axioms

MDPI

Article

Hybrid Fuzzy C-Means Clustering Algorithm Oriented to Big Data Realms

Joaquín Pérez-Ortega [1,*], Sandra Silvia Roblero-Aguilar [1,2], Nelva Nely Almanza-Ortega [2], Juan Frausto Solís [3], Crispín Zavala-Díaz [4], Yasmín Hernández [1] and Vanesa Landero-Nájera [5]

1 Tecnológico Nacional de México/Cenidet, Cuernavaca 62490, Mexico; sandra.ra@tlalnepantla.tecnm.mx (S.S.R.-A.); yasmin.hp@cenidet.tecnm.mx (Y.H.)
2 Tecnológico Nacional de México/IT Tlalnepantla, Tlalnepantla 54070, Mexico; nelva.ao@tlalnepantla.tecnm.mx
3 Tecnológico Nacional de México/IT Cd. Madero, Madero 89440, Mexico; juan.frausto@gmail.com
4 Faculty of Accounting, Administration and Informatic, Universidad Autónoma del Estado de Morelos, Cuernavaca 62209, Mexico; crispin_zavala@uaem.mx
5 Computer Systems, Universidad Politécnica de Apodaca, Apodaca 66600, Mexico; vlandero@upapnl.edu.mx
* Correspondence: jpo_cenidet@yahoo.com.mx

Abstract: A hybrid variant of the Fuzzy C-Means and K-Means algorithms is proposed to solve large datasets such as those presented in Big Data. The Fuzzy C-Means algorithm is sensitive to the initial values of the membership matrix. Therefore, a special configuration of the matrix can accelerate the convergence of the algorithm. In this sense, a new approach is proposed, which we call Hybrid OK-Means Fuzzy C-Means (HOFCM), and it optimizes the values of the membership matrix parameter. This approach consists of three steps: (a) generate a set of n solutions of an x dataset, applying a variant of the K-Means algorithm; (b) select the best solution as the basis for generating the optimized membership matrix; (c) resolve the x dataset with Fuzzy C-Means. The experimental results with four real datasets and one synthetic dataset show that HOFCM reduces the time by up to 93.94% compared to the average time of the standard Fuzzy C-Means. It is highlighted that the quality of the solution was reduced by 2.51% in the worst case.

Keywords: Fuzzy C-means; K-means; hybrid clustering algorithm; time complexity

MSC: 62H30; 90C70; 68W40; 91C20

Citation: Pérez-Ortega, J.; Roblero-Aguilar, S.S.; Almanza-Ortega, N.N.; Frausto Solís, J.; Zavala-Díaz, C.; Hernández, Y.; Landero-Nájera, V. Hybrid Fuzzy C-Means Clustering Algorithm Oriented to Big Data Realms. *Axioms* **2022**, *11*, 377. https://doi.org/10.3390/axioms11080377

Academic Editors: Oscar Castillo and Hsien-Chung Wu

Received: 24 June 2022
Accepted: 27 July 2022
Published: 31 July 2022

1. Introduction

The clustering of objects, according to their attributes, has been studied extensively in various areas, such as bioinformatics, pattern recognition, business, and image processing, among others [1,2]. Moreover, it has gained relevance due to an exponential increase in data in different areas of knowledge. In general, a complete dataset could provide a basis for their analysis, reasoning, and decision-making [3,4].

Clustering aims to partition a set of n objects in subsets, called clusters, in such a way that the objects in any cluster have similar attributes and, at the same time, are different from the objects in any other cluster. The above description corresponds to a hard or conventional clustering type, often related to the K-Means algorithm [5]. However, in many domains, each object needs to belong to two or more groups with different degrees of membership [6–8]. The algorithm that allows for the above is associated with Fuzzy C-Means (FCM). Nevertheless, the complexity of FCM is greater than that of K-Means. In [9], the time complexity of the FCM algorithm is $O(ndc^2t)$, where n is the number of objects, d is the number of dimensions, c is the number of clusters, and t is the number of iterations. In this paper, indistinct dimensions or attributes are named to represent the characteristics of the objects.

Clustering is one of the NP-hard problems most studied in the scientific community [10]. There are no efficient algorithms to solve large datasets in polynomial time. Such issues are considered computationally *intractable*. Hence, improving the FCM clustering algorithm is a relevant and open research problem.

