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

Numerical Computing **2023**

Student: Hun Rim Discussed with: Georgy Batyrev

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]

The table below is a record of edgecuts calculated during the graph partitioning of various mesh graphs using various partitioning algorithms. 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	140	118
Smallmesh	25	12	14	30
Tapir	55	23	58	49
Eppstein	42	41	47	45

Out of those 4 algorithms, 2 (Coordinate and Metis) were given whereas Spectral and Inertial had to be implemented. The following are the key factors of implementation for Spectral bisection according to the given psedo-code in the pdf file:

```
% 1. Construct the Laplacian.
G = graph(A,'omitselfloops');
L = laplacian(G);
% 2. the correspoding vector of second smallest eigen value
[V, ~] = eigs(L, 2, 'SM');
% 3. Label the vertices with the components of the Fiedler vector.
w = V(:,2);
% 4. Partition them around their median value, or 0.
threshold = median(w);
part1 = find(w < threshold);
part2 = find(w >= threshold);
```

The code snippet below shows the implementation of Inertial bisection:

```
% 1. Calculate the center of mass (lets call it 'com').
com = mean(xy);
% 2. Construct the matrix M.
Mxx = 0, Myy = 0, Mxy = 0;
for i = 1:length(xy)
    Mxy = Mxy + (xy(i,1)-com(1))*(xy(i,2)- com(2));
    Myy = Myy + (xy(i,2)-com(2))^2;
    Mxx = Mxx + (xy(i,1)-com(1))^2;
end
M = [Mxx, Mxy; Mxy, Myy];
% 3. Calculate the smallest eigenvector associated with M.
[Eigenvector, Eigenvalue] = eigs(M);
[~, minIndex] = min(diag(Eigenvalue));
smallest = Eigenvector(:, minIndex);
% 4. Find the line L on which the center of mass lies.
```

```
L = smallest(1) * (xy(:,2) - com(2)) - smallest(2) * (xy(:,1) - com(1));
% 5. Partition the points around the line L.
medianL = median(L);
part1 = find(L < medianL);
part2 = find(L >= medianL);
```

1.2. Recursively bisecting meshes [20 points]

Result of recursive bi-partitioning for $p = 8, l = 3$ can be seen in Table 2, and result for $p = 16, l = 4$ can be seen in Table 3.

Table 2: Edge-cut results for recursive bi-partitioning. $p = 8, l = 3$

Case	Spectral	Metis 5.1.0	Coordinate	Inertial
mesh3e1	58	57	63	59
bodyy4	1093	985	1065	1364
de-2010	742	491	929	1084
biplane-9	510	465	548	648
L-9	705	637	631	828

Table 3: Edge-cut results for recursive bi-partitioning. $p = 16, l = 4$

Case	Spectral	Metis 5.1.0	Coordinate	Inertial
mesh3e1	58	57	63	59
bodyy4	1839	1591	1951	2214
de-2010	1340	897	1796	2002
biplane-9	899	845	974	1093
L-9	1122	1019	1028	1377

The results above are generated using the Bench_rec_bisection.m script. The following are the key implementations in the script:

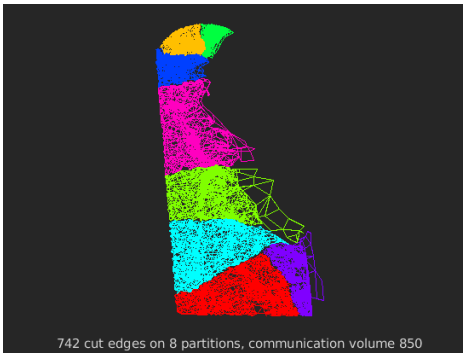
```
%  @bisection_function: A pointer to function which carries out the
bisection. i.e. bisection_inertial.
%  W: Original Square Matrix
%  Coord: Given coordinates of the vertices.

% Returns the map after the recursive bisection.
recursiveBisection = rec_bisection(@bisection_function, log2(p), W,...
coords, 0);

% Returns the cutsize of the bisection.
bisectCutSize = cutsize(W, recursiveBisection);

% Plots the resulting map of a bisection
gplotmap(W, coords, recursiveBisection);
```

As it can be seen in the key implementations, the result of case 'de-2010' is visualized in the next page.



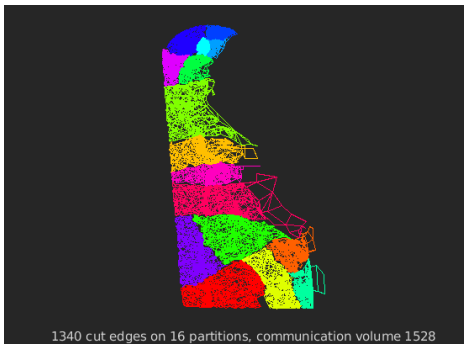
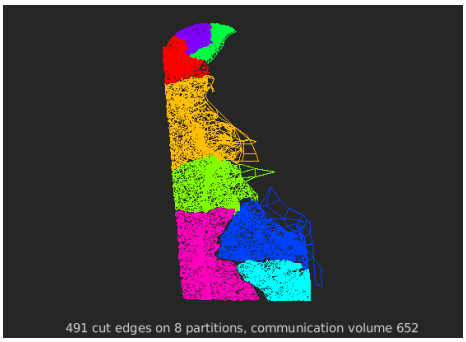


Figure 1: Recursive Spectral Bisection $p = 8, l = 3$ left and $p = 16, l = 4$ right



