Spectral clustering of graphs
Lecture notes for the 2016 CRM Summer School

Jean-Gabriel Young

Département de Physique, de Génie Physique, et d'Optique, Université Laval, Québec (Québec), Canada G1V 0A6

July 13, 2016

Abstract

Graph clustering designs the class of problem where one must identify a partition of the vertex set of graph which optimizes an objective function of the partition and the graph. Constrained min-cut is perhaps the most well known example of graph clustering problem. Like most graph clustering problem, it is known to belong to the NP-Hard complexity class. Heuristic methods are therefore desirable. One of the leading approach is known as spectral clustering. It regroups methods that exploit the spectral properties of the matrix representations of graphs. These algorithms work extremely well in practice, and theoretical support is only now catching up to their empirical success.

In this short tutorial, we approach graph clustering as a relaxed version of the NP-Hard optimization problem. We first formulate the graph clustering objective functions as quadratic forms. We then relax the clustering problem—we do not ask that nodes belong to discrete block. As a result, we can use well known results pertaining to the optimization of quadratic form, and thus of our objective functions. We then review a number of methods that allow us to reverse the relaxation and obtain a strict partition of the vertex set. To showcase the power of the spectral methods, we apply spectral algorithms to a number of real datasets and reproduce the key results of a number of recent landmark studies. We finish with a discussion of the limitations of spectral algorithms that draws from recent results of random matrix theory.

1 Introduction

In the age of big data, clustering algorithms are among the most useful tools available to data scientists. In a nutshell, these algorithms return the partition of a structured set of objects which optimizes an objective function of both the partition and the structure of the set. A typical use-case is coarse-graining: Clustering algorithms yield meaningful summaries of the structure of otherwise unmanageable datasets [6]. This information can be used to take design decisions[13, 18], uncover the underlying structure of the dataset [16], etc.

Because many complex datasets are relational, they are often represented as graphs [13] (also called networks). The associated clustering problem is that of graph clustering. Specifically, one looks for a partition of the vertex set of the graph which optimizes some objective function of the partition. Constrained min-cut is perhaps the most known example of graph clustering problem: The optimal partition is the one which minimizes the number of edges linking the different blocks of the partition, whilst maintaining a balanced number of vertex in each blocks [20].

Many problems of this kind known to belong to the NP-Hard complexity class, and an exact solution is therefore hopeless [4]. To address this issue, many of efficient (heuristics) alternatives have been proposed—graph clustering is still a very much active area of research [5]. A leading approach is that of spectral clustering, whereby we use the spectral properties of some matrix representation of the graph to solve the problem approximately [14, 20]. This class of algorithms have been known to exist for quite some time (see, e.g. Ref. [1**?**]). They work extremely well in practice [12, 16], and theoretical support is only now catching up to their empirical success [11, 14, 17, 18, 20].

In this short tutorial, we will approach graph clustering as a relaxed version of the HP-Hard optimization problem. This method has been used to establish a first-principle approach to spectral clustering in a number of special cases [14, 18, 20, 21]. We here present a generalized formulation of the principle. The structure of the tutorial is as follows. In section 2, we present a number of motivations for the spectral algorithm: Its efficiency, ease of use, and generality. In section 3, we obtain a first derivation of the algorithm, in the special case of graph bisection (partition in two blocks). This serves as an intuition, on which we build in Sec. 4, to formulate the general algorithm. Section 5 contains the results of a few experiments on real datasets. We gather our conclusion in Section 6.

2 Motivation

2.1 Objective functions and their exponential search space

Constrained graph clustering can be stated quite generally. Given the vertex set $V(G)$ of an undirected graph $G(V,E)$, graph clustering consists in identifying the partition $\mathcal{B}(V)$ of $V(G)$ which optimizes an *objective function* $f:\mathcal{B},G\to\mathbb{R}$ over the set of all partitions $\mathcal{B}(V)$. A partition $\mathcal{B}(V)$ is defined as a set of g disjoint vertex sets that covers V, i.e., for the partition $\mathcal{B}=\{B_1,...,B_g\}$, we must have that

$$B_r\cap B_s=\emptyset\ \ \forall r\neq s \qquad \text{and} \qquad B_1\cup...\cup B_g=V(G) \tag{1}$$

meaning that every vertex is assigned to a block, and blocks do not intersect. The objective function assigns a *score* to each partition and allows for an ordering of the partition from the most to the least desirable. The function is defined in term of the edge set $E(G)$ of the graph—the number of edges with both ends within the same block is a natural example.

The number of possible partition depends exponential on the number of vertices N in the graph G. This remains true even if when we impose some constraints on $\mathcal{B}$. For instance, in the easy scenario where we optimize f over the set of partitions with blocks which contain a fixed number of vertices $n_i=|B_i|$ $(i=1,..,g)$, the size of the search space equals $\binom{N}{n_1\ ...n_g}$, the multinomial coefficient. If we are looking for block of equal sizes $n_i=N/g\ \forall i$, then the number of partition scales approximately[1] as $\frac{N}{g}^{-g/2}g^{N+1/2}$. Therefore, even the simplest case entails an exponential number of possible solutions.

[1]We used the Stirling approximation for $n!$ to obtain this results.

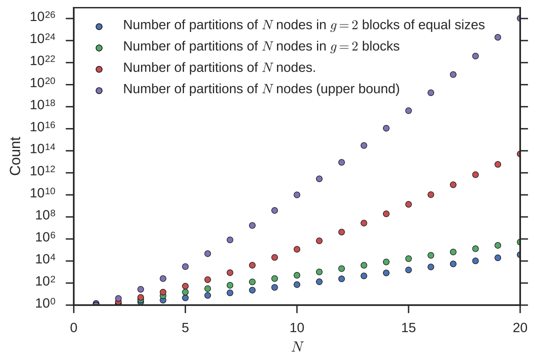


Figure 1: **Scale of the search space**. The size of the search space (number of possible partitions of $V(G)$) scales exponentially in $N = |V(G)|$, regardless of the constrains that are imposed on the solutions $\mathcal{B}$.

