Fast k-means with stable instance sets

Basso Madjoukeng, Ariel; Kenmogne, Edith Belise; Frénay, Benoît

Published in:
2024 International Joint Conference on Neural Networks, IJCNN 2024 - Proceedings

DOI:
10.1109/IJCNN60899.2024.10650923

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

Fast k-means with Stable Instance Sets

Conference Paper · June 2024

DOI: 10.1109/IJCNN60899.2024.10650923

3 authors, including:

Ariel Basso Madjoukeng
University of Namur
4 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Kenmogne Edith Belise
University of Dschang
15 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Fast k-means with Stable Instance Sets

Ariel Basso Madjoukeng
Fac. of Computer Science
University of Dschang
Dschang, Cameroon
arielbassodev@gmail.com

Edith Belise Kenmogne
Fac. of Computer Science
University of Dschang
Dschang, Cameroon
ebkenmogne@gmail.com

Benoît Frenay
Fac. of Computer Science
University of Namur
Namur, Belgium
benoit.frenay@unamur.be

***Abstract*—The k-means algorithm is used to group objects according to similarity or distance criteria. Due to its simplicity and operating principle, this algorithm has become one of the most widely used for clustering problems. However, despite its popularity, it has several limitations. One of these issues is the excessive convergence time on multidimensional datasets. To address this limitation, several fast variants of the k-means algorithm have emerged. Some optimize the process of updating centroids, while others optimize the process of assigning points to clusters and finally other methods propose to subdivide a dataset into batches and apply the algorithm to various batches to accelerate the speed of convergence of this algorithm. Despite these advancements, existing approaches are not always very optimal, and many of them optimize the algorithm while losing the efficiency of the k-means algorithm. This work address those challenges and proposes a new fast variant of the k-means algorithm capable of significantly optimizing the convergence time of the k-means algorithm while maintaining the efficiency of the naive k-means algorithm. This paper proposes a multi-level optimization of this algorithm. It proposes an efficient heuristic for determining stable and variant points from an iteration. Additionally, it optimizes the process of updating the centroid calculation process, which until now has been done using all the points in the cluster. Finally, it optimizes the assignment process using the notion of neighborhood clusters.**

***Index Terms*—Clustering , k-means, stable points , variant points.**

I. Introduction

Today, many clustering algorithms exist, among which the k-means algorithm holds a prominent position. Introduced for the first time by MacQueen [1], its objective is to group n observations into k clusters while minimizing intra-cluster inertia. The simplicity of this algorithm and its use in prototyping systems have contributed to its popularity, Leading to the development of variants that can be leveraged on real-world usecase. For a dataset of size n, a number of clusters k, and a maximum number of iterations t, the time complexity of the k-means algorithm is on the order of $\mathcal{O}(nkt)$ [2]. However, due to the continuous growth of modern datasets and its operational principles, this algorithm can sometimes require a considerable amount of time to converge. Faced with this challenge, fast variants of the k-means algorithm have emerged. These variants are specifically designed to minimize the algorithm's execution time while maintaining a high level of precision. This evolution addresses the growing need for efficiency in processing large volumes of data, providing solutions that are more tailored to the contemporary requirements of data analysis. Among these variants, some subdivide the dataset into multiple subsets and apply the algorithm to each of them, while others are based on intelligent initialization techniques capable of selecting points likely to make the algorithm converge more quickly, and finally, others rely on heuristics. The algorithm presented in this article is based on a heuristic that aims to improve the temporal efficiency of the k-means algorithm, Regardless the scale of the scenario, whether it is large or small.

The proposed algorithm optimizes the process of assigning points to clusters by using the concept of stable and variant points. It also proposes an efficient heuristic to determine stable and variant points at a specifically chosen iteration without having to recalculate the set of all stable points, nor the set of all variant points, at each further iteration. Furthermore, for clustering operations with a large number of clusters k, the proposed algorithm enhances the assignment of each points to its corresponding centroid. It does so by using a heuristic to compare each point only to neighboring clusters to optimize this crucial phase of clustering. Finally, the algorithm also refines the concept of cluster updating by relying on the concepts of stable and variable points. This contributes to accelerate the update process, ensuring better convergence of the algorithm in various situations.

This paper is organized as follows. Section II presents a brief review of the literature on the k-means clustering and its fast variants. Section III proposes a new fast k-means algorithm. Section IV shows experimental results and Section VI concludes this work with perspectives.

II. Related Work

This section offers an overview of the k-means algorithm, including initialization methods and accelerated variants.

A. The k-means algorithm

Let $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ a dataset of n observations $\mathbf{x}_i \in \Re^d$. The objective of the k-means algorithm is to group the dataset $\mathcal{D}$ into k distinct clusters, where each cluster is represented by a centroid. The algorithm aims to minimize intra-cluster inertia. Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ be a set of k clusters. The intra-cluster inertia W for these clusters is defined as the sum of

squared distances between each point $\mathbf{x}$ in a cluster $\mathcal{C}_j$ and the centroid $\mathbf{c}_j$ of that cluster, for all clusters:

$$W(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k) = \sum_{j=0}^{k} \sum_{\mathbf{x} \in \mathcal{C}_j} \|\mathbf{x} - \mathbf{c}_j\|^2. \quad (1)$$

In k-means (see Algorithm 1), initialization plays a crucial role, as inappropriate choices for the initial centers can result in significantly extended convergence times. To enhance the efficiency of this algorithm, several variants of initialization algorithms have been developed to improve this critical stage.

