Spectral Graph Approximation

Clemens Damke

1 Introduction

With the rise of Big Data applications over the recent years, working with large graph structures also became more important. Algorithms like PageRank or spectral clustering are commonly used to analyze the web graph or social networks. In order to run such algorithms on large graphs however, optimizations are required.

For this purpose we will specifically look at *graph coarsening*. Coarsening reduces the size of a given graph while preserving its overall structure via some notion of graph similarity that will be explained later. Graph algorithms can then be run on the smaller coarsened graph. Afterwards the algorithm's result for the coarsened graph can be iteratively refined to obtain an approximate result for the original graph. Figure 1 illustrates this approach.

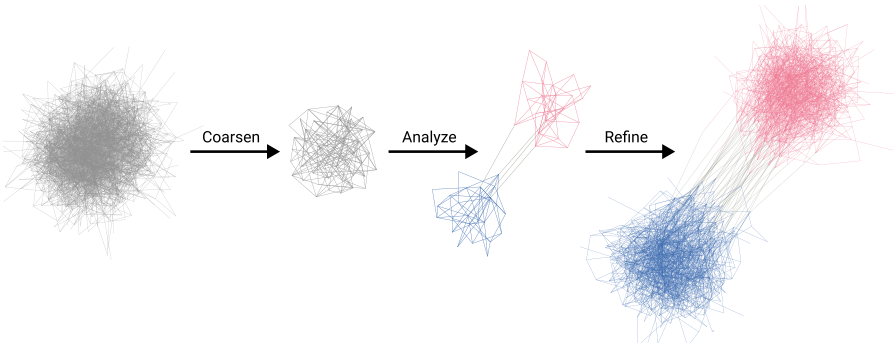


Figure 1: Using graph coarsening to speed up graph algorithms, e.g. Min-Cut.

The goal of this paper is to show how graph coarsening works and how it affects the results of graph algorithms. We will discuss this topic in the upcoming three sections, each of which aims to answer one main question:

1. *How can the structural properties of a graph be formally described?* To describe graph coarsening and its effects, the similarity between a graph G and its coarsened version G_c has to be quantified. We begin with an introduction to spectral graph theory. It allows us to describe the structure of graphs and provides the notion of the *graph spectrum*, a way to describe the overall structure of graphs.

2. *How does graph coarsening work?* Based on the notion of the graph spectrum, we will formally define the graph coarsening operation and describe a randomized coarsening algorithm.

3. *How does coarsening affect the result of graph algorithms?* Finally we will put bounds on how much the described coarsening algorithm is expected to change a graph's spectrum and what this implies for the spectral clustering algorithm.

2 Spectral Graph Theory

We start with an introduction to spectral graph theory, which will allow us to characterize and compare the structure of graphs. Graphs are most commonly described in their vertex base, i.e. the connection strength w_{ij} of pairs (v_i, v_j) of vertices.

$$G := (\mathcal{V}, \mathcal{E}, W) \quad \text{with } W \in \mathbb{R}^{N \times N}, N := |\mathcal{V}| \tag{1}$$

In this paper we will only consider undirected graphs with non-negative real weights and without self-loops ($\forall v_i : w_{ii} = 0$).

The core idea of spectral graph theory is to perform a change of basis of W and describe graphs in terms of their so called *spectral base* instead of their *vertex base*. To see what this means, we interpret the adjacency/weight matrix W as a linear operator that operates on so called signals $x \in \mathbb{R}^N$. A signal x can be interpreted as a function $x : \mathcal{V} \to \mathbb{R}$ that assigns a signal strength to each vertex. By applying Wx the given signal strengths x_i are shifted according to the connection strenghts w_{ij} to neighboring vertices v_j. This interpretation of graphs is very similar to that of Markov chains where signals represent probability distributions.

2.1 Relating Graph Signals to Real-valued Functions

Let us now compare discrete graph signals $x : \mathcal{V} \to \mathbb{R}$ to continuous real-valued functions $f : \mathbb{R} \to \mathbb{R}$. Both only differ in their domain. The domain $\mathbb{R}$ of f has an inherent structure, the real number line, which provides a strict ordering of its elements and a notion of distance between them. The domain $\mathcal{V}$ of x however has no such inherent structure, i.e. $v_1 < v_2$ for two vertices v_1, v_2 does not have a clear meaning. The structure of $\mathcal{V}$ fully depends on the graph G that is acting on it. Intuitively graph signals can thus be understood as a discretized generalization of real-valued functions, where the underlying structure of the input domain is not fixed but can be freely chosen.

