Optimizing Parallelization Strategies for the Big-means Clustering Algorithm

Ravil Mussabayev[1,2,3], Rustam Mussabayev[3]

[1] Department of Mathematics, University of Washington, Seattle, WA, USA
[2] Huawei Russian Research Institute, Moscow, Russia
[3] Laboratory for Analysis and Modeling of Information Processes, Institute of Information and Computational Technologies, Almaty, Kazakhstan

November 13, 2023

Motivation

- Rapid growth of data necessitates efficient yet effective clustering techniques;
- **Minimum Sum-of-Squares Clustering (MSSC)** is a critical and widely used model for many applications (e.g., image analysis, customer segmentation, etc.);
- Global minimizers accurately reflect the clustering structure. However, the pursuit of global minimizers is complicated by a high non-convexity of the MSSC objective function.

Problem formulation

Consider a set $X = \{x_1, \dots, x_m\}$, where $x_i \in \mathbb{R}^n,\ i = 1, \dots, m$.

Minimum sum-of-squares clustering (MSSC) solves the problem of finding k cluster centers (centroids) $C = (c_1, \dots, c_k) \in \mathbb{R}^{n \times k}$ that minimize the sum of squared Euclidean distances from each data point x_i to its nearest cluster center c_j:

$$\min_{C} \quad f\left(C, X\right) = \sum_{i=1}^{m} \min_{j=1,\dots,k} \|x_i - c_j\|^2 \qquad (1)$$

For general k and m, the MSSC is known to be an NP-hard problem [1].

[1] Aloise, D., Deshpande, A., Hansen, P., et al.: NP-hardness of euclidean sum-of-squares clustering. Machine Learning (2009)

Our contribution & RQs

In this work, we answer the following research questions:

RQ1: How can the state-of-the-art big data clustering algorithm – Big-means – be efficiently parallelized?

RQ2: Can we increase the accuracy of Big-means by optimizing its parallelization strategy? If yes, by how much?

RQ3: Can we turn the results of our experiments into generalizable knowledge and guidelines for practitioners using big data clustering algorithms on a parallel or distributed computing system?

Big-means algorithm

Algorithm 1: Big-means Clustering

Result: Compute the final centroids C and cluster assignments Y for a dataset X using the Big-means algorithm.

Initialization:

Initialize all k centroids C as degenerate;

$\hat{f} \leftarrow \infty$;

Set iteration counter $t = 0$;

while $t < T$ **do**

- Draw a random sample S of size s from X;
- **for** *each centroid* c *in* C **do**
  - **if** c *is the centroid associated with a degenerate cluster* **then**
    - Reinitialize c using K-means++ on S;
  - **end**
- **end**
- Compute new centroids C_{new} using K-means on S with initial centroids C;
- **if** $f(C_{new}, S) < \hat{f}$ **then**
  - $C \leftarrow C_{\text{new}}$;
  - $\hat{f} \leftarrow f(C_{\text{new}}, S)$;
- **end**
- $t \leftarrow t + 1$;

end

$Y \leftarrow$ Assign each point in X to nearest centroid in C;

Flowchart of Big-means

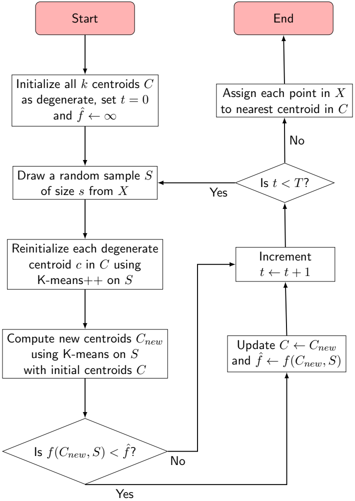

Figure 1: Flowchart of the Big-Means algorithm

Big-means — SOTA big data clustering algorithm [2]

The Big-means algorithm enjoys the following key properties:

- It avoids being trapped in suboptimal solutions and explores different parts of the solution space by:
  - restricting to random subsets of dataset during each iteration,
  - periodically re-initializing the centroids of degenerate clusters using K-means++;
- The time complexity of every iteration is $\mathcal{O}(s \cdot n \cdot k)$.