The standard FCM algorithm is sensitive to the random initial values of the membership matrix, which represent the values that express the degrees of membership of each object to each cluster. Therefore, a particular configuration of the membership matrix can accelerate the convergence of the algorithm. In the specialized literature, there have been publications trying to improve the FCM algorithm to create an initial configuration. However, we found no research that optimizes this matrix. In this paper, we aim to reduce the time complexity of the algorithm by optimizing the membership matrix. To achieve our goal, we propose an algorithm called Hybrid OK-Means Fuzzy C-Means (HOFCM).

This paper is organized as follows: Section 2 presents related work. Section 3 describes the algorithms used in this research. Section 4 shows the improvement proposal. Section 5 reports the results obtained. Section 6 presents the discussion. Conclusions and ideas for future research are given in Section 7.

2. Related Work

In [11], it is inferred that the FCM algorithm contains the following phases: initialization, calculation of the centroid, classification, and convergence. The initialization phase requires the definition of the following parameters: the number of clusters, the value of the weighting exponent, the random membership matrix, the convergence value (ε), and the maximum number of iterations. In [12–14], the initialization phase is different from that described. Instead of generating the membership matrix, these papers considered the initialization phase, calculating the initial centroids using the K-Means algorithm or some variant.

In [12], an algorithm called FCM++ was proposed. To generate the initial centroids, a variant of the K-Means++ algorithm [15] was implemented. Subsequently, it continues with the standard FCM algorithm. However, FCM++ has some limitations: the first concerns the IRIS and WINE datasets, where the number of iterations is higher than the standard FCM for $c = 8$ and $c = 10$, respectively; the second concerns the choice of parameter p, which is added to K-Means++ since its efficiency depends on the user's experience. A description of the K-Means++ algorithm is provided in Section 3.2.

In [13], the K-Means algorithm was implemented to generate the final centroids, which, through a transformation process called One-Hot encoding, generates the membership matrix required by the FCM algorithm. This research proposes selecting attributes through entropy weighting and Box–Cox transformation, so out of 2170 original dimensions, only 10 dimensions were used. A dataset called CSI 800, which stores the history of daily stock prices, was used to perform the experimental evaluation. A 15% decrease in execution time was observed in the results obtained.

In [14], an algorithm called NFCM was proposed. This algorithm implements the strategy of the K-Means++ algorithm [15], with the difference being that NFCM considers the information provided by the membership matrix to select the following centroids. Two datasets, namely, SPAM and IRIS, were used to evaluate its performance against the FCM++ algorithm. In IRIS, it was observed that the NFCM algorithm has a specific advantage in the number of iterations and CPU time when $c > 4$. With the SPAM dataset, when $c = 8$ and $c = 9$, the FCM++ algorithm had greater competitiveness.

There have been improvements to other phases of the standard FCM algorithm. However, this research focuses on the initialization phase. Readers interested in improvements to the FCM algorithm in the calculation of the centroid and classification phases may refer to [16–18]. For improvements that refer to the fuzzy factor, refer to [19,20].

Among the improvements reported to date that have had the greatest impact on the scientific community, it is observed that the initialization phase of the FCM algorithm is the least studied. It is essential to mention that the improvement proposals in references [12–14]

are the closest to the improvement proposal in this research. However, analytically, it has not been visualized whether the K-Means solution is on the path of convergence with FCM.

3. Materials and Methods

This section extensively describes the K-Means, K++, O-K-Means, and FCM algorithms.

3.1. K-Means Algorithm

The K-Means clustering algorithm is one of the most relevant, widely studied, and used algorithms [21]. Its popularity is mainly due to the ease of interpreting its results. This algorithm is an iterative method that consists of partitioning a set of n objects into $k \geq 2$ clusters; thus, the objects in one cluster are similar to each other and different from those in other clusters [22]. The formulation of the K-Means problem is described below:

Let $X = \{x_1, \ldots, x_n\}$ be the set of n objects to be partitioned by a criterion of similarity, where $x_i \in \mathrm{R}^d$ for $i = 1, \ldots, n$, and $d \geq 1$ is the number of dimensions. Furthermore, let $k \geq 2$ be an integer and $K = \{1, \ldots, k\}$. For a k-partition $P = \{G(1), \ldots, G(k)\}$ of X, denote v_j the centroid of cluster $G(j)$, for $j \in K$, and let $V = \{v_1, \ldots, v_k\}$ and $W = \{w_{11}, \ldots, w_{ij}\}$.