If we instead constrain the number of blocks g in the partition, but not the size of these blocks, then the size of the search space is (exactly) the Stirling number of the second kind $\left\{ {N \atop g} \right\}$. It goes up to N^g if we do not forbid empty blocks. Lastly, if we dot even constrain the number of blocks, then the number of potential partition is the N^{th} bell number $B_N = \sum_{j=1}^{N} \left\{ {N \atop j} \right\}$ (which is upper bounded by N^N). Figure 1 compares these different scaling rule.

The fact that the search space grows exponentially in N is not necessarily an issue—after all there exists problems with exponential search space which can be solved exactly and efficiently [10]. But many variants of graph clustering are known to be HP-Hard [4], which implies that we do not have an efficient algorithm unless P=NP[2]. It is therefore safe to assume that we cannot solve the problem both exactly and efficiently. This motivates the search for approximate, efficient, yet accurate approaches. Spectral clustering is an elegant method that fits these three criterion.

3 Clustering via the spectrum: a first look

We will first derive an intuition for spectral clustering in the simpler setting of graph bisection, i.e. by obtaining a spectral algorithm that splits the graph in two blocks B_1 and B_2. This approach is similar to that of, e.g. Refs. [18, 20].

3.1 Matrix formulation

Our end-goal is an eigenvalue equation. We will therefore need a matrix representation of the objective function. In so doing, we will focus on objective functions that can be written as

$$f(\{\sigma_i\}, G) = \frac{1}{2}\sum_{ij} h_{ij}^{(in)} \delta_{\sigma_i \sigma_j} + \sum_{ij} h_{ij}^{(out)} \bar{\delta}_{\sigma_i \sigma_j} \tag{2}$$

where δ_{rs} is the Kronecker delta and $\bar{\delta}_{rs} = 1 - \delta_{rs}$. The terms $\{h_{ij}^{in}\}$ quantify the change in "score" associated with putting vertices v_i and v_j in the same block, and $\{h_{ij}^{(out)}\}$ quantify the change in "score" associated with putting vertices v_i and v_j in different block. The 1/2 leading factor is added for convenience[3]. It is

[2] The proof is by reduction of the decision problem to other problems in NP-Complete. It is necessarily on a case by case basis, i.e. it depends on the specifics of the objective function. For instance, if the objective function is the Newman modularity [12], then reduction is from 3-partition to the problem of deciding whether a graph has a partition of modularity $\geq K$ [2]. The maximization version of the decision problem is NP-Hard because it at least as hard as any problem in NP-Complete. Proofs of NP-hardness generally follow the above structure.

[3] It can also be seen as accounting for the fact that edges are undirected and that $h_{ij}^{(x)} = h_{ji}^{(x)}$ by construction, such that every contribution is counted twice.

immediately clear that, up a irrelevant additive constant, f can be rewritten as

$$f(\{\sigma_i\}, G) = \frac{1}{2}\sum_{ij}[h_{ij}^{(in)} - h_{ij}^{(out)}]\delta_{\sigma_i\sigma_j} \; . \tag{3}$$

This merely states that we need not specify the two changes in score separately: If two nodes are in the same block, then they are necessarily *not* in two different block, and vice-versa.

In the particular case of graph bisection, $\sigma(v_i)$ can only take on two values, depending on whether $v_i \in B_1$ or $v_i \in B_2$. We will express this fact by using an indicator variable $s_i = \pm 1$ whose value depend on the block to which vertex v_i is assigned in the partition[4]. With this choice of indicator variables, we have that

$$s_i s_j = \begin{cases} 1 & \text{if } \sigma(v_i) \neq \sigma(v_j) \\ -1 & \text{otherwise} \end{cases} \quad , \tag{4}$$

which allows us to rewrite the Kronecker delta more simply as $\delta_{\sigma_i\sigma_j} \equiv (s_i s_j + 1)/2$. Substituting in Eq. (3), we find

$$\begin{aligned} f(\{\sigma_i\}, G) &= \frac{1}{4}\sum_{ij}[h_{ij}^{(in)} - h_{ij}^{(out)}](s_i s_j + 1) \\ &= \frac{1}{4}\sum_{ij} s_i[h_{ij}^{(in)} - h_{ij}^{(out)}]s_j + C \\ &\equiv \frac{1}{4}\boldsymbol{s}^T\boldsymbol{H}\boldsymbol{s} + C \; . \end{aligned} \tag{5}$$

where we have switched to the matrix notation, and $C = \frac{1}{4}\sum_{ij}[h_{ij}^{(in)} - h_{ij}^{(out)}]$ is an additive constants that do not change the position of the optimum, when one optimizes over all possible $\boldsymbol{s}$. By switching to the matrix notation, we have defined the *objective matrix* $\boldsymbol{H}$. It embodies our clustering objective. An element h_{ij} gives the change in score $h_{ij}^{(in)} - h_{ij}^{(out)}$ associated to putting vertices v_i, v_j in the same block. Notice how the diagonal of $\boldsymbol{H}$ does not affect the objective of a partition—all its element are summed regardless of the choice of $\{\sigma(v_i)\}$. We are therefore free to set the diagonal as we please to, e.g, simplify the calculations or interpretation of the objective matrix.