[2] Mussabayev, R., Mladenovic, N., Jarboui, B., Mussabayev, R.: How to Use K-means for Big Data Clustering? Pattern Recognition 137, 109269 (2023).

Parallelization strategies

We conduct a comparative analysis of the following parallelization schemes:

1. Inner parallelism (Big-means-inner)
2. Competitive parallelism (Big-means-competitive)
3. Collective parallelism (Big-means-collective)

Inner parallelism

Big-means-inner:

The main loop of the Big-means algorithm remains sequential (data samples are processed in a sequence), but all the internal loops of K-means and K-means++ are executed in parallel.

Competitive parallelism

Big-means-competitive:

- Each ‘worker’ processes start simultaneously on each available processor by initializing their own data samples from the big dataset;
- Each worker keeps clustering its individual stream of data samples completely independently from other workers, employing the sequential versions of K-means and K-means++;
- For every iteration, workers only use their own preceding best centroids for initialization;
- Once the stopping criterion for every worker is met, the best solution among all workers is selected.

Competitive parallelism

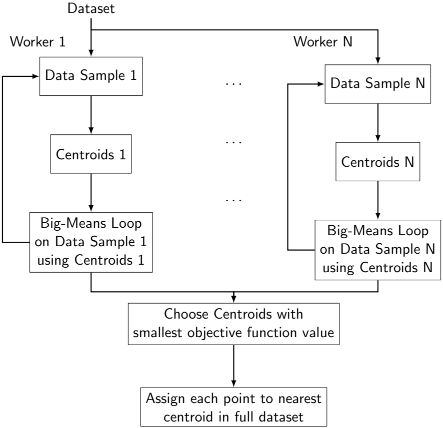

Figure 2: Flowchart of the Big-Means algorithm with the competitive parallelization

Competitive parallelism

Algorithm 2: Competitive Big-means Clustering

Result: Compute the final centroids C and cluster assignments Y for a dataset X using the competitive Big-means algorithm.

Initialization:
$C_w \leftarrow$ Mark all k centroids as degenerate for each worker w;
$\hat{f}_w \leftarrow \infty$ for each worker w;
$t_w \leftarrow 0$ for each worker w;
while $t_w < T$ *for any worker* w **do**
    for *each parallel worker* w **do**
        $S_w \leftarrow$ Random sample of size s from X;
        for *each* $c \in C_w$ **do**
            if c *is the centroid associated with a degenerate cluster* **then**
                Reinitialize c using K-means++ on S_w;
            end
        end
        $C_{\text{new},w} \leftarrow$ K-means clustering on S_w with initial centroids C_w;
        if $f(C_{\text{new},w}, S_w) < \hat{f}_w$ **then**
            $C_w \leftarrow C_{\text{new},w}$;
            $\hat{f}_w \leftarrow f(C_{\text{new},w}, S_w)$;
        end
        $t_w \leftarrow t_w + 1$;
    end
end
$C_{\text{best}} \leftarrow$ Centroids of the worker with the smallest $\hat{f}_w$ value;
$Y \leftarrow$ Assign each point in X to nearest centroid in C_{best};

Collective parallelism

Big-means-collective:

- Similar to competitive parallelism, workers begin clustering their individual data samples in parallel on each available processor;
- However, after independently initializing their first sample, each worker initializes every subsequent sample using the best set of centroids observed so far from all previous iterations across all workers;
- This parallelization mode is termed collective since the workers share information about the best solutions;
- Once the stopping criterion for every worker is met, the best solution among all workers is chosen.

