Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Student: KONRAD HANDRICK
Discussed with: Felix Illes

Project 5 – Large Scale Graph Partitioning
Due date: 4 May 2020, 12pm (midnight)

Introduction

To familiarize myself with the assignment I read up on the mathematical background provided in the graph theory / graph partitioning PDF provided from the course, lecture notes from the course "Advanced Graph Algorithms and Optimization", given at ETH this semester and lecture notes from the course "Spectral Graph Theory" given at Yale University. To start off, I'd like to present some linear algebra facts.

Theorem 0.1 (Spectral theorem)**.** *For a matrix $\mathbf{A} = \mathbf{A}^\top \in \mathbb{C}^{n\times n}$ it holds that*

1. *all eigenvalues $\lambda_i \in \mathbb{R} \forall i$*
2. *$\mathbf{A}$ is diagonizable*
3. *the eigenvectors of $\mathbf{A}$ form an orthogonal basis*

Theorem 0.2 (Eigenvectors with Rayleigh quotients, assuming unit vectors)**.** *For a matrix $\mathbf{M}$, its Rayleigh quotient is defined as $R(M) = \frac{\mathbf{x}^\top \mathbf{M}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}}$ for a vector $\mathbf{x}$.*
Let $\mathbf{M} = \mathbf{M}^\top \in \mathbb{C}^{n\times n}, x \neq \mathbf{0}$. Then, maximizing the Rayleigh quotient will yield an eigenvector with the largest eigenvalue of $\mathbf{M}$.

Proof. $\nabla R(\mathbf{M}) = 0 \implies \mathbf{M}\mathbf{x} = R(M)\mathbf{x}$ which holds iff $\mathbf{x}$ is an eigenvector with eigenvalue $\lambda = \max_{\mathbf{x}} R(M)$. □

Theorem 0.3 (Eigenvector expansion of symmetric matrices)**.** *Let $\mathbf{M} = \mathbf{M}^\top \in \mathbb{C}^{n\times n}$ with eigenvalues $\lambda_1, ..., \lambda_n$ and eigenvectors $\psi_1, ...$ Further let $\mathbf{x} = \sum_i c_i\psi_i$. Then we have $\mathbf{x}^\top\mathbf{M}\mathbf{x} = \sum_i c_i^2\lambda_i$.*

Proof.

$$\mathbf{x}^\top\mathbf{M}\mathbf{x} = \left(\sum_i c_i\psi_i\right)^\top \mathbf{M}\left(\sum_j c_j\psi_j\right) = \left(\sum_i c_i\psi_i\right)^\top \left(\sum_j \lambda_j c_j\psi_j\right) = \sum_{i,j} c_i c_j \lambda_j \psi_i^\top\psi_j = \sum_i c_i^2\lambda_i$$

□

Theorem 0.4 (Courant-Fischer theorem, eigenbasis formulation)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ with corresponding eigenvalues $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ and orthonormal eigenvectors $\psi_1, \psi_2, ...$ Then we have*

1. $$\lambda_i = \min_{x\perp x_1,...,x_{i-1}, x\neq 0} R(\mathbf{A})$$
2. $$\lambda_i = \max_{x\perp x_{i+1},...,x_n, x\neq 0} R(\mathbf{A})$$

Here we have to introduce the basic concepts of graph theory for the nomenclature in the latter theorems to make sense.
A graph $G = (E, V)$ is a pair of edges and vertices. We define its **adjacency matrix** $\mathbf{A}$ as $\mathbf{A}_{ij} = w(i,j)\chi_{(i,j)\in E}$ where w denotes the weight of edge (i, j).
Analogously we can define the **edge-vertex incidence matrix** B with $B(v, e) = \chi_{e=(u,v)} - \chi_{e=(v,u)}$.
Lastly, we define the **Laplacian** L with $L = D - A$ where $D = \text{diag}(d)$ and $d = \sum_{(u,v)\in E} w(u, v)$ (row-wise).

We can now derive some interesting properties from this. We assume an undirected graph.

1. $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$
2. $\mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{(u,v)\in E} w(u,v)\left(\mathbf{x}(u) - \mathbf{x}(v)\right)^2$
3. $\left(\sum_j \mathbf{L}_{ij}\right) = \left(\sum_i \mathbf{L}_{ij}\right) = \mathbf{0}$
4. $0 \leq \lambda_i$ for all eigenvalues $\lambda_i \in \mathbb{R}$
5. $0 \leq R(\mathbf{L}) \leq \lambda_n$ for any vector, that is, $\mathbf{L}$ is spd and bounded by its spectral radius
6. $e = (1, ..., 1)^\top \implies \mathbf{L}e = 0 = 0v$ so the smallest eigenvalue is 0 with eigenvector e
7. $\lambda_2 = \min_{x, x\perp x^\top e} R(\mathbf{L})$
8. we define $\lambda_2(G)$ as the algebraic connectivity of a graph and its corresponding eigenvector the Fiedler vector.

