Università della Svizzera italiana

Institute of Computing
CI

High-Performance Computing Lab **Institute of Computing**

Student: Paolo Deidda Discussed with: Lino Candian

Solution for Project 6

HPC Lab — Submission Instructions
(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, Matlab). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  Project_number_lastname_firstname
  and the file must be called:
  project_number_lastname_firstname.zip
  project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

Contents

1. Graph Partitioning with Matlab: Load balancing for HPC

$\sim$ *No tasks here*

2. Graph partitioning with Matlab: Exercises [85 points]

2.1. Task: Install METIS 5.0.2, and the corresponding Matlab mex interface

2.2. Task: Construct adjacency matrices from connectivity data [10 points]

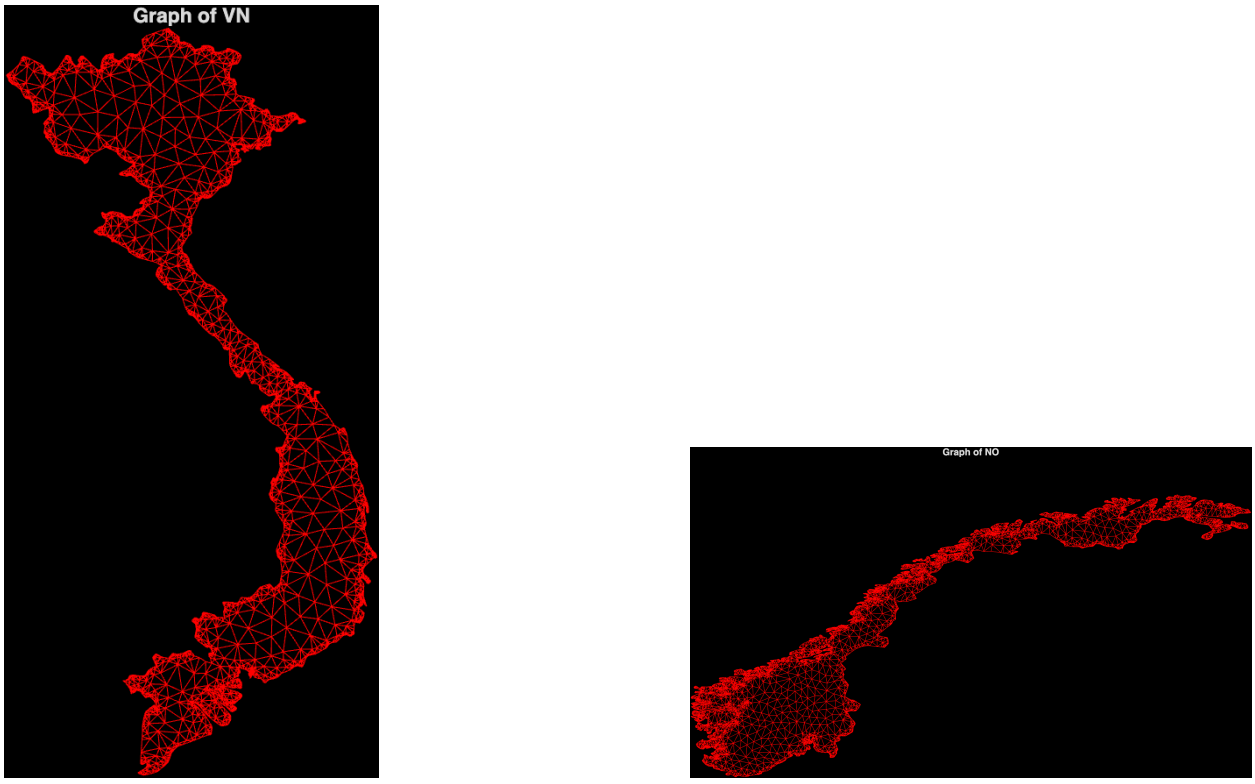
We used the script `Source/read_csv_graphs.m` to handle the raw connectivity data that was given in CSV format. This dataset contains road networks from various countries, where the nodes represent intersections and the edges signify road segments.

