Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Network Theory
Lecture 4.07

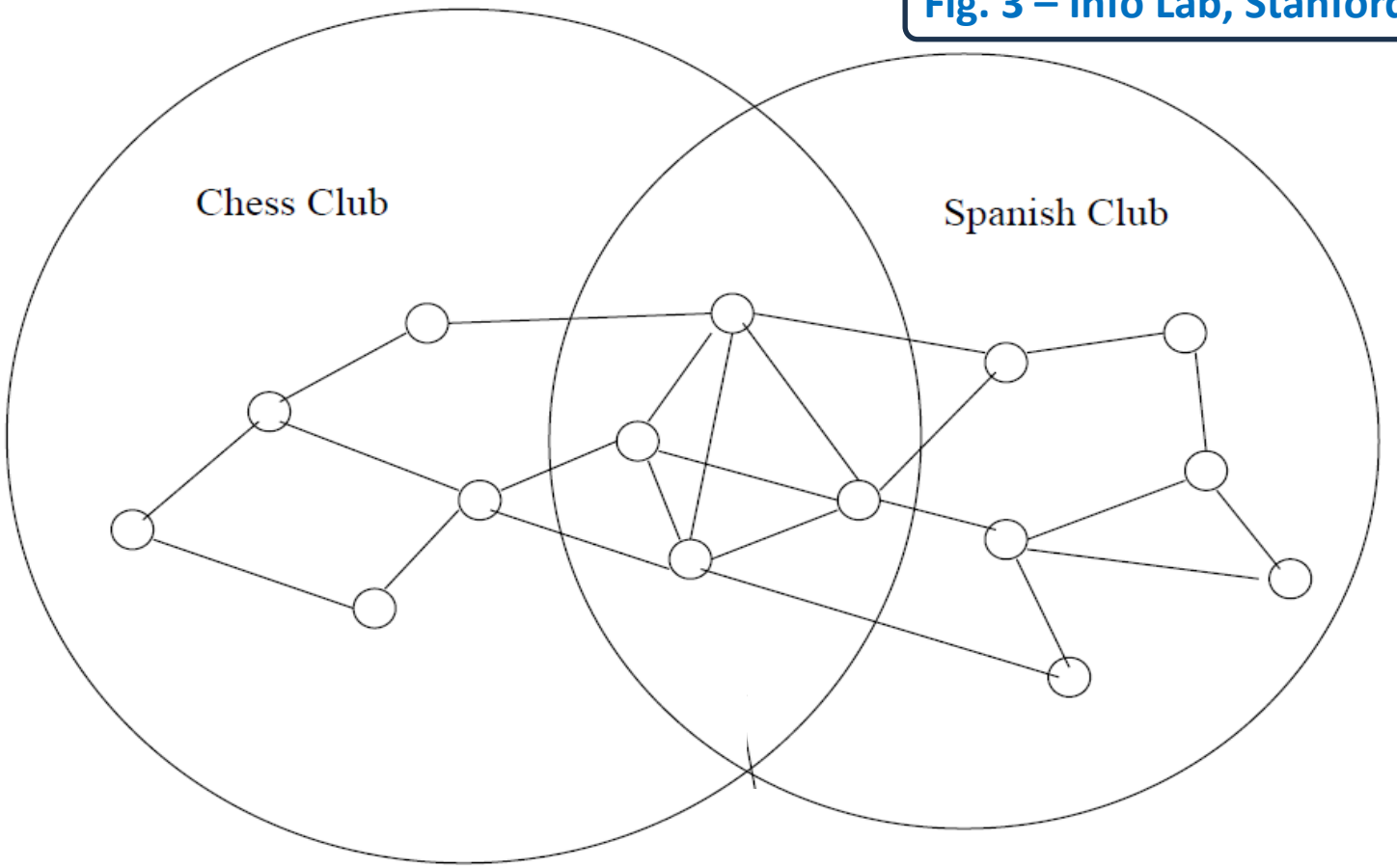
EEU45C09 / EEP55C09
Self Organising Technological Networks

Nicola Marchetti
nicola.marchetti@tcd.ie

Community detection

Graph partitioning and community detection both refer to the division of vertices of a network into groups according to the pattern of edges in the network.

Fig. 3 – Info Lab, Stanford Uni



Community detection

By constructing a network like that in Fig. 3 and then observing its clustered structure, one can deduce the existence of groups within the department.

The ability to discover groups or cluster in a network can be a useful tool for revealing structure and organisation within the network at a scale larger than the single vertex.

In this example, the network is small enough that we could probably identify the communities by eye, however, for much larger or denser networks we need numerical tools.