The second-smallest eigenvalue is of vital importance looking at the following theorem.

Theorem 0.5 (Isoperimetric number and λ_2)**.** *Let $S \subseteq V$. Define the cut by $\partial(S) := \{(u,v) \in E : u \in S, v \notin S\}$. Further, define the **isoperimetric ratio** as $\theta(S) := \frac{|\partial(S)|}{|S|}$ and the **isoperimetric number** as $\theta_G := \min_{|S|\leq |V|/2} \theta(S)$. Then we have $\theta(S) \geq \lambda_2(1-s)$ and $\theta_G \geq \lambda_2/2$ for $s := |S|/|V|$.*

Proof. $\lambda_2 = \min_{x, x\perp x^\top e} R(\mathbf{L}) \implies \mathbf{x}^\top \mathbf{L}\mathbf{x} \geq \lambda_2 \mathbf{x}^\top \mathbf{x}$
We take the characteristic vector of S and use the beforementioned facts we found out about:
$\chi_S^\top \mathbf{L}\chi_S = \sum_{(u,v)\in E}\left(\chi_S(u) - \chi_S(v)\right)^2 = |\partial(S)|$ but the characteristic vector is not orthogonal. We take $x := \chi_S - s\mathbf{1} \implies x \perp \mathbf{1}$, so $\mathbf{x}(a) = \chi_{a\in S}(1-s) + \chi_{a\in V\setminus S}s$ (here as a characteristic function and not as the characteristic vector). We then find
$\mathbf{x}^\top \mathbf{L}\mathbf{x} = |\partial(S)|$ and $\mathbf{x}^\top \mathbf{x} = |S|(1-s)$. We can now conclude
$\lambda_2 \leq \frac{\mathbf{x}^\top \mathbf{L}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \frac{|\partial(S)|}{(1-s)|S|}$ □

Looking at the makeup of the formula we have now proven we can see that the larger $\lambda_2(G)$ the "more connected" our graph is - in our case, we have "local" graphs, ie. graphs which have geometric information associated with them. We thus expect for such sparse graph rather low numbers for λ_2. Lastly a theorem that provides the reasoning behind spectral partitioning.

Theorem 0.6. *Let $G = (V, E)$ be a connected graph and u its Fiedler vector. Take $r \geq 0, V_1 := \{v_i \in V : u_i \geq -r\}$. Then the induced subgraph V_1 is connected.*

We can equivalently formulate the balanced partitioning as an LP

$$\min \frac{1}{4} x^\top \mathbf{L} x$$
$$\text{s.t.} x \in \{\pm 1\}^n, x^\top e = 0$$

and alternatively, for a relaxed version in a weighted graph

$$\min \frac{1}{4} x^\top \mathbf{L} x$$
$$\text{s.t.} x^\top V x = e^\top V e, x^\top V e = 0$$

For the algorithms used in the assignment, we can thus exploit the Fiedler eigenvectors or the geometric information supplied in the graph object. To benchmark the performance of the given algorithms, the metric of the cut (related to the theorem above) is used: $\text{cut}(V_1, V_2) := \sum_{u \in V_1, v \in V_2} w_{uv}$.
The first idea is to immediately use the Fiedler eigenvector which we obtain by either applying the Lanzcos method which works in the induced Krylov subspace of a symmetric matrix or apply the Rayleigh quotient iteration. Both are iterative methods which converge fairly quickly and from what I gathered, Matlab uses a Cholesky-type iteration but mixtures of Krylov methods with the Rayleigh quotient iteration for sparse matrices can be utilized.
The second idea is to use our readily available geometric information and find a least squares solution to a "balanced" partitioning of the points.
In further parts of the assignment we want to expand our bisection methods to find k-way partitionings. There we use a supplied recursive version and algebraic k-way partitioning methods also supplied by Metis which use refinement and upscaling.

Algorithm 1 Recursive bisection.