The procedure consisted of parsing the coordinate files provided to build the list $C \in \mathbb{R}^{n\times 2}$ and the adjacency files to construct the sparse matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$. A crucial part of this construction was making sure the adjacency matrix was symmetric, which is necessary for the spectral partitioning algorithms we would be using later on. In the source code, we achieved this symmetry by applying the following transformation:

$$\mathbf{W} = \frac{\mathbf{W} + \mathbf{W}^T}{2} \tag{1}$$

This step fixes any directional issues in the raw data, ensuring that the graph is treated as undirected.

We saved the processed sparse matrices and coordinates into binary .mat files for quicker loading in the next tasks. Then, we visualized the road networks for Vietnam and Norway using the plotting tools provided. The visualizations of these networks are displayed in Figure 1.



(a) Vietnam road network

(b) Norway road network

Figure 1: Visual representation of the constructed graphs for Vietnam and Norway. The nodes are plotted according to their geographical coordinates.

2.3. Task: Implement various graph partitioning algorithms [25 points]

Script `Bench_bisection.m` runs a series of benchmark tests and visualizations on different toy meshes. Most of the intermediate displays are meant for later use, so I won't go into detail about them here. If you take a look at the script, you'll see how each mesh is loaded and processed, which helps make sure that all the meshes included in the assignment show up in the final table.

The spectral bisection method was implemented in `bisection_spectral.m`. This method is based on the graph Laplacian, as outlined in the theoretical framework, so it uses a symmetric adjacency matrix. The relevant partitioning information is extracted from the Fiedler eigenvector, obtained by computing the two smallest eigenvalues via `eigs` with the option `'smallestreal'` to improve numerical robustness.

The inertial bisection routine was completed in `bisection_inertial.m`. In this method, the dividing line is set up to go through the center of mass of the coordinates. The direction of this line is guided by the eigenvector linked to the smallest eigenvalue of the inertia matrix, helping to achieve a well balanced geometric partition.

Table 1: Bisection results

Mesh	**Coordinate**	**Metis 5.0.2**	**Spectral**	**Inertial**
mesh1e1	18	18	18	19
mesh2e1	37	34	35	37
mesh3e1	19	20	22	19
mesh3em5	19	20	142	19
airfoil1	94	93	132	124
netz4504_dual	25	19	23	39
stufe	16	17	16	40
3elt	172	96	117	182
barth4	206	111	127	253
ukerbe1	32	36	28	88
crack	353	220	233	323

2.4. Task: Recursively bisecting meshes [15 points]

The recursive bisection algorithm is implemented in `rec_bisection.m`. Using the script `Bench_rec_bisection.m`, each mesh is recursively partitioned into 8 and 16 subgraphs. At each level of recursion, the chosen bisection method comes into play, helping us see how different strategies like spectral, Metis, coordinate-based, and inertial affect the overall edge-cut during the hierarchical breakdown. You can find a summary of the edge-cut values for all scenarios in Table 2. The visual inspection requested for the case *crack* with $p = 16$ is also performed within the same script.

Table 2: Edge-cut results for recursive bi-partitioning.

Graph	Spectral		Metis 5.0.2		Coordinate		Inertial	
	8	16	8	16	8	16	8	16
airfoil1	397	631	318	561	516	819	1276	2531
netz4504_dual	111	185	96	159	127	198	237	483
stufe	128	243	108	193	123	227	265	495
3elt	469	752	418	699	733	1168	881	1608
barth4	549	835	470	743	875	1306	1299	2353
ukerbe1	126	236	147	245	225	374	435	785
crack	883	1419	808	1275	1343	1860	1706	3447

2.5. Task: Comparing recursive bisection to direct k-way partitioning [10 points]

Script `Bench_metis.m` is used to compare two Metis strategies for obtaining a k-way partition: recursive bisection and direct multiway partitioning. We apply both methods to the chosen graphs for $k = 16$ and $k = 32$. The recursive strategy involves splitting the graph into two subgraphs repeatedly, while the direct method calculates the entire k-way partition using Metis' multilevel refinement process. The goal is to evaluate how the two methods differ in terms of edge-cut and determine whether the observed behavior aligns with expectations based on the underlying algorithmic design. The resulting cut values are reported in Table 3.

Table 3: Comparing the number of cut edges for recursive bisection and direct multiway partitioning in Metis 5.0.2.

