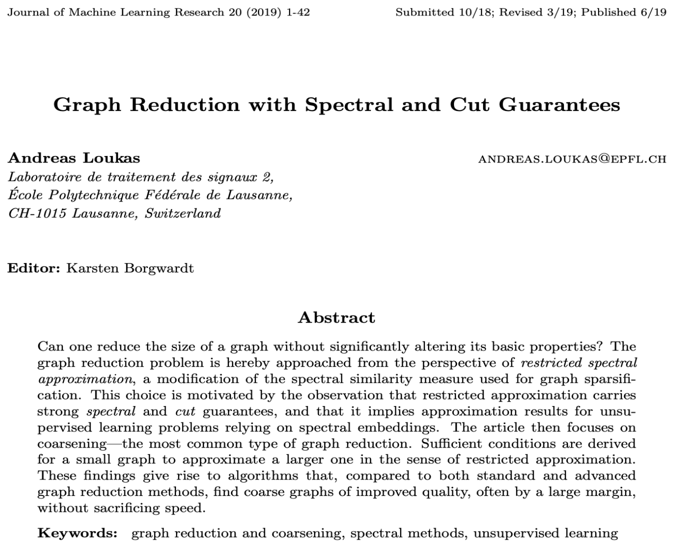
Graph Reduction with Spectral and Cut Guarantees

Vanderbilt University

Anwar Said
Postdoctoral Research Fellow
February 17, 2023



Journal of Machine Learning Research 20 (2019) 1-42 Submitted 10/18; Revised 3/19; Published 6/19

Graph Reduction with Spectral and Cut Guarantees

Andreas Loukas ANDREAS.LOUKAS@EPFL.CH
Laboratoire de traitement des signaux 2,
École Polytechnique Fédérale de Lausanne,
CH-1015 Lausanne, Switzerland

Editor: Karsten Borgwardt

Abstract

Can one reduce the size of a graph without significantly altering its basic properties? The graph reduction problem is hereby approached from the perspective of *restricted spectral approximation*, a modification of the spectral similarity measure used for graph sparsification. This choice is motivated by the observation that restricted approximation carries strong *spectral* and *cut* guarantees, and that it implies approximation results for unsupervised learning problems relying on spectral embeddings. The article then focuses on coarsening—the most common type of graph reduction. Sufficient conditions are derived for a small graph to approximate a larger one in the sense of restricted approximation. These findings give rise to algorithms that, compared to both standard and advanced graph reduction methods, find coarse graphs of improved quality, often by a large margin, without sacrificing speed.

Keywords: graph reduction and coarsening, spectral methods, unsupervised learning

Simplifying Graphs

A common solution

- Solve a small similar problem instead of solving the original (large) problem
- Refine the solution (if needed)

Two methods

- **Sparsification**: reduce the number of edges M
- **Reduction**: reduce the number of vertices N as well as M (graph coarsening, Kron reduction)

A Generic Graph Reduction Scheme

Let L be an $N \times N$ PSD matrix, s.t, $L(i,j) \neq 0$ only if e_{ij} as an edge of $G = (\mathcal{V}, \mathcal{E}, W)$:

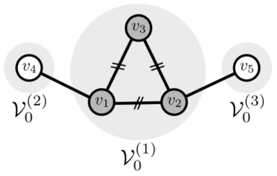
Let $L_0 = L$ and $x_0 = x$ and proceed according to the two recursive equations:

$$L_l = P_l^{\mp} L_{l-1} P_l^{+} \quad \text{and} \quad x_l = P_l x_{l-1}$$

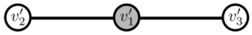
where $P_l \in \mathbf{R}^{N_l \times N_{l-1}}$ are matrices with more columns than rows, $l = 1, 2, \ldots, c$ is the level of the reduction. Symbol $\mp$ denotes the transposed pseudoinverse, and N_l is the dimensionality at level l such that $N_0 = N$ and $N_c = n \ll N$.

