Note - 2017 Summer Research

July 19, 2017

Contents

1 Questions

1.1 MMF Related

1. Finding the optimal Givens rotation:
   (a) Proposition 4 proposes to find the roots of an optimality condition derived in the supplement. However, the pMMF library appears to avoid the A matrix altogether. We calculated that if we ignore A then the optimal alpha should be $\arctan \frac{b}{2c}$.
   (b) If this is what is being done, how do we interpret the choice in terms of optimization?
   (c) Why is A in Proposition 4 supposed to be a diagonal matrix
2. Finding the optimal rotations when $k > 2$

   In Section 2.2 of second paper, "actual rotation angle is determined by diagonalizing the Gram matrix." How?

   In their code, they diagonalize the the Gram matrix and take the transpose of the eigenvectors as the rotation matrix. The code appears to avoid the first error term in equation 19 and solve the optimization problem described in section **??** (Courant-Fischer-Weyl min-max problem).
3. In equation (10) on the multiresolution, should the second to last term be $dim(V_l), i$ instead of $dim(V_{l-1}), i$?
4. How is the clustering done? Function called compressionMap in `MMFmatrix.hpp`.
5. Algorithm 2
   (a) If A is a tall, skinny matrix, how can it be reset by multiplying on left and right by a square matrix of same dimensions?
   (b) Why is it the rows of A that are used to determine which one to eliminate? (or this is because A is always symmetric). In this case, A is referring to square matrix, not the tall skinny matrix defined in the input.

   $$n_1 = \text{Gram}_{i_1,i_1} - A_{i_1,i_1} * A_{i_1,i_1}$$
   $$n_2 = \text{Gram}_{i_2,i_2} - A_{i_2,i_2} * A_{i_2,i_2}$$

   If $n_1 < n_2$, eliminate i_1. Otherwise, eliminate i_2.

1.2 Other

1. How to quantify the relationship between reduced and original Laplacian eigenvectors?
2. How to construct a graph from a given heat diffusion? (convex optimization, see [**?**])
3. Why the core obtained from pMMF has nonzero diagonals? Where do they come from?

   The rotation matrix Q affects all rows and columns invovled in the rotation, contributing errors to the diagonal entries.

2 Givens Rotations, Sparse QR - May 26

2.1 A 2×2 Review

A 2-by-2 orthogonal matrix Q is a **rotation** if it has the form

$$Q = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

For $x \in \mathbb{R}^2$, $Q^T x$ rotates x counterclockwise through θ.
A 2-by-2 orthogonal matrix Q is a **reflection** if it has the form

$$Q = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$

For $x \in \mathbb{R}^2$, $Q^T x = Qx$ reflects x with respect to

$$\text{span}\left\{ \begin{bmatrix} \cos\theta/2 \\ \sin\theta/2 \end{bmatrix} \right\}$$

2.2 General Givens Rotations

Givens rotations are used to introducing zero elements into a matrix selectively. In general, they are rank 2 corrections to the identity. Denote $c = \cos\theta$ and $s = \sin\theta$, a givens rotation has the following form

$$G(i,k,\theta) = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & s & & \\ & & -s & c & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix}$$

Clearly, G is orthogonal. G rotates $x \in \mathbb{R}^n$ through θ in (i,k) coordinate plane. Also, givens rotations preserve sparsity of a matrix, since $G^T \cdot A \cdot G$ only affects the i^{th} and k^{th} rows/columns of matrix A.
In practice, there are efficient algorithms to compute c and s given a vectors in $\mathbb{R}^2$, See [**?**] for technical details.

2.3 Sparse QR using Givens Rotation

Given a matrix A, the QR factorization is unique. In Q, some of the columns have to span a particular subspace and the remaining columns have to form an orthogonal basis for the complement to this subspace. Thus, the orthogonal matrix cannot be sparse. Sparse QR method represents Q as a product of Givens rotations, which are extremely sparse/structured matrices.(https://mathoverflow.net/questions/94198/sparsity-of-qr-decomposition)[**?**]

2.4 Sparse QR Challenge

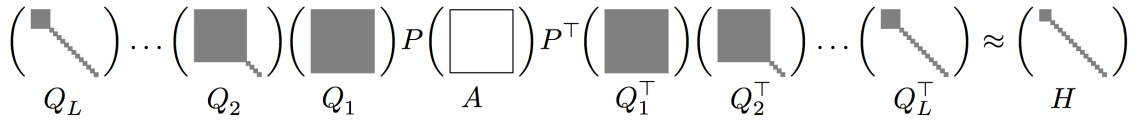
- Given a sparse matrix $A \in \mathbb{R}^{m \times n}$, determine a permutation p of $[n]$ such that, if $P = I_n(:,p)$, then the R-factor in the thin QR-factorization $AP^T = QR$ is close to being optimally sparse.
- Use orthogonal transformations to determine R from AP^T.
- Reduces the problem $Ax = b$ down to $R^T RPx = PA^T b$, which is easier computationally when R is optimally sparse. Additionally, we now no longer have to compute Q.

3 pMMF

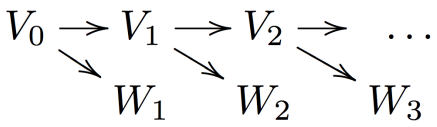
pMMF is a multiresolution matrix factorization method, which tries to find a sequence of sparse and orthogonal rotation matrices Q_i such that

$$H = Q_L \cdots Q_2 Q_1 A Q_1^T Q_2^T \cdots Q_L^T.$$

Graphically, the structure ca be represented as



Each orthogonal transform eliminates a small set of vertices by introducing zeros in corresponding rows/columns. Notice that after the first stage, $Q_1 A Q_1^T$ captures the finest scale structure in A. It splits $\mathbb{R}^n$ into two subspaces: the subspace spanned by scaling functions (kept vertices) and the subspace spanned by wavelets (eliminated vertices). Then, another stage of transform splits off another subspace spanned by level two wavelets and so on. This is the nested multiresolution of function spaces and can be represented graphically as



Each V_i is a subspace spanned by scaling functions at level i, which provide global insight into structure of signals, while each W_i is a subspace spanned by wavelet functions at level i, which have higher frequency than scaling functions and capture specialized local information of signals. Essentialy, pMMF is an approcimation algorithm and it tries to solve the following optimization problem

$$\underset{Q_1 \cdots Q_l \in \mathcal{Q}}{\text{minimize}} \; ||A - Q_1^T \cdots Q_L^T H Q_L \cdots Q_1||_{\text{Frob}}^2$$

3.1 pMMF Detailed Pseudocode

Algorithm 1 ComputeRotationMatrix

Input: current cluster A_1, row/column indices i and j, current Gram Matrix G
set $b = G(i, j)$
set $c = (G(j, j) - G(i, i))/2$;
if $g \neq 0$ **then**
    set $\beta = c/b$
    set $tan(\alpha) = \frac{1}{|\beta| + \sqrt{\beta * \beta + 1}}$
    set $cos(\alpha) = \frac{1}{\sqrt{(t * t + 1)}}$;
    set $Q = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$
else
    $Q = I$
Output: Q

Algorithm 1 finds an orthogonal rotation matrix $Q = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$ such that the second term of equation 20 is minimized when $k = 2$. That is to find an angle $\theta = 2\alpha$ such that

$$\sin(\theta + \omega + \pi/2) = 0, \text{, where } \omega = \arctan(\frac{G(i,i) - G(j,j)}{2G(i,j)})$$

Let $b = G(i,j)$ and $c = \frac{G(j,j)-G(i,i)}{2}$;
Equivalently,

$$\cos(\theta + \omega) = 0$$

$$\theta = \pi/2 - \omega = \pi/2 - \arctan(\frac{c}{b})$$

$$\tan(\alpha) = \tan(\theta/2) = \tan(\pi/4 - \frac{1}{2}\arctan(\frac{c}{b})) \text{ (if } \frac{c}{b} > 0)$$

$$\tan(\alpha) = \frac{1}{\frac{c}{b} + \sqrt{(\frac{c}{b})^2 + 1}}$$

$$\cos(\alpha) = \frac{1}{\sqrt{\tan(\alpha)^2 + 1}}$$

$$\sin(\alpha) = \tan(\alpha) * \sin(\alpha)$$

When $k > 2$, the rotation matrix Q is the transpose of eigenvectors of the $G_{I,I}$, where I contains k selected column indices. (why?)
The code appears to ignore the first error term as well. Similarly, only the second term of equation 20 is minimized. That is,

$$\min\ [QG_{I,I}Q^T]_{k,k}$$

Equivalently,

$$\min\ x^T[G_{I,I}]x \text{ such that } ||x||_2 = 1.$$

By Courant-Fischer theorem,

$$\min\ x^T[G_{I,I}]x = \lambda_1$$

Let x be the smallest eigenvector v_1 of $G_{I,I}$. Then,

$$\min\ x^T[G_{I,I}]x = \min\ v_1^T[G_{I,I}]v_1 = v_1\lambda_1 v_1 = \lambda_1$$

To find k rows/columns involved in each rotation, pMMF uses the randomized greedy method. First, randomly select a column i, and then compute the inner product of this column with all other columns. The selected column set will include $k-1$ columns that have the largest inner products with column i. Algorithm 2 in "Parallel MMF: a Multiresolution Approach to Matrix Computation" gives a good description of this process.
Note that after we get the rotation matrix Q, we have to update the original matrix $A = QAQ^T$ and the Gram matrix $G = QGQ^T$ before next channel. We can do this since A is assumed to be a symmetric square matrix.