Require: $\mathcal{G}(V, E)$,
Ensure: A p-way partition of $\mathcal{G}$

```
1  G_p = {C_1, ..., C_p}
2  p = 2^l                                              ▷ p is a power of 2
3  function RECURSIVEBISECTION(graph G(V,E), number of parts p)
4      function RECURSION(C', p', index)
5          if p' is even then                           ▷ If p' is even, a bisection is possible
6              p' ← p'/2
7              (C'_1, C'_2) ← bisection(C')
8              RECURSION(C'_1, p',index)
9              RECURSION(C'_2, p',index+k')
10         else                                         ▷ No more bisection possible, C' is in a partition of G
11             C_index ← C'
12         end if
13     end function
14     RECURSION(C, p, 1)
15     return G_p                                       ▷ G_p is a partition of G in p = 2^l parts
16 end function
```

Other methods
The algebraic multigrid methods (ie. k-Way partitioning which we tried later on) have several steps where we contract the graph (following the PDF, using maximally independent sets) and refine the eigenvalues we find for distinct sets using a prolongation step.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Student: KONRAD HANDRICK
Discussed with: Felix Illes

1. Implementation

Since I tried developing why spectral clustering makes sense (mostly using theory found in the ”Graph Partitioning” pdf) during the introduction I’ll stay with implementation for now.
The spectral and inertial partitioning algorithms were pretty straightforward except for the function that needed to be used for the inertial partitioning algorithm, where we needed to normalize and correct for the means when partitioning. At first I built up the matrices manually but checked with Matlab’s internal functionality and saw that ”graph()” and ”laplacian” built faster objects to operate on. Other than that, the laplacian function needed to be corrected for self-loops.
Recursively bisecting the meshes involved just calling the provided routine so not much had to be done there.
”help metismex” gives a precise manual of how to work with METIS’s functions which was also more or less straightforward (except now, the factors for the recursion had to be set differently). Interestingly, applying this functionality on the sparse datasets (usroads and luxembourg_osm) showed that METIS is parallelized but not to maximal efficiency. Seeing the software’s author explaining that METIS should still be one of the fastest graph partitioning software shows again that handling sparse matrices in parallel is, in fact, a **very** hard problem that is worth working on.
The part that took the most time for me was figuring out how to plot correctly and nicely, eg. forcing Matlab to save correct backgrounds or finding the right scaling for a given axis.

2. Results

I built METIS from source (latest rebuild on 28th of April) and included a 64-bit executable to run on my local i7 machine, this is maybe why some results differ from the calculations in the project PDF.

2.1. Bisection benchmark

For clarity, I highlighted the ”best” (according to the cut-edge metric) algorithm results in the tables below. For illustration purposes I included some graphic results below.
Interestingly, Spectral Bisection never yielded the single best result but fared well at all times - this seemed very good since it reassures our reasoning behind the first nonzero eigenvalue. Interestingly, METIS is not always optimal which underlines the point that handling graphs is, in fact, a hard problem.

Table 1: Bisection results

Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
grid5rect(10,100)	10	10	10	10
grid5rect(100,10)	10	10	10	10
gridt(40)	58	60	60	58
gridt9(30)	88	93	94	88
Smallmesh	25	12	14	22
Tapir	55	23	58	68
Eppstein	42	41	46	55