The elements of the objective matrix can be defined directly in terms of the graph (e.g., favour bisection which cluster connected vertices), but could also incorporate some metadata not contained in it: Say a particular vertex represents Alice, and another Bob, and suppose that they like each other very much. We could set $h_{ab}^{(in)} = K \gg 1$ for this particular edge, and favour bisections which put them in the same block.

3.2 Example of a objective matrix for graph clustering

Before progressing further, it will be instructive to give a number of example of objective matrices $\boldsymbol{H}$. This will root the rest of our discussion in concrete examples.

Suppose, for instance, that we are given a *Barbell graph* (See Fig. 2). Moreover, suppose that we are asked to recover the two cliques of this graph. Of course, we could trivially solve this problem by hand, but let us construct an objective function that allows us to distinguish the bisection in two cliques. Notice how there is only 1 edge linking the two cliques, and that any other bisection of the graph will have more than 1 edge running between the blocks. We can therefore look for the cliques by identifying a bisection of the graph which minimize the number of edges running between blocks.

It turns out that this information is encoded in the combinatorial Laplacian $\boldsymbol{L}$ of the graph, which we now construct. To define the graph Laplacian, we first need to define the adjacency matrix $\boldsymbol{A}$. Simply put each vertex corresponds to a row / column of $\boldsymbol{A}$, and the element a_{ij} of $\mathbf{A}$ is taken to be equal to 1 if there is an edge between v_i and v_j, and 0 otherwise. If edges are undirected—as is the case here—then $\boldsymbol{A}$ is

[4]This choice is more or less arbitrary—we only to be able to construct a representation of the Kronecker delta. We could, for instance, rescale the variables [20].

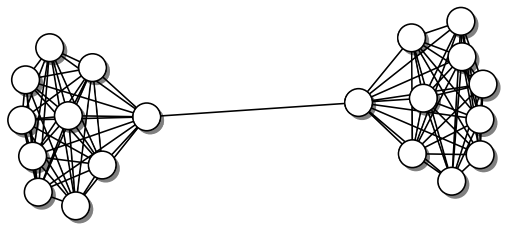


Figure 2: **Barbell graph $B(n_1, n_2)$ with $n_1 = 10$, $n_2 = 0$**. It consists of two cliques of n_1 vertices, connected by a path of length n_1. The bisection that minimizes the number of edges running between the two blocks is evidently the cut which removes the path between the blue and red cliques.

symmetric. The *degree* k_i of a vertex is the number of vertices to which is is connected. This information is also contained in $\boldsymbol{A}$:

$$k_i = \sum_{j=1}^{N} a_{ij} \ . \tag{6}$$

From the vector $\boldsymbol{k}$, we also construct the diagonal matrix of degrees $\boldsymbol{D} = \text{diag}(\boldsymbol{k})$, i.e., a matrix where k_i appears on the i^{th} element of the diagonal, and which is zero everywhere else. Combining these two matrices, we finally obtain the graph Laplacian

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \ . \tag{7}$$

This matrix is extensively studied. See, e.g., Refs. [7, 8] for a survey of the property of the Laplacian.

A popular approach is to take $\boldsymbol{L}$ to be our objective matrix $\boldsymbol{H}$. Intuitively, $\boldsymbol{L}$ gives a score of -1 to an edge that connects two vertices $v_i \in B_r, v_j \in B_r$ in the same block B_r (hereafter *internal edges*), and it does not count edges which connect two blocks (hereafter *external edges*). A minimum of the objective function f_{Lap} associated to $\boldsymbol{L}$ should therefore reveal a bisection with many internal edges, and, conversely, few external edges. This corresponds to the clustering objective enunciated above.

We can be more precise about this statement. Substituting the definition of the Laplacian in the the objective function, we find

$$f_{\text{Lap}} = \frac{1}{4}\boldsymbol{s}^T\boldsymbol{L}\boldsymbol{s} = \frac{1}{4}\boldsymbol{s}^T\boldsymbol{D}\boldsymbol{s} - \frac{1}{4}\boldsymbol{s}^T\boldsymbol{A}\boldsymbol{s} \tag{8}$$

Note that we here have $C = \frac{1}{4}\sum_{ij} h_{ij} = 0$ because the rows and columns of $\boldsymbol{L}$ sum to 0 [see Eq. (5)]. This will often be the case if construct $\boldsymbol{H}$ carefully. We can achieve $C = 0$ by choosing an appropriate diagonal that cancels out the off-diagonal elements in the computation of C [5]. But this need not be the case, since C is, after all, irrelevant from the clustering point of view.

The nice thing about the graph Laplacian is that it is easily interpretable. First, notice that

$$\frac{1}{4}\boldsymbol{s}^T\boldsymbol{D}\boldsymbol{s} = \frac{1}{4}\sum_{i=1}^{N} k_i s_i^2 = \frac{m}{2} \ , \tag{9}$$

where m is the number of edges in the graph. Second, if one defines $m(B_1, B_2)$ as the number of edges that

[5]Recall that the diagonal does not affect the ordering of partitions with respect to their score.