Figure 2 shows that all real-valued functions f can be seen as signals x of the graph described by the real number line[1]. Both, real-valued functions and graph signals, can be described as vectors in their time and vertex base respectively:

$$f = \int_{\mathbb{R}} f(t) b_t dt \Rightarrow \langle b_t, f \rangle = f(t) \quad \text{and} \quad x = \sum_{i=1}^{N} x_i b_i \Rightarrow \langle b_i, x \rangle = x_i \tag{2}$$

[1]Technically this is not correct, since $\mathbb{R}$ is continuous whereas all vertex sets $\mathcal{V}$ have to be discrete. To build an intuition for graph signals, this detail can however be ignored.

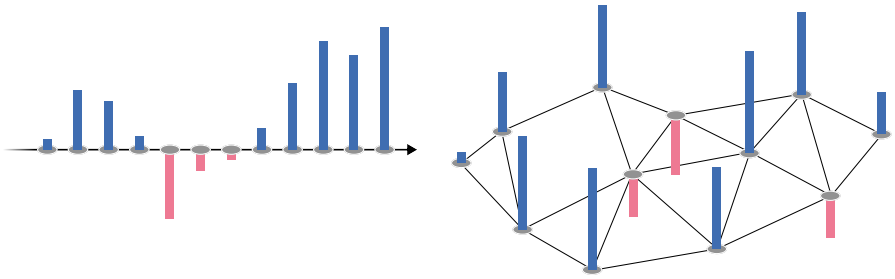


Figure 2: Illustration of how the discretized real number line can be interpreted as an infinite linear graph, compared to some arbitrary finite non-linear graph. The bars show the signal strengths $f(t)$ and x_i at the vertices b_t and b_i respectively.

In this equation $\{b_i\}_{v_i \in \mathcal{V}}$ denotes the standard basis of the adjacency matrix W, i.e. $b_i \in \mathbb{R}^N$ has a signal strength of 0 for all $v_{j \neq i}$ and a signal strength of 1 for v_i. Similarly $\{b_t\}_{t \in \mathbb{R}}$ denotes the infinite dimensional standard basis of the space of real functions, where $\langle b_t, \cdot \rangle := \delta_t(\cdot)$, with δ denoting the Dirac delta function.

2.2 Extending the Fourier Transform to Graphs

As mentioned at the beginning of this section, the core idea of spectral graph theory is to express graph signal vectors x in the spectral basis $\{u_i\}_{i=1}^N$ instead of the standard vertex basis $\{b_i\}_{v_i \in \mathcal{V}}$. This idea is analogous to the classical Fourier transform on real-valued functions. In the first step we are now going to give an intuition for the classical Fourier transform. Afterwards we will extend this intuition to the graph domain.

The Fourier basis $\{u_\xi\}_{\xi \in \mathbb{R}}$ describes a function f not in terms of the values $f(t) = \langle b_t, f \rangle$ it takes at position t but instead in terms of the amplitudes $\hat{f}(\xi) = \langle u_\xi, f \rangle$ of the complex exponentials $u_\xi(t) = e^{2\pi i \xi t}$. Using this perspective, the Fourier transform can be viewed as a change of basis operator from the orthonormal standard basis $\{b_t\}_{t \in \mathbb{R}}$ to the also orthonormal Fourier basis $\{u_\xi\}_{\xi \in \mathbb{R}}$:

$$f = \int_{\mathbb{R}} \underbrace{\langle b_t, f \rangle}_{f(t)} b_t dt \quad = \quad \int_{\mathbb{R}} \underbrace{\langle u_\xi, f \rangle}_{\hat{f}(\xi)} u_\xi d\xi \tag{3}$$

This property of the Fourier transform by itself is not special; in fact there are infinitely many orthonormal bases on the space of real-valued functions. The distinguishing property of $\{u_\xi\}_{\xi \in \mathbb{R}}$, making it useful in many domains, is that it is an *eigenbasis* of the *Laplacian* Δ, i.e. $\Delta u_\xi = \lambda_\xi u_\xi$ for the eigenvalue $\lambda_\xi \in \mathbb{R}$. The Laplacian is a multidimensional generalization of the second derivative. For real-valued functions this boils down to $\Delta u_\xi = \frac{\partial^2}{\partial t^2} u_\xi = -(2\pi\xi)^2 e^{2\pi i \xi t}$ with the eigenvalue $\lambda_\xi = -(2\pi\xi)^2$ depending solely on the frequency ξ. The original motivation to express functions in terms of the Laplacian's eigenbasis was to solve the physical heat equation[2]. For

[2]More generally the Fourier basis turns out to be meaningful for all *linear time-invariant* (LTI) systems, of which the heat equations are only one instance.

this reason it is a useful intuition to think about signals as temperature distributions that will converge to an equilibrium state over time. This intuition also works in the graph setting where heat only flows between neighboring vertices in proportion to their connection strengths.