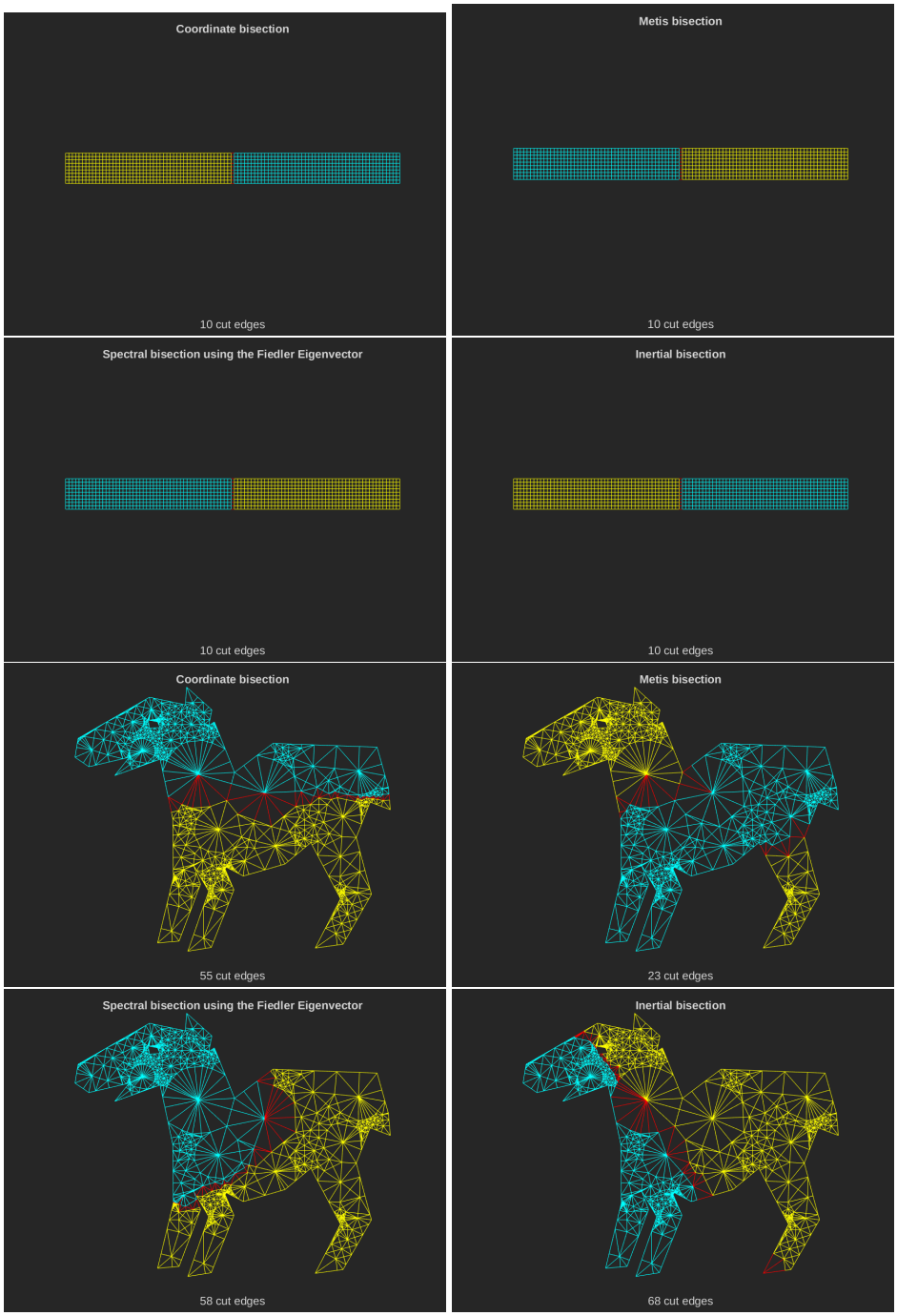

Figure 1: Example partitioning results for the first and last graphs in 1

2.2. Recursive bisection benchmark

For larger problems it is the moment for METIS to shine: it performs the best for every case in our comparison. The eigenvalue iteration for the case ”crack” does not converge which hints at an ill-conditioned problem/matrix. It does produce consistent results but listening to Matlab’s hints seems useful. In the last exercise I tried alleviating the problem introducing ”format long e;” s.t. Matlab has access to higher precision numbers.

Spectral decomposition fares extremely well - exploiting our mathematical facts approximates METIS's solution to a certain percent range which is good (compared to the other methods which have cut sizes up to an order of magnitude larger). Again, the best-performing algorithm results have been highlighted in the table. In general we can observe that METIS's strength lies absolutely in irregular sparse grids with high node/edge count.

Table 2: Edge-cut results for recursive bi-partitioning, p = 8

Case	Spectral	Metis 5.0.2	Coordinate	Inertial
mesh3e1	75	74	75	76
airfoil1	397	333	516	670
3elt	469	383	733	814
barth4	549	463	875	977
crack	883	800	1343	1351

Table 3: Edge-cut results for recursive bi-partitioning, p = 16

Case	Spectral	Metis 5.0.2	Coordinate	Inertial
mesh3e1	120	113	122	121
airfoil1	631	565	819	1081
3elt	752	693	1168	1230
barth4	837	713	1306	1492
crack	1419	1253	1860	1884