Collective parallelism

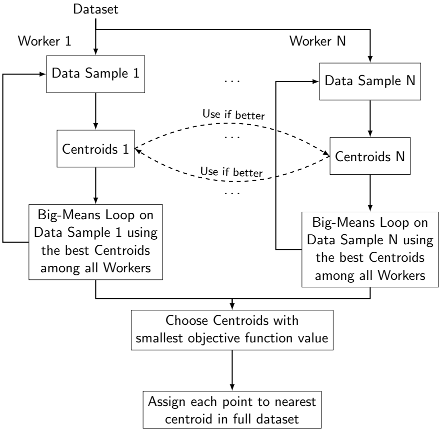

Figure 3: Flowchart of the Big-Means algorithm parallelization using a collective strategy

Collective parallelism

Algorithm 3: Collective Big-means Clustering

Result: Compute the final centroids C and cluster assignments Y for a dataset X using the collective Big-means algorithm.

Initialization:
$C_w \leftarrow$ Mark all k centroids as degenerate for each worker w;
$\hat{f}_w \leftarrow \infty$ for each worker w;
$t_w \leftarrow 0$ for each worker w;
while $t_w < T$ *for any worker* w **do**
    for *each parallel worker* w **do**
        $S_w \leftarrow$ Random sample of size s from X;
        $C_{\text{best}} \leftarrow$ Centroids of the worker with the smallest $\hat{f}_w$ value;
        for *each* $c \in C_{best}$ **do**
            if c *is the centroid associated with a degenerate cluster* **then**
                Reinitialize c using K-means++ on S_w;
            end
        end
        $C_{\text{new},w} \leftarrow$ K-means clustering on S_w with initial centroids C_{best};
        if $f(C_{new,w}, S_w) < \hat{f}_w$ **then**
            $C_w \leftarrow C_{\text{new},w}$;
            $\hat{f}_w \leftarrow f(C_{\text{new},w}, S_w)$;
        end
        $t_w \leftarrow t_w + 1$;
    end
end
$C_{\text{best}} \leftarrow$ Centroids of the worker with the smallest $\hat{f}_w$ value;
$Y \leftarrow$ Assign each point in X to nearest centroid in C_{best};

Experiments: hardware & software

Hardware & software:

- Ubuntu 22.04 64-bit;
- AMD EPYC 7663 56-Core Processor;
- 1.46 TB of RAM;
- Python 3.10.11 along with NumPy 1.24.3 and Numba 0.57.0.

Experiments: datasets

Datasets:

- 23 real-world publicly available datasets;
- The number of attributes ranges from 2 up to 5,000;
- The number of instances varied from thousands (smallest 7,797) to tens of millions (largest 10,500,000).
- Each of the 23 datasets was clustered using each algorithm n_{exec} times into clusters of sizes: 2, 3, 5, 10, 15, 20, 25. Total number of conducted individual clustering processes for the main experiment reached 18,415.

Experiments: datasets

Table 1: Brief description of the datasets

Datasets	No. instances m	No. attributes n	Size $m \times n$	File size
CORD-19 Embeddings	599616	768	460505088	8.84 GB
HEPMASS	10500000	28	294000000	7.5 GB
US Census Data 1990	2458285	68	167163380	361 MB
Gisette	13500	5000	67500000	152.5 MB
Music Analysis	106574	518	55205332	951 MB
Protein Homology	145751	74	10785574	69.6 MB
MiniBooNE Particle Identification	130064	50	6503200	91.2 MB
MFCCs for Speech Emotion Recognition	85134	58	4937772	95.2 MB
ISOLET	7797	617	4810749	40.5 MB
Sensorless Drive Diagnosis	58509	48	2808432	25.6 MB
Online News Popularity	39644	58	2299352	24.3 MB
Gas Sensor Array Drift	13910	128	1780480	23.54 MB
3D Road Network	434874	3	1304622	20.7 MB
KEGG Metabolic Relation Network (Directed)	53413	20	1068260	7.34 MB
Skin Segmentation	245057	3	735171	3.4 MB
Shuttle Control	58000	9	522000	1.55 MB
EEG Eye State	14980	14	209720	1.7 MB
Pla85900	85900	2	171800	1.79 MB
D15112	15112	2	30224	247 kB