Now that we have looked at the Fourier transform of real-valued functions, we will extend this notion to graphs. Just like the classical Fourier transform, the graph Fourier transform performs a change of basis of a signal x from the vertex basis $\{b_i\}_{v_i \in \mathcal{V}}$ to the Fourier basis $\{u_k\}_{k=1}^N$. This Fourier basis again is characterized by it being an eigenbasis of the Laplacian, more specifically the so called *combinatorial graph Laplacian* L in this case:

$$L := D - W \quad \text{with the degree matrix } D := \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_N \end{pmatrix}, d_i := \sum_{j=1}^{N} w_{ij} \tag{4}$$

Using this definition, the application Lx is a discrete generalized analogue of the second derivative $\frac{\partial^2}{\partial t^2} f$. Putting both Laplacian variants, $\frac{\partial^2}{\partial t^2}$ and L, side-by-side gives an intuition for why this is the case:

$$\left(-\frac{\partial^2}{\partial t^2} f\right)(t) = \lim_{h \to 0} \frac{1}{h^2} (\underbrace{f(t) - f(t-h)}_{\Delta_{t,t-h}} + \underbrace{f(t) - f(t+h)}_{\Delta_{t,t+h}}) \quad \Big| \quad (Lx)_i = \sum_{j=1}^{N} w_{ij} \underbrace{(x_i - x_j)}_{\Delta_{i,j}} \tag{5}$$

The second derivative of a function f essentially just averages the differences in signal strength in the neighborhood of a point t. For real-valued functions this neighborhood only consists of the two infinitesimally close points to the left and to the right of t, i.e. $t - h$ and $t + h$. The graph Laplacian represents the same operation, where each point/vertex might however have more than two neighbors that need to be averaged.

Based on the graph Laplacian L we just defined, the graph Fourier basis $\{u_k\}_{k=1}^N$ is just the set of eigenvectors of L. In the rest of this paper we will assume that these eigenvectors u_k are sorted in ascending order w.r.t. their eigenvalues λ_k, i.e. $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. Figure 3 shows how this definition generalizes the Fourier basis from the real number line to an arbitrary graph. It also shows that the eigenvalues λ_k of the graph Fourier basis encode some notion of frequency, just like they do for the classical Fourier basis. Analogous to the classical Fourier transform, the eigenvalues of a graph's Laplacian are therefore also called its *spectrum*.

2.3 Spectral Properties of Graphs

We will now give an intuition for the relation between the spectrum of a graph and its structural properties[3]. For any graph the smallest eigenvalue always is $\lambda_1 = 0$ with the associated eigenvector $u_1 = \frac{1}{\sqrt{N}}(1, \ldots, 1)^\top$ being a uniform signal over all vertices. Using the heat analogy, this simply means that any system in which everything has the same temperature is in an equilibrium state and no heat is flowing. More generally for graphs with m separate connected components the first m eigenvalues are all 0 since each component can have its own equilibrium temperature without causing heat flow. If the spectrum of a graph is known, this fact can be used to quickly determine whether a graph is connected or not.

[3]Only a general overview will be given. For a more detailed discussion we refer to Shuman et al. [3].

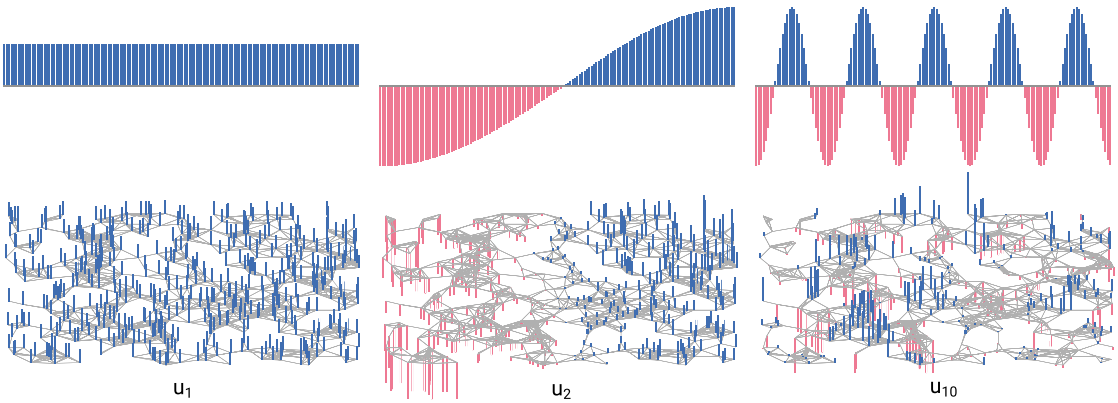


Figure 3: Comparison between the basis functions/vectors of the classical Fourier transform and the graph Fourier transform. For the eigenfunctions on the upper half only the real cosine components of the complex exponentials are shown. BASED ON: [3]