Vector x_c is lifted back to $\mathcal{R}^N$ by recursion $\hat{x}_{l-1} = P^{+} \hat{x}_l$ where $\hat{x}_c = x_c$

Graph coarsening as a special case



(a) Graph G



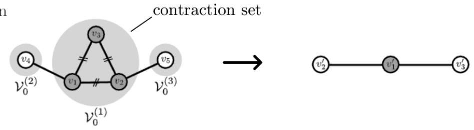
(b) Coarse graph G_c

To coarse $G_{l-1} = (V_{l-1}, E_{l-1})$ into $G_l = (V_l, E_l)$:

- Partition G_{l-1} into N_l connected subgraphs $G_{l-1}^{(r)} = \left(V_{l-1}^{(r)}, E_{l-1}^{(r)}\right)$ where $V_{l-1}^{(r)}$ is a contraction set
- Form a vertex $v'_r \in V_l$ for every contraction set
- The weight of edges (v'_r, v'_p) is equal to $cut(V_{l-1}^{(r)}, V_{l-1}^{(p)})$

Toy Coarsening Example

L is the Laplacian



$$P_1 = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad P_1^{+} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \Pi = P_1^{+} P_1 = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and coarsening results in

$$L_c = P_1^{\mp} L P_1^{+} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad x_c = P_1 x = \begin{bmatrix} (x(1) + x(2) + x(3))/3 \\ x(4) \\ x(5) \end{bmatrix}.$$

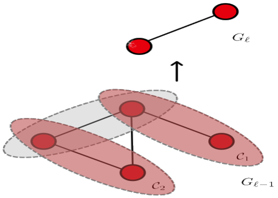
Let $\mathbf{R}$ be a k dimensional subspace of $\mathbb{R}^N$, $V \in \mathbb{R}^{(N \times k)}$ being an orthonormal basis of $\mathbf{R}$ and $A_0 = VV^T L^{+1/2}$, then the variation cost is:

$$\sigma_l = \|\Pi_l^{\perp} A_{l-1}\|_{L_{l-1}} \quad where \quad \Pi_l^{\perp} = I - P_l^{+} P_l$$

Local Variation Algorithm

To go from level $l-1$ to l:

1. Find many, small, possibly overlapping candidate sets $\mathcal{F}_l = \{\mathcal{C}_1, \mathcal{C}_1, \ldots, \mathcal{C}_z\}$
2. Compute local variation cost $\sigma_l(G_{\mathcal{C}i}, \mathbf{R})$
3. Sort $\mathcal{F}_l$ in terms of increasing cost
4. contract each set from $\mathcal{F}_l$ unless $N_l > n$ and cost $< \sigma$
5. form coarsening matrix P_l and return



Algorithm

Algorithm 2 Single-level coarsening by local variation

```
1: input: Combinatorial Laplacian $L_{\ell-1}$, threshold $\sigma'$, and target size $n$.
2: Form the family of candidate sets $\mathcal{F}_\ell = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \ldots\}$ (algorithm-specific step).
3: $N_\ell \leftarrow N_{\ell-1}$, marked $\leftarrow \varnothing$, $\sigma_\ell^2 \leftarrow 0$.
4: Sort $\mathcal{F}_\ell$ in terms of increasing $\text{cost}_\ell(\mathcal{C})$.
5: while $|\mathcal{F}_\ell| > 0$ and $N_\ell > n$ and $\sigma_\ell \leq \sigma'$ do
6:     Pop the candidate set $\mathcal{C}$ of minimal cost $s$ from $\mathcal{F}_\ell$.
7:     if all vertices of $\mathcal{C}$ are not marked and $\sigma' \geq \sqrt{\sigma_\ell^2 + (|\mathcal{C}| - 1)s}$ then
8:         marked $\leftarrow$ marked $\cup \mathcal{C}$, $\mathcal{P}_\ell \leftarrow \mathcal{P}_\ell \cup \mathcal{C}$, $N_\ell \leftarrow N_\ell - |\mathcal{C}| + 1$, $\sigma_\ell^2 \leftarrow \sigma_\ell^2 + (|\mathcal{C}| - 1)s$
9:     else
10:        $\mathcal{C}' \leftarrow \mathcal{C} \setminus$ marked
11:        if $|\mathcal{C}'| > 1$ then
12:            Compute $\text{cost}_\ell(\mathcal{C}')$ and insert $\mathcal{C}'$ into $\mathcal{F}_\ell$ while keeping the latter sorted.
13: Form the $N_\ell \times N_{\ell-1}$ coarsening matrix $P_\ell$ based on $\mathcal{P}_\ell$.
14: return $L_\ell \leftarrow P_\ell^{\mp} L_{\ell-1} P_\ell^{+}$ and $\sigma_\ell$
```