Metrics:

- The result of each experiment was analyzed for error gap ε, spent CPU time t, and baseline time $\bar{t}$. For each algorithm A, dataset choice X and number of clusters k, error gap ε is defined as:
$$\varepsilon(\%) = \frac{100 \times (f - f^*)}{f^*}$$
where $f^* = f^*(X, k)$ is the best value of the objective function observed on the whole dataset X using k clusters from the available past experiments and history records;

Experiments: metrics

- For the Big-means algorithm, CPU time t is considered to be the time of the last change of the incumbent solution C. The CPU time t is measured in seconds;
- For the parallel multi-worker Big-means versions, t is defined to be the time of the last change of the incumbent solution C_w for the worker w that achieved the best value of the objective function $f(C_w, S_w)$ on the sample S_w;

Experiments: metrics

- For a fixed pair (X, k), baseline time $\bar{t}$ of algorithm A_i is defined to be the time of achieving the baseline sample objective value $\overline{f}_s$:

$$\overline{f}_s = \max_{j \in \{1,2,3,4\}} med\left(f_s^*(A_j)\right) \tag{2}$$

where $f_s^*(A_j)$ is the best value of the objective function obtained on a sample of size s using algorithm A_j. The median $med(\cdot)$ is calculated across n_{exec} executions of algorithm A_j for the fixed pair (X, k).

Experiments: parameters

Parameters:

- The maximum CPU time of Big-means was capped at t_{max} seconds;
- The clustering process on each sample was stopped when the number of iterations exceeded 300, or the relative tolerance between two consecutive objective function values was less than 10^{-4};
- For K-means$++$, three candidate points were considered when generating the next centroid, choosing only the best one;

Experiments: parameters

- The rule of thumb was used to determine the sample size s of Big-means in our experiments. We adjusted s until neither increasing nor decreasing it improved the objective function values;
- For each pair (X, k), the choice of parameters t_{max} and n_{exec} precisely matched the values specified in the original Big-means paper [3].

---

[3] Mussabayev, R., Mladenovic, N., Jarboui, B., Mussabayev, R.: How to Use K-means for Big Data Clustering? Pattern Recognition 137, 109269 (2023).

Preliminary experiments

We have conducted two preliminary experiments:

1. The number of employed CPUs (workers) was varied in the range 2, 4, 8, 12, 16, and the resulting values of the error gap ε and CPU time t were measured 3 times for every pair (X, k).

   We established that having **8 CPUs** would be the optimal value for the subsequent experiments;

2. Three parallelized versions of Big-means, along with the fully sequential version, were run over all datasets according to the methodology described above. Then, the baseline sample objective values $\overline{f}_s$ were computed. These values served as baselines in the main experiment.

Experimental results

Table 2: Relative clustering accuracies ϵ (%) for different algorithms. The highest accuracies for each experiment (algorithm, data pair (X, k)) are displayed in bold. Success is indicated when an algorithm's performance matches the best result among all algorithms for the current experiment.