Another meaningful eigenvalue is λ_2 and its eigenvector u_2, also called the *Fiedler value* and *Fiedler vector* respectively. As we have just seen, a Fiedler value of $\lambda_2 = 0$ means that a graph is not connected. More generally the Fielder value can be interpreted as a measure of overall graph connectivity[4]. If there are two clusters of vertices $\mathcal{V}_+$ and $\mathcal{V}_-$ in a graph that are connected via relatively few edges (compared to the overall number of edges), the Fielder value is small; for well connected graphs on the other hand the Fiedler value is large. Using the heat analogy, the Fielder value essentially measures the "width" of a bottleneck between two parts of a graph through which heat will only flow very slowly or even not at all in case of $\lambda_2 = 0$. The partitioning of vertices into $\mathcal{V}_+$ and $\mathcal{V}_-$ can be retrieved via the Fiedler vector u_2; vertices with a positive signal strength in u_2 are in $\mathcal{V}_+$, the other vertices are in $\mathcal{V}_-$. The so called *spectral clustering* algorithm is based on this relationship. Figure 3 shows how the Fiedler vector u_2 partitions a graph via the signs of the vertex signal strengths.

As we have just seen, expressing a graph G in terms of the spectrum of its Laplacian L gives access to graph characteristics like its overall connectivity. Similarly to how the low-frequency components of the Fourier transform of a function represent the overall shape of that function, the eigenvectors associated with the small eigenvalues of L represent overall characteristics of G. The details of a function or graph on the other hand are encoded in the high-frequency components of its Fourier transform, i.e. the eigenvectors associated with large eigenvalues. This perspective already hints at how spectral graph analysis might be useful to approximate graphs.

3 Graph Coarsening

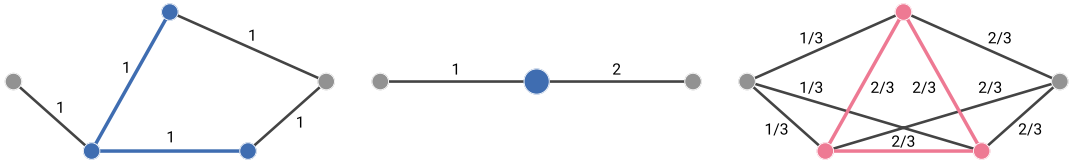
We will now see how the size of a graph G can be reduced via *graph coarsening*. The resulting coarsened graph G_c should ideally be structurally similar to G, i.e. it should have a similar

[4]Formally this measure is called *algebraic connectivity*, as opposed to regular *graph connectivity*, which is defined as $\min_{v_i \in \mathcal{V}} \deg(v_i)$.

spectrum. In this section we will first define the class of coarsening operators C and give an intuition on how they change the shape of a graph. Then we will describe a randomized algorithm to compute such a coarsening.

3.1 Definition of the Coarsening Operator

The core idea of graph coarsening is to replace clusters of vertices in the original graph G by single vertices in the coarsened graph G_c. Formally this means that the original vertices $\mathcal{V} = \{v_1, \dots, v_N\}$ are mapped to a smaller vertex set $\mathcal{V}_c = \{v'_1, \dots, v'_n\}$ via a surjective mapping $\varphi : \mathcal{V} \to \mathcal{V}_c$. The original edges $(v_i, v_j) \in \mathcal{E}$ are mapped to $(\varphi(v_i), \varphi(v_j))$, resulting in a reduced edge set $\mathcal{E}_c$ that contains every edge for which $\varphi(v_i) \neq \varphi(v_j)$. Figures 4a and 4b show an exemplary graph coarsening. To reverse the coarsening mapping, we define $\varphi^{-1} : \mathcal{V}_c \to \mathcal{P}(\mathcal{V})$ as the mapping from a coarsened vertex to the set of original vertices it represents.



(a) Original G **(b)** Coarsened G_c **(c)** Approximated $\widetilde{G}$

Figure 4: Example showing the effect of coarsening G when merging the blue vertices into a single vertex. The graph $\widetilde{G}$ on the right shows the result of re-expanding G_c.

In the last section we saw how graphs can be described via their Laplacian L. Since our goal is to analyze the effects of coarsening on the overall characteristics of a graph, we will now describe a coarsening φ as an operation that acts directly on L. The so called *coarsening matrix* $C \in \mathbb{R}^{n \times N}$ is essentially just a matrix representing φ:

$$\varphi(v_i) = v'_j \Leftrightarrow Cb_i = n_j b'_j \quad \text{with the standard basis vectors } b_i \in \mathbb{R}^N,\ b'_j \in \mathbb{R}^n \text{ and normalization factor}^5\ n_j := \left|\varphi^{-1}(v'_j)\right|^{-\frac{1}{2}} \tag{6}$$