Graph partitioning vs Community Detection

There are two general classes of techniques which can be used to separate a network into groups or cluster:

- Graph Partitioning
- Community Detection

Graph Partitioning: relies on the experimenter fixing the number and size of groups. (Node partitioning in parallel numerical simulations.)
Community Detection: leaves the number and size of groups unspecified. (Find the natural fault lines along which a network separates.)

Graph Partitioning

Graph partitioning is dividing the network into two groups, fixing the number and size of groups.

This is an easy problem to state, however, in practice can be very difficult.

The simplest graph partitioning is the division of the network into just two parts called graph bisection. This division is performed in such a way as to minimise the number of edges cut by the partition.

Graph Partitioning

An exhaustive search of looking through all divisions of the network into two parts and choosing the one with the smallest cut size is very computationally costly.

The number of ways of dividing the network of n nodes into two groups of n_1 and n_2 nodes is

$$\frac{n!}{n_1!n_2!}.$$

By Stirling's approximation $x! = \sqrt{2\pi x}(x/e)^x$. Therefore,

$$\frac{n!}{n_1!n_2!} \approx \frac{\sqrt{2\pi n}(n/e)^n}{\sqrt{2\pi n_1}(n_1/e)^{n_1}\sqrt{2\pi n_2}(n_2/e)^{n_2}} \approx \frac{n^{n+1/2}}{n_1^{n_1+1/2}n_2^{n_2+1/2}}. \quad (3)$$

Graph Partitioning

If we want to divide the network into two parts of equal size, then Eq. 3 is roughly,

$$\frac{n^{n+1/2}}{(n/2)^{n+1}} = \frac{2^{n+1}}{\sqrt{n}}.$$

Therefore the time required to look through all of these divisions scales roughly exponentially with the size of the network.

Therefore, we must look to heuristic methods such as the Kernighan-Lin algorithm.

Kernighan-Lin Algorithm

The Kernighan-Lin algorithm is one of the best known graph bisection methods. It works on the premise that we try to partition the graph into two disjoint subsets A and B in a way that minimises the number of the sum T of edges between the two subsets.

For a connected graph, divide the vertices into two groups of the required size A and B (often two groups of equal size).

Let I_a be the number of edges between node $a \in A$ and other nodes in A. Also let E_a be the number of edges between node $a \in A$ and nodes in B.

Furthermore, let $D_a = E_a - I_a$

Kernighan-Lin Algorithm

Now, we aim to swap two nodes a and b, to reduce the number of external edges and increase the number of internal edges.

Prior to the swap, the total external cost T between A and B is

$$T = z + E_a + E_b - c_{ab},$$

where z is the total cost between A and B that does not involve a and b. Also let c_{ab} be the adjacency matrix element to prevent over-counting.

After exchanging a and b, the new external cost is

$$T' = z + I_a + I_b + c_{ab}.$$

The difference in these terms is the gain g_{ab} due to this swap,

$$g_{ab} = T - T' = D_a + D_b - 2c_{ab}$$

Kernighan-Lin Algorithm

For all $a \in A$ and $b \in B$, calculate g_{ab}, which is an order $O(N^2)$ operation. Next, select the largest gain $\hat{g}_1 = \max g_{ab}$, corresponding to some swap of nodes $x \in A$ and $y \in B$.

Next, recalculate the differences D_a and D_b excluding the nodes x and y,

Recalculate g_{ab} and find its maximum ($\hat{g}_2$).

Finally, perform the partial sum which maximises

$$\sum_{i=1}^{k \leq N} \hat{g}_i.$$

The first k pairs of nodes are then swapped and a KL sweep has been completed.

Kernighan-Lin Algorithm

However, this has a complexity of roughly $O(N^3)$. For this reason we will focus on Spectral Clustering.

What does Spectral mean?

The spectrum of a matrix is its set of eigenvalues.

The eigenvalues of an $n \times n$ matrix M are the scalar values λ which satisfy the following equation for x, a non-zero $n \times 1$ vector,

$$Mx = \lambda x \tag{4}$$

To find the eigenvalues, λ, of M we can rewrite Eq. 4,

$$Mx - \lambda Ix = 0, \tag{5}$$

$$(M - \lambda I)x = 0, \tag{6}$$

where I is the identity matrix. Let $S = M - \lambda I$. Since x is non-zero, S must be singular,

$$\det(M - \lambda I) = 0.$$

Diffusion

Diffusion is the movement of gas from regions of high density to low by the difference in pressure.

We can consider diffusion on a network also, as a simple model of spread across a network. This spread could be:

- a substance,
- an idea,
- an opinion,
- a disease.