Dataset	Big-means-sequential				Big-means-inner				Big-means-competitive				Big-means-collective			
	#Succ	MinGap	MedianGap	MaxGap	#Succ	MinGap	MedianGap	MaxGap	#Succ	MinGap	MedianGap	MaxGap	#Succ	MinGap	MedianGap	MaxGap
CORD-19 Embeddings	5/49	0.01	0.32	2.7	9/49	0.0	0.14	4.03	9/49	0.01	**0.05**	**0.43**	11/49	0.01	0.09	0.49
HEPMASS	5/49	0.0	0.33	1.7	4/49	0.0	0.27	1.27	19/49	**-0.06**	0.12	0.34	10/49	0.0	0.17	1.25
US Census Data 1990	22/140	0.01	3.75	163.36	15/140	0.03	4.05	163.52	35/140	0.05	**1.94**	**5.62**	24/140	0.03	2.74	8.41
Gisette	14/105	-1.72	**0.01**	0.43	18/105	-1.65	**0.01**	0.55	15/105	-1.68	**0.01**	0.28	25/105	**-1.76**	**0.01**	**0.19**
Music Analysis	22/140	0.02	1.16	26.17	40/140	**0.01**	1.02	9.24	31/140	0.04	**0.62**	**3.29**	26/140	0.03	0.87	4.01
Protein Homology	24/105	0.03	0.96	18.83	20/105	**0.01**	**0.64**	18.53	31/105	0.07	0.84	3.12	15/105	0.06	1.01	2.89
MiniBooNE Particle Identification	14/105	-0.49	0.18	116.8	22/105	-0.45	0.06	21.68	17/105	-0.51	**0.01**	4031563.99	31/105	-0.48	**0.01**	**1.0**
MiniBooNE Particle Identification (normalized)	29/140	**0.0**	0.8	7.12	32/140	**0.0**	0.66	7.06	27/140	0.01	0.68	3.64	32/140	**0.0**	**0.52**	**3.16**
MFCCs for Speech Emotion Recognition	21/140	**0.01**	1.21	5.62	30/140	**0.01**	0.94	5.03	24/140	0.02	0.13	2.29	33/140	0.02	**0.11**	**2.14**
ISOLET	10/105	-0.1	0.82	2.59	14/105	-0.01	0.59	3.39	29/105	**-0.15**	0.23	**1.44**	25/105	-0.1	0.32	1.89
Sensorless Drive Diagnosis	42/280	-2.41	1.45	100.19	40/280	**-2.42**	1.12	100.2	63/280	-2.41	0.03	21.95	75/280	**-2.42**	**-0.0**	**8.31**
Sensorless Drive Diagnosis (normalized)	39/280	0.02	3.59	14.45	52/280	**0.01**	3.47	13.83	77/280	0.02	**1.9**	7.74	58/280	0.02	2.46	**7.24**
Online News Popularity	36/140	0.0	2.96	29.08	31/140	**-0.15**	2.57	31.05	25/140	0.01	**0.87**	**9.16**	27/140	0.0	0.97	20.25
Gas Sensor Array Drift	31/210	-0.08	2.83	33.08	31/210	-0.79	3.42	33.04	56/210	**-0.85**	0.15	8.46	34/210	-0.77	0.4	12.48
3D Road Network	53/280	**0.0**	0.24	5.87	67/280	**0.0**	**0.19**	5.49	65/280	**0.0**	**0.19**	2.85	49/280	**0.0**	0.21	**2.67**
Skin Segmentation	29/210	-1.18	4.83	18.67	33/210	**-1.22**	3.95	22.34	40/210	-1.15	**0.2**	**9.12**	53/210	-1.06	0.23	12.36
KEGG Metabolic Relation Network (Directed)	24/140	-0.97	2.55	124.91	19/140	-1.09	2.61	124.83	24/140	-1.27	0.04	157.01	34/140	**-1.29**	0.03	**17.47**
Shuttle Control	18/120	-1.87	5.03	78.35	19/120	-2.81	4.99	91.65	23/120	**-3.62**	1.41	**24.3**	27/120	-3.11	**0.48**	152.42
Shuttle Control (normalized)	28/160	0.03	2.09	31.05	24/160	**0.02**	1.97	31.98	43/160	0.07	1.5	**9.26**	27/160	0.07	1.55	16.75
EEG Eye State	34/160	-0.01	0.57	29.91	39/160	-0.01	0.24	29.91	34/160	**-0.1**	0.02	29.91	23/160	-0.01	0.02	**4.25**
EEG Eye State (normalized)	42/240	-0.33	0.42	65.96	34/240	-0.34	0.43	185.07	56/240	-0.33	**0.0**	65.96	41/240	-0.34	**0.0**	**0.75**
Pla85900	45/280	**0.0**	0.36	2.85	34/280	**0.0**	0.42	2.8	71/280	**0.0**	**0.12**	**1.46**	55/280	**0.0**	0.24	2.01
D15112	19/105	**0.0**	0.71	4.66	13/105	**0.0**	0.16	16.71	30/105	**0.0**	**0.1**	1.78	23/105	**0.0**	0.13	2.12
Overall Results	606/3683	-0.39	1.82	38.45	640/3683	-0.47	1.66	40.14	844/3683	**-0.52**	**0.69**	175301.45	758/3683	-0.48	0.72	**12.37**