Algorithm 2 pMMF

Input: weighted adjacency matrix A, number of stages P, number of clusters M, compression ratio μ, k-point rotation k

Set the current active columns $A_0 = A$
Set the current column indices $orig\_idx1 = 1 : G.N$
    // $orig\_idx\{p+1\}$ keeps track of the columns indices at stage p
Set the current permutation $idx1 = 1 : G.N$
    // $idx\{p+1\}$ records the permutation applied to active columns at stage p

for each stage $p = 1 : P$ **do**
    Cluster the active columns into M clusters;
    Set $perm$ = permutation that will permute A_{p-1} into correct cluster order
    Permute $A = A(perm, perm)$
    Update $orig\_idx\{p+1\} = orig\_idx\{p\}(perm)$
    Update $idx\{p+1\} = perm$

    for each cluster $m = 1 : M$ **do**
        Set $Qs\{m\}$ = FindRotsInCluster(cluster, μ, k, cInds)

    Merge Qs together to get the rotation $Q\{p\}$ for stage p
    Update $A = Q\{p\}AQ\{p\}^T$
    Set $perm$ = permutation that puts active columns together
    Permute $A = Q\{p\}AQ\{p\}^T$
    Update $orig\_idx\{p+1\} = orig\_idx\{p+1\}(perm)$
    Update $idx\{p+1\} = idx\{p+1\}(perm)$

Output: core A

3.2 Wavelet Transform - June 5-7

First, we describe how the dictionary is built from pMMF. Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, a L level pMMF factorization will result in

$$H = P_{L+1}(\overline{Q}_L P_L)\cdots(\overline{Q}_2 P_2)(\overline{Q}_1 P_1)A(P_1^T\overline{Q}_1^T)(P_2^T\overline{Q}_2^T)\cdots(P_L\overline{Q}_L^T)P_{L+1}^T,$$

where each P_l is a permutation matrix and each $\overline{Q}_l$ is an orthogonal rotation matrix at level l. Denote $Q_l = \overline{Q}_l P_l$ and we get

$$H = P_{L+1}Q_L\cdots Q_2Q_1AQ_1^TQ_2^T\cdots Q_L^TP_{L+1}^T$$

Notice that $(P_{L+1}Q_L\cdots Q_2Q_1)^T$ forms the dictionary for level L. The first $\dim(V_L)$ columns are scaling functions and the rest of its columns are wavelets. The dictionary has the form of:

$$\left[\begin{array}{c|c|c|c|c} & & & & \\ V_L & W_L & W_{L-1} & \cdots & W_1 \\ & & & & \end{array}\right]$$

where V_L denotes the scaling functions at level L and each W_l denotes the wavelet functions at level l.

The scaling and wavelet functions have the following properties:

1. Scaling functions and wavelets are orthonormal. That is, the dictionary is an orthonormal matrix.
2. Scaling and wavelet functions are localized in both vertex and frequency domain.
3. Wavelets centered at the very first nodes eliminated have highest frequencies - these are the wavelet functions which make up W_1.

4. Last vertices remaining are the centers of the scaling functions in V_L.

5. $\text{span}(V_i) = \text{span}(V_{i+1} \oplus W_{i+1})$

In our code, P_n is split into two separate permutations – C_n, which occurs when the rows and columns are clustered, and A_n, which occurs when we group the remaining active columns together. However, for a given $\overline{A}_n$, the grouping of active columns and the clustering take place in successive iterations of a loop. So, for each $\overline{Q}_n$, we must undo the permutations C_{n+1} and A_n. The only exception is $n = 1$ – in this case, we only undo C_1 since the active columns have not yet been grouped together. So, our dictionary (W) is

$$W = A_n \overline{Q}_n C_n A_{n-1} \cdots \overline{Q}_2 C_2 A_1 \overline{Q}_1 C_1$$

3.2.1 Wavelet Transform Example 1: 16×16 Community Graph

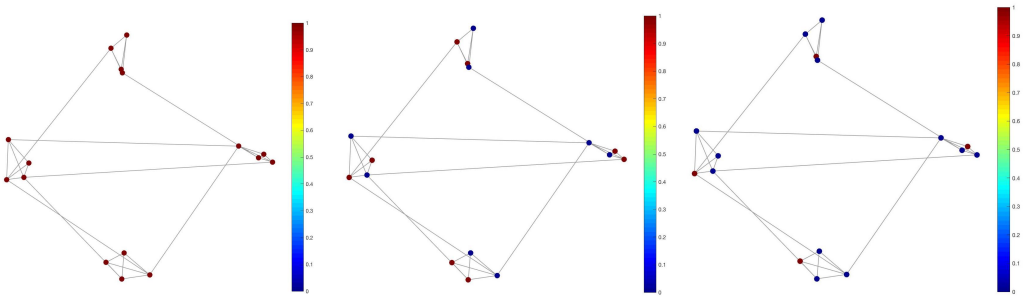


Figure 1: Downsampling 16-by-16 community graph. Red vertices are kept after each stage. Notice that after 2 stages, 4 kept vertices are from 4 different communities.

Dictionary Atoms:

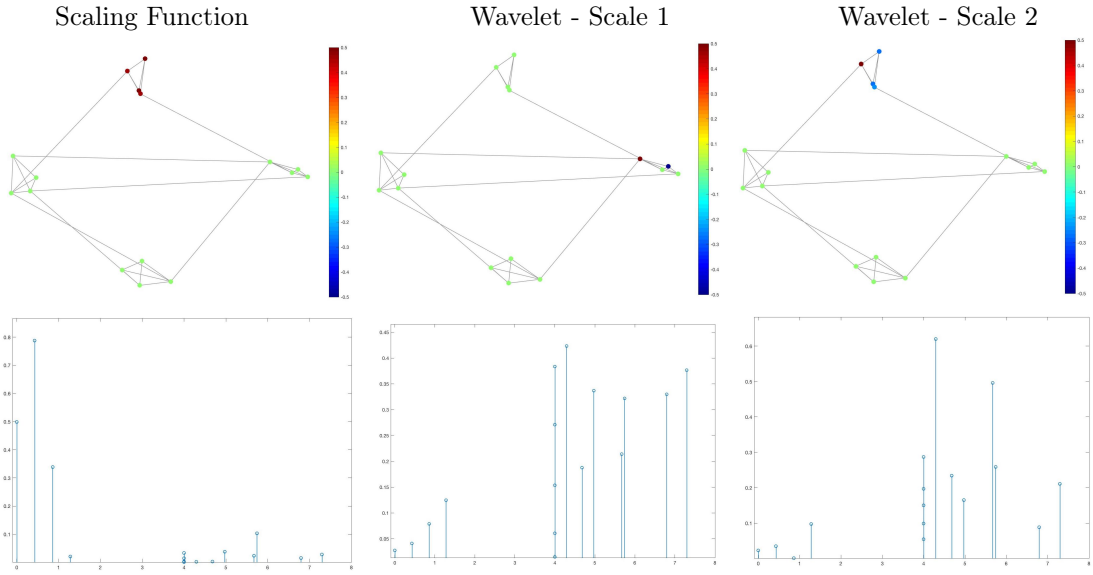


Figure 2: Dictionary atoms in both vertex and frequency domains.