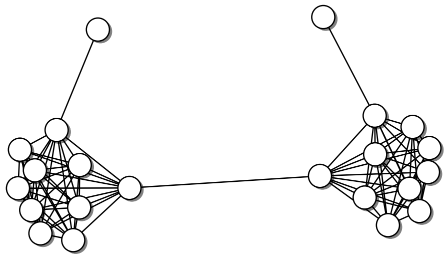


Figure 3: **Modified Barbell graph $B(n_1, n_2)$ with $n_1 = 10$, $n_2 = 0$**. It is the Barbell graph $B(10, 0)$ of Fig. 2, to which we attached two degree one vertices. This graph now admits three optima of f_{Lap}: A cut that isolates the degree vertices of degree ones is as good as the cut that separates the two cliques in different block.

connects the vertices of blocks B_1 and B_2, then it easy to see that

$$\frac{1}{4}\boldsymbol{s}^T\boldsymbol{A}\boldsymbol{s} = \frac{1}{4}\sum_{ij} s_i a_{ij} s_j \equiv \frac{1}{4}\left[\sum_{i\in B_1, j\in B_1} a_{ij} + \sum_{i\in B_2, j\in B_2} a_{ij} - \sum_{i\in B_1, j\in B_2} a_{ij}\right]$$
$$= \frac{m(B_1,B_1) + m(B_2,B_2) - m(B_1,B_2)}{2}\ . \tag{10}$$

Since $m - m(B_1, B_1) - m(B_2, B_2) = m(B_1, B_2)$ [we count all edges], it follows that f_{Lap} counts the number of external edges, which is known as the *cut-size* $m(B_1, B_2)$ of the bisection $\mathcal{B}(V) = \{B_1, B_2\}$ of $G(V, E)$. This shows rigorously that we can identify the two cliques of the Barbell graph of Fig. 2 by minimizing

$$f_{\text{Lap}} = \frac{1}{2}\boldsymbol{s}^T\boldsymbol{L}\boldsymbol{s} \tag{11}$$

over the the set of all possible bisections, i.e. over the set of all possible $\boldsymbol{s} \in \{-1, 1\}^N$. Alternatively, we could have used Eq. (10) directly. We would then seek to *maximize* this objective function over the set of all $\boldsymbol{s}$, because it encodes the *difference* between the number of internal and external edges, up to a multiplicative constant.

Let us note in passing that many standard objective functions have been defined over the course of the past decades. See, for instance Ref. [1] for a discussion of the adjacency matrix as an objective function, Ref. [20] for a review of the variants of the Laplacian, and Ref. [12, 19] for a discussion of the *modularity* matrix[6].

3.3 Constrained clustering

So far, the optimization problem can be summarized as

$$\text{Optimize } f(\{\sigma_i\}, G) = \boldsymbol{s}^T\boldsymbol{H}\boldsymbol{s} \text{ subject to } \boldsymbol{s} \in \{-1, 1\}^N\ . \tag{12}$$

(we have dropped all irrelevant constants). It will often be desirable to incorporate additional constraints in the optimization problem[7]. For instance, consider the modified Barbell graph of Fig. 3. This graph is exactly the same as the Barbell graph of Fig. 2, with the crucial difference that we added two vertices of degree one.

[6]The modularity matrix $\boldsymbol{Q} = \boldsymbol{A} - \langle\boldsymbol{A}\rangle_P$ is defined as the difference between the adjacency matrix $\boldsymbol{A}$, and the *expected* adjacency matrix $\langle\boldsymbol{A}\rangle_P$, assuming that G is drawn from an ensemble of random graphs P [9, 15].

[7]These constraints are actually what makes the problem NP-Hard [20]. The unconstrained problem generally admits an efficient algorithm. For instance, unconstrained min-cut is in P [10].

This trivial modification of the graph structure lead to a dramatic difference in the optimization landscape. Namely, with respect to the objective function f_{Lap}, a cut that isolates the vertices of degree one is just as good as the cut that puts the two cliques in different blocks. This example is admittedly contrived, but similar connectivity pattern often arise “in the wild”: Many relational datasets can be modelled by graphs which contain an overwhelming majority of vertices of low degree [3].

Intuitively, the bisection that we would rather find should consist of decently sized blocks, not a few vertices separated from the rest of the graph. A popular way of incorporating this type of constraints is to redefine the objective function altogether. For instance, the following re-normalized objective functions are often used [20]:

$$\widetilde{f}_{\text{Lap}} := \frac{f_{\text{Lap}}}{|B_1||B_2|} \,,$$
$$\bar{f}_{\text{Lap}} := \frac{f_{\text{Lap}}}{\text{vol}(B_1)\text{vol}(B_2)} \,.$$

Here $|B_r|$ is the number of vertices in B_r, and $\text{vol}(B_r)$ it the sum of the degrees of the vertices $v_i \in B_r$. It is easy to see that the factor $(xy)^{-1}$ has its minima at $x = y$, such that small / non-uniform blocks will be assigned large scores (recall that we are *minimizing* the objective function in this instance). The resulting “normalized” objective functions therefore favour bisections which are balanced with respect to the degree or size of the blocks.

Another alternative—which we shall use in this contribution—is to constrain $\boldsymbol{s}$ directly. Notice that if the blocks are perfectly balanced in terms of size, then

$$\boldsymbol{s}^T\mathbf{1} = 0 \tag{13}$$

where $\mathbf{1}$ is the length N vector of ones. In general, we will be content with blocks which are not exactly of the same size. Our formulation of the constrained optimization problem will therefore take on the form

$$\text{Optimize } f(\{\sigma_i\}, G) = \boldsymbol{s}^T\boldsymbol{H}\boldsymbol{s} \text{ subject to } \boldsymbol{s} \in \{-1, 1\}^N \text{ and } \boldsymbol{s}^T\mathbf{1} \leq \epsilon, \text{ with } \epsilon \geq 0 \,. \tag{14}$$

3.4 The spectral algorithm for graph bisection

So far, we have expressed the solution of the bisection problem in terms of the solution of the optimization problem of the quadratic form

$$f(\{\sigma_i\}, G) = \frac{1}{4}\boldsymbol{s}^T\boldsymbol{H}\boldsymbol{s} + C \tag{15}$$

over a constrained set of length N vectors $\boldsymbol{s} \in \{-1, 1\}^N$. Clearly, this problem is no easier than the original problem: We merely encoded the objective function in matrix form. It turns out, however, that dropping constraints turn the problem into an easy one, in the sense that there exists a polynomial time algorithm to solve it. If we drop the constraint $\boldsymbol{s}^T\mathbf{1} \leq \epsilon$, then variants of the min-cut / max-flow algorithm will solve the problem efficiently (see Ref. [10] for a discussion of these algorithms). More interestingly, if we drop the constraint $\boldsymbol{s} \in \{-1, 1\}^N$, then we can can find the optimal bisection with a spectral approach.