Algorithm 1 k-means Algorithm

1: **Input:** dataset $\mathcal{D}$, number of clusters k
2: **Output:** final cluster assignments
3: **Procedure:**
4: Select k points as initial cluster centroids.
5: **repeat**
6: Assign each data point to the cluster with the nearest centroid.
7: Update the centroids by taking the average of points in each cluster.
8: Check for convergence by comparing the new centroids with the old centroids.
9: **until** Convergence

B. Initialization methods

Initialization is a crucial step in k-means, involving the selection of starting points for the algorithm. In this research area, several approaches have been proposed. Among these, the Random Initialization [1] method involves randomly selecting k points from the dataset as initial centers. While this method has the advantage of simplicity, its major drawback lies in the fact that, with the chosen points, the algorithm may require a significant amount of time to achieve convergence.

To improve the convergence time of k-means, the k-means++ [3] algorithm was proposed by Arthur and Vassilvitskii, and operates in two phases. First, it randomly selects an initial cluster centroid from the available data points. Subsequently, it calculates the distance between each data point and its nearest cluster center. Data points are weighted based on their proximity to the nearest cluster center, meaning that points farther away have a higher probability to be chosen as cluster centers. This process is repeated until all cluster centers are initialized. See Algorithm 2 for a detailed presentation.

k-means++ reduces the risk of converging to a local minimum by intelligently selecting the initial centroids. Furthermore, it significantly improves the initialization process by selecting initial centroids in a more thoughtful and balanced manner. This smart initialization reduces the algorithm's reliance on random starting points, typically resulting in quicker convergence and more stable clustering solutions.

C. Fast k-means variants

In the optimization of the execution time of the k-means algorithm, various initiatives have been undertaken.

Algorithm 2 k-means++ initialization

1: **Input:** dataset $\mathcal{D}$, number of clusters k
2: **Output:** Initial centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$
3: **Procedure:**
4: Choose the first centroid $\mathbf{c}_1$ uniformly at random from the data points.
5: **for** $j = 2$ to k **do**
6: Compute the distance $d(\mathbf{x})$ from each data point $\mathbf{x}$ to the nearest existing centroid.
7: Select the next centroid $\mathbf{c}_j$ from the data points with probability proportional to $d(\mathbf{x}^2)$.
8: **end for**

The mini-batch k-means algorithm was developed by Sculley [4]. In contrast to the naive k-means algorithm, the mini-batch k-means version utilizes smaller datasets. At each iteration, a mini-batch is selected, and update operations are exclusively performed on its elements. This enhances the execution speed of the naive k-means algorithm. Several enhanced variants of this algorithm have emerged.

The Nested mini-batch k-means algorithm by Newling and Fleuret [5], is an enhancement of Sculley mini-batch k-means algorithm that utilizes the Elkan distance bounding approach [6]. This algorithm improves mini-batch k-means by limiting batch size while maintaining balanced contribution to centroids. Peng *et al.* introduce a variation capable of identifying neighboring clusters of a given cluster [7]. They significantly reduce the number of operations for reassigning a point from one cluster to another, using only the neighboring clusters for updates. The main limitation is that, at each iteration, all points from each cluster are used in the update process, even though they are only used in relation to their neighboring clusters.

Xia *et al.* propose an accelerated version of the k-means algorithm [8] capable of optimizing the assignment of points to clusters. Their approach works as follows: At each iteration and for each cluster, they draw a line connecting the centroid of the cluster with those of the other clusters. Once this step is completed, they construct several circles centered on the centroid of each cluster. To reduce the number of points during the assignment, they assume that points not located in an intersection of circles are unlikely to move.

Despite all the approaches in the literature, there is still room for improvement. First, approaches that optimize the process of calculating centroids at each iteration do not take into account all other aspects, such as optimizing the process of assigning an observation from one cluster to another, as well as the process of reducing the number of points to be assigned at each iteration. Approaches seeking to develop fast variants using only a subset of points seek stable sets at each iteration, which can be inefficient overall.

Subsequently, this paper proposes an approach to accelerate the k-means algorithm while maintaining a high level of precision. It relies on the concepts of stable points and variant points within a cluster and presents a heuristic able to detect these points from an iteration. Additionally, the proposed approach

includes a metric for optimizing the update of centroids at each iteration. Finally, the assignment of variant points is also optimized thanks to a concept of neighborhood.

III. Fast k-means with Stable Instance Sets

The k-means algorithm uses all points at each iteration to assign points to clusters. This can lead to slow convergence for large values of n, as the calculations apply to all n observations at each iteration. This limitation makes k-means particularly inefficient when many points remain in the same cluster from one iteration to another. Figure 1 shows the number of points moving between clusters over successive iterations of the k-means algorithm. The information about the various datasets is in Table I. The value of k for each curve was selected using the elbow method [9], [10]. The graphical representation in Figure 1 illustrates the dynamics of point movement for six datasets of different sizes. In general, the number of iterations to reach convergence varies between 4 and 20 consecutive iterations. In addition, beyond a certain iteration, very few points move from one cluster to another for each dataset. Therefore, based on this observation, we define a *stable point* within a cluster $\mathcal{C}$ as a point likely to remain within its cluster $\mathcal{C}$ from one iteration to another. *A variant point*, on the other hand, is a point likely to move from cluster $\mathcal{C}$ to another during one iteration. By identifying stable and variant points from a certain iteration onwards, our k-means algorithm shall significantly reduce execution time.