Experimental results

Table 3: Resulting clustering times $\bar{t}$ (sec.) with respect to baseline sample objective values $\overline{f}_s$. The lowest clustering times for each experiment (algorithm, data pair (X, k)) are displayed in bold.

Dataset	Big-means-sequential			Big-means-inner			Big-means-competitive			Big-means-collective		
	MinGap	MedianGap	MaxGap	MinGap	MedianGap	MaxGap	MinGap	MedianGap	MaxGap	MinGap	MedianGap	MaxGap
CORD-19 Embeddings	**0.76**	30.21	41.4	1.8	**20.0**	**38.83**	5.06	28.28	46.18	5.52	30.49	42.26
HEPMASS	**1.13**	18.31	30.44	1.7	21.48	30.29	2.54	**16.6**	30.92	2.22	20.23	**29.61**
US Census Data 1990	0.14	1.9	3.13	**0.1**	1.82	**3.07**	0.17	**1.74**	3.12	0.19	1.88	3.16
Gisette	2.69	16.65	41.01	**1.15**	**4.98**	**9.26**	3.4	21.76	65.31	3.39	20.34	61.89
Music Analysis	**0.22**	4.76	9.49	0.29	**4.42**	**8.51**	0.63	6.18	9.79	0.36	5.89	10.43
Protein Homology	0.17	2.72	5.92	**0.11**	**2.49**	**3.54**	0.83	3.35	9.71	0.43	3.37	11.35
MiniBooNE Particle Identification	**0.16**	3.11	10.56	0.29	**2.14**	**3.06**	0.46	5.32	22.96	0.48	5.08	20.04
MiniBooNE Particle Identification (normalized)	0.03	0.72	1.58	**0.02**	**0.58**	**1.04**	0.13	0.85	1.99	0.21	0.88	1.72
MFCCs for Speech Emotion Recognition	**0.08**	0.65	1.29	**0.08**	**0.52**	**1.03**	0.19	0.92	1.88	0.15	0.86	1.85
ISOLET	0.31	3.09	5.06	**0.15**	**2.2**	**4.98**	0.5	4.05	5.56	0.95	3.63	5.72
Sensorless Drive Diagnosis	0.06	0.9	4.4	**0.03**	**0.59**	**1.09**	0.06	1.72	7.01	0.13	1.78	6.69
Sensorless Drive Diagnosis (normalized)	**0.01**	**0.19**	0.35	**0.01**	**0.19**	**0.32**	**0.01**	0.32	0.72	**0.01**	0.31	0.67
Online News Popularity	**0.02**	0.49	0.84	**0.02**	**0.33**	**0.74**	0.18	0.65	1.48	0.14	0.59	1.45
Gas Sensor Array Drift	0.05	1.22	2.06	**0.03**	**1.15**	**2.02**	0.29	1.75	2.44	0.26	1.73	2.89
3D Road Network	0.03	**0.36**	1.6	**0.02**	0.39	**0.71**	**0.02**	0.76	3.08	0.06	0.79	2.94
Skin Segmentation	**0.01**	**0.12**	**0.21**	**0.01**	**0.12**	**0.21**	0.03	0.21	0.5	**0.01**	0.18	0.43
KEGG Metabolic Relation Network (Directed)	0.09	0.64	1.4	**0.05**	**0.54**	**1.02**	0.21	0.96	2.28	0.33	0.93	2.16
Shuttle Control	0.07	**0.74**	1.54	**0.01**	0.87	**1.5**	0.34	1.09	1.7	0.2	1.12	1.96
Shuttle Control (normalized)	**0.0**	**0.22**	**0.4**	0.01	0.23	**0.4**	0.02	0.34	0.41	0.02	0.31	**0.4**
EEG Eye State	**0.02**	0.86	**1.5**	**0.02**	**0.78**	**1.5**	0.22	0.94	**1.5**	0.15	1.02	**1.5**
EEG Eye State (normalized)	**0.0**	**0.54**	1.02	0.01	0.58	**1.0**	0.01	0.71	1.04	0.1	0.75	1.01
Pla85900	**0.02**	0.87	**1.5**	**0.02**	**0.8**	1.51	0.03	0.99	1.51	**0.02**	0.89	**1.5**
D15112	0.03	**0.63**	1.5	0.05	0.83	**1.49**	**0.01**	0.96	**1.49**	0.14	0.95	1.5
Overall Results	0.27	3.92	7.31	**0.26**	**2.77**	**5.09**	0.67	4.45	9.68	0.67	4.63	9.27

