**GROUP NUMBER:** 36

**GROUP MEMBERS:** Francesco Visonà, Alessandro Dario, Luca Pellegrini

**AVAILABLE INPUTS:** Input files are available in the hdfs file system: /data/BDC2425/artificial1M7D100K.txt and /data/BDC2425/artificial4M7D100K.txt

**PART 1:** The goal of this test is to assess the scalability of the standard and fair implementations. The test must be performed on file artificial4M7D100K.txt. However, if your implementation is slow (i.e., taking more than 10 minutes for the slowest run), you can use the smaller file artificial1M7D100K.txt. You must use the following parameters: L=16, K=100, M=10.

Fill in the following table.

**Name of used file:** /data/BDC2425/artificial4M7D100K.txt

|  |  |  |  |
| --- | --- | --- | --- |
| **SCALABILITY WITH RESPECT TO NUMBER OF EXECUTORS** | | | |
| **Number of executors** | **Spark Lloyd’s implementation** | **MRFairLloyd** | **MRComputeFairObjective** |
| 2 | 11802 | 41330 | 1905 |
| 4 | 10010 | 28395 | 1263 |
| 8 | 6822 | 16986 | 795 |
| 16 | 6821 | 7802 | 346 |

**General hints:**

* Remember that Spark uses the lazy evaluation for constructing an RDD. Therefore, be sure to include an action on the final RDD when you take running times.
* Any used RDD in your program should be cached.
* Do not include the reading of the input in your running times.

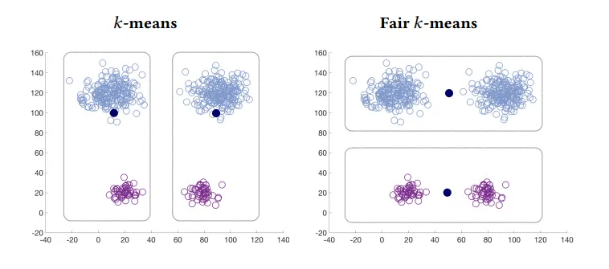
**PART 2:** Describe the program GxxGEN.java or GxxGEN.py that you have implemented for point 5 of the specifications. Include a brief high-level description of your program and the constraints, if any, on the input parameters (e.g., the minimum number of points *N*).

Figure 1. Example of difference between k-means standard algorithm and fair k-means algorithm when clustering 4 clusters with K=2

Following what was described in the paper that introduced Fair K-Means [1] and represented in *Figure 1*, the G36GEN.java program first divides the number K of clusters to generate by 2. This will be the number of times we replicate the image shown in *Figure 1.* Then, if the number K is odd, it creates another cluster that resembles half of the clusters of *Figure 1*.

More in detail, we first divide N for the number of clusters, obtaining:   
 points for every cluster (notice that the points that are left out by this division will be put in the first cluster or in the cluster of points that is left alone).

Then we create times the following balls:

* One big ball centered in [0,120] of points only of group A with radius approximately 5
* One small ball centered in [0,20] of points only of group B with radius approximately 1
* One big ball centered in [200,120] of points only of group A with radius approximately 5
* One small ball centered in [200,20] of points only of group B with radius approximately 1

These four balls will be clustered in 2 clusters. The centers will probably be the ones represented in *Figure 1* with the standard k-means that treats the 2 small balls as outliers as they don’t weight much and the fair k-means that puts the centers between the clusters.

Then the last cluster, if needed (if K is odd), will consist of:

Then we create times the following balls:

* One big ball centered in [0,120] of points only of group A with radius approximately 5
* One small ball centered in [0,20] of points only of group B with radius approximately 1

The idea is the same as above: the standard k-means will treat the small ball as outliers.

Requirements: N >= 2K.

**REFERENCES**

Mehrdad Ghadiri, Samira Samadi, and Santosh Vempala. 2021. Socially Fair k-Means Clustering. <https://doi.org/10.1145/3442188.3445906>