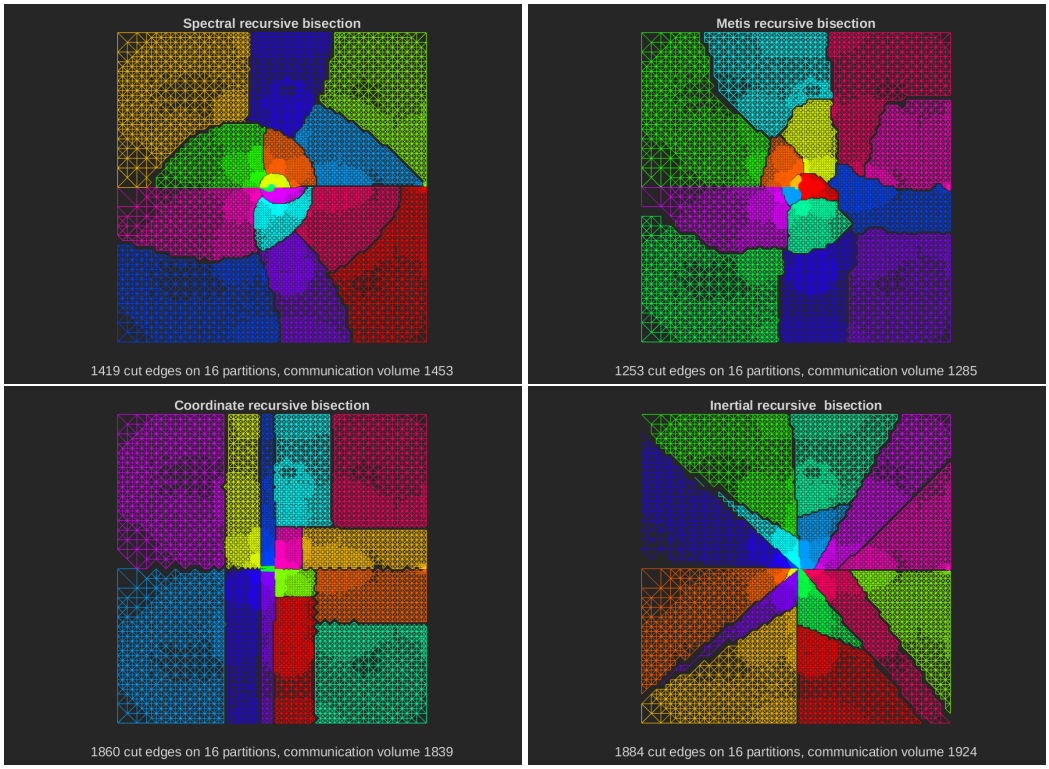

Figure 2: ”crack” recursive partitioning for p=16

2.3. Comparing recursive bisection to direct k-way partitioning

For the sparse, location-based matrices representing the road networks we expect a partitioning somewhat similar to the counties or states in a given country (from amateur economical reasoning). The results indeed indicate a somewhat similar distribution of the partitions according to states / administrative divisions. Unsurprisingly, multigrid refinement methods yielded consistently stronger results than the globally-blind recursive methods.
The interesting part not mentioned before is the ”communication” metric that is now being shown on the graphics. For example, this can either give us a sense of geographical connectedness (in this actual case) or could be used to find good matrix partitions for sparse matrices to be used for matrix-vector multiplications (cf. Project 4).
The balanced relaxations applied in the k-Way partitioning give them a) time advantages and b) better results according to our chosen metrics. The basic idea is smoothing out errors while going down (”higher” levels, but finer grid) and reapply the prolongations. That way, the partitions of the road networks may look absolutely different because the algorithm differs in that the different levels of the multigrid on completely different indices compared to the recursive partitioning, due to the method applied - which may be finding a maximally independent set, as I mentioned above.
METIS’s author probably found a near-optimal cycle that converges quickly with not too many levels.
For a further look at how METIS’s algorithms partition the road networks, have a look at the supplied folder. I generated most combinations but did not want to bloat the report too much.

Table 4: Comparing the number of cut edges for recursive bisection in Metis 5.0.2.

Partitions	Luxemburg	usroads-48
8	122	348
16	198	605
32	321	968

Table 5: Comparing the number of cut edges for direct multiway partitioning in Metis 5.0.2.