A. Methodology

To enhance k-means, this work proposes to identify stable and variant points through a heuristic. The efficiency of this heuristic lies in the fact that stable points are not (re)determined at each iteration, as in some methods in the literature. This significantly optimizes convergence time by further reducing the computational load. Since variant points are likely to move from one cluster to another, our algorithm reduces the computational load related to the movements of related points by limiting their movements to neighboring clusters. Finally, to optimize the number of operations needed to update the centroids of each cluster, our algorithm relies on stable points to reduce the computational load at each iteration.

B. Fast cluster assignment

Given the significant impact of stable points on the k-means algorithm, it becomes imperative to establish a method for their detection. The objective of k-means is to minimize intra-cluster inertia, which causes points close to the centroids to contribute less to the intra-cluster inertia. As a consequence, the closer a point is to its centroid during an iteration, the less likely it is to move from one cluster to another. Our approach to determining stable and variant points is therefore based on their proximity to the centroids: the $n_{\text{stable}}(j)$ nearest neighbors of centroid $\mathbf{c}_j$ are chosen as stable points $\mathcal{S}_j$ for cluster $\mathcal{C}_j$, whereas the remaining point in cluster $\mathcal{C}_j$ form the set of variants points $\mathcal{C}_j$ (see Algorithm 3).

In the classical k-means algorithm, point stability can occur very quickly, even in the initial iterations. Figure 1 shows how the rate of movement of points (from one cluster to another) evolves over iterations. In most datasets, starting from a certain iteration m (typically, $m = 3$), the percentage of points whose cluster membership changes is minor (2% to 10%). Based on Figure 1, we conclude that a set of stable points can be determined quite early. In this article, the determination of stable and variant points occurs only once at a given iteration m, and is no longer repeated at each iteration. This greatly enhances the convergence time of the algorithm.

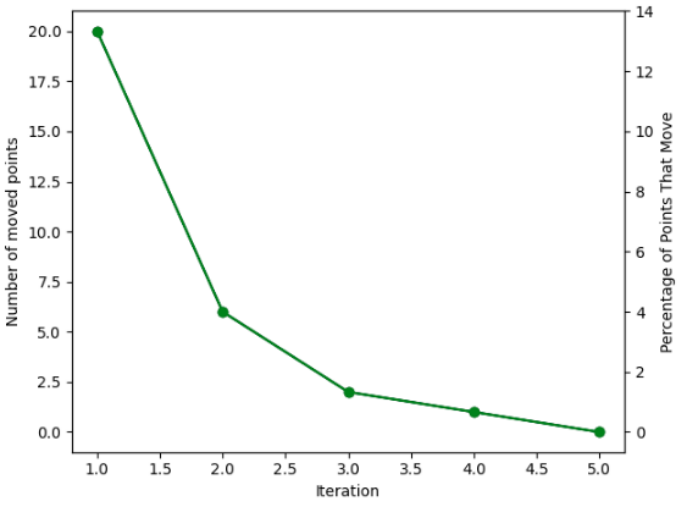
Algorithm 3 Variants and Stable Points

Require: cluster $\mathcal{C}_j$ containing n_j elements, $n_{\text{stable}}(j)$ the number of neighbor points.
Ensure: $\mathcal{S}$: the set of stable point and $\mathcal{V}$ is the set of variant points.
1: **Begin**
2: c_j the centroid of cluster $\mathcal{C}_j$
3: $\mathcal{S}_j \leftarrow$ the set of P nearest neighbors to c_j
4: $\mathcal{V}_j \leftarrow N \setminus \mathcal{S}_j$
5: **Return** $\mathcal{S}_j$ and $\mathcal{V}_j$
6: **End**

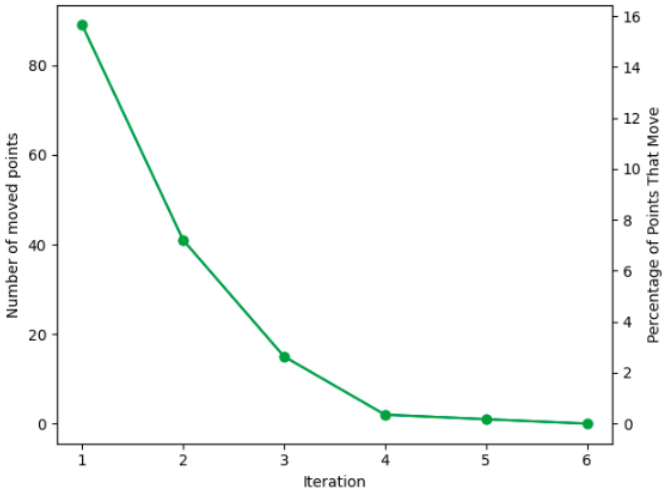
The proposed algorithm is highly dependent on the number $n_{\text{stable}}(j)$ of nearest neighbors of centroid $\mathbf{c}_j$ that are chosen as stable points $\mathcal{S}_j$ for cluster $\mathcal{C}_j$, as well as the number of iterations m to execute before stable points can be identified. The larger $n_{\text{stable}}(j)$ and the smaller m, the faster the algorithm will be. However, clustering performance will also decrease, so a compromise needs to be chosen. In the following, we propose an approach to determine them efficiently.

