is competitive for both image segmentation and cluster recovery. The work on cluster recovery using pairs of graphs has provided its own interesting experimental results which warrant their own further study.

5.2 Future Work

With new definitions and notions there is always a bounty of new questions and work. We will briefly enumerate some that are of specific interest to us.

First, with it established that it is possible for $\rho^k_{G,H}$ to be less than $\phi^k_{G,H}$, is there any bound on $\phi^k_{G,H} - \rho^k_{G,H}$, the difference of the two expressions. We conjecture that there is a bound on this but do not provide any guess as to what it may be. Another question that arises is about higher order Cheeger inequalities in relation to $\rho^k_{G,H}$. It is interesting to ask about how tight the inequalities on $\phi^k_{G,H}$ would be on $\rho^k_{G,H}$. The answer to this question may be related to the answer to the previous question. A third and final question about $\rho^k_{G,H}$ is if there exist algorithms which can be analyzed from

end to end in the style of [Peng et al., 2017].

In terms of practical work, experiments with clustering algorithms that directly incorporate CL and ML graphs could be explored. This would in effect yield an algorithm which is from start to finish aware of the constraints. We also believe that the work done on cluster recovery is particularly interesting and future work could be done with experiments of this type.

Bibliography

[Abbe, 2017] Abbe, E. (2017). Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531.

[Aittokallio and Schwikowski, 2006] Aittokallio, T. and Schwikowski, B. (2006). Graph-based methods for analysing networks in cell biology. *Briefings in bioinformatics*, 7(3):243–255.

[Anderson et al., 1999] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' guide*, volume 9. Siam.

[Basu et al., 2008] Basu, S., Davidson, I., and Wagstaff, K. (2008). *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.

[Belkin and Niyogi, 2002] Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591.

[Biemann, 2006] Biemann, C. (2006). Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the association for computational linguistics: student research workshop*, pages 7–12. Association for Computational Linguistics.

[Bilmes et al., 1998] Bilmes, J. A. et al. (1998). A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126.

[Brandes et al., 2003] Brandes, U., Gaertler, M., and Wagner, D. (2003). Experiments on graph clustering algorithms. In *European Symposium on Algorithms*, pages 568–579. Springer.

[Chung, 1997] Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

[Condon and Karp, 2001] Condon, A. and Karp, R. M. (2001). Algorithms for graph partitioning on the planted partition model. *Random Structures & Algorithms*, 18(2):116–140.

[Cucuringu et al., 2019] Cucuringu, M., Davies, P., Glielmo, A., and Tyagi, H. (2019). Sponge: A generalized eigenproblem for clustering signed networks.

[Cucuringu et al., 2016] Cucuringu, M., Koutis, I., Chawla, S., Miller, G. L., and Peng, R. (2016). Scalable constrained clustering: A generalized spectral method. *CoRR*, abs/1601.04746.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

[Dunford and Schwartz, 1957] Dunford, N. and Schwartz, J. T. (1957). *Linear Operators. Part 1: General Theory*. New York Interscience.

[Erdős and Rényi, 1961] Erdős, P. and Rényi, A. (1961). On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, 12(1-2):261–267.

[Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181.

[Garey and Johnson, 1990] Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[Gilbert, 1959] Gilbert, E. N. (1959). Random graphs. *Ann. Math. Statist.*, 30(4):1141–1144.

[Gross and Yellen, 2005] Gross, J. L. and Yellen, J. (2005). *Graph theory and its applications*. Chapman and Hall/CRC.

[Harary, 1953] Harary, F. (1953). On the notion of balance of a signed graph. *Michigan Math. J.*, 2(2):143–146.

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

[Hung and Ho, 1999] Hung, Y. and Ho, H. (1999). Errata: Corrections to "a kalman filter approach to direct depth estimation incorporating surface structure". *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 15(10):1101.

[Kamvar et al., 2003] Kamvar, K., Sepandar, S., Klein, K., Dan, D., Manning, M., and Christopher, C. (2003). Spectral learning. In *International Joint Conference of Artificial Intelligence*. Stanford InfoLab.

[Knyazev, 2001] Knyazev, A. V. (2001). Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2):517–541.

[Kunegis et al., 2010] Kunegis, J., Schmidt, S., Lommatzsch, A., Lerner, J., De Luca, E. W., and Albayrak, S. (2010). Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 559–570. SIAM.

[Lawson et al., 1979] Lawson, C. L., Hanson, R. J., Kincaid, D. R., and Krogh, F. T. (1979). Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Softw.*, 5(3):308–323.

[Lee et al., 2014] Lee, J. R., Gharan, S. O., and Trevisan, L. (2014). Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37.

[Lu and Carreira-Perpinan, 2008] Lu, Z. and Carreira-Perpinan, M. A. (2008). Constrained spectral clustering through affinity propagation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[Lu and Leen, 2008] Lu, Z. and Leen, T. K. (2008). Pairwise constraints as priors in probabilistic clustering. *Constrained clustering: advances in algorithms, theory, and applications*, page 59.

[Nahshon, 2018] Nahshon, T. (2018). Spectral clustering from scratch.

[Nascimento and de Carvalho, 2011] Nascimento, M. C. and de Carvalho, A. C. (2011). Spectral methods for graph clustering a survey. *European Journal of Operational Research*, 211(2):221 – 231.

[Ng et al., 2001] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 849–856, Cambridge, MA, USA. MIT Press.

[Peng et al., 2013] Peng, B., Zhang, L., and Zhang, D. (2013). A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 46(3):1020–1038.

[Peng et al., 2017] Peng, R., Sun, H., and Zanetti, L. (2017). Partitioning well-clustered graphs: Spectral clustering works! *SIAM Journal on Computing*, 46(2):710743.

[Rohe et al., 2011] Rohe, K., Chatterjee, S., and Yu, B. (2011). Spectral clustering and the high-dimensional stochastic blockmodel. *Ann. Statist.*, 39(4):1878–1915.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905.

[Smith and Guild, 1931] Smith, T. and Guild, J. (1931). The cie colorimetric standards and their use. *Transactions of the optical society*, 33(3):73.

[Strehl and Ghosh, 2003] Strehl, A. and Ghosh, J. (2003). Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230.

[von Luxburg, 2007] von Luxburg, U. (2007). A tutorial on spectral clustering. *CoRR*, abs/0711.0189.

[Wagstaff et al., 2001] Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584.

[Wu and Leahy, 1993] Wu, Z. and Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (11):1101–1113.

[Yan et al., 2006] Yan, R., Zhang, J., Yang, J., and Hauptmann, A. G. (2006). A discriminative learning framework with pairwise constraints for video object classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):578–593.