Diffusion

In order to monitor this spreading, we monitor the amount ψ_i of some commodity at vertex i in the network. The commodity can move only along the edges, flowing from one vertex j to an adjacent one i at a rate $C(\psi_j - \psi_i)$, where C is called the diffusion constant. (Contains all of the real-world units.)

In a very small interval of time, the amount of this commodity flowing from j to i is $C(\psi_j - \psi_i)dt$.

Diffusion

With this notation, in our graph we can say that the rate at which ψ_i is changing is given by,

$$\frac{d\psi_i}{dt} = C \sum_j A_{ij}(\psi_j - \psi_i), \tag{1}$$

which is the rate at which the commodity at all neighbours of i is flowing into (or out of) i. This condition on the neighbours is enforced by the adjacency matrix A_{ij}.

Diffusion

If we multiply the adjacency matrix elements through the brackets from Eq. 1, we find:

$$\frac{d\psi_i}{dt} = C\sum_j A_{ij}\psi_j - C\psi_i\sum_j A_{ij} \tag{2}$$

$$= C\sum_j A_{ij}\psi_j - C\psi_i k_i \tag{3}$$

$$= C\sum_j (A_{ij} - \delta_{ij}k_i)\psi_j, \tag{4}$$

where we use the fact that $k_i = \sum_j A_{ij}$.

Diffusion

Finally, we can rewrite Eq. 4 where ψ is the vector whose components are numbers ψ_i,

$$\frac{d\psi}{dt} = C(A - D)\psi$$

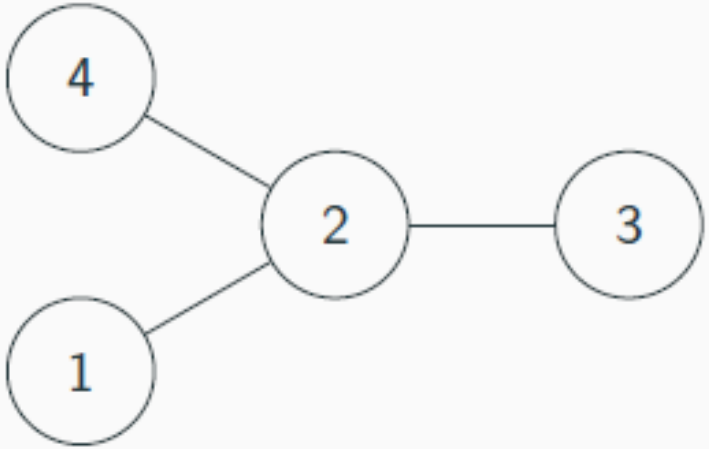
A is the adjacency matrix of the graph in question and D is the degree matrix, which has the degree of node i in the i-th diagonal position,

$$D_{ii} = k_i$$

and zeros otherwise.

Diffusion

Explicitly, for a given graph,



the degree and adjacency matrices are,

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Graph Laplacian

It is common the define the new matrix,

$$L = D - A,$$

and Eq. 4 takes the form,

$$\frac{d\psi}{dt} + CL\psi = 0$$

which has a similar form as the ordinary diffusing equation for a gas, except the Laplacian operator ∇^2 has been replaced by L. We will see how we can use the graph Laplacian L not only to describe diffusion, but also to help with graph clustering.

Graph Laplacian: Properties of the Eigenvalues

Here, we will discuss some properties of the eigenvalues of the Graph Laplacian. The Graph Laplacian is a real, symmetric matrix and therefore has real eigenvalues.

In fact, the eigenvalues are also nonnegative

$$\lambda_i \geq 0,$$

for all i.

The smallest eigenvalue corresponds to the eigenvalue equation, where $x = (1, 1, \ldots 1)$,

$$L_i x = \sum_j L_{ij} \times 1 = \sum_j (\delta_{ij} k_i - A_{ij}) = k_i - \sum_j A_{ij} = k_i - k_i = 0.$$

Graph Laplacian: Properties of the Eigenvalues

In vector notation, $L \cdot 1 = 0$. Thus the vector $(1, 1, 1 \ldots 1)$ is always an eigenvector of the graph Laplacian with eigenvalue zero. Since there are no negative eigenvalues, this is the smallest eigenvalue.