Analogous to φ, the coarsening matrix C maps vertex basis vectors of G to vertex basis vectors of G_c. By linearity any signal $x \in \mathbb{R}^N$ on G can thus be downsampled to a signal $x_c := Cx \in \mathbb{R}^n$ on the coarsened graph G_c; the signal strengths of merged vertices will simply be added up. Similarly a downsampled signal x_c can be upsampled again to an approximation $\widetilde{x} \in \mathbb{R}^N$ of the original signal x. Upsampling uniformly distributes the signal strength of each $v'_j \in \mathcal{V}_c$ among $\varphi^{-1}(v'_i)$, where the inverse φ^{-1} can be represented by $C^\top$:

$$\widetilde{x} := C^\top x_c = C^\top C x = \Pi x \quad \text{with the projector } \Pi := C^\top C \tag{7}$$

[5] n_j is required for technical reasons. It normalizes $\Pi = C^\top C$ so that it is a projector, i.e. so it has eigenvalues in $\{0, 1\}$. This ensures that G and $\widetilde{G}$ have the same total weight.

Since both the coarsening matrix C and the Laplacian L are operators acting on signals, we can combine them to define the *coarsened Laplacian* L_c and also the *approximate Laplacian* $\widetilde{L}$:

$$L_c := CLC^\top \quad \text{and} \quad \widetilde{L} := C^\top L_c C = \Pi L \Pi \tag{8}$$

L_c represents the Laplacian of the coarsened graph G_c[6]. $\widetilde{L}$ represents the Laplacian of the graph $\widetilde{G}$, which is the result of re-expanding the coarsened graph G_c. During re-expansion, every vertex $v'_i \in \mathcal{V}_c$ is replaced by the complete graph on the vertex set $\varphi^{-1}(v'_i)$. The neighbors of each replaced v'_i are connected to the vertices that replace it. Figure 4c shows how a graph might look like after such a re-expansion.

3.2 The Randomized Edge Contraction Algorithm

Now that we have defined the coarsening operator, we will look at a simple randomized algorithm which finds a coarsened graph G_c that is spectrally similar to some given graph G. The so called *Randomized Edge Contraction* (REC) [1] algorithm is a variant of the well-known greedy algorithm for maximal matching generation. It contracts random edges until the vertex count has been reduced by a ratio r or until there are no more neighboring pairs of unmerged vertices. The contracted edges are chosen with a probability proportional to their weight.

Algorithm 1 Randomized Edge Contraction

1: **function** REC($G = (\mathcal{V}, \mathcal{E}, W), r \in [0, \frac{1}{2}]$)
2: $\quad \mathcal{C} \leftarrow \mathcal{E}$, $G_c \leftarrow G$
3: $\quad Z \leftarrow \sum_{e_{ij} \in \mathcal{E}} w_{ij}$
4: $\quad$ **while** $|\mathcal{C}| > 0$ and $\frac{|\mathcal{V}_c|}{|\mathcal{V}|} > 1 - r$ **do**
5: $\quad\quad$ Select some $e_{ij} \in \mathcal{C}$ with prob. $p_{ij} = \frac{w_{ij}}{Z}$.
6: $\quad\quad \mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{N}_{ij}$
7: $\quad\quad Z \leftarrow \sum_{e_{ij} \in \mathcal{C}} w_{ij}$
8: $\quad\quad G_c \leftarrow \mathit{contract}(G_c, e_{ij})$
9: $\quad$ **return** G_c

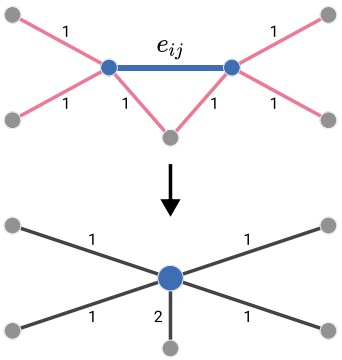


Figure 5: $\mathit{contract}(G_c, e_{ij})$.

We define the neighborhood $\mathcal{N}_{ij}$ as the set of incident edges of e_{ij}, including e_{ij} itself; this is shown as the red and blue edges in fig. 5. REC only merges vertices that have not been merged with some other vertex yet. Thus the reduction ratio $r = \frac{N-n}{N}$ is at most $\frac{1}{2}$, i.e. $|\mathcal{V}_c| \geq \frac{1}{2}|\mathcal{V}|$. If the node count needs to be further reduced, REC has to be applied multiple times.

4 Effects of Coarsening

We have just seen how to compute a coarsened graph G_c via the REC algorithm. What remains to be answered now is how coarsening affects the performance of graph algorithms when G_c is

[6] L_c is actually not a proper combinatorial Laplacian, i.e. its rows do not generally add up to 0. To fix this, L_c could be normalized, which is however not necessary for our analysis. We refer to Loukas and Vandergheynst [1, Sec. 2.1] for the details.

used as a proxy for the original graph G. In the first step we will present a graph similarity measure. Using this measure we will then put bounds on the dissimilarity of G and G_c. Finally we will use those bounds to analyze the influence of coarsening on the performance of the *spectral clustering algorithm*. The results described in this section were found by Loukas and Vandergheynst [1].