C. When and how large should the stable sets be built?

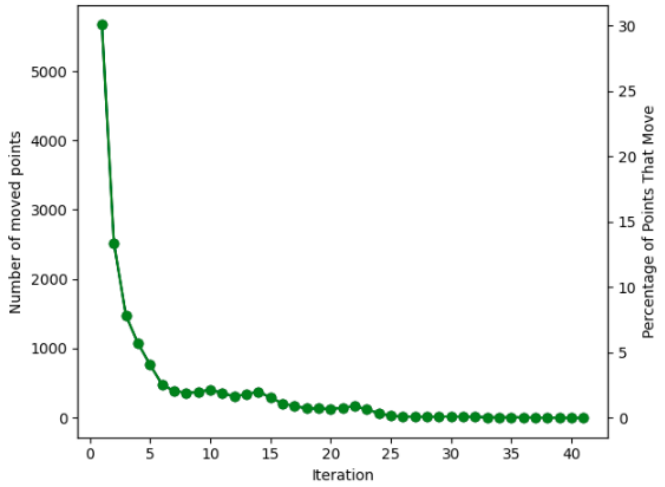
To determine the size of each set of stable points $\mathcal{S}_j$ and at which iteration they should be built, we conducted an empirical study using various datasets. Figure 2 graphically represents the evolution of intra-cluster inertia over time for three datasets from the literature (Boston, Breast Cancer and KDD Cup 1999). Each point on the graph corresponds to an execution of our algorithm with a given proportion of the points in clusters chosen as stable points (80%, 90%, 95% and 99% of each cluster). Each line corresponds to stable sets being chosen at a given iteration m (2^{nd}, 3^{rd} or 4^{th} iteration). When the stable point percentage is set to 99%, the algorithm converges quickly, but the intra-cluster inertia does not reach its optimum. On the other hand, with a threshold of 80% stable points, intra-cluster inertia is optimal, but the execution time increases. However, by opting for a proportion of stable point of 90% in each cluster, the best compromise between execution time and intra-cluster inertia is observed. Also, Figures 2 confirm that $m = 3$ is a good choice, as already shown by Figure 1. Thus, we retain the choice of a stable point percentage of 90% at the third iteration. Experiments in Section IV will further assess the validity of this heuristic.



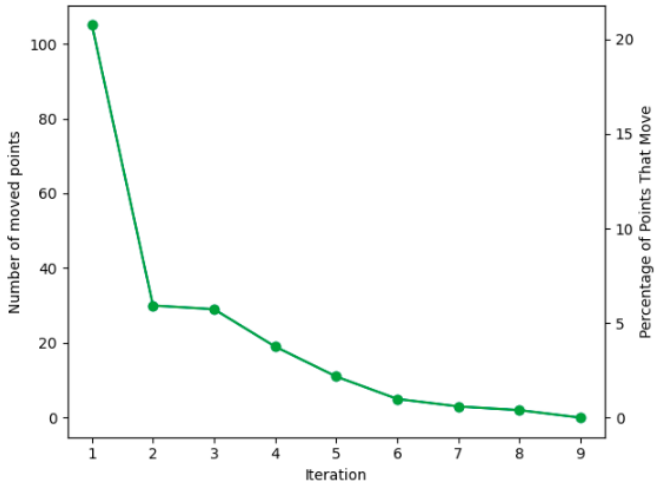
(a) Iris ($n = 150$)



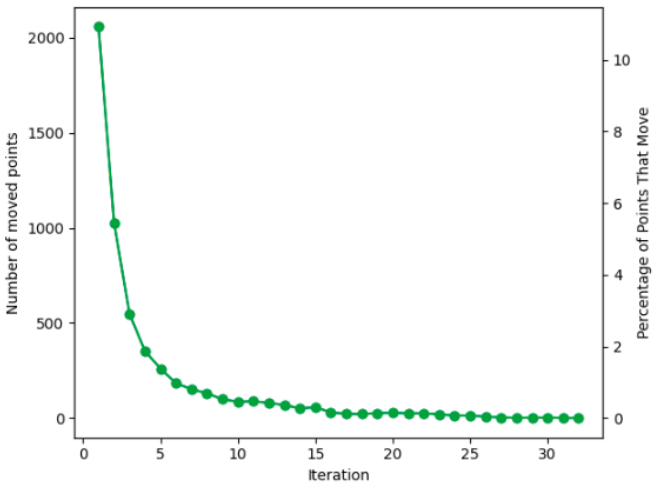
(b) Breast Cancer ($n = 569$)



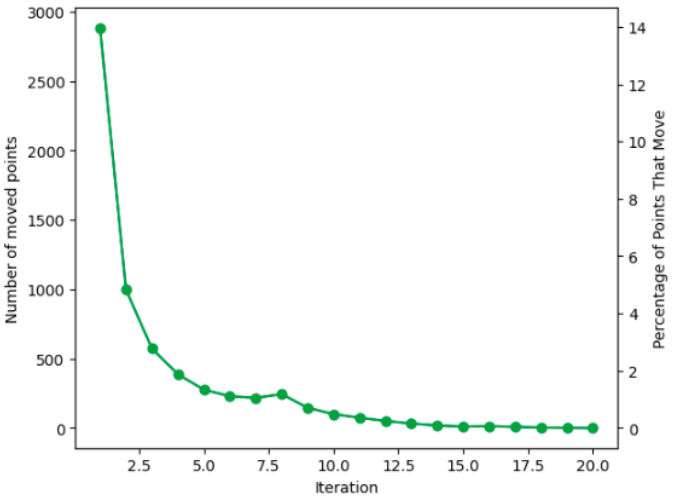
(c) Newsgroups ($n = 18{,}846$)



(d) Boston ($n = 506$)