Partitions	Luxemburg	usroads-48
8	104	321
16	174	553
32	291	932

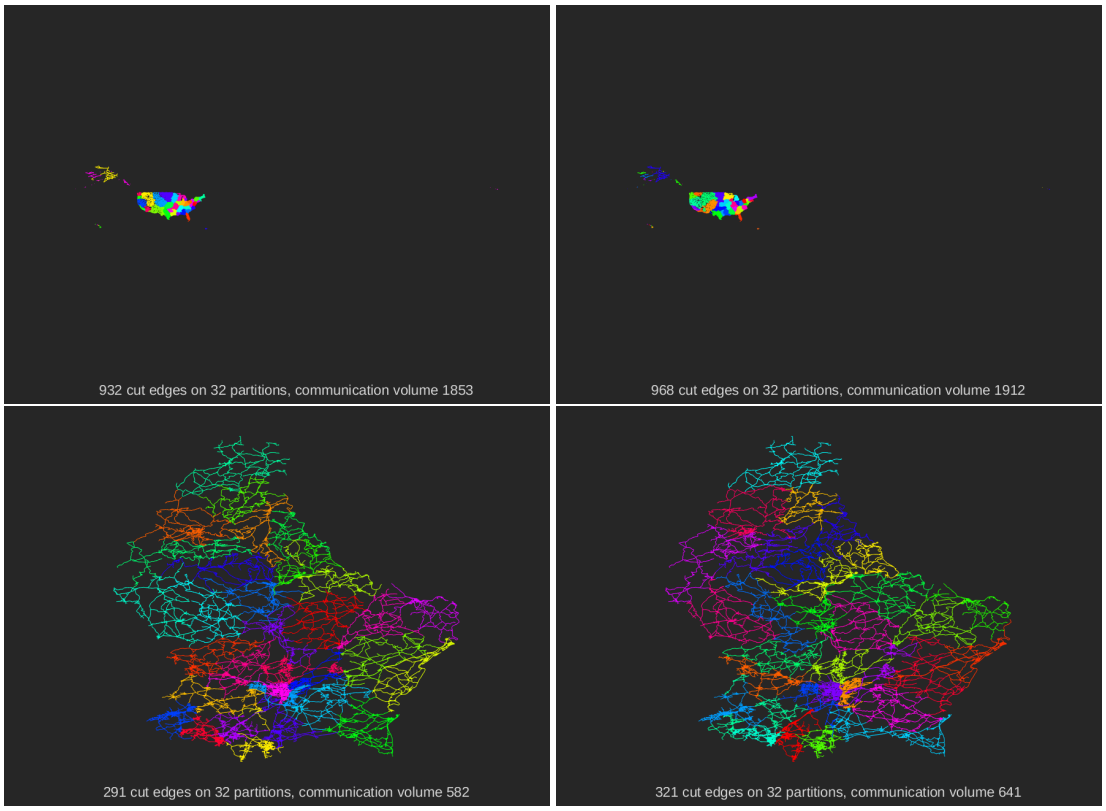

Figure 3: comparing the map graph partitions (kway vs recursive) ”usroads” and ”luxembourg_osm” for p=32

2.4. Visualizing graphs using eigenvectors

2.5. 5a

For the eigenvectors corresponding to the first two smallest eigenvectors should yield a) a constant first one b) and a ”wave-like” second one looking at the graph ”airfoil”: the graph is a) connected and that b)locally (seeing the geometric makeup of the graph).
Indeed, this is what we observe. The theorem developed above does apply to normalized Laplacians which we do not

treat here so the smallest eigenvector's entries should be constant one up to multiplication by a scalar.

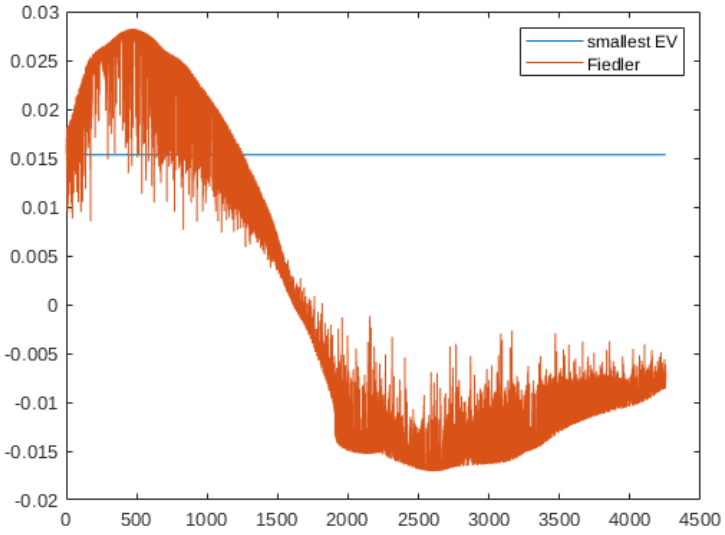

Figure 4: Entries of first and Fiedler eigenvector of "airfoil" graph

2.6. 5b

Projecting the Fiedler eigenvector onto the coordinate system associated with a geometric graph is an interesting exercise in testing one's own assumption about the understanding of a given mesh. For example "barth4" is densely connected in a small area which we do not observe just looking at its geometry. The separation happens unintuitively. Scattering the points across the graphs representation proved the hardest part of the assignment to me but was, like most implementation details, solved by simply extending the range in the z direction and holding the layout upon scattering for three dimensions.
Obtaining a useful representation of the Fiedler eigenvector's entries proved also difficult in that the scaling had to be adjusted according to 'DataAspectRatio' and 'PlotBoxAspectRatio'.
Overall, the result manifests the ideas that have been manifesting since looking at the theory; topology stays during projection and the theorem 0.6 gives us a "balanced" way of partitioning a graph using global information without using too much locality.