If we number the n eigenvalues of the Laplacian in ascending order,

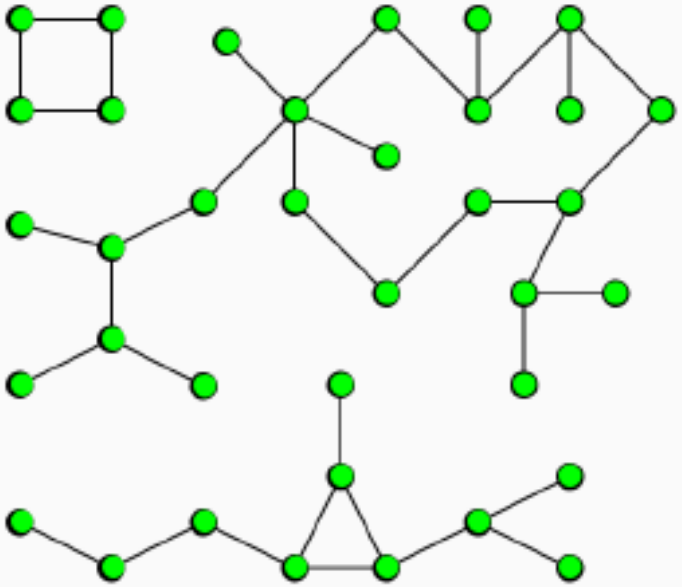
$$\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n,$$

we have $\lambda_1 = 0$.

Note that the determinant of a matrix is the product of its eigenvalues. Therefore the determinant of L is always zero and by definition, this make L singular (non-invertible).

Graph Laplacian: Components and Algebraic Connectivity

Suppose we have a network which is divided into c different components of sizes $n_1, n_2, \ldots n_c$. For simplicity, let's make the n_1 to be the number of vertices in component 1.



With this choice, the Laplacian of the network will be a block diagonal matrix.

Graph Laplacian: Components and Algebraic Connectivity

Along the diagonal, the Laplacian has the degree of the vertices in that component and -1 in each position corresponding to an edge.

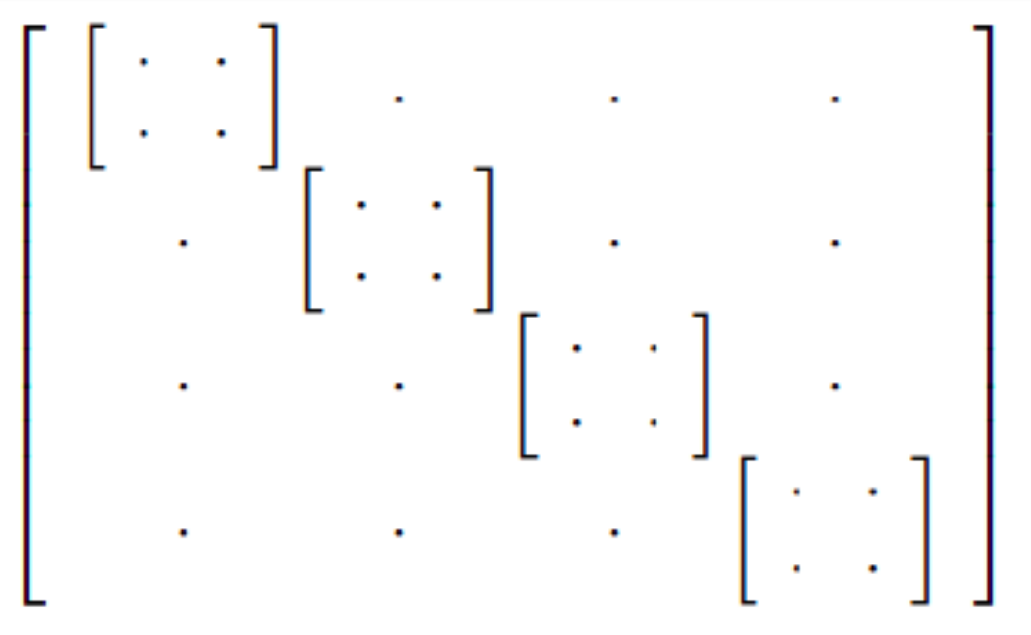


Figure 1: The block diagonal form of the graph Laplacian will have square blocks along the diagonal and zeros elsewhere.

Graph Laplacian: Components and Algebraic Connectivity

In this form, we can immediately (by inspection) write down c different eigenvectors of L with eigenvalue 0. These vectors have ones in all the positions corresponding to vertices in a single component and zeros elsewhere.

For instance

$$v = (\underbrace{1, 1, 1}_{n_1 \text{ ones}}, \ldots, 0, 0, 0, \ldots, 0)$$

is an eigenvector with eigenvalue zero.

Thus in a network with c components there are always at least c eigenvectors with eigenvalue zero.

Graph Laplacian: Components and Algebraic Connectivity

In fact, it can be rigorously shown that the number of zero eigenvalues is always exactly equal to the number of components.

