**15-640 Project 4 Report**

Name 1: Rui Zhang

Andrew ID 1: ruiz1

Name 2: Jing Gao

Andrew ID 2: jinggao

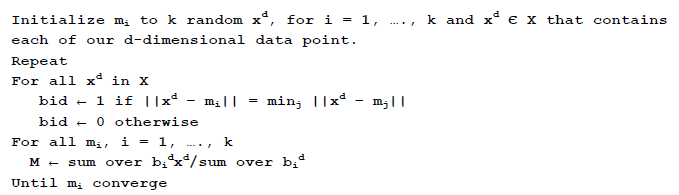
**Part I: Design and Features**

The project implements both the sequential and MPI-based K-Means algorithm.

**Introduction of Sequential version of K-Means:**

The sequential K-Means algorithm has three phase: First we need to select k random points from the dataset to be the centroids. Then for each d-dimensional data point, we compute the distance from it to every centroid and label it as the cluster with smallest distance from the point to that centroid. Then we compute k new centroids based on the labels we have computed.

The pseudocode is as follows:



**Strategy to parallelize the K-Means algorithm:**

Since the computation of distance for every data point is independent to each other, we divide the data points of datasets into **P** parts and each of which is taken care of one process. The strategy can be explained in the following phases:

1. The Master node divides the dataset according to the number of processes. We use MPI\_Scatterv to distribute the data to different processes
2. Use MPI\_BCast to broadcast the random selected centroids.
3. Based on minimum distance strategy (2D points) or most common occurrence strategy (DNA strands), each process will then determine which cluster each point it has belongs to. Meanwhile it will sum them up (2D points) or compute the occurrence of each character in every dimension (DNA strands) for each cluster.
4. When all the processes finishes the computing, we use MPI\_Gatherv method to let the Master to collect results from each process. The Master will then be responsible for using all the results to compute the new centroids.
5. If the new centroids differ from the old centroids under a pre-defined threshold, we will terminate the whole process. Otherwise we continue from step 2 until the termination condition meets.

The pseudocode is as follows (Take the 2D points as an example):

**On Master Process [1]:**

Initialize k different random centroids ci (i = 1….k) selected from the datasets.

**On Master Process [2]:**

BroadCast ci d(i = 1….k) to other P processes.

Divide the xid (i = 1….n) into P parts (the first P – 1 parts have equivalent size and the last one has the remaining part) and scatter them to corresponding process

**On each Process [3]:**

Initialize cntj = 0 for every j = 1….k

Initialize sumjd = 0 for every j=1…..k

For all xid in set X received from Master process

labeli = minj||xid – cjd||

cntj++

sumjd = sumjd + xjd

End for

Send all the cntj and sumjd to the Master Process

**On Master Process [4]:**

Collect the cntj[i] and sumjd [i] (i = 1….P which represents which process it comes from) from every process.

Initialize the new centroids NCjd (j=1….k)

Sum all the cntj[i] and sumjd[i[ to be cntj and sumjd (Use MPI\_Reduce here)

For j=1….k

Update the centroid NCjd = sumjd / cntj

End for

If sum(||NCjd - cjd||) < threshold then

Terminate all processes

Else

Set cjd as NCjd

Continue from step [2]

End if