3.2.2 Wavelet Transform Example 2: 500×500 Sensor Network

Dictionary Atoms:

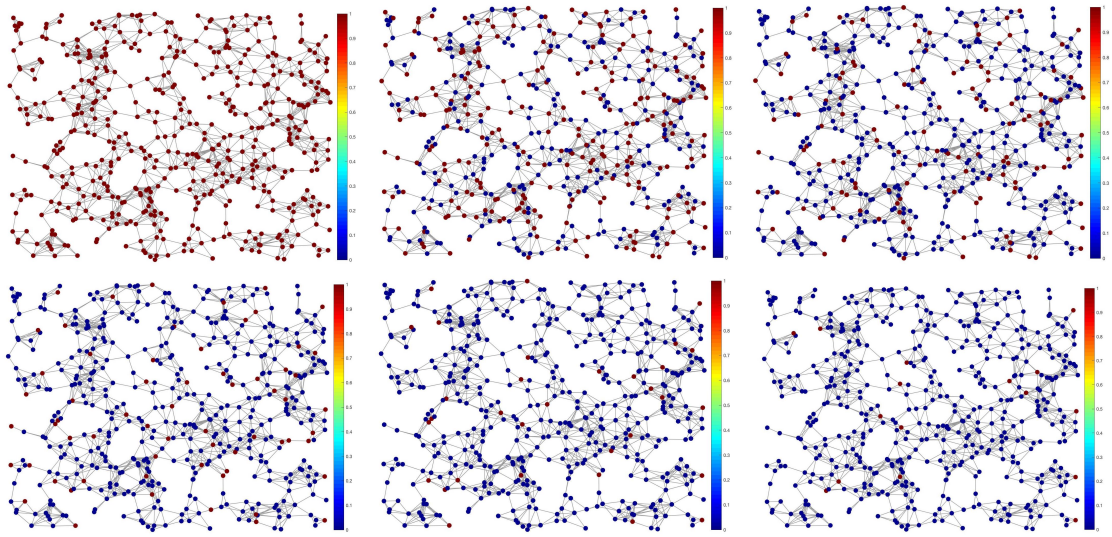


Figure 3: Downsampling 500-by-500 sensor network.
Notice that after each stage, kept vertices cover the whole graph instead of centering at a specific place.

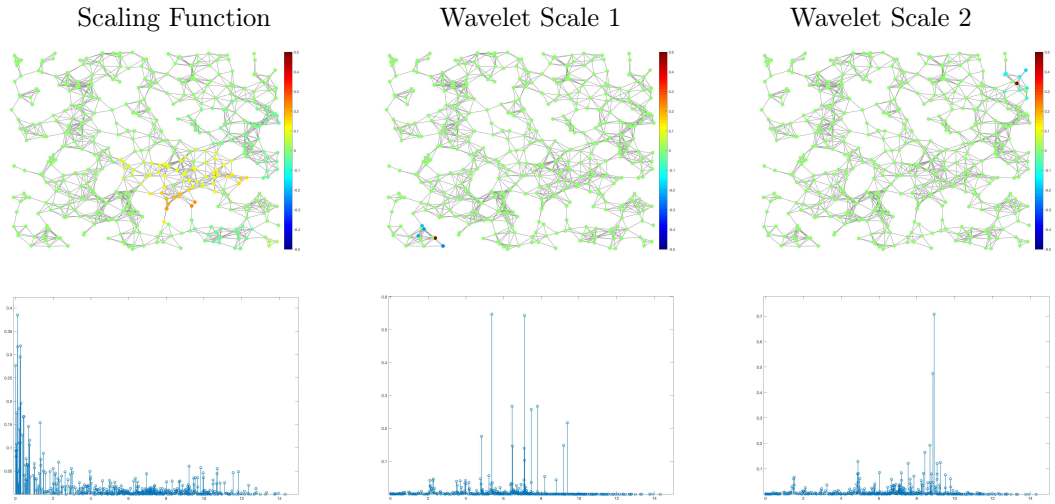


Figure 4: Dictionary Atoms in both vertex and frequency domain.

3.3 Check scaling spaces

We are interested in how scaling spaces are related to each other. We know that V_i are nested. If we interpolate scaling function is V_{l-1}, will we get scaling function in V_l?

3.4 Plot core

Solve for x in $V_L(1:4,:)x = h_1$. Then plot $V_L x$ on original graph. (Do least square if not full rank)

3.5 Other stuff

1. Check if the Laplacian eigenvectors form a subspace of the old eigenvectors.
2. Plot interpolation coefficients. $Ax = B$.

4 pMMF-Graph Multiresolution

The goal is to rewire the downsampled graph based on the information provided by pMMF.
pMMF Levers:

- Clustering methods: spectral clustering, hierarchical clustering
- k-point rotations
- vertex selection for rotations

Graph Levers:

- starting matrix - $W, \mathcal{L}$, or heat diffusion operator

4.1 MMF on Diffusion Operator - June 8-12

Given graph G, let $G.U$ be eigenvectors of its graph Laplacian and $G.e$ be corresponding eigenvalues. Let $A = G.U * f(G.e) * G.U^T$, where $f(\lambda) = e^{-\tau\lambda}$. One level of MMF on matrix A results in

$$H = P_2 Q_1 A Q_1^T P_2^T$$

Assume H is a diffusion operator on some reduced graph. We tried to

1. Diagonalize $H = VDV^T$
2. Construct a graph Laplacian $\mathcal{L} = Vf^{-1}(D)V^T$

However, $\mathcal{L}$ is not a graph Laplacian on downsampled vertices (row sums not equal to 0).
Are there other methods to find a Laplacian $\mathcal{L}$ that minimizes $|f(\mathcal{L}) - H|$?
Segarra et al. [**?**] suggest to infer the network topology based on spectral templates. Essentially, it is a convex optimization problem, see section **??** for more details.

4.2 MMF on Weighted Adjacency Matrix W - June 12

Some general observations:

1. The core H has non-zero diagonals.
2. The new graph is not necessarily connected.
3. The new graph is still pretty sparse.
4. The current clustering method doesn't yield great downsampling results for path and ring graphs.
5. The clusters have great influences on how downsampled graphs are rewired.

4.2.1 Experiments on path graphs

MMF works well for path graphs in general. The core has zero diagonals (surprisingly) and the new graph is connected (if # clusters = 1). The reduced graph captures the path structure.

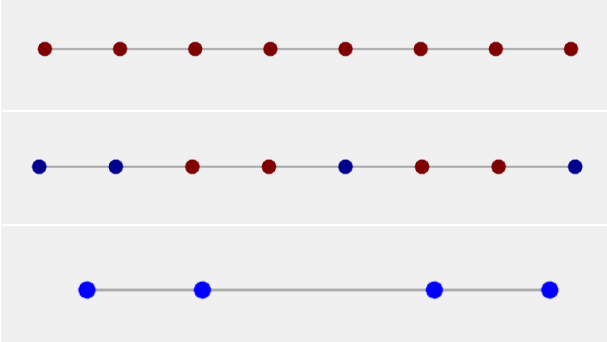


Figure 5: a. original graph b. downsampled graph (red vertices are kept) c. reduced graph

4.2.2 Experiments on ring graphs

For ring graphs, the core has zero diagonals, but the new graph is not necessarily connected. The new graph captures the ring structure.

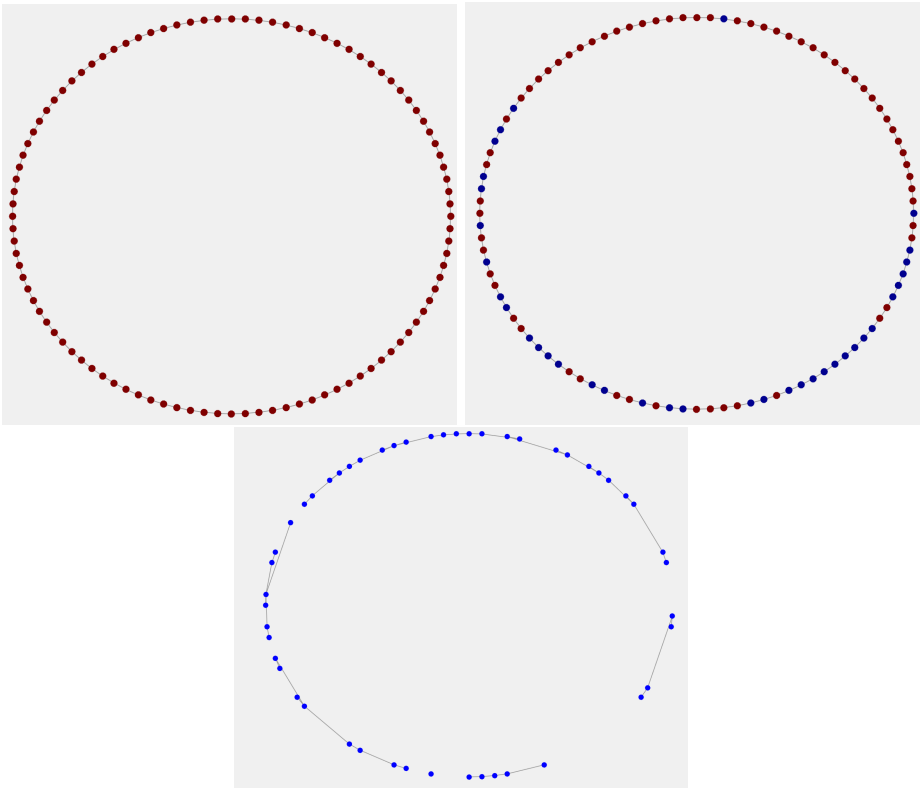


Figure 6: a. original graph b. downsampled graph (red vertices are kept) c. reduced graph