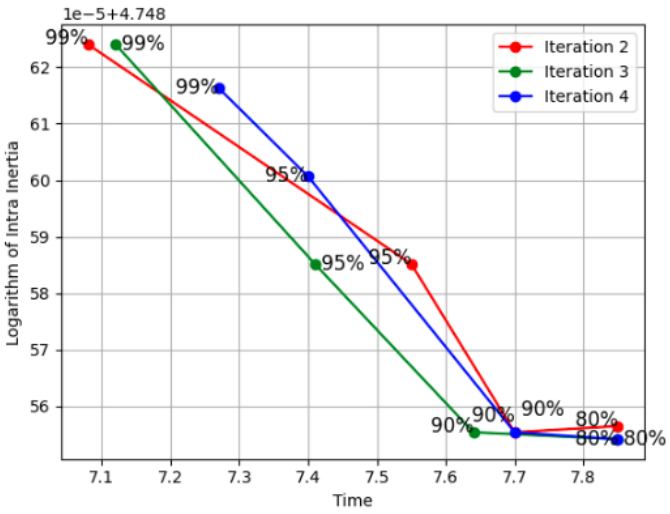


(e) KDD Cup 1999 ($n = 4{,}898{,}431$)

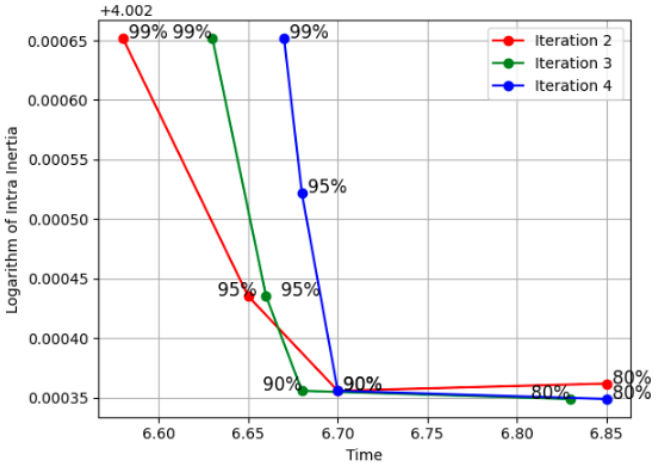


(f) California Housing ($n = 20{,}640$)

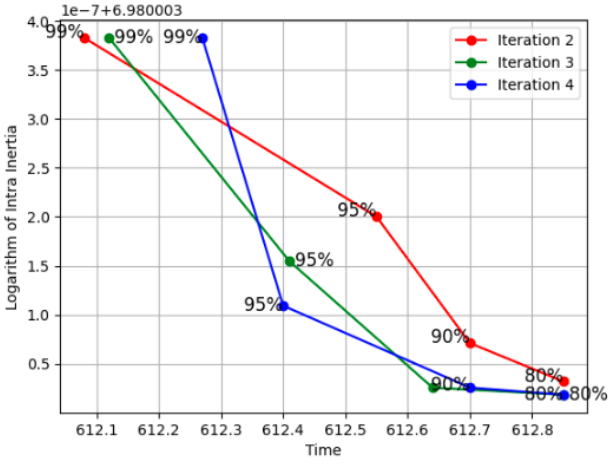
Fig. 1: Number of points that moved from one cluster to another at each iteration with standard k-means.



(a) Boston dataset



(b) Breast Cancer dataset



(c) KDD CUP 1999 dataset

Fig. 2: Relationship between the intra-cluster inertia and the execution time for different stable point percentages and choices of when (i.e., at which iteration) to choose them.

D. Further reducing the cost of cluster assignment

Whereas stable points remain after iteration m in the same cluster for all subsequent iterations, variants points are free to move from one cluster to another. In standard k-means, this would require to consider all possible clusters for each variant point. To avoid this, This article adopts the approach proposed by Peng *et al.* [7]. Their method aims to determine the clusters closest to a given cluster $\mathcal{C}_j$ and to use only these neighboring clusters during the assignment phase for points in cluster $\mathcal{C}_j$. In practice, to find all the neighboring clusters, two k-means iterations are performed . If, during an iteration, an observation $\mathbf{x}$ moves from cluster $\mathcal{C}_1$ to cluster $\mathcal{C}_2$, then cluster $\mathcal{C}_1$ is considered a neighbor of cluster $\mathcal{C}_2$. This enables the determination of the neighbors of cluster $\mathcal{C}_j$ as the clusters that exchanged at least one instance with cluster $\mathcal{C}_j$ in the two iterations. Only the neighbors of cluster $\mathcal{C}_j$ are considered to update the assignment of variant points in cluster $\mathcal{C}_j$, reducing the complexity of the whole assignment step when k is large.

E. Fast centroid update algorithm

In the k-means algorithm, updating centroids at each iteration is a crucial step. Depending on the cluster size, this step can take a considerable amount of time. To optimize this centroid update process, this section proposes an intelligent optimization of centroids that take advantage of the identification of stable and variant points to avoid unnecessary operations.

In a cluster $\mathcal{C}$, stable points are not likely to move to another cluster. If their assignment remain unchanged, their contribution to the centroid update remain the same at each k-means iteration and does not need to be recalculated each time. In the k-means algorithm and its variants, the expression for calculating the centroid of a cluster $\mathcal{C}_j$ is given by

$$\mathbf{c}_j = \frac{1}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} \mathbf{x}_i. \tag{2}$$

To optimize this process, we divide this equation into two parts that depends on stable and variants points, respectively:

$$\mathbf{c}_j = \mathbf{c}_j^{\mathcal{S}} + \frac{1}{|\mathcal{V}_j|} \sum_{i \in \mathcal{V}_j} \mathbf{x}_i, \tag{3}$$

where the contribution of the set of stable points $\mathcal{S}_j$ is computed only once at iteration $m = 3$ and then keep fixed:

$$\mathbf{c}_j^{\mathcal{S}} = \frac{1}{|\mathcal{S}_j|} \sum_{i \in \mathcal{S}_j} \mathbf{x}_i. \tag{4}$$

Since $|\mathcal{S}_j| \gg |\mathcal{V}_j|$ as $\mathcal{S}_j$ contains 90% of cluster j, Equation 4 allows us to significantly reduce the cost of evaluating Equation 3, hence expediting the centroid update step.