4.1 Restricted Spectral Similarity

To compare a graph G with its coarsened version G_c we will use the notion of *spectral similarity*. As we have seen before, G can be viewed as an operator that transforms an input signal $x \in \mathbb{R}^N$. We described this transform in terms of the graph's Laplacian L, more specifically in terms of the Laplacian's eigenbasis $\{u_k\}_{k=1}^N$ and spectrum $\{\lambda_k\}_{k=1}^N$. Similarly the coarsened graph G_c can be described in terms of its Laplacian L_c. Since $L \in \mathbb{R}^{N\times N}$ and $L_c \in \mathbb{R}^{n\times n}$ act on signal spaces of different dimensionality, they cannot be compared directly however. Instead we compare L with $\widetilde{L} = C^\top L_c C$, the upsampled version of L_c. We say $\widetilde{L}$ is an *ε-approximation* of L iff. $\widetilde{L}$ scales the eigenvectors u_k of L by a factor of roughly λ_k in the direction of u_k:

$$\forall k \leq K: \ (1-\varepsilon)\underbrace{u_k^\top L u_k}_{\lambda_k} \leq u_k^\top \widetilde{L} u_k \leq (1+\varepsilon)\underbrace{u_k^\top L u_k}_{\lambda_k} \text{ with } \varepsilon \geq 0 \quad (9)$$

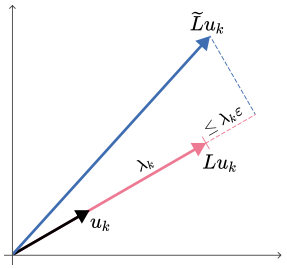


Figure 6: See eq. (9).

The reason we require eq. (9) to hold only for the first K eigenvectors of L is, that $\text{rank}(L) = N - c$, whereas $\text{rank}(\widetilde{L}) = n - c$; i.e. $\widetilde{L}$ has a higher dimensional null space[7]. Thus eq. (9) cannot hold for a signal x that is in the null space of $\widetilde{L}$ but not in that of L. Therefore the similarity condition is restricted to the first K eigenvectors, as they represent the most important "low-frequency" components of G. This restricted condition is called *restricted spectral similarity* (RSS).

The choice of K in RSS depends on the level of detail that should be considered when comparing L and $\widetilde{L}$. If G has c' clusters, i.e. connected subgraphs with relatively few or even no edges going out of it, a choice of $K = c'$ is reasonable. That way RSS checks whether the clusters of G are preserved in the coarsened graph G_c and how much the connectedness between the clusters changes. Details like the connections within clusters on the other hand do not have a strong influence on the RSS similarity if $K = c'$, since they are described by the "high-frequency" eigenvectors u_ℓ of L where $\ell > K$.

4.2 Bounding REC via RSS

Now we will use RSS to bound the eigenvalue distortion caused by a single application of the REC coarsening algorithm. It can be shown that the probability of satisfying the RSS condition for a single eigenvector u_k with a sufficiently small eigenvalue λ_k is lower-bounded by

$$\forall \lambda_k \leq \frac{\nu_{\min}}{4}: \ P\Big((1-\varepsilon)\lambda_k \leq u_k^\top \widetilde{L} u_k \leq (1+\varepsilon)\lambda_k\Big) \geq 1 - \frac{r\nu_{\max}}{2\varepsilon d_{\text{avg}}}\left(1 + \frac{3-4\lambda_k}{\nu_{\min}}\right) \quad (10)$$

[7] Here c denotes the number of connected components of G. They reduce the rank since $\lambda_1 = \cdots = \lambda_c = 0$.

where $r = \frac{N-n}{N}$ is the graph reduction ratio, $d_{\text{avg}} = \frac{1}{N}\sum_{v_i \in \mathcal{V}} d_i$ is the average weighted vertex degree and $\nu_{\min}$, $\nu_{\max}$ are the minimum and maximum of the neighborhood weights $\{d_i + d_j - w_{ij}\}_{e_{ij} \in \mathcal{E}}$. For a formal proof of this bound we refer to Loukas and Vandergheynst [1, Suppl. 2]. Here we will instead give an intuition for its key statements:

1. **ε-term:** Unsurprisingly the RSS bound is inversely proportional to ε. Relaxing the RSS bound makes it is more likely to be satisfied.

2. **r-term:** The RSS bound is linearly dependent on the reduction ratio r. The more the graph size is reduced, the smaller the probability of satisfying the bound.