4.2.3 Experiments on community graphs

For community graphs or more complex graphs, the pMMF has a severe problem. The diagonals of reduced matrices are not zero. In other words, we can construct graph from the core directly. In this section, we force the diagonals of the reduced matrices to be 0 at each stage and create the reduced graph from it.
The reduced graph preserves the community structure. Disjoint communities are still disjoint in new graphs. The sparsity is also maintained.

Figure 7: a. original graph b. downsampled graph (red vertices are kept) c. reduced graph

4.2.4 Experiments on sensor networks

Observations are similar to those of community graphs.

Figure 8: a. original graph b. downsampled graph (red vertices are kept) c. reduced graph

Note: The downsampled vertices greatly influence how the reduced graph is rewired. It seems like a disconnection occurs if most of the vertices between the two are eliminated.
With a evenly downsampled graph (every other vertex is kept), can we get better results? Maybe experiment with different clustering method to get a evenly downsampled graph?
Note: If we perform MMF on the Laplacian matrix, the reduced matrix we get is not a valid graph Laplacian.

4.3 $A = L$ vs. $A = I - L$

- When $A = L$, our weighted adjacency matrix is $W = \text{diag}(\text{diag}(H))$ - H.
- When $A = I - L$, our Laplacian is $L = I - H$ and our weighted adjacency matrix is $W = \text{diag}(\text{diag}(L))$ - L.

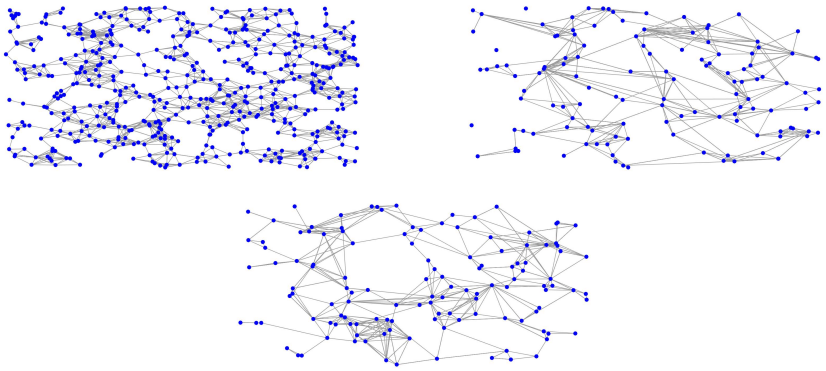


Figure 9: 1. Original graph (gsp_david _sensor_network) with 500 vertices. 2. Downsampled graph using $A = L$. 3. Downsampled graph using $A = I - L$.

- The two are very comparable, but it seems like the downsampling using $A = I - L$ captures the sparsity/density of the original graph a bit better.

5 Force Zero Diagonals

The structure of the weighted adjacency matrix can be lost through downsampling – we are often left with a core that has non-zero diagonal entries. We can maintain a zero diagonal, however, by being selective when we rotate columns within a certain cluster. If columns i and j only rotate with one another, and if $[A_{n-1}]_{i,i} = 0$, then

$$[A_n]_{i,i} = [Q_n A_{n-1} Q_n^T]_{i,i} = 0$$

if vertices i and j are disconnected in the graph represented by the weighted adjacency matrix A_{n-1}:

$$\begin{aligned}
[A_n]_{i,i} &= [Q_n A_{n-1} Q_n^T]_{i,i} \\
&= [Q_n]_{i,:}[A_{n-1}Q_n^T]_{:,i} \\
&= [Q_n]_{i,:} < [A_{n-1}]_{1,:}[Q_n^T]_{:,i} + \cdots + [A_{n-1}]_{s,:}[Q_n^T]_{:,i} > \\
&= [Q_n]_{i,1}[A_{n-1}]_{1,:}[Q_n^T]_{:,i} + \cdots + [Q_n]_{i,s}[A_{n-1}]_{s,:}[Q_n^T]_{:,i}
\end{aligned}$$

Since column i only rotates with column j, the i^{th} row of Q_n will only have nonzero entries in the i^{th} and j^{th} columns. So, the above equation simplifies to

$$[A_n]_{i,i} = [Q_n]_{i,i}[A_{n-1}]_{i,:}[Q_n^T]_{:,i} + [Q_n]_{i,j}[A_{n-1}]_{j,:}[Q_n^T]_{:,i}$$

Likewise, the i^{th} column of Q_n^T will only have nonzero entries in the i^{th} and j^{th} rows. We now have

$$\begin{aligned}
[A_n]_{i,i} &= [Q_n]_{i,i}([A_{n-1}]_{i,i}[Q_n^T]_{i,i} + [A_{n-1}]_{i,j}[Q_n^T]_{j,i}) + [Q_n]_{i,j}([A_{n-1}]_{j,i}[Q_n^T]_{i,i} + [A_{n-1}]_{j,j}[Q_n^T]_{j,i}) \\
&= [Q_n]_{i,i}(0 + [A_{n-1}]_{i,j}[Q_n^T]_{j,i}) + [Q_n]_{i,j}([A_{n-1}]_{j,i}[Q_n^T]_{i,i} + 0)
\end{aligned}$$

We have $[A_{n-1}]_{i,j} = [A_{n-1}]_{j,i}$ since A is symmetric and $[Q_n]_{i,j} = [Q_n^T]_{j,i}$, so we are left with

$$[A_n]_{i,i} = 2[A_{n-1}]_{i,j}[Q_n]_{i,i}[Q_n]_{i,j}$$

Since both entries of Q_n are nonzero, $[A_n]_{i,i} =$ if and only if $[A_{n-1}]_{i,j} = 0$.

The case is complicated further when, say, column i rotates with both column j and column k. Then, Q_n has nonzero entries in the i^{th}, j^{th}, and k^{th} columns, so we would end up with

$$\begin{aligned}
[A_n]_{i,i} &= [Q_n]_{i,i}([A_{n-1}]_{i,i}[Q_n^T]_{i,i} + [A_{n-1}]_{i,j}[Q_n^T]_{j,i} + [A_{n-1}]_{i,k}[Q_n^T]_{k,i}) + [Q_n]_{i,j}([A_{n-1}]_{j,i}[Q_n^T]_{i,i} + [A_{n-1}]_{j,j}[Q_n^T]_{j,i} \\
&\quad + [A_{n-1}]_{j,k}[Q_n^T]_{k,i}) + [Q_n]_{i,k}([A_{n-1}]_{k,i}[Q_n^T]_{i,i} + [A_{n-1}]_{k,j}[Q_n^T]_{j,i} + [A_{n-1}]_{k,k}[Q_n^T]_{k,i}) \\
&= 2[A_{n-1}]_{i,j}[Q_n]_{i,i}[Q_n]_{i,j} + 2[A_{n-1}]_{i,k}[Q_n]_{i,i}[Q_n]_{i,k} + 2[A_{n-1}]_{j,k}[Q_n]_{i,j}[Q_n]_{i,k}
\end{aligned}$$

We see that $[A_{n-1}]_{j,k}$ now contributes as well. So, not only must the pairs (i,j) and (i,k) be disconnected, vertices j and k must also be disconnected even if they are not one of the pairs of columns that is rotated.
If we want to guarantee $[A_n]_{i,i} = 0$ for all i in the cluster, the vertices in the cluster must all be mutually disconnected. However, this seems counterintuitive, since we normally would want to rotate similar vertices with one another. This method does the exact opposite, and as a result, downsampling results in edges between communities that were previously disconnected.