Note that the vector $v = (1, 1, \ldots, 1)$ is just equal to the sum of the other c eigenvectors and not an independent eigenvector itself.

From what we learned earlier, we can say that the second eigenvalue of the graph Laplacian λ_2 is non-zero if and only if the network is connected (a single component).

For this reason, the second eigenvalue of the Laplacian is called the algebraic connectivity.

Spectral Clustering - Motivation

The Kernighan-Lin algorithm can be used as a heuristic method to partition a graph into two subgraphs of sizes n_1 and n_2 nodes, where $n = n_1 + n_2$. However, the algorithm scales like $O(n^3)$ and therefore is only useful up to a few hundred or thousand nodes.

Spectral clustering, developed by Fiedler, uses the matrix properties of the graph Laplacian. In this description, we will divide the graph into two parts of specified sizes, analogous to the Kernighan-Lin.

Set-up

Consider a network with n vertices and m edges and a division of these vertices into two groups. The cut-size R for this division (the number of edges between the two groups) is,

$$R = \frac{1}{2} \sum_{i,j \text{in different groups}} A_{ij}.$$

Now, let us define some quantity s_i which represents a division on the network

$$s_i = \begin{cases} +1 & \text{if vertex } i \text{ belongs to group 1,} \\ -1 & \text{if vertex } i \text{ belongs to group -1.} \end{cases}$$

Cut-Size

Then we can define a quantity,

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in different groups,} \\ 0 & \text{if } i \text{ and } j \text{ are in the same group.} \end{cases}$$

Thus we can rewrite the cut-set,

$$R = \frac{1}{4} \sum_{ij} A_{ij}(1 - s_i s_j).$$

The first term in the sum,

$$\sum_{ij} A_{ij} = \sum_i k_i = \sum_i k_i s_i^2 = \sum_{ij} k_i \delta_{ij} s_i s_j,$$

since $\sum_j A_{ij} = k_i$ and $s_i^2 = 1$.

Cut-Size

Substituting back into the cut-size definition, we have

$$R = \frac{1}{4}\sum_{ij}(k_i\delta_{ij} - A_{ij})s_i s_j = \frac{1}{4}\sum_{ij} L_{ij}s_i s_j,$$

where L_{ij} is the ij-th element of the graph Laplacian.

This equation can be rewritten as a matrix equation,

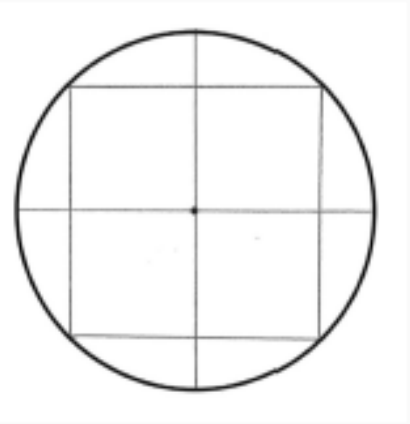
$$R = \frac{1}{4}s^T L s,$$

where s is the vector with elements s_i.

The approach is now to minimise R by finding the vectors s for L which minimise this product.

Relaxation of the s vectors

This minimisation is quite hard however, since the elements of s must take values $+1$ or -1. To make this somewhat easier, we can relax this constraint a little to find approximate solution vectors for s which minimises R.



This solution vector must have length $\sqrt{n}$.

The constraints of s are that the elements can take only values $s_i \in [+1, -1]$, which means they lie on the 2^n corners of a hypercube.

Relaxation of the s vectors

If we allow the s vectors to take any value on the hypersphere bounding the hypercube, we enforce,

$$\sum_i s_i^2 = n.$$

The second constraint is that there should be $n_1 \times (+1)$ values and $n_2 \times (-1)$ values. Therefore, we must constrain the values of s such that,

$$\sum_i s_i = n_1 - n_2$$

Using the method of Lagrange multipliers to minimise R, with these two constraints, we find that

$$R = \frac{n_1 n_2}{n} \lambda.$$

Minimising R

Therefore, in order to minimise R, we must use the smallest eigenvalue possible. However, during the derivation, we find that we may not use the smallest eigenvalue, λ_1, corresponding to the eigenvectors, $v_1 = (1, 1, \ldots, 1)$. Therefore, we use the second eigenvalue, λ_2.

Thus, after a relatively lengthy derivation, we find that we must find the eigenvector v_2 corresponding to the eigenvalue λ_2 and place the n_1 most positive values in v_2 into group one and all others into group 2 in order to minimise the cut set R.

This eigenvector, v_2, is often called the Fiedler vector in honour of the creator of this method.

Ackowledgement

- Dr Neal McBride, Arista Networks