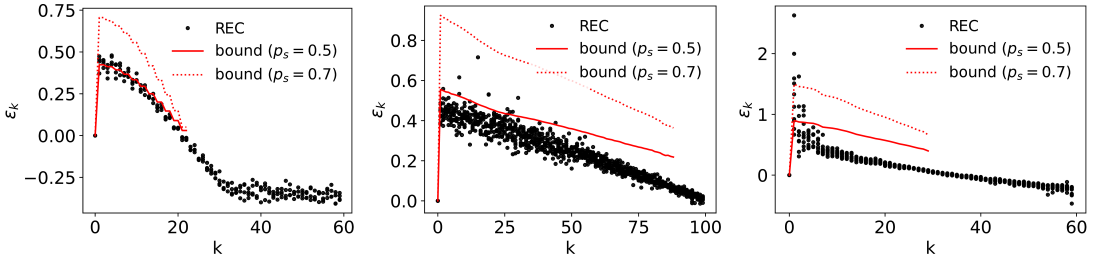
3. **$\lambda_k/\nu_{\min}$-term:** The bound is also linearly dependent on the eigenvalue λ_k. Additionally, since λ_k is proportional to the arbitrarily large weights w_{ij}, it is normalized via $\nu_{\min}$. To get an intuition for the influence of the eigenvalue on the RSS bound, two opposing effects have to be considered:

   (i) **Negative effect:** The eigenvector u_k for a large λ_k represents a "high-frequency" component of G that is more easily distorted by coarsening than a smooth "low-frequency" eigenvector u_ℓ with eigenvalue $\lambda_{\ell \ll k}$ (see fig. 3). Formally this means that the approximated eigenvector $\widetilde{u}_k := \Pi u_k$ has a lower overlap with the original u_k than $\widetilde{u}_\ell$ has with its original u_ℓ, i.e. $\langle u_k, \widetilde{u}_k \rangle < \langle u_\ell, \widetilde{u}_\ell \rangle$. ⇒ This effect causes the RSS probability bound to *decrease* with increasing, more easily distorted, eigenvalues.

   (ii) **Positive effect:** Recall that $u_k^\top \widetilde{L} u_k = \widetilde{u}_k^\top L \widetilde{u}_k$ and $\lambda_k = u_k^\top L u_k$. RSS checks whether L transforms u_k similarly to its approximate $\widetilde{u}_k$. When writing the latter transform in terms of L's eigenbasis we get $\widetilde{u}_k^\top L \widetilde{u}_k = \sum_{m=1}^{N} \lambda_m \langle u_m, \widetilde{u}_k \rangle^2$. For large eigenvalues λ_k the self-overlap $\langle u_k, \widetilde{u}_k \rangle^2$ is weighted more strongly than the overlap with low-valued eigenvectors $u_{\ell \ll k}$. ⇒ This effect causes the RSS probability bound to *increase* with increasing, more strongly weighted, eigenvalues.

   As long as the eigenvector distortion described in (i) is not too large ($\lambda_k \leq \frac{\nu_{\min}}{4}$), the positive effect described in (ii) dominates, i.e. $\lambda_k \langle u_k, \widetilde{u}_k \rangle^2 > \lambda_\ell \langle u_\ell, \widetilde{u}_\ell \rangle^2$ for $k \gg \ell$ despite $\langle u_k, \widetilde{u}_k \rangle < \langle u_\ell, \widetilde{u}_\ell \rangle$. This is why the RSS probability bound increases with increasing λ_k's, up to the point where they become too large and the bound becomes undefined. This can be seen in fig. 7a. It shows the RSS bound for a 20-regular graph, which is defined up to λ_{20}, after that the eigenvector distortion becomes too large.

4. **$\nu_{\max}/d_{\text{avg}}$-term:** Finally the bound also depends on the quotient between the maximum weight $\nu_{\max}$ within an edge neighborhood $\mathcal{N}_{ij}$ and the average weighted vertex degree d_{avg}. This quotient can be interpreted as a measure of how much the weight distribution within G varies when comparing local clusters with the global average. Regular graphs minimize this quotient, as all their local clusters have the same weight. The key implication is that the distortion of the graph Laplacian caused by coarsening is proportional to the regularity of the coarsened graph, i.e. coarsening distorts regular graphs less than highly irregular graphs. This can be seen in figs. 7a–7c for three increasingly irregular example graphs.