K- time	k = 16		k = 32	
Country	Rec	Kway	Rec	Kway
Luxemburg	191	185	317	304
US (usroads-48)	585	545	983	921
Greece	318	301	500	486
Switzerland	318	301	500	486
Vietnam	270	231	445	418
Norway	271	241	509	436
Russia	572	551	941	931

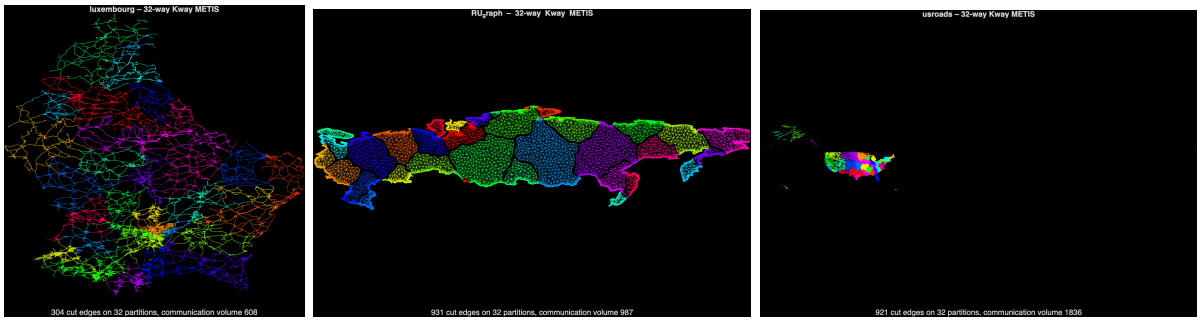


Figure 2: Metis recursive vs. direct k-way visualizations for $k = 32$.

2.6. Task: Utilizing graph eigenvectors [25 points]

In this section we analyze the spectral properties of the graph Laplacian matrix L and their application to graph drawing and partitioning.

We computed the first two eigenvectors (v_1, v_2) associated with the smallest eigenvalues of the Laplacian for the `airfoil1` graph. The results are shown in Figure 3.

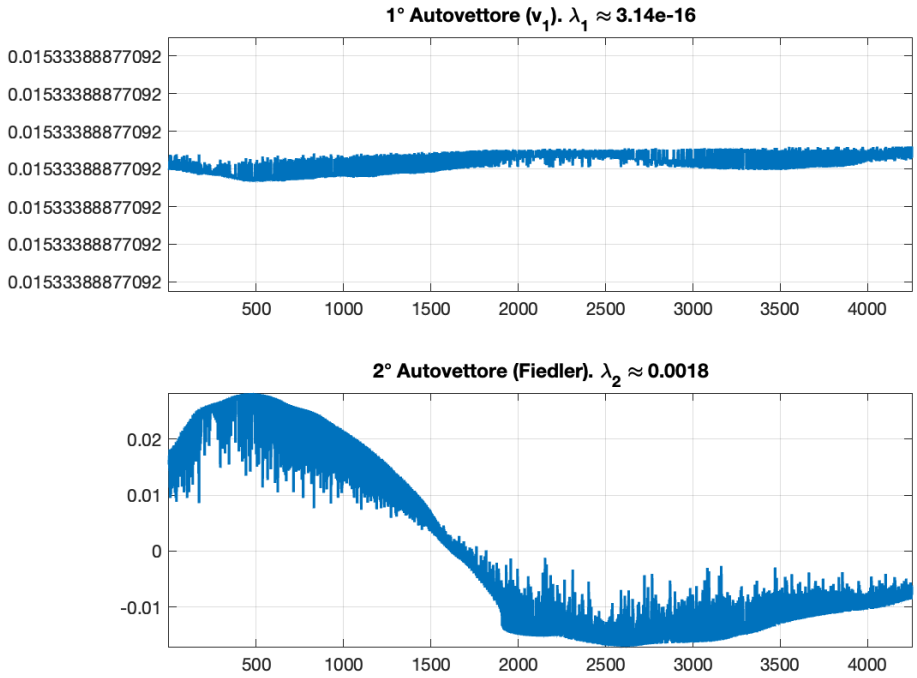
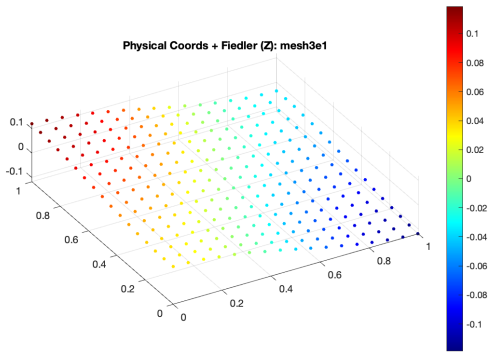


Figure 3: First and Second Eigenvectors of the `airfoil1` graph Laplacian.