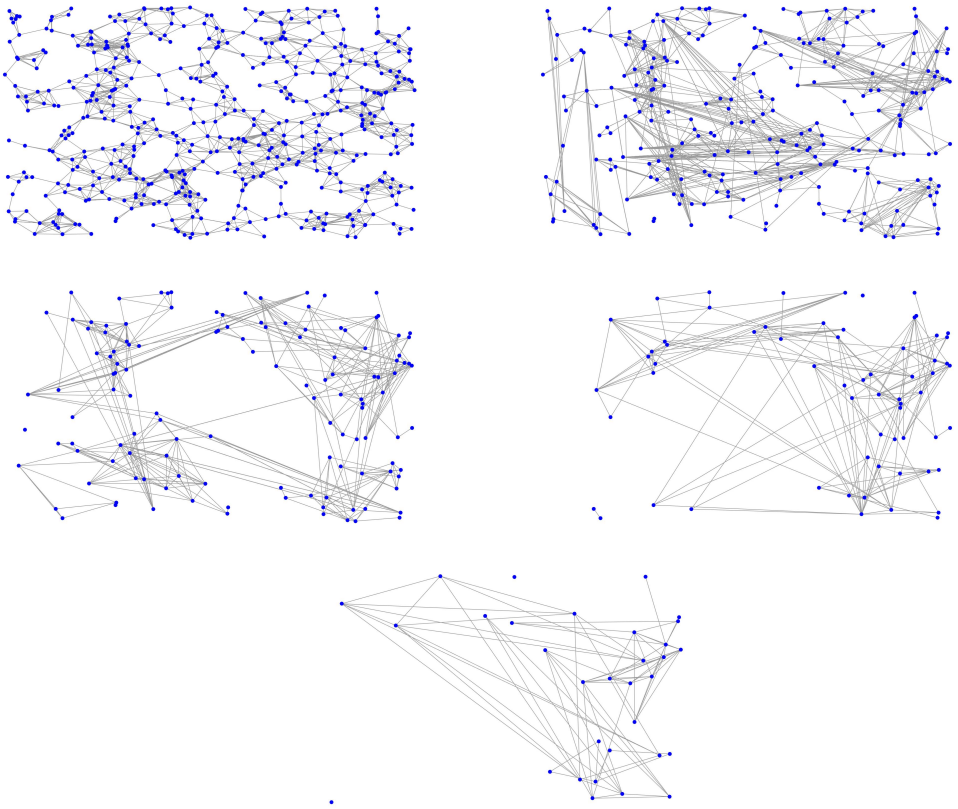


Figure 10: pMMF performed on a 500 vertex sensor network with 4 stages, 4 clusters, 2-point rotations, and compression ratio 0.5. Rotations are performed only on disconnected vertices.

6 k-point Rotations, $k > 2$

As k increases, the graphs become denser. When $k = n - 1$, it is equivalent to double procrustes problem.

6.1 Double Procrustes Problem

Double Procrustes problem considers finding an orthogonal matrix Q, which minimizes

$$||Q^T AQ - B||$$

for symmetric A and B.
Since both A and B are symmetric, we can diagonalize these two matrices

$$A = V_1 D_1 V_2^T \text{ and } B = V_2 D_2 V_2^T.$$

Then in this case, $Q = V_1 V_2^T$. (Detailed solution can be found in "Procrustes Problems", Oxford University Press).
In the graph setting, we want to find an orthogonal Q, which minimizes $||Q^T LQ - L'||$, where L is the original graph Laplacian and L' is the reduced graph Laplacian. If we do multile levels, we could generate scaling and wavelet functions similar to pMMF. That is, the first $dim(V_L)$ rows of Q are scaling functions and the rest of its rows are wavelets.
Some general observations;

1. This method connects Kron reduction and pMMF.
2. Q is not sparse.

3. The energy of scaling functions and wavelets are not localized.

4. Kron reduction produced dense sub-graphs.

7 Spectral Clustering

Another lever is the clustering method used in pMMF. In C++ library, they randomly cluster the columns until some criterion is satisfied. We would like to try another clustering method such that the graph is better cover with downsampled vertices.
Spectral clustering is a combination of polarity of eigenvectors and k-means. The other levers are kept the same. Then, we construct a downsampled graph at each stage, taking the core to be a weighted adjacency matrix. Note that the core has non-zero diagonals. we simply set the diagonals to zero here.
Parameters:

- G = gsp_david_sensor_network
- 500 vertices
- $A = G.W$
- 4 stages
- 2 clusters
- 2-point rotations

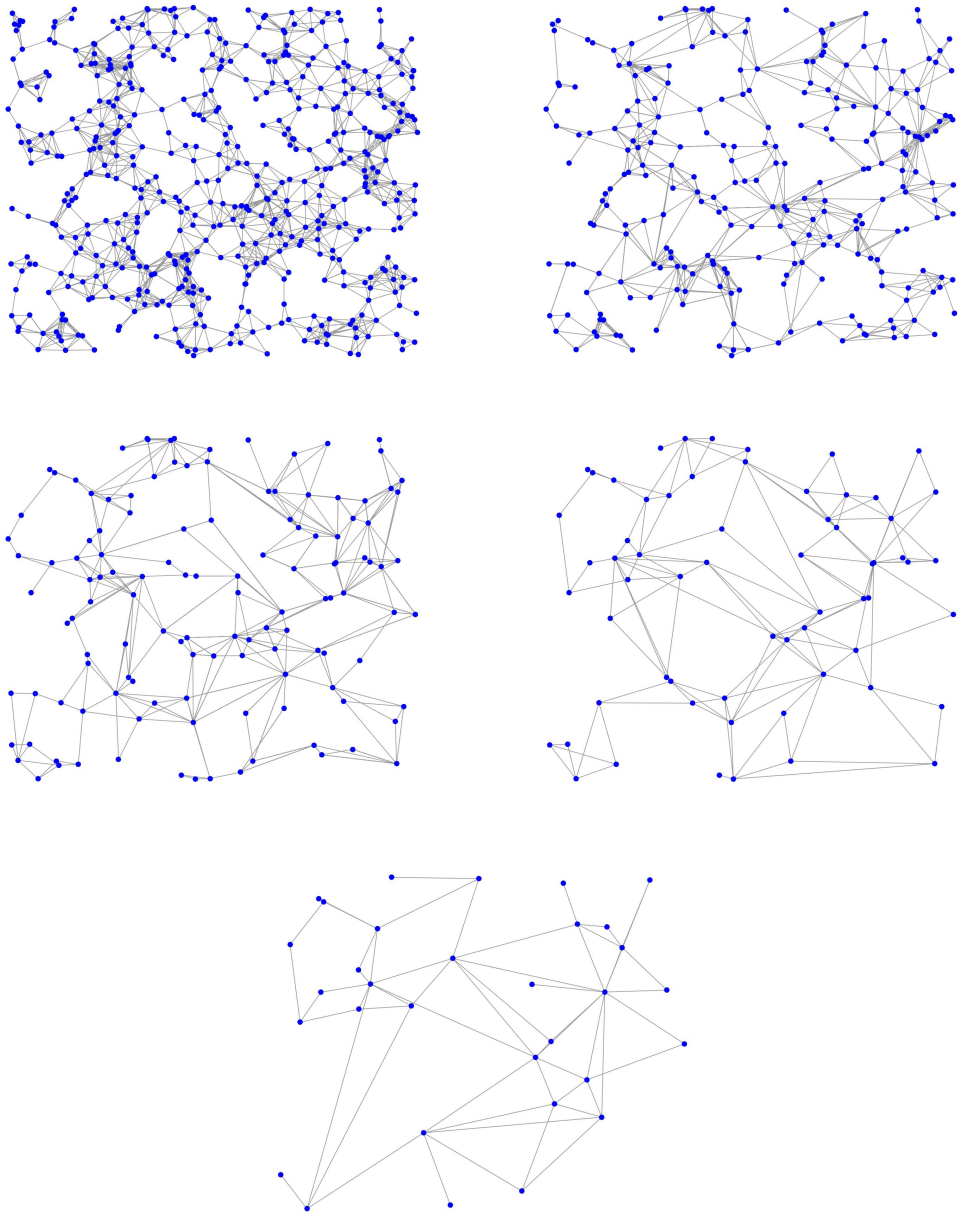


Figure 11: The top left figure is the original graph, gsp_david_sensor_network with 500 vertices, and each subsequent figure is the result of one stage of downsampling.