To see this, suppose that $\boldsymbol{x}$ is a vector in $\mathbb{R}^N$ and that x_i is no longer constrained to $\{-1, 1\}^N$. The relaxed objective function is simply $f = \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$. We then take $\boldsymbol{x}_i$ to be a normalized eigenvector of $\boldsymbol{H}$ and obtain

$$f = \boldsymbol{x}_i^T\boldsymbol{H}\boldsymbol{x}_i = \lambda_i\boldsymbol{x}_i^T\boldsymbol{x}_i = \lambda_i \,, \tag{16}$$

where we have used the fact that eigenvectors are normalized in the last step. For ordered eigenvectors (accounting for multiplicities),

$$\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N \tag{17}$$

this mean that the maximum of f equals λ_N and its minimum equals λ_1. The objective function is therefore maximized by the eigenvector $\boldsymbol{x}_N$ associated to λ_N, and minimized by the eigenvector $\boldsymbol{x}_1$ associated to

λ_1. Sometime, the extremal eigenvector does not satisfy the balance condition $\boldsymbol{x}^T\mathbf{1} \leq \epsilon$; We ignore this eigenvector and check if the second best eigenvector satisfies the condition, until we encounter a good vector[8]

Once we have obtained the optimal eigenvector $\boldsymbol{x}_i$, it is only a matter of finding the closest constrained solution, i.e. of obtained the "quenched" solution $\boldsymbol{s}$ from the the approximate solution $\boldsymbol{x}_i$. We can then read of the bisection from the quenched vector $\boldsymbol{s}$. If the graph contains a good bisection with respect to $\boldsymbol{H}$, it can be shown that elements of $\boldsymbol{x}_i$ will form two tight clusters in $\mathbb{R}$ [16, 20]. In idealized cases, these clusters are in fact centred around $\pm\alpha$, where α a function of the block size [18]. A powerful heuristic is therefore to round x_i to $s_i = -1$ if $x_i < 0$ and $s_i = 1$ otherwise [12]. Our algorithm is therefore the following

Algorithm 1: Spectral bisection

```
Input  : An objective matrix $\boldsymbol{H}$, a tolerance $\epsilon$, and a direction of optimization
Output: A bisection $\mathcal{B}(V) = \{B_1, B_2\}$ of the vertices of $G$
1  Set $\ell = 0$
2  if Maximize then
3  |   Compute the eigenvector $\boldsymbol{x}$ associated to the eigenvalue $\lambda_{N-\ell}$ of $\boldsymbol{H}$.
4  else
5  |   Compute the eigenvector $\boldsymbol{x}$ associated to the eigenvalue $\lambda_{1+\ell}$ of $\boldsymbol{H}$.
6  end
7  if $\boldsymbol{x}^T\mathbf{1} < \epsilon$ then
8  |   Set $\ell = \ell + 1$ and go back to statement 2.
9  end
10 if $x_i < 0$ then
11 |   Assign vertex $v_i$ to block $B_1$.
12 else
13 |   Assign vertex $v_i$ to block $B_2$.
14 end
15 return $\{B_1, B_2\}$,
```

Alternatively, one may use the K-Means algorithm during the quenching step [20]. It clusters points in the euclidean space by minimizing the difference between the points in a cluster and their centre. The function that is minimized is specifically

$$\mathrm{argmin}_{\mathcal{B}} \sum_{r=1}^{g} \sum_{i \in B_r} ||\boldsymbol{x}_i - \boldsymbol{\mu}_r||^2 \tag{18}$$

where $\boldsymbol{\mu}_r$ is the centre of cluster B_r in $\mathbb{R}^g$ (average of $\boldsymbol{x} \in B_r$), and g is the number of clusters. Clustering the result of a clustering algorithm might seem weird. The idea is that the spectral step separates the cluster nicely in the $\mathbb{R}^g$; The K-Means step will usually be efficient and accurate. Moreover, we can afford to use an approximated variant of K-Means with little impact on the quality of the clusters if the clusters are well separated in G, and therefore in $\mathbb{R}^g$ [16].

3.5 Continuous optimization perspective

We briefly touch on an alternative perspective which will help us build our intuition for the general case of Sec. 4 (partition in $g \geq 2$ blocks). Consider again the relax objective function

$$f = \boldsymbol{x}_i^T \boldsymbol{H} \boldsymbol{x}_i = \sum_{ij} h_{ij} x_i x_j \tag{19}$$

[8]If the constraint ϵ is too stringent, this algorithm will yield a vector which is not correlated with the *unrelaxed* bisection. The bulk of the information is contained in the extremal eigenvectors, and $\boldsymbol{x}_j$ for $1 \ll j \ll N$ will be a poor predictor of the optimal bisection [11].