F. Final algorithm

Algorithm 4 incorporates all the improvements proposed in this section. The initialization of the algorithm is based on k-means++ due to its efficient selection of centroids, as it is standard. The following section evaluates and validates the effectiveness of Algorithm 4, demonstrating that it improves execution times without sacrificing intra-cluster inertia.

Algorithm 4 k-means algorithm with stable and variant points

Inputs: cluster data $\mathcal{C}$
Outputs: optimized centroids
Begin
Use k-means++ as initialization method
for $i \leftarrow 1$ to 3 **do**
    Execute one iteration of the k-means algorithm on cluster data $\mathcal{D}$
end for
for each cluster $\mathcal{C}_j$ **do**
    $n_{\text{stable}}(j) \leftarrow 0.9\times$ size of cluster $\mathcal{C}_j$
    $\mathcal{S}, \mathcal{V} \leftarrow$ DetermineStableAndVariantPoints($\mathcal{C}_j$, $n_{\text{stable}}(j)$)
end for
repeat
    Apply one iteration of the k-means algorithm only on elements of $\mathcal{V}$
    Apply the solution by Peng *et al.* to optimize the update process of variant points
    Efficient centroid update using only variant points
until Convergence
End

IV. Experiments

This section undertakes the evaluation and comparison of the proposed k-means algorithm on real datasets. The implementation of this algorithm was carried out in the Python language, on a computer running the Windows 10 64-bit operating system, equipped with a Core i5 processor and 8 GB of RAM. It first assesses and compares the proposed k-means algorithm based on the time required to achieve convergence across various datasets referenced (see Table I for details). Subsequently, it evaluates this algorithm by measuring the intra-cluster inertia, aiming to appreciate the quality of clustering obtained at each iteration.

TABLE I: Details of the datasets used in Section III and IV.

Dataset	Number of Observations	Dimensionality
Newsgroups Text	18,846	1
Poker Hand	1025010	11
LFW People	13,233	5,828
Forest Cover Types	581,012	54
3D Road Network	434,874	4
KDD Cup 1999	4,898,431	41
Iris Dataset	150	4
Breast Cancer	569	40
Boston	506	13
California Housing	20,640	8

A. Evaluation of the algorithm in terms of convergence speed

To assess the proposed algorithm in terms of convergence time, it is executed on various datasets. Algorithms from the literature have also been implemented for comparison. Table II presents results obtained with the optimal value of k determined through the elbow method. For each algorithm, the data has been normalized before applying the respective algorithm. Our algorithm is systematically faster than other alternatives, showing the efficiency of the harnessed heuristics.

TABLE II: Execution times of k-means variants.

Dataset	Algorithm	Execution Time (s)
poker-hand-training-true	k-means	1,200
	Xia *et al.*	1,000
	Peng *et al.*	900
	Our algorithm	**400**
Newsgroups Text	k-means	28,800
	Xia *et al.*	11,200
	Peng *et al.*	14,400
	Our algorithm	**9,800**
KDD Cup	k-means	6,200
	Xia *et al.*	1,900
	Peng *et al.*	1,300
	Our algorithm	**612**
LFW People	k-means	2,000
	Xia *et al.*	1,650
	Peng *et al.*	1,090
	Our algorithm	**720**
Forest covertypes	k-means	900
	Xia *et al.*	500
	Peng *et al.*	800
	Our algorithm	**300**
3D Road Network	k-means	33,569
	Xia *et al.*	1,100
	Peng *et al.*	1,390
	Our algorithm	**849**
Iris	k-means	10
	Xia *et al.*	8
	Peng *et al.*	10
	Our algorithm	**5**
Breast Cancer	k-means	17
	Xia *et al.*	15
	Peng *et al.*	10
	Our algorithm	**6**
Boston	k-means	19
	Xia *et al.*	11
	Peng *et al.*	9
	Our algorithm	**7**
California housing	k-means	890
	Xia *et al.*	670
	Peng *et al.*	660
	Our algorithm	**320**

B. Evaluation of our algorithm in terms of clustering quality

The proposed algorithm and its competitors are also evaluated on two main aspects: the quality of the clustering in terms of intra-cluster inertia and the conformity of the prototypes with respect to those of the naive k-means algorithm.

Table III shows the different intra-cluster inertia values obtained by the different algorithms on various datasets. We note that among the algorithms in the literature, our algorithm is the closest to the k-means algorithm in terms of intra-cluster inertia. Our algorithm represents a better compromise between efficiency and speed. To conclude, we will examine the prototypes obtained by the traditional k-means algorithm and our algorithm to assess their concordance.

Figures 3 ,4 , 5, 6 depict the prototypes generated by two distinct approaches: the traditional k-means algorithm and

TABLE III: Intra-cluster inertia of k-means variants.