(a) Regular graph ($d = 20$) **(b)** Yeast (protein network) **(c)** Bunny (point cloud)

Figure 7: Comparison of the RSS-bound and the actual error constants ε for the eigenvectors of multiple coarsened graphs ($r = 0.4$). Confidence bounds for $p_s \in \{0.5, 0.7\}$ are shown. SOURCE: [1]

4.3 Implications for Spectral Clustering

Lastly we will now evaluate the effect coarsening has on the performance of the *spectral clustering algorithm* [2]. As we have seen in section 2.3, the positive and negative vertex signal strengths of the Fiedler vector u_2 describe a 2-clustering of the graph G where the cluster boundary can be interpreted as a heat-flow bottleneck. This idea can be extended to the K-clustering scenario by considering the first K eigenvectors $U_K = \{u_2, \dots, u_K\}$ instead of only looking at u_2. To obtain a vertex clustering, the vertex basis vectors $\{b_i\}_{v_i \in \mathcal{V}}$ are projected onto the spectral basis U_K, i.e. the Fourier transform $\widehat{b}_i = (\langle u_2, b_i \rangle, \dots, \langle u_K, b_i \rangle)^\top$ is computed for each vertex. Afterwards regular K-means clustering is performed on the Fourier-transformed vertex basis $\{\widehat{b}_i\}_{v_i \in \mathcal{V}}$ to get the clustering $S = (S_1, \dots, S_K)$ with $S_k \subseteq \mathcal{V}$.

For large graphs G, computing the eigenbasis U_K of $L \in \mathbb{R}^{N \times N}$ is expensive. To speed this up, one can compute the lower dimensional eigenbasis $\widetilde{U}_K = \{C^\top u'_2, \dots, C^\top u'_K\}$ instead, where u'_k are the eigenvectors of the coarsened Laplacian $L_c \in \mathbb{R}^{n \times n}$. Since we want a clustering on the original vertex set $\mathcal{V}$, not the coarsened vertices $\mathcal{V}_c$, the coarse eigenvectors $u'_k \in \mathbb{R}^n$ need to be upsampled to $C^\top u'_k \in \mathbb{R}^N$. In total this is often cheaper than computing U_K directly.

The question now is how much worse the spectral clustering results become when the vertex basis vectors $\{b_i\}_{v_i \in \mathcal{V}}$ are projected onto $\widetilde{U}_K$ instead of U_K. To answer this question, it was shown by Loukas and Vandergheynst [1, Coroll. 5.1] that the RSS bound also implies a bound on the absolute spectral clustering error; therefore the intuitions we discussed in section 4.2 also hold for spectral clustering. Most notably this implies that the absolute clustering error introduced by coarsening depends on the regularity of the graph, i.e. regular graphs are best suited to speed up spectral clustering via coarsening. Figure 8 shows the relative performance decrease caused by coarsening for an exemplary regular graph.

5 Conclusion

We have now discussed the three main topics of this paper: 1. How the structural properties of graphs can be described via spectral graph theory. 2. How graphs can be coarsened via the REC algorithm. 3. How to bound the effects of coarsening via RSS and what this implies for spectral clustering.

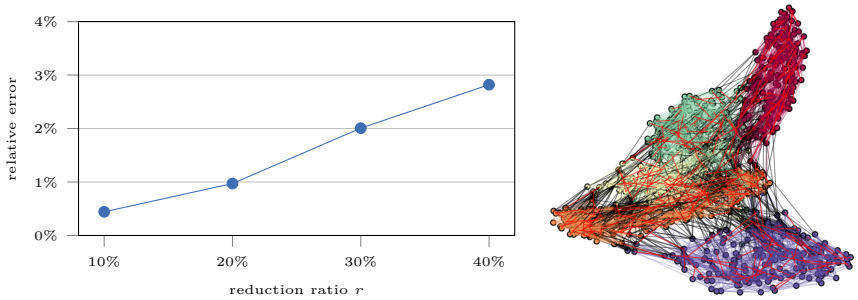


Figure 8: (Left) Relative spectral clustering error on a coarsened 12-nearest neighbor similarity graph for $N = 1000$ images from the MNIST dataset. Only images of the digits 0 to 4 were sampled, thus $K = 5$. (Right) The full MNIST similarity graph G with vertex colors indicating the clustering obtained via $\widetilde{U}_K$. Edges that are contracted in G_c are shown in red. SOURCE: [1]

Based on the work we presented, there are two main open questions for future research: 1. The described RSS bound assumes a single application of the REC algorithm. For multiple REC applications with a total reduction ratio of $r > \frac{1}{2}$, the RSS bound still needs to be generalized. 2. Currently the RSS bound has only been applied to the analysis of spectral clustering. To make the results more generally applicable, the implications for other graph algorithms, e.g. graph convolutional neural networks, are still to be considered.

References

[1] Andreas Loukas and Pierre Vandergheynst. "Spectrally Approximating Large Graphs with Smaller Graphs". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 3237–3246. arXiv: `1802.07510v1` (cit. on pp. 7–11).

[2] Ulrike von Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (2007), pp. 395–416. arXiv: `0711.0189v1` (cit. on p. 10).

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (2013), pp. 83–98 (cit. on pp. 4, 5).