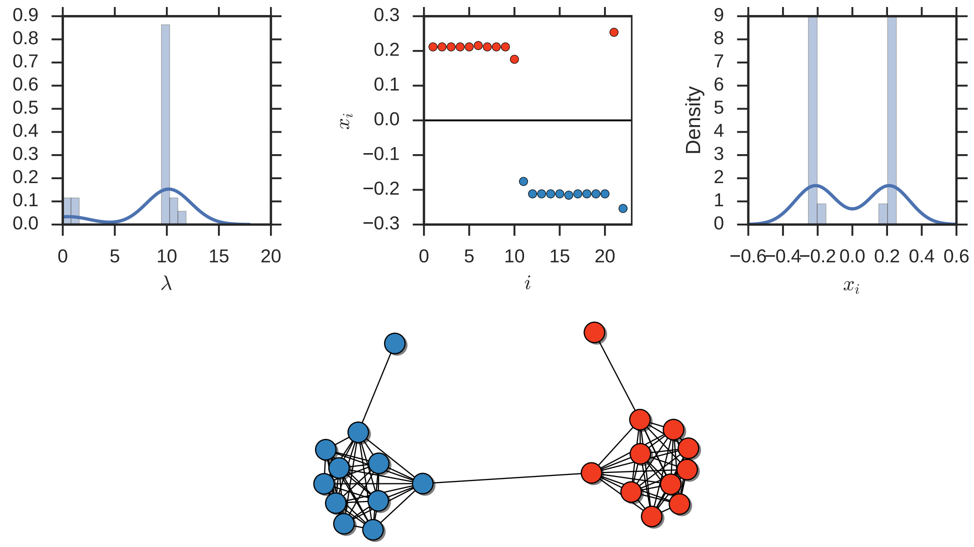


Figure 4: **Spectral clustering of the modified barbell graph.** We apply the spectral bisection algorithm to the Barbell graph, using the graph Laplacian $\boldsymbol{L}$ as the objective matrix. (*left*) Density of the eigenvalues of $\boldsymbol{L}$. (*middle*) Values of the elements of $\boldsymbol{x}_2$. We plot the value $x_i^{(2)}$ against the index of the nodes to which it is associated. (*right*) Distribution of the elements of $\boldsymbol{x}_2$ in $\mathbb{R}^1$. (*bottom*) Bisection corresponding to the quenched $\boldsymbol{s}$ associated to $\boldsymbol{x}$. Block assignments are indicated by the colour of the vertices. We used the sign heuristics to quench $\boldsymbol{x}$.

Since the elements of $\boldsymbol{x}$ are in $\mathbb{R}$ (and or no longer constrained values), the optimum of f will be located at the point where the derivatives with respect to $\{x_i\}$ are collectively equal to zero. Because there is a trivial solution at $x_i = 0\ \forall i$, we will further constraint the solution by asking that $\sum_i x_i^2 = \Delta$ where Δ is a positive constant. This can be incorporated in the optimization condition with a Langrange multiplier. Putting this all together, we have find that the optimal solutions will satisfy

$$\frac{\partial}{\partial x_r}\left[\sum_{ij} h_{ij}x_ix_j - \lambda\left(\sum_i x_i^2 - \Delta\right)\right] = 0 \qquad \text{where } \Delta > 0\,. \tag{20}$$

Using $\partial_{x_r}[x_i] = \delta_{ir}$, we find that

$$\sum_j H_{ij}x_j = \lambda x_i \tag{21}$$

which can be rewritten as

$$\boldsymbol{H}\boldsymbol{x} = \lambda\boldsymbol{x} \tag{22}$$

in matrix notation. This last equation tells us more rigorously that the optimal relaxed solution are eigenvectors of $\boldsymbol{H}$. The condition $\boldsymbol{x}^T\boldsymbol{1} > \epsilon$ allow us to reject trivial eigenvectors.

3.6 Illustration of spectral bisection

To illustrate the algorithm, we will apply the spectral bisection algorithm to the modified Barbell graph of Fig. 3, using the graph Laplacian $\boldsymbol{L}$ as an objective matrix. The results are shown in Fig. 4.

The fist panels shows the density of the eigenvalues of $\boldsymbol{L}$. The first eigenvector is associated to the eigenvalue $\lambda = 0$ and it does not satisfy the constraint $\boldsymbol{x}^T\boldsymbol{1} > \epsilon$ for $\epsilon < N$. This comes from the fact that the rows and columns of $\boldsymbol{L}$ sum to zero, such that $\mathbf{1}$ is always an eigenvector with associated eigenvalue $\lambda = 0$. We therefore look at the second smallest eigenvector for information about the bisection.

The second panels shows the values of the elements of $\boldsymbol{x}_2$ (see caption of the figure for details). The two extreme values on the right-hand of the panel (corresponding to $i = 21, 22$) are associated to the degree 1 vertices. There is also two entries of value closer to zero—they are associated to the vertices which participate in the path. The distance from 0 can be seen as a indicator of the node affiliation to a block; The degree one vertices are deeply within the block whereas the nodes close to the bisection are coupled to both blocks.

The third panels shows the distribution of the elements of $\boldsymbol{x}_2$ in $\mathbb{R}^1$. The clusters are neatly separated and K-Means will do a good job—just like the simple sign based heuristics. For clusterings in g blocks, we will have points in $\mathbb{R}^g$.

4 General case

We now turn to the general case of spectral clustering: We now look for separation of the graph in $g \geq 2$ blocks.

We represent the $g \geq$ blocks with the corner of a $g-1$ simplex vector. The indicator variables $s_i = \pm 1$ could have been seen as pointing towards the "corners" of the 1-simplex. This choice is somewhat arbitrary but will illustrate the method. We could, again, have rescaled each vector, or even used a higher dimensional representation. This can all be shown to be equivalent.