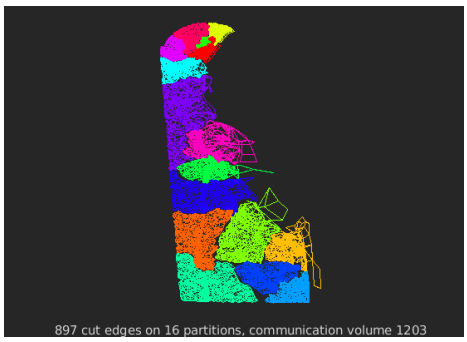
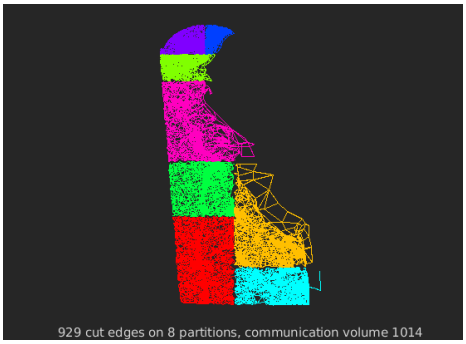


Figure 2: Recursive Metis Bisection $p = 8, l = 3$ left and $p = 16, l = 4$ right



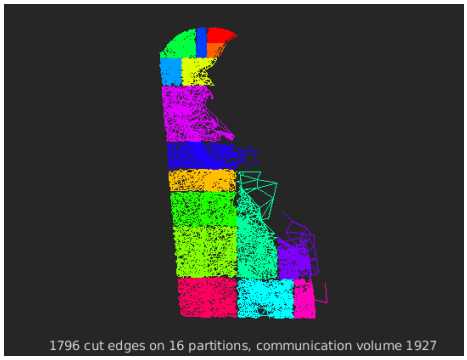
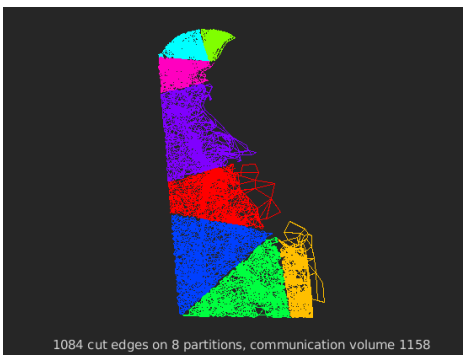


Figure 3: Recursive Coordinate Bisection $p = 8, l = 3$ left and $p = 16, l = 4$ right



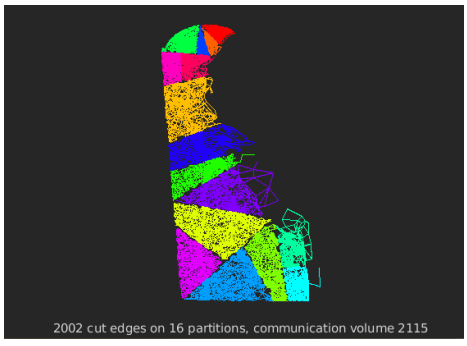


Figure 4: Recursive Inertial Bisection $p = 8, l = 3$ left and $p = 16, l = 4$ right

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

Results on table 4 shows a similar performance as the number of edgecuts are very close to each other in all cases and the number of edge cuts almost doubled in all of the examples when the k increased from 16 to 32. There might be some execution(calculation) time difference but number of edgecuts showed a very similar and predictable performance as suspected.

Table 4: 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

The following are the visualization of *helicopter* and *skirt* models using different multiway partitioning methods given by the Metis 5.1.0.

Figure 5: Visualization of Helicopter using 32-recursive bisection (left) and 32-way direct bisection(right).

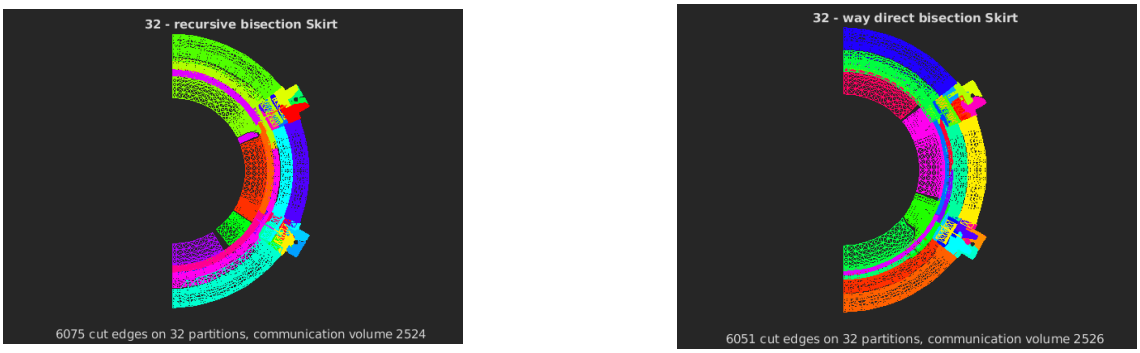


Figure 6: Visualization of Skirt using 32-recursive bisection (left) and 32-way direct bisection(right).

The information on the table and visualization are obtained through running Bench_metis.m script. In the next page are the key factors of implementation for the Bench_metis.m script.

```
%To get a result of recursive bisection on MatrixA.
[Map, edgeCutCount] = metismex('PartGraphRecursive', MatrixA, level);

%To get a result of K-way bisection on MatrixA.
[Map, edgeCutCount] = metismex('PartGraphKWay', MatrixA, level);

%Resultant map of partition can be partitioned as the following
gplotmap(MatrixA, CoordinatesA, Map);

%To turn on the 3D view
rotated3d on; % has to be called on a figure.
```

Please view the Bench_metis.m script to see the full implementation of the task. The graphs generated from the script allows the viewer to rotate the graph around and view it in 3D perspective.