2.7. 5c

The overall topology of a given graph is preserved. The overall method of using the second and third eigenvectors can give us a lot of information about our assumptions of the topology of a given graph. In general we can observe that our geometric graphs are "well-separable", that is, a "balanced" amount of nodes has $v_2^i \leq 0$ or $v_2^i \geq 0$. The intuition being developed on the spectral properties of a graph with localisation information is thus once again reassured. The structure of the graph overall is projected in the eigenvalues, that is, we can observe the features somewhat curved according to the entries in the Fiedler eigenvectors. Additionally we observe that the more regular the distancing of a grid, the more the structure of the graph is preserved in the spectral space.

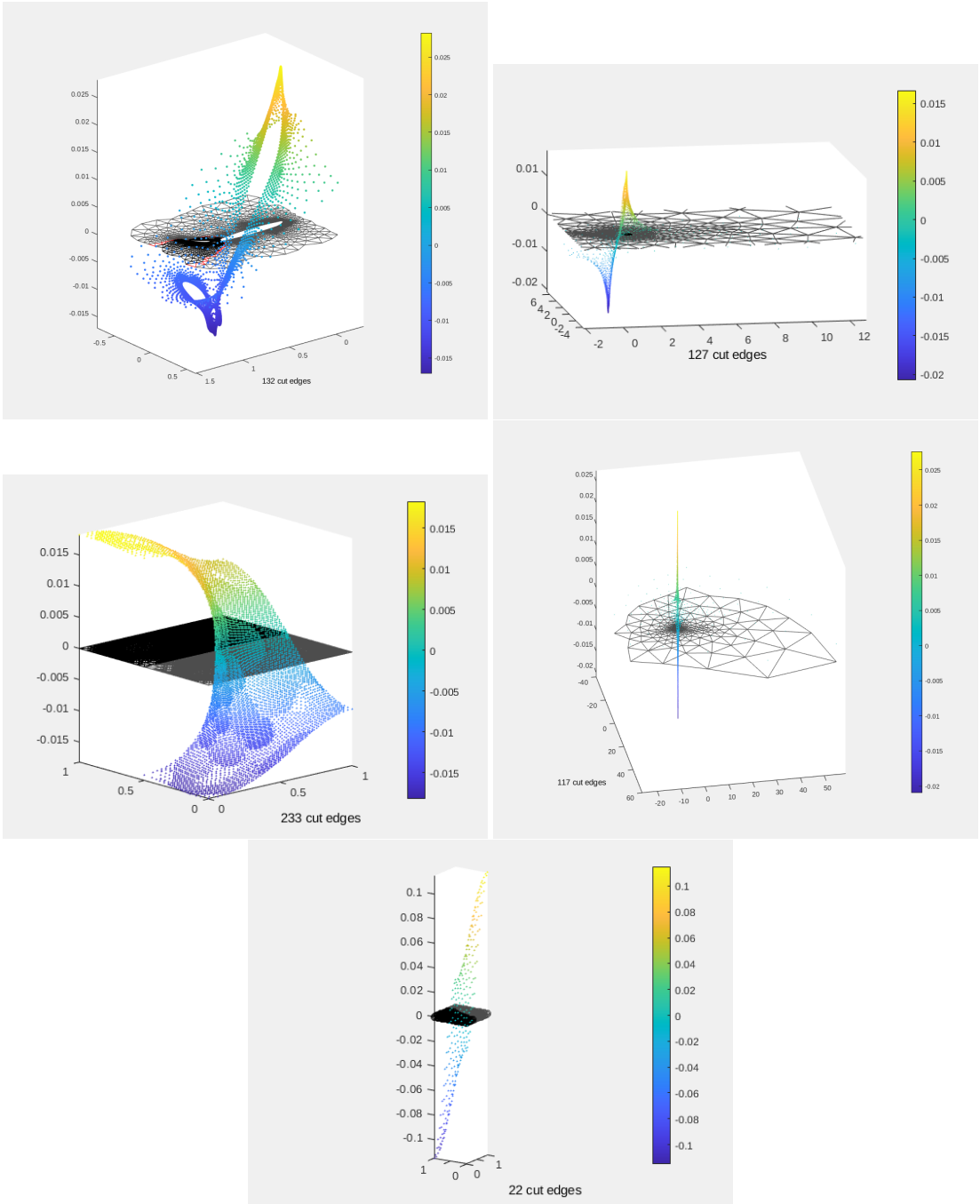

Figure 5: Fiedler vector projected onto geometric graphs ”airfoil”, ”barth4”, ”crack”, ”3elt”, ”mesh3e1”. The scaling of the Fiedler eigenvector had to be done manually to give a visual representation - that is, the scaling of the z-axis varies throughout.