Multi-level Coarsening

Algorithm 1 Multi-level coarsening

```
1: input: Combinatorial Laplacian $L$, threshold $\epsilon'$, and target size $n$.
2: Set $\ell \leftarrow 0$, $L_\ell \leftarrow L$, and $\epsilon_\ell \leftarrow 0$.
3: while $N_\ell > n$ and $\epsilon_\ell < \epsilon'$ do
4:     $\ell \leftarrow \ell + 1$
5:     Coarsen $L_{\ell-1}$ using Algorithm 2 with threshold $\sigma' = \frac{1+\epsilon'}{1+\epsilon_{\ell-1}} - 1$ and target size $n$.
   Let $L_\ell$ be the resulting Laplacian of size $N_\ell$ with variation cost $\sigma_\ell$.
6:     $\epsilon_\ell \leftarrow (1 + \epsilon_{\ell-1})(1 + \sigma_\ell) - 1$.
7: return $L_\ell$
```

Constructing contraction sets

Edge-based: $\mathcal{F}_l$ contains candidate set for each edge of G_{l-1}
Drawbacks:

- The size of the graph can only be reduced by a factor of 2 at each level
- computationally complex

Heavy edge matching: The contraction family is obtained by computing a maximum weight matching with the weight of each contraction set (v_i, v_j) calculated as $w_{ij}/max(d_i, d_j)$
Neighborhood-based: a more attractive choice to construct one candidate set from the neighborhood of each vertex

- The size of the graph can be reduced to the desired scale at a single-level
- Experimentally shows better results

Implications

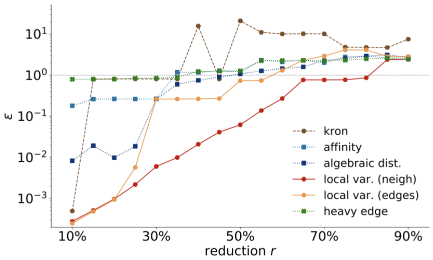
- Good cuts are preserved
- Good multi-way cuts are preserved
- Spectrum is preserved
- Spectral clustering works

Numerical Results

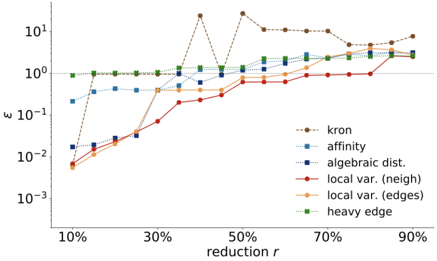
	r	heavy edge	local var. (edges)	local var. (neigh.)	algebraic distance	affinity	Kron reduction
yeast	30%	0.284	0.123	**0.003**	0.126	0.164	0.054
	50%	1.069	0.460	**0.034**	0.759	0.877	1.321
	70%	5.126	3.920	**0.409**	3.395	3.140	1.865
airfoil	30%	0.278	**0.036**	0.065	0.219	0.258	0.345
	50%	0.527	0.201	**0.197**	1.221	1.291	0.900
	70%	3.954	1.042	**0.926**	5.562	5.145	2.027
bunny	30%	0.015	**0.006**	0.061	0.244	0.070	0.335
	50%	0.064	**0.046**	0.190	0.401	0.137	0.801
	70%	0.122	**0.080**	0.323	0.694	0.304	1.812
minnesota	30%	0.332	0.088	**0.078**	0.220	0.295	0.324
	50%	1.363	0.431	**0.310**	2.394	2.676	0.873
	70%	7.452	4.553	**1.892**	8.412	9.354	2.068

Table 1: Mean relative error for the first $k = 10$ eigenvalues, for different graphs, reduction ratios, and coarsening methods.

Numerical Results



(a) yeast ($k = 10$)



(b) yeast ($k = 40$)