Università della Svizzera italiana | Faculty of Informatics | Institute of Computing CI

Numerical Computing **2023**

Student: Kyla Kaplan

Discussed with:

Solution for Project 2

Due date: Wednesday, 25 October 2023, 11:59 PM

Numerical Computing 2023 — 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.

1. The assignment

1.1. Implement various graph partitioning algorithms [50 points]

Summarize your results in table 1.

Table 1: Bisection results

Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
grid5rec(12,100)	12	12	12	12
grid5rec(100,12)	12	12	12	12
grid5recRotate(100,12,-45)	22	12	12	12
gridt(50)	72	82	78	72
grid9(40)	118	127	128	118
Smallmesh	25	12	14	30
Tapir	55	23	58	49
Eppstein	42	41	47	45

We were given 4 algorithms to benchmark: **Coordinate**, **Metis**, **Spectral** and **Inertial**. While Coordinate and Metis have been implemented, we were instructed to re-implement non-naively the algorithms for Spectral and Inertial partitioning. Their respective implementations are attached in the accompanying Zip file.

During execution, I was able to observe that although Metis and Coordinate are perhaps, not the most accurate in terms of partitioning, they are faster; with Spectral being the most accurate, but the slowest. This is due to the fact that it is much more computationally expensive to find the eigenvectors/eigenvalues for the larger input matrices.

1.2. Recursively bisecting meshes [20 points]

Summarize your results in table 2.

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

Case	Spectral ($p=8$)	Spectral ($p=16$)	Metis ($p=8$)	Metis ($p=16$)	Coord. ($p=8$)	Coord. ($p=16$)	Inertial ($p=8$)	Inertial ($p=16$)
mesh3e1	58	58	57	57	63	63	59	59
bodyy4	1093	1839	985	1591	1065	1951	1363	2212
de-2010	742	1340	491	897	929	1796	1080	1996
biplane-9	510	899	465	845	548	974	647	1092
L-9	705	1122	637	1019	631	1028	828	1378

The code for execution and visualization can be found in the accompanying Zip file.

Following are the visual results p = 8 and p = 16 for the case "de-2010":

(a) (b)

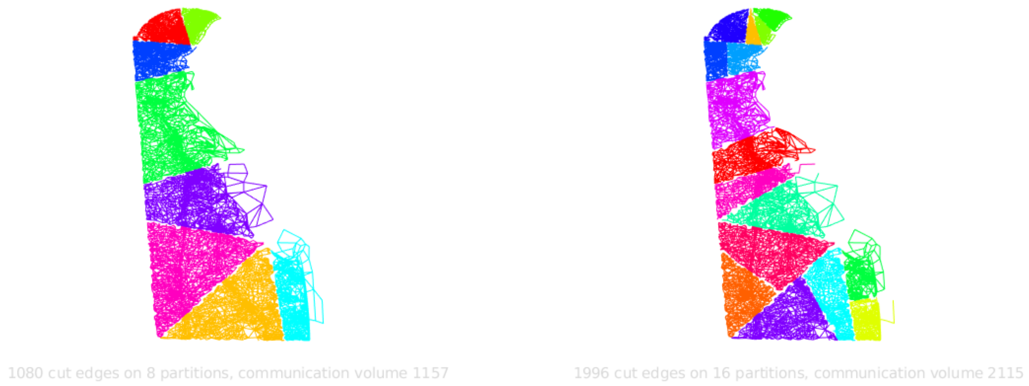
Figure 1: Spectral algorithm (a) p = 8 (b) p = 16

(a) (b)

Figure 2: Metis 5.1.0 algorithm (a) p = 8 (b) p = 16

(a) (b)

Figure 3: Coordinate algorithm (a) p = 8 (b) p = 16



(a) (b)

Figure 4: Inertial algorithm (a) p = 8 (b) p = 16

Recursively running the algorithms splits them into graphs/subgraphs for each respective input case. Recursion is advantageous especially if there is a possibility to be able to parallelize/scale the computations. In this case, p is the levels of partitioning/levels of recursion. In the code, the following variables...

```
nlevels_a = 3;
nlevels_b = 4;
```

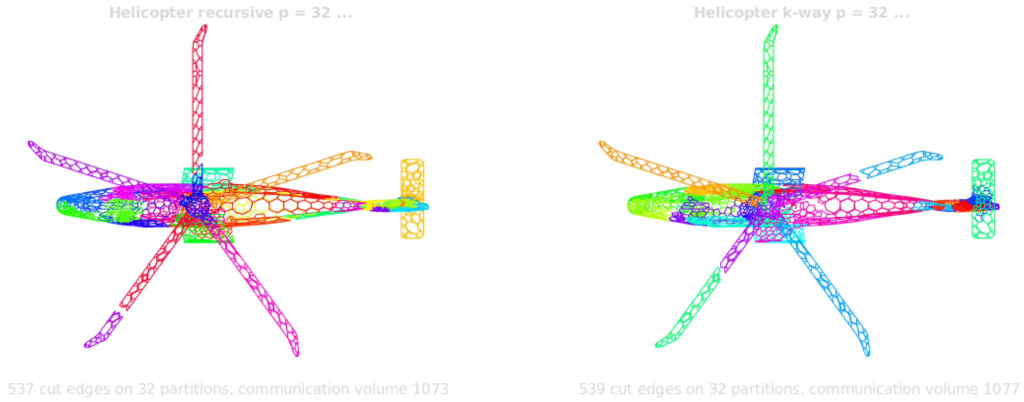
...define whether the graph is split into 8 or 16 partitions respectively. We can notice from the table that the different algorithms behave differently when it comes to edge-cut performance. The recursive approach generally results in greater edge-cuts. Inertial yields low edge cuts as it always computes eigenvectors of matrix $M \in R^{2\times 2}$.

1.3. Comparing recursive bisection to direct k-way partitioning [15 points]

Summarize your results in table 3.

Table 3: Comparing the number of cut edges for recursive bisection and direct multiway partitioning in Metis 5.1.0.

Partitions	Helicopter	Skirt
16 - recursive bisection	343	3119
16-way direct partition	324	3393
32 - recursive bisection	537	6075
32-way direct partition	539	6051



(a) (b)

Figure 5: Helicopter (p = 32): (a) Recursive algorithm (b) K-way algorithm

(a) (b)

Figure 6: Skirt (p = 32): (a) Recursive algorithm (b) K-way algorithm

As the results and visualizations show, adjusting the partition by double does yield a higher edge cut result. Additionally, it is expected that the k-way/direct bisection would perform better in both cases due to it being an algorithm with higher computational efficiency (with minor computational deviations/exceptions).

As mentioned in the project description, a better view of the graph bi-sectioning can be seen upon calling the following code in the MATLAB command window (already included in code execution):

```
>> rotate3d on;
```