Dataset	Algorithm	Intra-cluster inertia
poker-hand-training-true	k-means	2.03×10^5
	Xia *et al.*	2.32×10^5
	Peng *et al.*	2.12×10^5
	Our algorithm	2.03×10^5
Newsgroups Text	k-means	1.8×10^4
	Xia *et al.*	1.98×10^4
	Peng *et al.*	1.86×10^4
	Our algorithm	1.8×10^4
KDD Cup	k-means	9.31×10^{13}
	Xia *et al.*	9.88×10^{13}
	Peng *et al.*	9.52×10^{13}
	Our algorithm	9.33×10^{13}
LFW People	k-means	3.02×10^8
	Xia *et al.*	3.02×10^8
	Peng *et al.*	3.02×10^8
	Our algorithm	3.02×10^8
Forest covertypes	k-means	2.69×10^8
	Xia *et al.*	2.88×10^8
	Peng *et al.*	2.70×10^8
	Our algorithm	2.69×10^8
3D Road Network	k-means	3.36×10^2
	Xia *et al.*	3.35×10^2
	Peng *et al.*	3.35×10^2
	Our algorithm	3.35×10^2
Iris	k-means	155
	Xia *et al.*	155
	Peng *et al.*	155
	Our algorithm	**155**
Boston	k-means	$5,72 \times 10^5$
	Xia *et al.*	$5,76 \times 10^5$
	Peng *et al.*	$5,72 \times 10^5$
	Our algorithm	$5,72 \times 10^5$
Breast Cancer	k-means	$7,81 \times 10^7$
	Xia *et al.*	$7,81 \times 10^7$
	Peng *et al.*	$7,81 \times 10^7$
	Our algorithm	$7,81 \times 10^7$
California house	k-means	$1,1 \times 10^5$
	Xia *et al.*	$1,5 \times 10^5$
	Peng *et al.*	$1,3 \times 10^5$
	Our algorithm	$1,1 \times 10^5$

our variant, applied to the Iris and California dataset. We randomly selected two axes on which we visualized the data to observe the concordance of prototypes. We observe that the visualization generated by our algorithm appears to yield results that are substantially equivalent to those provided by the k-means algorithm.

V. ACKNOWLEDGMENTS

This work was supported by the ARIAC by Digital Wallonia4.AI project number 2010235 of the SPW Recherche / Wallonia Research. The authors thank Jerome Fink, Valentin Delchevalerie, Pierre Poitier and Clementin Tayou Djamegni for their comments and the fruitful discussions on this paper. Their feedback enhances the clarity of the arguments, ultimately strengthening the quality of the paper.

VI. CONCLUSION

This work present a new variant of the k-means algorithm incorporating three levels of optimization. We introduced the concept of stable and variant points, demonstrating that considering these points could lead to faster convergence of the k-means algorithm while maintaining a high level of precision. Our study also explored methods to optimize variant points. Our algorithm is tested on various dataset, comparing it to benchmark algorithms from the literature. At each iteration, we measured the execution time, and our algorithm consistently demonstrated superior performance. Additionally, we assessed the algorithm in terms of computing intra-cluster inertia, finding that our method exhibited intra-cluster inertia equivalent to that of the k-means algorithm. Finally, we conducted a comparative study between the prototypes generated by our algorithm and those produced by the k-means algorithm. We also observed that the prototypes from our algorithm matched those generated by the k-means algorithm. Through these experiments, we observe that our algorithm represents a better compromise between efficiency and execution time. Anticipating the future importance of variant points, we plan to propose new approaches to determine stable points.

REFERENCES

[1] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 2, pp. 281–297, 1967.

[2] S. Xia, D. Peng, D. Meng, C. Zhang, G. Wang, E. Giem, W. Wei, and Z. Chen, "Some methods for classification and analysis of multivariate observations," *IEEE transactions on pattern analysis and machine intelligence*, no. 99, pp. 1–1, 2020.

[3] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, p. 1027–1035, 2007.

[4] D. Sculley, "Web-scale k-means clustering," *Proceedings of the 19th international conference on World wide web*, p. 1177–1178, 2010.

[5] J. Newling and F. Fleuret, "Nested mini-batch k-means," *NIPS*, p. 1360–1368, 2016.

[6] C. Elkan, "Using the triangle inequality to accelerate k-means," *International Conference on Machine Learning*, p. 147–153, 2003.

[7] D. Peng, Z. Chen, and J. Fu, "Fast k-means clustering based on the neighbor information," *International Symposium on Electrical, Electronics and Information Engineering*, p. 551–555, 2021.

[8] S. Xia, D. Peng, D. Meng, C. Zhang, G. Wang, E. Giem, and W. Wei, "A fast adaptive k-means with no bounds," *International Symposium on Electrical, Electronics and Information Engineering*, pp. 1–1, 2020.

[9] R. L. Thorndike, "Who belongs in the family?" *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[10] D. J. Ketchen, Jr and C. L. Shook, "The application of cluster analysis in strategic management research: An analysis and critique," *Strategic Management Journal*, vol. 17, no. 6, pp. 441–458, 1996.