Two simplex vectors satisfy

$$\boldsymbol{s}_i^T \boldsymbol{s}_i = \begin{cases} 1 - \frac{1}{g} & \text{if } \sigma_i \neq \sigma_j \\ -\frac{1}{g} & \text{otherwise} \end{cases} . \tag{23}$$

This implies for the objective function:

$$\begin{aligned} f(\{\sigma_i\}, G) &= \frac{1}{2} \sum_{ij} [h_{ij}^{(in)}(G) - h_{ij}^{(out)}(G)] \delta_{\sigma_i \sigma_j} \\ &= \frac{1}{2} \sum_{ij} h_{ij} \left(\boldsymbol{s}_i^T \boldsymbol{s}_j + \frac{1}{g} \right) \\ &= \frac{1}{2} \sum_{ij} \boldsymbol{s}_i^T h_{ij} \boldsymbol{s}_j + C \\ &= \frac{1}{2} \mathrm{Tr}\big(\boldsymbol{S}^T \boldsymbol{H} \boldsymbol{S}\big) + C \end{aligned}$$

where $\boldsymbol{S}$ is the matrix whose i^{th} row is the simplex vector $\boldsymbol{s}_i$.

The balance constraint now reads

$$\boldsymbol{S}^T \mathbf{1} \leq \mathbf{1} \epsilon \qquad \epsilon \geq 0 \tag{24}$$

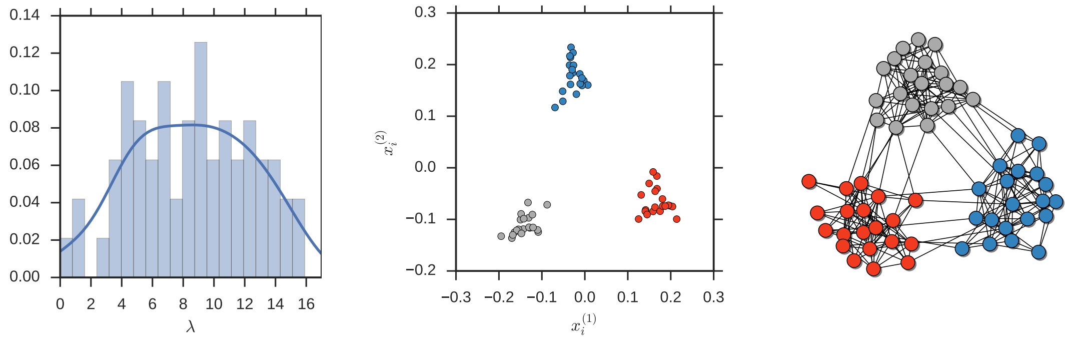
where $\mathbf{1}$ is the length $g-1$ vector of ones.

Again, we relax the solutions $\boldsymbol{S} \to \boldsymbol{X}$ and allow $\boldsymbol{X} \in \mathbb{R}^{g-1}$. Suppose that $\boldsymbol{X}$ is a matrix of normalized eigenvectors of $\boldsymbol{H}$ such that

$$\boldsymbol{H}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda} \tag{25}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. Then,

$$\begin{aligned} f(\{\sigma_i\}, G) &= \frac{1}{2} \mathrm{Tr}\big(\boldsymbol{X}^T \boldsymbol{H} \boldsymbol{X}\big) \\ &= \frac{1}{2} \mathrm{Tr}\big(\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{\Lambda}\big) \\ &= \frac{1}{2} \sum_{i=1}^{g-1} \lambda_i \end{aligned}$$



][h]

Figure 5: (*left*) Density of the eigenvalues of $\boldsymbol{L}$. (*centre*) Elements of $\boldsymbol{x}_2$ versus the element of $\boldsymbol{x}_3$ in $\mathbb{R}^2$. (*(*right) Clustered graph

4.1 Continuous optimization perspective

$$f = \frac{1}{2}\mathrm{Tr}\big(\boldsymbol{X}^T\boldsymbol{H}\boldsymbol{X}\big)$$

optima of f are found by setting $\{\partial_{X_{rs}}[f]\}$ to zero.

We avoid trivial solutions $X_{rs} = 0\ \forall i$, by asking $\boldsymbol{X}^T\boldsymbol{X} = \Delta\boldsymbol{I}$

$$\frac{\partial}{\partial \boldsymbol{X}}\left[\frac{1}{2}\mathrm{Tr}\big(\boldsymbol{X}^T\boldsymbol{H}\boldsymbol{X}\big) - \mathrm{Tr}\big(\boldsymbol{X}(\boldsymbol{\Lambda}+\Delta\boldsymbol{I})\boldsymbol{X}^T\big)\right] = 0 \qquad (\Delta > 0)$$

Here $\partial/\partial\boldsymbol{X}$ is understood as

$$\frac{\partial y}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{12}} & \cdots & \frac{\partial y}{\partial x_{1n}} \\ \frac{\partial y}{\partial x_{21}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{2n}} \\ \frac{\partial y}{\partial x_{31}} & \frac{\partial y}{\partial x_{32}} & \cdots & \frac{\partial y}{\partial x_{3n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{m1}} & \frac{\partial y}{\partial x_{m2}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix}, \tag{26}$$

with y a scalar.

Because we have the identities

$$\frac{\partial}{\partial \boldsymbol{X}}\mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{X}) = (\boldsymbol{A}+\boldsymbol{A}^T)\boldsymbol{X}$$
$$\frac{\partial}{\partial \boldsymbol{X}}\mathrm{Tr}(\boldsymbol{X}\boldsymbol{A}\boldsymbol{X}^T) = \boldsymbol{X}\big(\boldsymbol{A}+\boldsymbol{A}^T\big)$$

we find

$$\boldsymbol{H}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda} \tag{27}$$

The other steps of the algorithm are the same, we can apply the method out of the box to a graph which has 3 blocks.