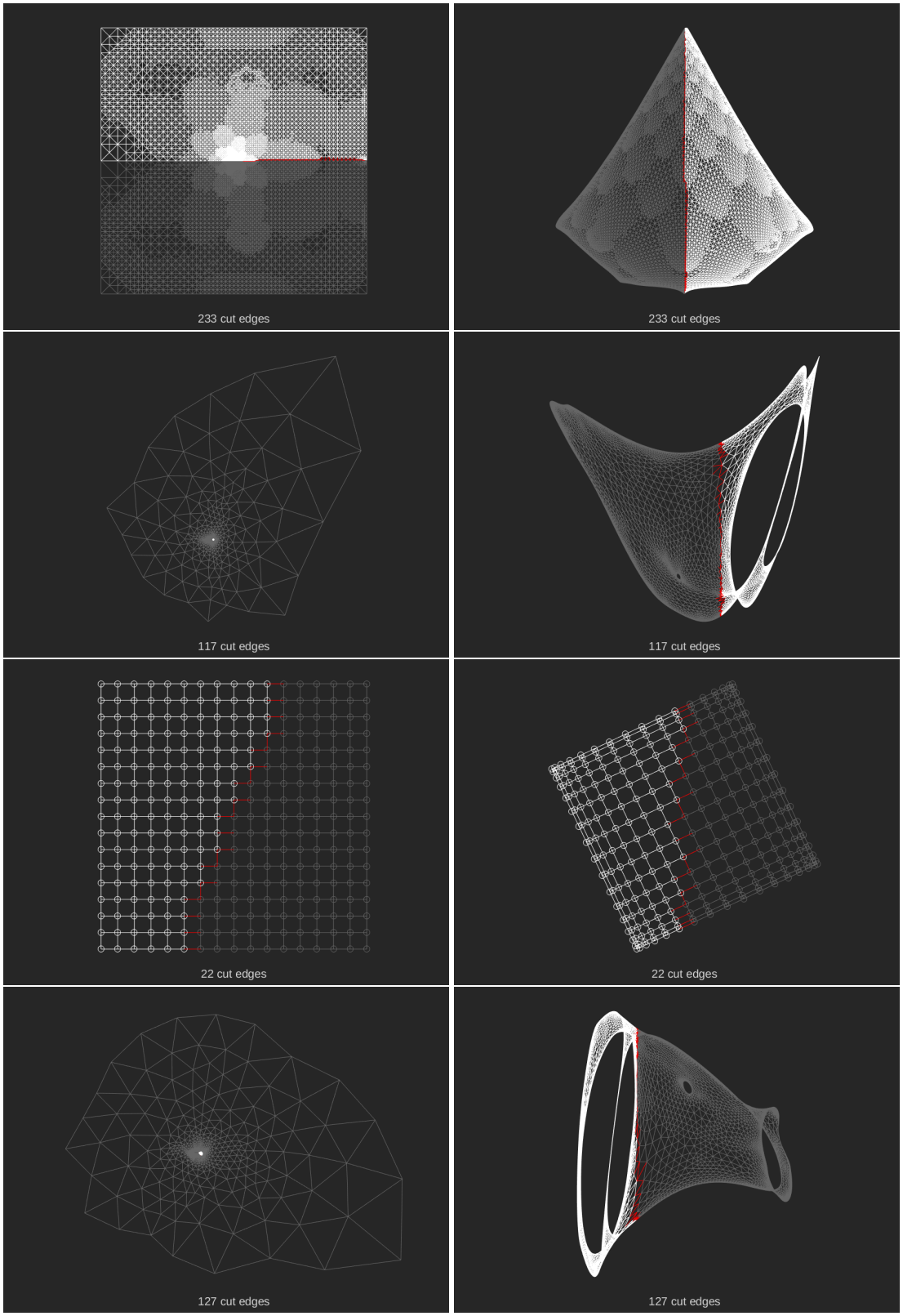

Figure 6: Plotting as presented in the project PDF - usual representation on xy axis versus (-)v2v3 axis (where v2, v3 are the eigenvectors associated to second and third largest eigenvalue respectively): ”crack”, ”3elt”, ”mesh3e1”, ”barth4”.