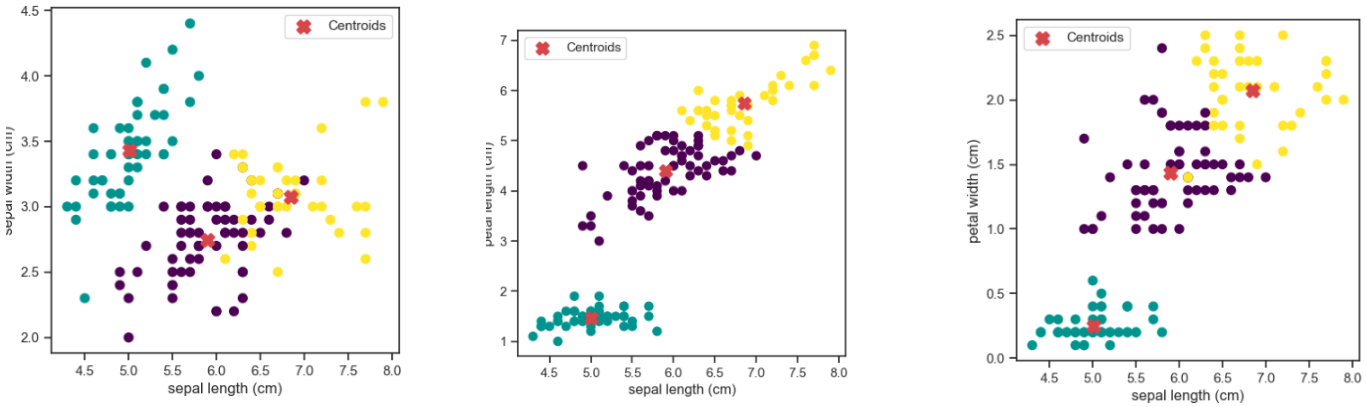


Fig. 3: Visualization of prototypes with standard k-means on the Iris dataset.

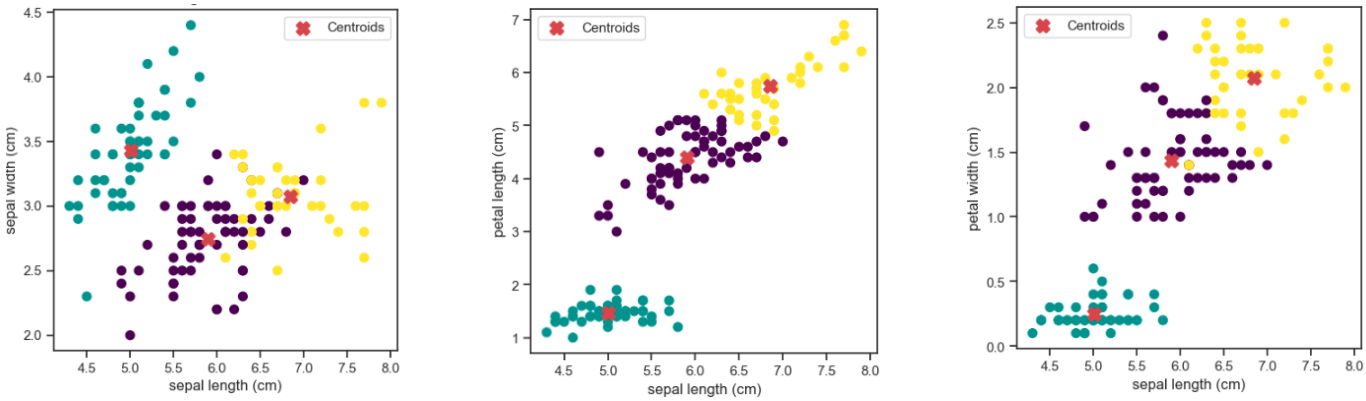


Fig. 4: Visualization of prototypes with our fast k-means algorithm on the Iris dataset.

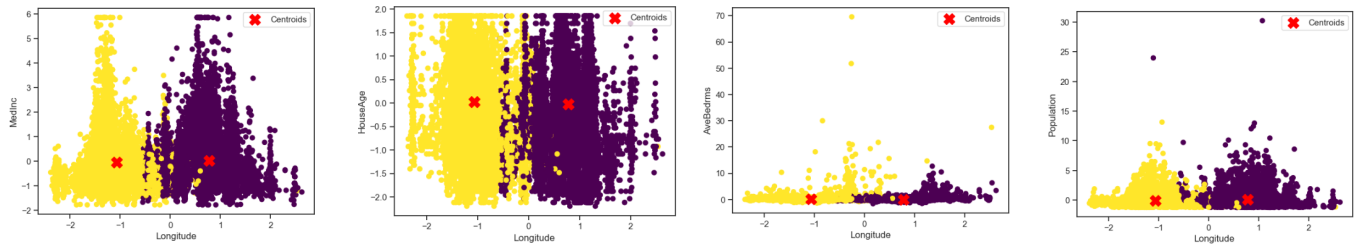


Fig. 5: Visualization of prototypes with standard k-means algorithm on the California dataset.

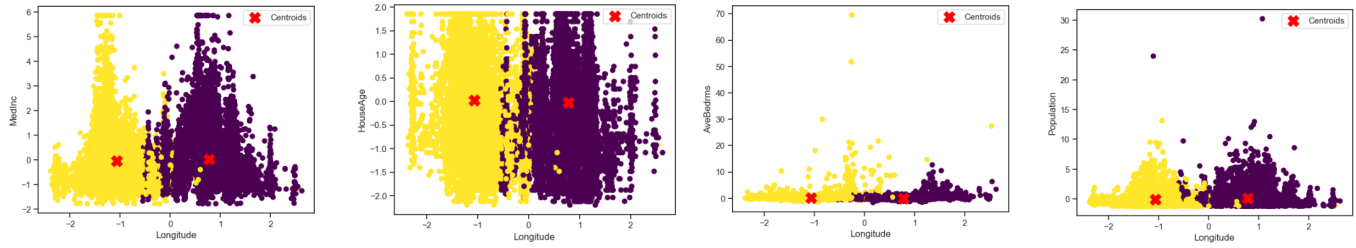


Fig. 6: Visualization of prototypes with our fast k-means algorithm on the California dataset.