Spectral Graph Approximation

Clemens Damke

Intelligent Systems and Machine Learning Group (ISG)
Heinz Nixdorf Institute
Paderborn University
Warburger Straße 100
33098 Paderborn

Abstract. TODO

Keywords: Machine Learning · Spectral Graph Theory · Graph Coarsening

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 defined later. Graph algorithms can then be run on the smaller coarsened graph. Afterwards the result for the coarsened graph can be iteratively refined to obtain an approximate result for the original graph. Figure 1 illustrates this approach.

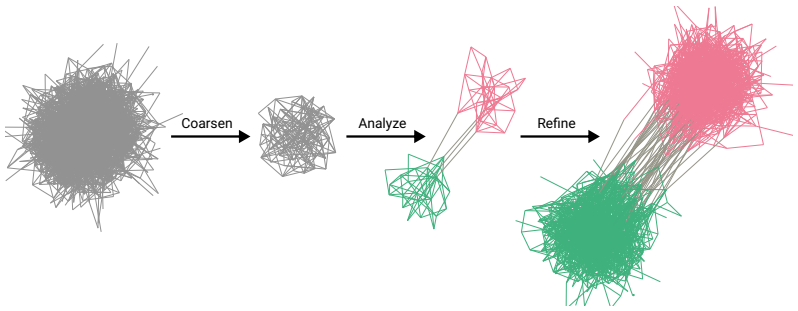


Fig. 1: Using graph coarsening to speed up graph algorithms, e.g. Min-Cut.

The goal of this paper is to show how graph coarsening works and how it affects the results of graph algorithms. This introduction consists of 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 increase the error of 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 strength w_{ij} with which pairs (v_i, v_j) of vertices are connected.

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

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 functions, where the underlying structure of the input domain is not fixed but can be freely chosen. Figure 2 shows that all real functions f can be seen as signals x of the graph described by the real number line[1]. Both, real functions and graph signals, can be described as vectors in their vertex base:

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

[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.

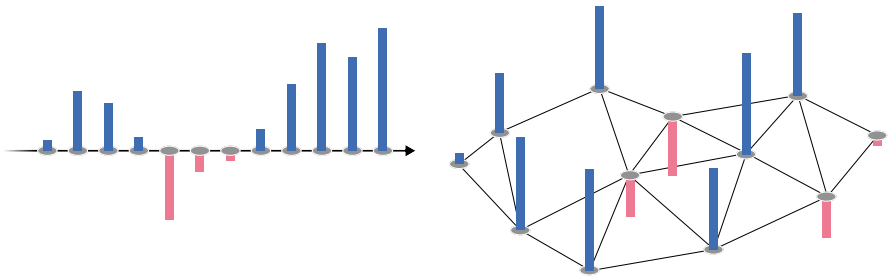


Fig. 2: Illustration of how the discretized real number line can be interpreted as an infinite linear graph, compared to some arbitrary finite non-linear graph. The bars show the signal strengths $f(t)$ and x_i at the vertices b_t and b_i respectively.

In this equation $\{b_i\}_{v_i \in \mathcal{V}}$ denotes the standard basis of the adjacency matrix W. 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 $\{u_\xi\}_{\xi \in \mathbb{R}}$:

$$f = \int_{\mathbb{R}} \underbrace{\langle b_t, f \rangle}_{f(t)} b_t dt = \int_{\mathbb{R}} \underbrace{\langle u_\xi, f \rangle}_{\hat{f}(\xi)} u_\xi d\xi \tag{2}$$

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 eigenvalues $\lambda_\xi = -(2\pi\xi)^2$ 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 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

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

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 $\{u_k\}_{1 \leq k \leq N}$. 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.

$$L := D - W \text{ with the degree matrix } D := \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_N \end{pmatrix}, d_i := \sum_{j=1}^{N} w_{ij} \tag{3}$$

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, L and $\frac{\partial^2}{\partial t^2}$, side-by-side gives an intuition for why this is the case:

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

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

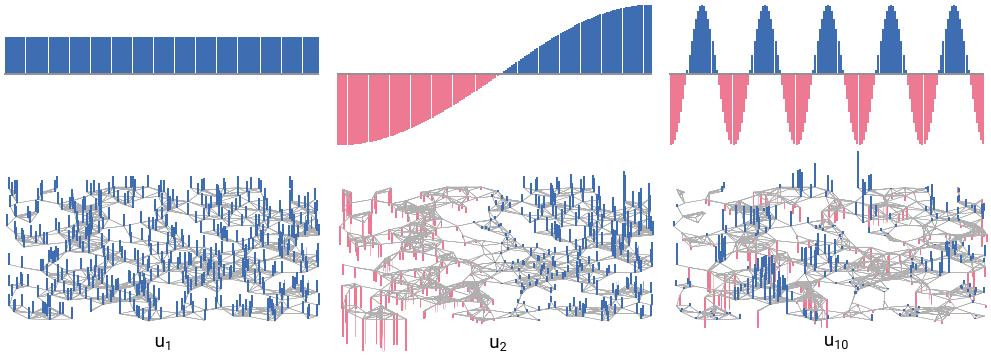


Fig. 3: Comparison between the basis functions/vectors of the classical Fourier transform and the graph Fourier transform. For the eigenfunctions on the upper half only the real cosine components of the complex exponentials are shown.