8 Interpreting Results

8.1 Edge Weights

Compared the proportion of edges with high weight (> 0.9) in a gsp_david_sensor_network graph with 500 nodes and the downsampled graph using A = W, A = L, and A = I - L.
22.15% of the edges on the original graph had high weight, and

- using A = W, 29.28% of the edges had high weight
- using A = L, 21.82% of the edges had high weight
- using A = I - L, 22.00% of the edges had high weight

When we look at the n highest-weighted edges (however many have weight > 0.9) in the original graph G and the $n \cdot \frac{Gdownsampled.N}{G.N}$ highest-weighted edges in the downsampled graph, we see that the areas of the downsampled graph which contain high-weighted edges most closely mirror the analogous areas on the original graph when $A = I - L$. $A = W$ does okay, but when $A = L$, the downsampled graph seems to ignore some high-weight areas from the original graph.
These judgments are all just visual, so next steps would be to find a way to quantify high-weight edge retention numerically.

8.2 Plot Reduced Laplacian Eigenvectors

Parameters:

- G: gsp_david_sensor_network
- 2 stages
- 1 cluster

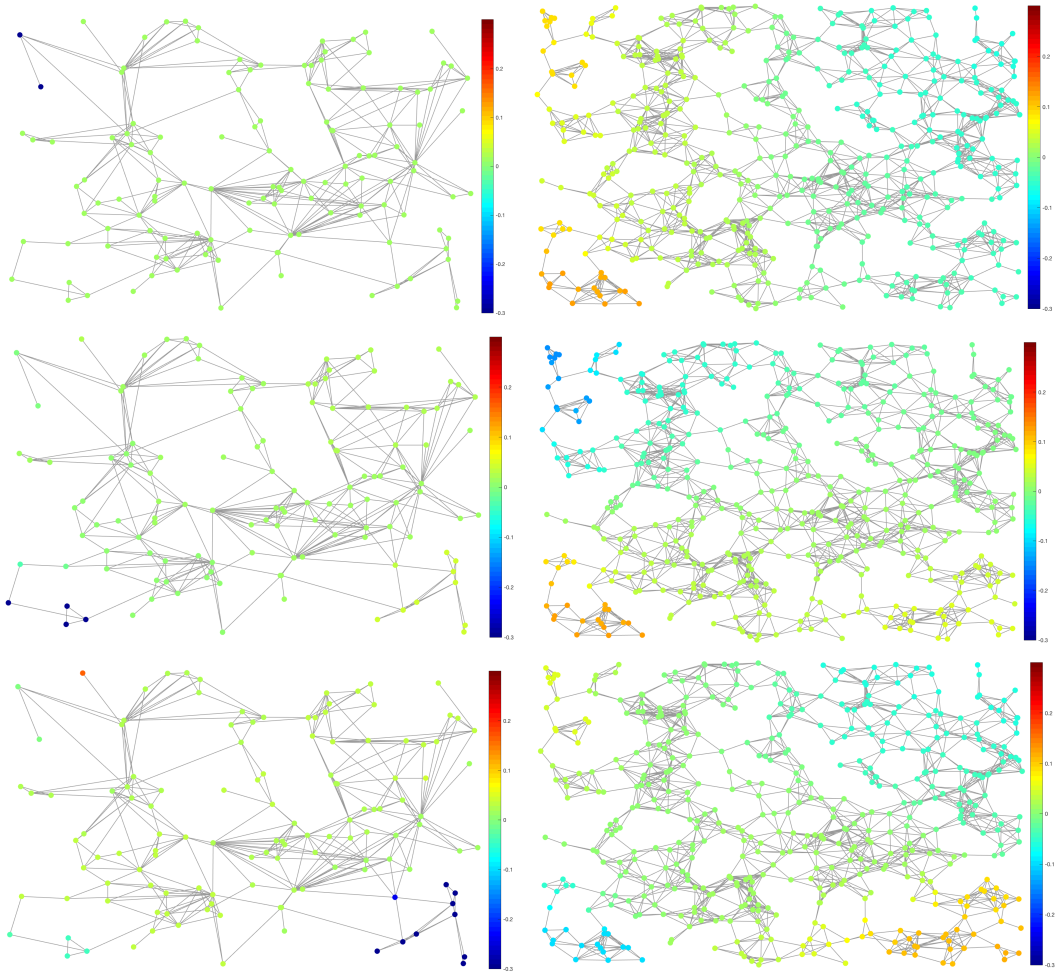


Figure 12: The left column shows the 2nd, 3rd and 4th eigenvectors of the reduced graph Laplacian. The right column shows the 2nd, 3rd and 4th smallest eigenvectors of the original graph Laplacian.

8.3 Interpolation

The adjacency matrix of the reduced graph is obtained from $H - Diag(H)$. Then we interpolate the eigenvectors of the reduced graph Laplacian and compare them to the eigenvectors of the original graph Laplacian.

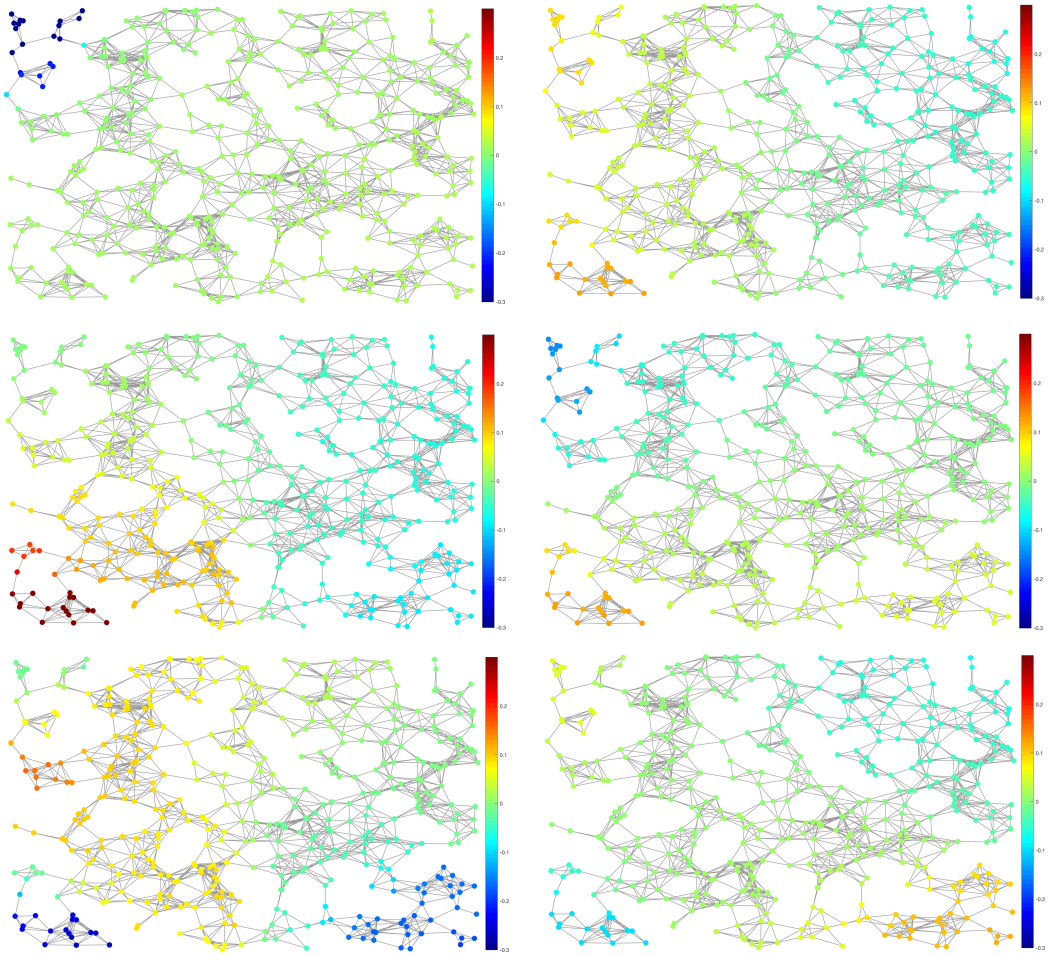


Figure 13: The left column shows interpolations of the 2nd, 3rd and 4th smallest eigenvectors. The right column shows the 2nd, 3rd and 4th smallest eigenvectors of the original graph

8.3.1 Interpolation Coefficient Matrix

Take A as the the weighted adjacency matrix, we notice that the reduced matrix has non-zero diagonals. In this section, we zero out the diagonals at each stage. Also, we set the compression ratio $\mu = 0.2$. Smaller compression ratio seems to give better interpolation results.
First example is on the path graph. Parameters:

1. 128 vertices
2. compression ratio $\mu = 0.2$
3. 4 stages
4. 2 cluster

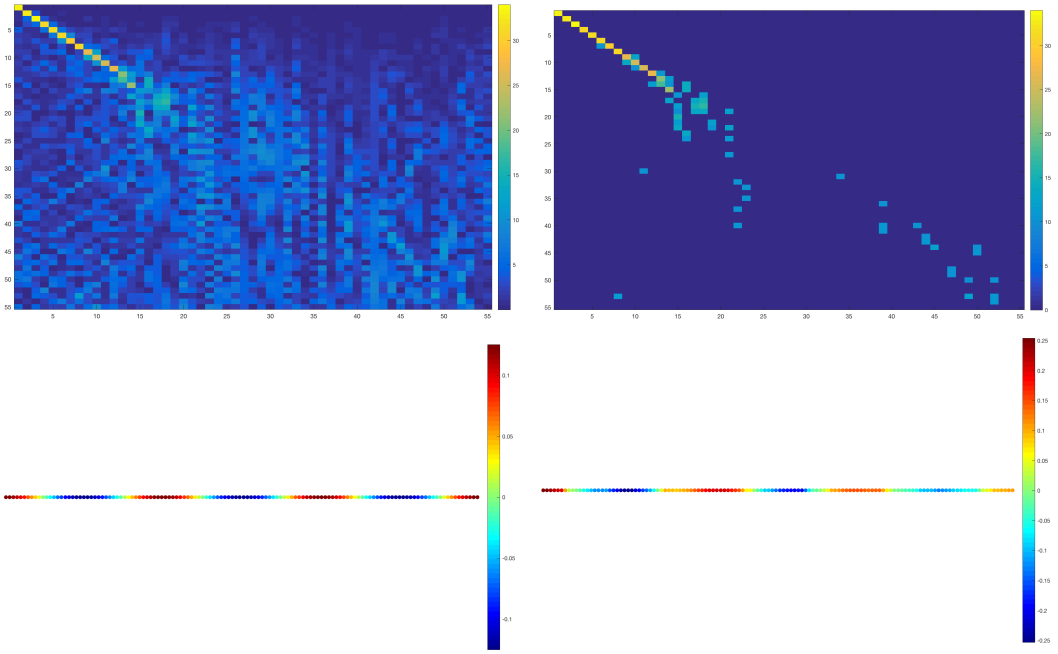


Figure 14: (a) interpolation coefficient matrix (b) threshold at $x > 10$
(c) original 7th eigenvector (d) interpolated 7th eigenvector

The following images are produced under same process except the diagonals of Abars are kept non-zero at each stage.

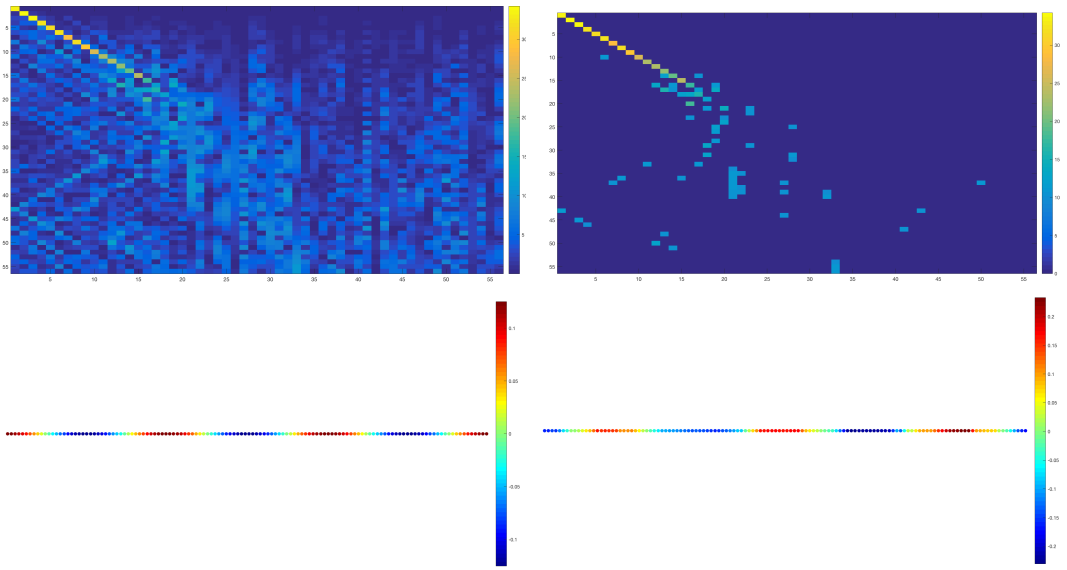


Figure 15: (a) interpolation coefficient matrix (b) threshold at $x > 10$
(c) original 7th eigenvector (d) interpolated 7th eigenvector

The second example is on david_sensor_network graph. Parameters:

1. 500 vertices
2. compression ratio $\mu = 0.2$
3. 4 stages

4. 2 cluster

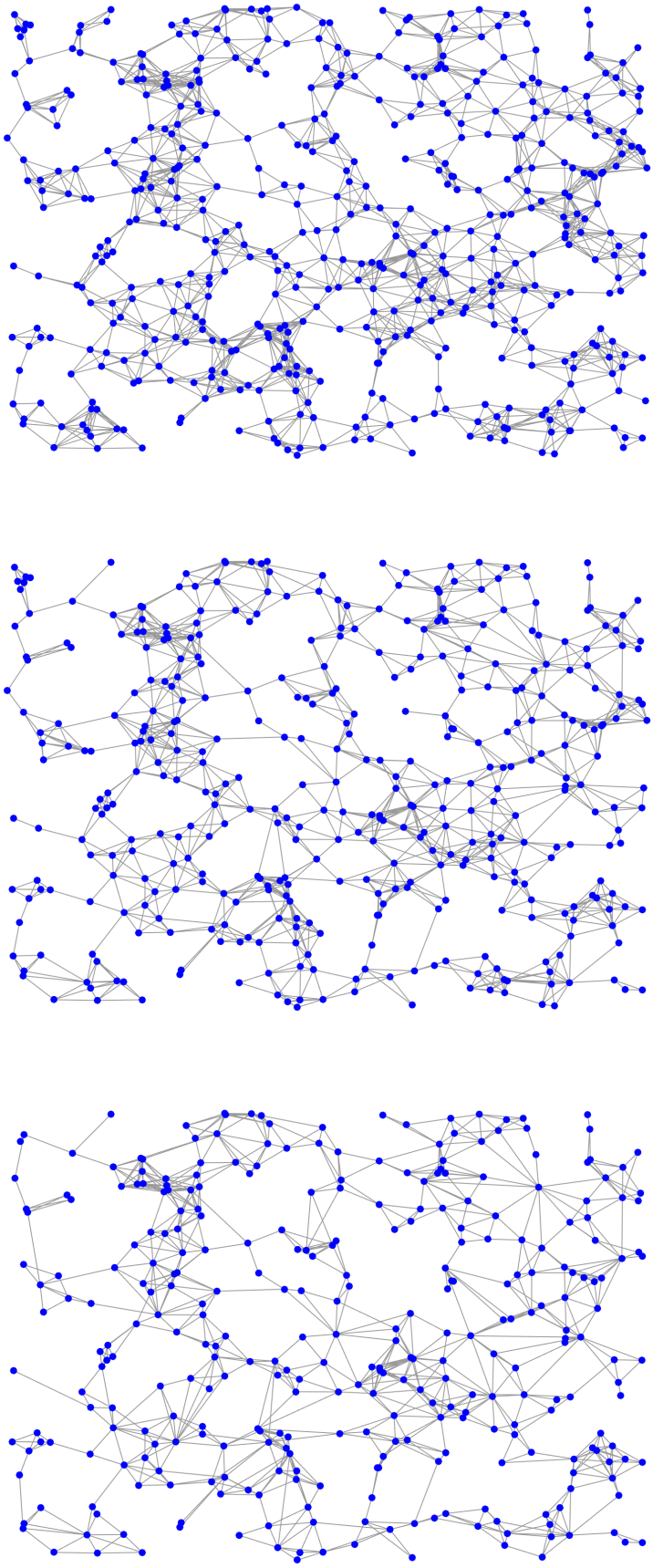


Figure 16: (a) original graph (b) 1st reduced graph (c) 2nd reduced graph

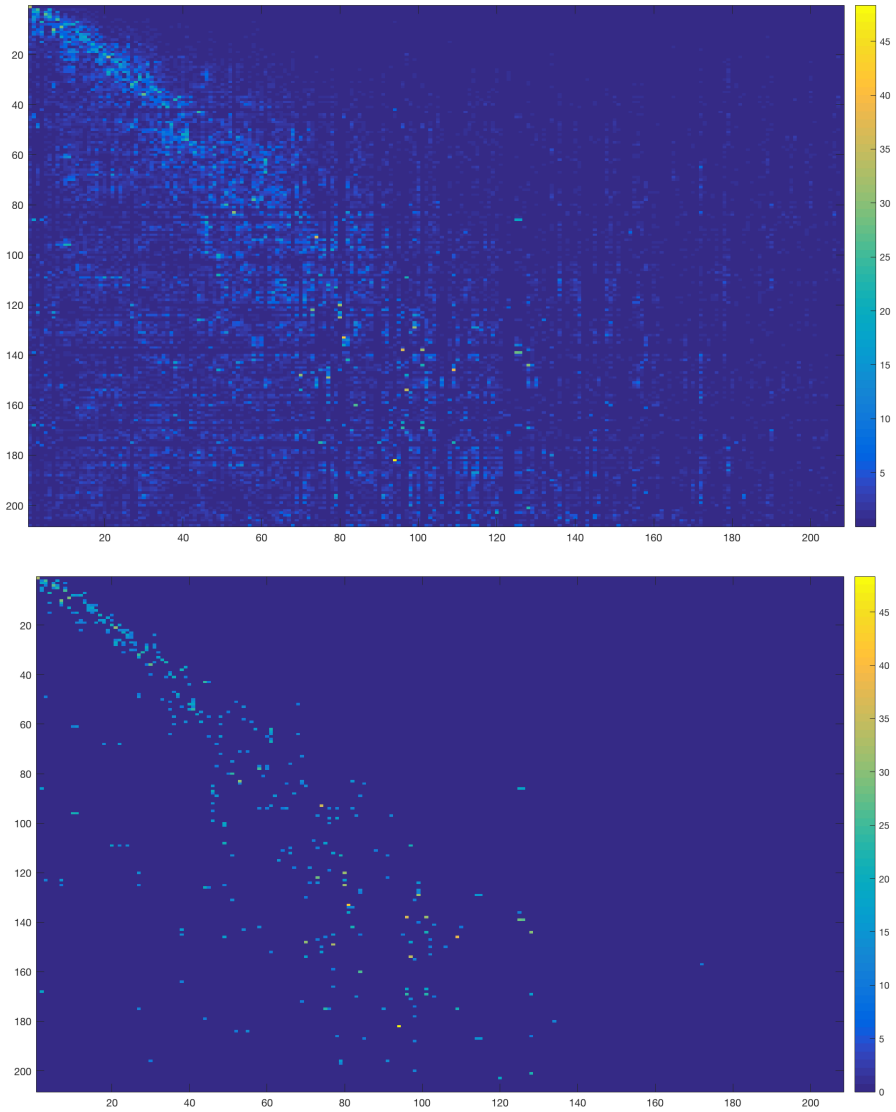


Figure 17: (a) interpolation coefficient matrix (b) threshold at $x > 10$

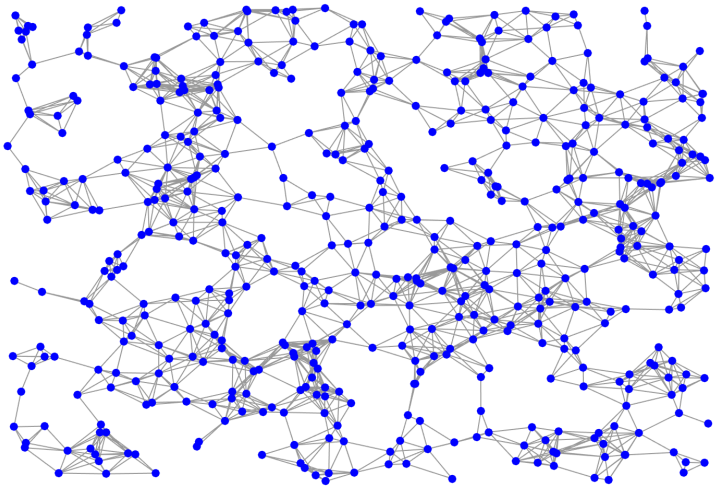
9 Other Methods

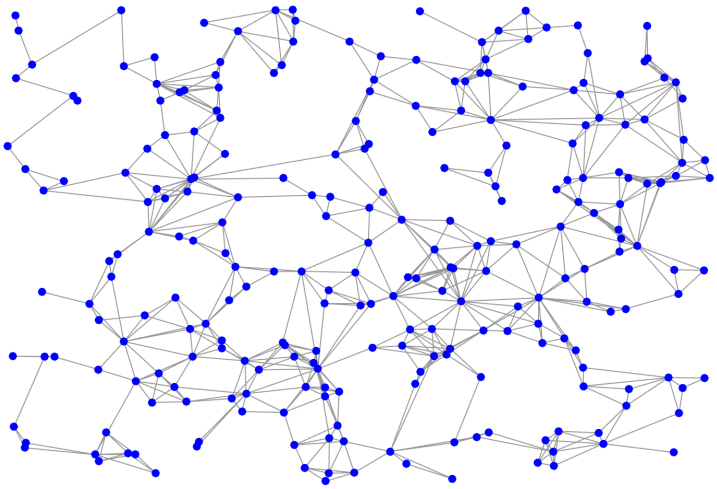
9.1 Ron et al.

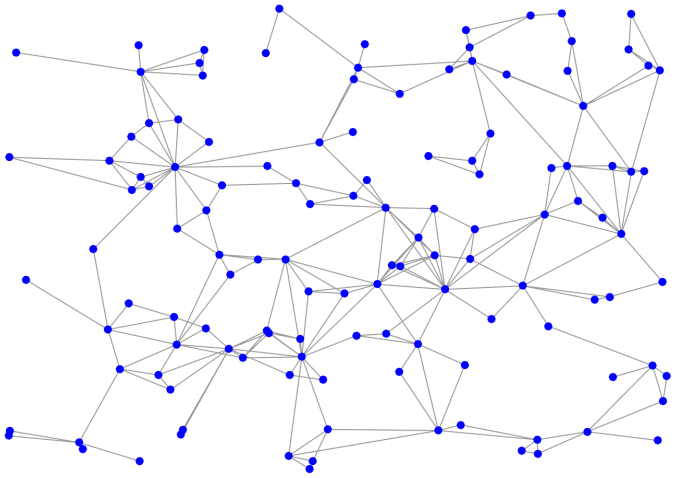
Ron et al. assign a fraction P_{ij} to a vertex's neighbors when it is eliminated. We combine this idea with pMMF. We use the downsampled graph generated by pMMF. If a givens rotation is performed between vertices i and j and j is eliminated, the $P_{j,i} = 1$. The final reduced graph is

$$W_{j,j'} = \sum PmjPnj'W_{mn}.$$

Reduced graphs:







9.2 Network Topology Identification from Spectral Templates

Background:
Consider problem P0:

$$\text{minimize}||X||_0 \text{ subject to } AX = b$$

Its relaxation is problem P1:

$$\text{minimize}||X||_1 \text{ subject to } AX = b$$

Let the graph shift operator $S \in \mathbb{R}^{N \times N}$ be the adjacency matrix of a graph. Given the spectral templates $V = v_1, \cdots, v_N$, Segarra et al try to find

$$\min_{S,\lambda} ||S||_0$$

$$\text{subject to } S = V\lambda V^T$$

$$\text{and } \lambda_1 = 0, S_{i,j} \in [-1, 0], S_{i,i} = 1$$

Let A be a heat diffusion on a given graph G, $A = G.Uf(G.e)G.U$. Perform pMMF on A and we get a core H. Our goal is to infer a graph structure based on H. Diagonalize $H = UVU^T$. We use U as our spectral templates. Our goal is to find

$$\min_{S,\lambda} ||S||_0$$

$$\text{subject to } S = V\lambda V^T$$

$$\text{and } \lambda_1 = 0, S_{i,j} \in [-1, 0], S_{i,i} = 1$$

Relaxation: l_1 norm penalized problem

$$\min_{S,\lambda} \sum\sum W_{i,j}(p)|S_{i,j}|$$

$$\text{subject to } S = V\lambda V^T$$

$$\text{and } \lambda_1 = 0, S_{i,j} \in [-1, 0], S_{i,i} = 1$$