Discussion of results

We have obtained the following results:

- Big-means-sequential strategy performed consistently worse than the Big-means-inner parallelization strategy for all datasets, as measured by both time and accuracy;
- The Big-means-inner strategy consistently demonstrated faster convergence to baselines compared to other strategies across most datasets;

Discussion of results

- While employing Big-means-competitive and Big-means-collective strategies resulted in improved final solutions, the coordination of multiple processors and the complexities introduced by the Numba library led to increased convergence times. On average, using 8 CPUs, the convergence times were up to twice as long compared to the Big-means-inner version;
- The Big-means-competitive scheme demonstrates slightly better clustering quality compared to the Big-means-collective scheme. This improvement can be attributed to multiple initializations at the beginning.

Conclusive thoughts

- Big-means-inner achieved accelerated processing times compared to the sequential worker in other parallel Big-means strategies. These finding highlights the substantial impact of dataset characteristics on the efficiency-accuracy trade-off, reinforcing the importance of balancing sample size and quality of clusters;

- If the convergence time is not a critical factor, both the Big-means-competitive and Big-means-collective strategies exhibit considerably improved global clustering quality compared to other versions of Big-means. On average, using 8 CPUs, the resulting quality is up to three times better;

Conclusive thoughts

- There exist two ways to clustering of samples by multiple workers: either spending a significant amount of time on local search with a single initialization or conducting multiple different initializations. Our experiments suggest that the latter approach seems to be more advantageous than the former;
- At some point, Big-means-collective transitions to processing the results of a single initialization, although it is not guaranteed to be the best choice (it may only be good at the beginning). On the other hand, the alternative approach continues to cluster a multitude of different K-means++ initializations, from which the best one is selected at the end.

Recommendation for practitioners

This study reveals that there is no one-size-fits-all parallelization strategy for the Big-means algorithm.

Instead, the optimal strategy appears to be data-dependent, suggesting the need for adaptive techniques that can select the most suitable strategy based on the characteristics of the dataset.

Nevertheless, in the majority of cases, we advise practitioners to utilize the competitive parallelization strategy of the Big-means algorithm.

Future research ideas

For future work, we plan to:

- investigate adaptive techniques that can dynamically select the optimal parallelization strategy based on a dataset at hand;
- delve deeper into the trade-offs observed in this study to gain a better understanding of their impacts on algorithmic performance and accuracy.

Thank you!