Now that we have defined the graph Laplacian L, the graph Fourier basis $\{u_k\}_{1 \leq k \leq N}$ 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. The figure also shows that the eigenvalues λ_k of the graph Fourier basis encode some notion of frequency, just like they do in the classical Fourier basis. Those eigenvalues describe many characteristics of a graph and are called its *spectrum*.

2.3 Spectral Properties of Graphs

We will now see how the spectrum of a graph describes its structural properties. For any graph the smallest eigenvalue always is $\lambda_1 = 0$ with the associated eigenvector u_1 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.

Another meaningful eigenvalue is λ_2 and its eigenvector u_2, also called the *Fiedler value* and *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[3]. 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 us access to graph characteristics like its overall connectivity. Similar to how the low-frequency components of the Fourier transform of a function represent the overall shape of that function, the eigenvectors associated to the small eigenvectors of L represent overall characteristics of G. The details of a Fourier-transformed function or graph are encoded in the high-frequency components, i.e. the eigenvectors associated to large eigenvalues. This perspective already hints at how spectral graph analysis might be useful to approximate graphs.

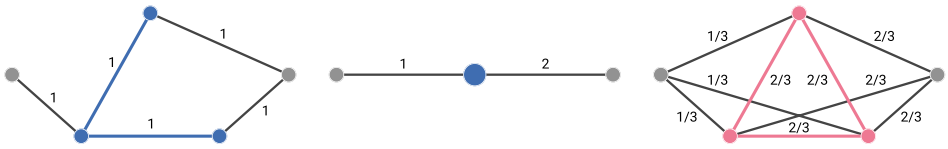
3 Graph Coarsening

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

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

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 $\widetilde{G}$

Fig. 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 representation of φ:

$$\varphi(v_i) = v'_j \Leftrightarrow Cb_i = n_j b'_j \quad \text{with the standard basis vectors } b_i \in \mathbb{R}^N,\ b'_j \in \mathbb{R}^n \quad (4)$$
$$\text{and normalization factor}^4\ n_j := |\varphi^{-1}(v'_j)|^{-\frac{1}{2}}$$

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 $C^\top$:

$$\widetilde{x} := C^\top x_c = C^\top C x = \Pi x \quad \text{with the projector } \Pi := C^\top C \quad (5)$$

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

$$L_c := CLC^\top \quad \text{and} \quad \widetilde{L} := C^\top L_c C = \Pi L \Pi \quad (6)$$

[4] 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.

L_c represents the Laplacian of the coarsened graph G_c[5]. $\widetilde{L}$ 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) algorithm is a variant of the well-known greedy algorithm for maximal matching generation. It contracts up to T random edges from the candidate set $\mathcal{C}$ as long as there are neighboring pairs of unmerged vertices. The contracted edges are chosen with a probability proportional to their weight.

Algorithm 1 Randomized Edge Contraction

function REC($G = (\mathcal{V}, \mathcal{E}, W), T \in \mathbb{N}$)
  $\mathcal{C} \leftarrow \mathcal{E}$, $G_c \leftarrow G$
  $\Phi \leftarrow \sum_{e_{ij} \in \mathcal{E}} w_{ij}, t \leftarrow 0$
  while $|\mathcal{C}| > 0$ and $t < T$ **do**
    $t \leftarrow t + 1$
    Select some $e_{ij} \in \mathcal{C}$ with prob. $p_{ij} = \frac{w_{ij}}{\Phi}$ or **continue** with prob. $1 - \sum_{e_{ij} \in \mathcal{C}} p_{ij}$.
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{N}_{ij}$
    $G_c \leftarrow \mathit{contract}(G_c, e_{ij})$
  return G_c

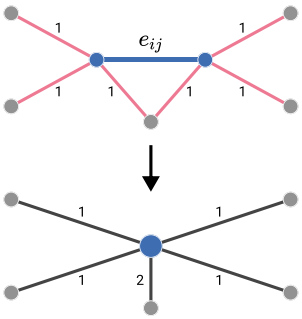


Fig. 5: $\mathit{contract}(G_c, e_{ij})$.

We define the neighborhood $\mathcal{N}_{ij}$ as the set of incident edges of e_{ij} including e_{ij} itself; 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 used as a proxy for the original graph G. In the first step we will introduce 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*.

[5] 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] for the details.

4.1 Restricted Spectral Similarity

4.2 Putting an RSS-bound on REC

4.3 Implications for Spectral Clustering

5 Conclusion

TODO

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 (cit. on p. 7).