4.2 Example

See Fig. 5.

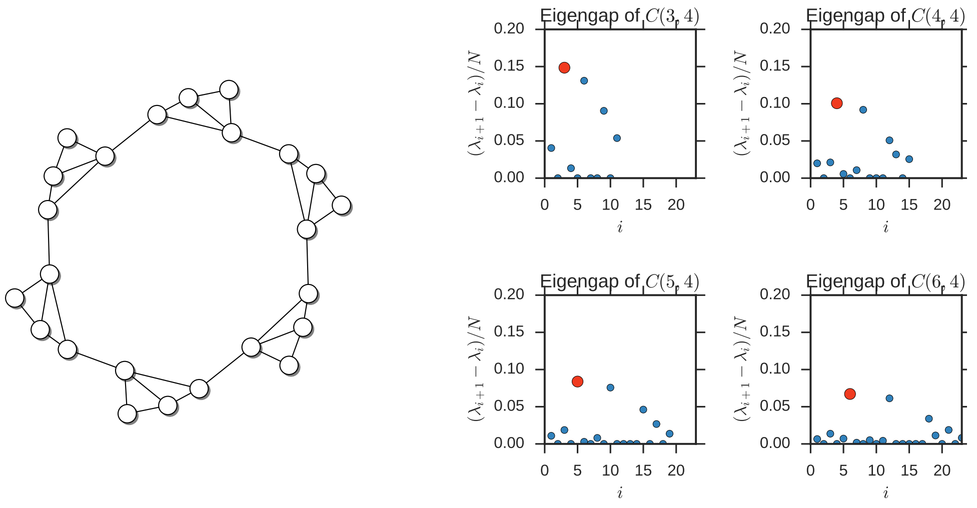


Figure 6: (*left*) Caveman graph $C(6,4)$ (*right*) Eigengap of $C(\ell,4)$, $\ell = 4,..,6$.

4.3 Eigengap statistics

The eigengap can predict the optimal number of clusters.

$$\Delta_i = \frac{(\lambda_{i+1} - \lambda_i)}{N}$$

The *largest* eigengap predicts the optimal number of clusters, and the other close gaps predict nearly optimal partitions.

5 Experiments

To be added.

6 Conclusion

- ◇ Constrained clustering is hard (NP-Hard)
- ◇ Relaxing the discrete constraint $\Longrightarrow$ spectral algorithm
- ◇ The spectral approach arises from the continuous optimization perspective
- ◇ The framework is general, arbitrary $\boldsymbol{H}$.

References

[1] E. R. Barnes, *An algorithm for partitionning the nodes of graphs*, SIAM J. Alg. Disc. Disc. Meth., 3 (1982), p. 541.

[2] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, *Maximizing modularity is hard*, arXiv:0608255, (2006).

[3] A. Clauset, C. R. Shalizi, and M. E. Newman, *Power-law distributions in empirical data*, SIAM review, 51 (2009), pp. 661–703.

[4] P. Crescenzi and V. Kann, *A Compendium of NP Optimization Problems*, Università di Roma "La Sapienza", 1995.

[5] S. Fortunato, *Community detection in graphs*, Phys. Rep., 486 (2010), pp. 75–174.

[6] D. Gfeller and P. De Los Rios, *Spectral coarse graining and synchronization in oscillator networks*, Phys. Rev. Lett., 100 (2008), p. 174104.

[7] R. Merris, *Laplacian matrices of graphs: a survey*, Linear algebra and its applications, 197 (1994), pp. 143–176.

[8] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann, *The laplacian spectrum of graphs*, Graph theory, combinatorics, and applications, 2 (1991), p. 12.

[9] M. Molloy and B. A. Reed, *A critical point for random graphs with a given degree sequence*, Rand. Struct. Alg., 6 (1995), pp. 161–180.

[10] C. Moore and S. Mertens, *The Nature of Computation*, Oxford University Press, 2011.

[11] R. R. Nadakuditi and M. E. J. Newman, *Graph spectra and the detectability of community structure in networks*, Phys. Rev. Lett., 108 (2012), p. 188701.

[12] M. E. J. Newman, *Modularity and community structure in networks*, PNAS, 103 (2006), pp. 8577–8582.

[13] ———, *Networks: An Introduction*, Oxford University Press, 2010.

[14] ———, *Spectral methods for community detection and graph partitioning*, Phys. Rev. E, 88 (2013), p. 042822.

[15] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, *Random graphs with arbitrary degree distributions and their applications*, Phys. Rev. E, 64 (2001), p. 026118.

[16] A. Y. Ng, M. I. Jordan, Y. Weiss, et al., *On spectral clustering: Analysis and an algorithm*, Advances in neural information processing systems, 2 (2002), pp. 849–856.

[17] T. P. Peixoto, *Eigenvalue spectra of modular networks*, Phys. Rev. Lett., 111 (2013), p. 098701.

[18] M. A. Riolo and M. E. J. Newman, *First-principles multiway spectral partitioning of graphs*, J. Complex Netw., 2 (2014), pp. 121–140.

[19] V. A. Traag, P. Van Dooren, and Y. Nesterov, *Narrow scope for resolution-limit-free community detection*, Phys. Rev. E, 84 (2011), p. 016114.

[20] U. Von Luxburg, *A tutorial on spectral clustering*, Statistics and computing, 17 (2007), pp. 395–416.

[21] X. Zhang and M. Newman, *Multiway spectral community detection in networks*, Phys. Rev. E, 92 (2015), p. 052808.