Numerical Computing **2023**

Student: Harkeerat Singh Sawhney

Solution for Project 2

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

1. The assignment

1.1. Implement various graph partitioning algorithms [50 points]

In this Question we were asked to implement two function, one for Spectral Bisection adn the other for Inertial Bisection.

```matlab
    % 1. Construct the Laplacian matrix
    n = size(A,1);
    D = diag(sum(A,2));
    L = D - A;

    % 2. Calculate its eigensdecomposition.
    [V,D] = eigs(L,2,'sm');

    % 3. Label the vertices with the components of the Fiedler vector.
    fiedler = V(:,2);

    % 4. Partition them around their median value, or 0.
    median_value = 0;
    part1 = find(fiedler < median_value);
    part2 = find(fiedler >= median_value);
```

The above code is the implementation of Spectral Bisection. As it can be seen from the above code there are 4 steps to implement it. The first step is to construct the Laplacian matrix, which is done by the equation $L = D - W$, where D is a diagonal matrix with the degree of each vertex on the diagonal and W is the adjacency matrix. The second step is to calculate the eigensdecomposition of the Laplacian matrix. This was done by using the MATLAB function `eigs`. The function lets us choose the number of eigenvalues and eigenvectors we want, in which we get the second smallest eigenvalue and its corresponding eigenvector. The third step is to label the vertices with the components of the Fiedler vector, which is done by taking the second smallest eigenvector. At last we are supposed to partition the vertices around the median value. Median value can either be 0 or the median of the Fiedler vector. In our case for simplicity we have chosen the median value to be 0. The vertices with the Fiedler vector value less than 0 are assigned to one partition and the vertices with the Fiedler vector value greater than or equal to 0 are assigned to the other partition. With this we are able to partition the graph into two parts.

```matlab
    % 1. Calculate the center of mass.
    x = xy(:,1);
    y = xy(:,2);

    x = sum(x)/length(x);
    y = sum(y)/length(y);

    center_of_mass = [x,y];

    % 2. Construct the matrix M.
    %  (Consult the pdf of the assignment for the creation of M)

    Sxx = sum((xy(:,1) - center_of_mass(1)).^2);
    Syy = sum((xy(:,2) - center_of_mass(2)).^2);
    Sxy = sum((xy(:,1) - center_of_mass(1)).*(xy(:,2) - center_of_mass(2)));
    M = [Sxx, Sxy; Sxy, Syy];

    % 3. Calculate the smallest eigenvector of M.
    [V,D] = eigs(M,2,'sm');
    v = V(:,1);


    % 4. Find the line L on which the center of mass lies.
    orthogonal_vector = [v(2), -v(1)];

    % 5. Partition the points around the line L.
    %   (you may use the function partition.m)
    [part1,part2] = partition(xy,orthogonal_vector);
```

The above code is the implementation of Inertial Bisection. The main logic behind this algorithm is to find the center of mass of the graph and then find the line on which the center of mass lies. The vertices are then partitioned around the line. Hence the first step is to calculate the center of mass. This is done by taking the x and y coordinates and dividing them by their length. With this we can get the center of mass. After that we need to construct the matrix M which is a matrix of sum of distances. With that we can calculate the smallest eigenvector of M. The reasons we do that is because the smallest eigenvector of M is the orthogonal vector to the line on which the center of the mass line. With that we can partition the vertices around the line. The vertices with the orthogonal vector value less than 0 are assigned to one partition and the vertices with the orthogonal vector value greater than or equal to 0 are assigned to the other partition. With this we are able to partition the graph into two parts.

Hence with this we are able to implement both Spectral Bisection and Inertial Bisection. Table 1 shows the number of cut edges with different techniques for each of the graphs.

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	70	72
grid9(40)	118	127	128	118
Smallmesh	25	12	12	30
Tapir	55	23	18	49
Eppstein	42	41	42	45

1.2. Recursively bisecting meshes [20 points]

In this Question we were asked to implement a function that recursively bisects a mesh. The main advantage for this is to parallelize the computation. With recursion we are able to take advantage of it, by parallelizing the system. With that we can split the graphs into 2^n parts, where n is the level of the depth. In Table 2 and Table 3 we can see the number of cut edges for recursive bisection with 8 and 16 partitions respectively. The results are compared with Spectral Bisection, Metis 5.1.0, Coordinate Bisection and Inertial Bisection.

Table 2: Edge-cut results for recursive bi-partitioning with 8 Partitions

Case	Spectral	Metis 5.1.0	Coordinate	Inertial
mesh3e1	51	57	63	59
bodyy4	1000	985	1065	1364
de-2010	809	491	929	1084
biplane-9	411	465	548	648
L-9	718	637	631	828

Unfortunately, I was not able to visualize the "de-2010" mesh because of a bug which by the time of the deadline I was not able to fix it.

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

The implementation for `bench mestic` was very similar to the implementation of `bench recursivebisection.` The only difference was that we uses metismex to get the ParGraphRecursive and PartGrapKway

Table 3: Edge-cut results for recursive bi-partitioning with 16 Partitions

Case	Spectral	Metis 5.1.0	Coordinate	Inertial
mesh3e1	61	57	63	59
bodyy4	1675	1591	1951	2214
de-2010	1512	897	1796	2002
biplane-9	794	845	974	1093
L-9	1121	1019	1028	1377

methods to do the partition for the graphs. The results for the number of cut edges for recursive bisection and direct multiway partitioning in Metis 5.1.0 are shown in Table 4.

From the results we can see that there is not much difference in terms of the number of cut edges in the helicopter mesh where the mesh is low. However when we look at Skirt mesh we can see that the number of cut edges is a bit higher for way direct partition.

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

Again unfortunately I was not able to visualize this in 3d because of a bug which by the time of the deadline I was not able to fix it.