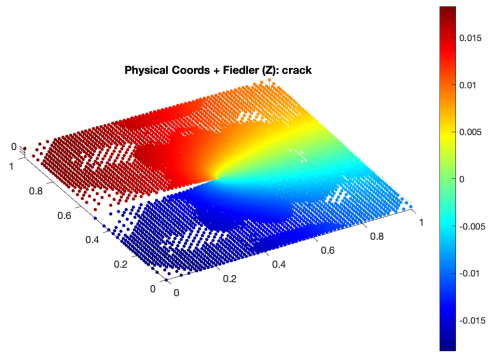
- The first eigenvector v_1 (top plot) corresponds to the eigenvalue $\lambda_1 \approx 0$ (computed as $3.14e^{-16}$). Since the graph is connected, the multiplicity of the zero eigenvalue is 1, and the associated eigenvector is constant ($v_1 = c \cdot \mathbf{1}$).
- The second eigenvector v_2 (bottom plot) is the *Fiedler Vector*. It is orthogonal to v_1 and its values vary smoothly across the graph indices, crossing zero. This variation provides the heuristic for spectral bisection: nodes with $v_2 < 0$ belong to one partition, and nodes with $v_2 \geq 0$ to the other.

Fiedler Vector Projection (Physical Space)

We visualized the values of the Fiedler vector (v_2) by projecting them onto the physical coordinate system of four different meshes. The value of v_2 is represented by both the Z-axis height and the color map.



(a) mesh3e1

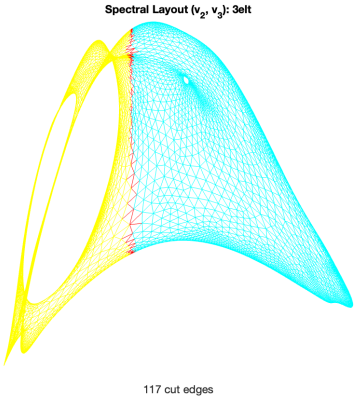


(b) crack

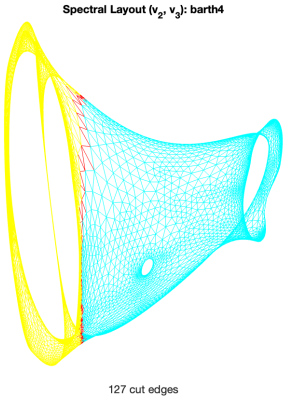
Figure 4: Projection of the Fiedler vector (v_2) values onto the physical coordinates of the meshes. The color gradient shows how the spectral value varies smoothly across the geometry, identifying the optimal cut.

Spectral Graph Drawing

Performing spectral graph drawing by utilizing the second (v_2) and third (v_3) eigenvectors as Cartesian coordinates (x, y) for the nodes. The results for `3elt` and `barth4` are shown below.



(a) 3elt Spectral Layout



(b) barth4 Spectral Layout

Figure 5: Spectral Graph Drawing using coordinates (v_2, v_3). The colors (Cyan/Yellow) represent the bisection cut. In this spectral space, nodes are clustered by connectivity rather than physical distance, making the partition boundary appear like a clear line.

In this layout, the nodes that are closely connected end up near each other "unfolding" the graph such that the bisection cut looks more geometrically simple.

3. Task: Quality of the Report [15 Points]

Additional notes and submission details

Submit the source code files (together with your used `Makefile`) in an archive file (tar, zip, etc.), and summarize your results and the observations for all exercises by writing an extended Latex report. Use the Latex template from the webpage and upload the Latex summary as a PDF to iCorsi.

- Your submission should be a gzipped tar archive, formatted like project_number_lastname_firstname.zip or project_number_lastname_firstname.tgz. It should contain:
  - all the source codes of your MATLAB solutions;
  - your write-up with your name project_number_lastname_firstname.pdf.
- Submit your .zip/.tgz through Icorsi.