In Equation (1), the clustering problem is shown as an optimization problem [23]:

$$P: \text{ minimize } z(W, V) = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} D(x_i, v_j) \tag{1}$$

$$\text{Subject to } \sum_{j=1}^{k} w_{ij} = 1, \text{for } i = 1, \ldots, n,$$

$$w_{ij} = 0 \text{ or } 1, \text{for } i = 1, \ldots, n \text{ y } j = 1, \ldots, k,$$

where $w_{ij} = 1 \Leftrightarrow$ the object x_i belongs to cluster $G(j)$, and $D(x_i, v_j)$ denotes the Euclidean distance between x_i and v_j for i =1, . . . , n and j = 1, . . . , k. The pseudocode of the standard K-Means algorithm is shown in Algorithm 1 [24].

Algorithm 1: Standard K-Means

Initialization:
  X: = $\{x_1, \ldots, x_n\}$;
  V: = $\{v_1, \ldots, v_k\}$;
Classification:
  For $x_i \in X$ and $v_k \in V$ {
    Calculate the Euclidean distance from each x_i to the k centroids;
    Assign the x_i object to the nearest v_k centroid;}
Calculate centroids:
  Calculate the centroid v_k;
Convergence:
  If V: = $\{v_1, \ldots, v_k\}$ does not change in two consecutive iterations:
    Stop the algorithm;
  Otherwise:
    Go to Classification
End of algorithm

3.2. K++ Algorithm

The K++ algorithm, proposed in [15], initializes the clustering centroids of the K-Means algorithm by selecting objects from the dataset that are the farthest from each other in a probabilistic manner. This method accelerates the convergence speed, theoretically guaranteeing $O(\log k)$ and therefore being competitive with the optimal solution. The pseudocode of the K++ algorithm is shown in Algorithm 2.

Algorithm 2: K++

Initialization:
X: = $\{x_1, \ldots, x_n\}$;
Assign the value for k;
V: = Ø;
Select the first randomly uniform k_1 centroid V: = V U $\{v_1\}$;
For i = 2 to k:
Select the i-th centroid v_i of X with probability $D(x_i, v_j)/\sum_{x \epsilon X} D(x_i, v_j)$;
V: = V U $\{v_i\}$;
End of for
Return V
End of algorithm

In Algorithm 2, the initialization of the centroids with the K++ algorithm is shown. Given dataset X and the value of k, the next step is to select the first centroid k_1 in a uniform random manner from dataset X. Subsequently, assign it to the set of centroids V. For the second centroid, and until completing the total of k, select the *i-th* centroid with a probability distribution. In this paper, the algorithm is referred to as K++ when it is only about initializing the centroids.

3.3. O-K-Means Algorithm

There have been different improvements to the K-Means algorithm. In [22], O-K-Means is proposed, which accelerates the convergence process, stopping the algorithm when the total number of objects that change the cluster in an iteration is less than a threshold. This value expresses a relationship between the computational effort and the quality of the solution. The pseudocode of the O-K-Means algorithm is shown in Algorithm 3.

Algorithm 3: O-K-Means

Initialization:
X: = $\{x_1, \ldots, x_n\}$;
V: = $\{v_1, \ldots, v_k\}$;
εok: = Threshold value for determining O-K-Means convergence;
Classification:
For x_i ϵ X and v_k ϵ V\{
Calculate the Euclidean distance from each x_i to the k centroids;
Assign the x_i object to the nearest v_k centroid;
Compute γ \};
Calculate centroids:
Calculate the centroid v_k;
Convergence:
If ($\gamma \leq \varepsilon ok$):
Stop the algorithm;
Otherwise:
Go to **Classification**
End of algorithm

The indicator γ represents the percentage of objects that change the cluster in an iteration t, which is calculated as $\gamma_t = 100(o_t/n)$, where o_t is the number of objects that change the cluster.

3.4. FCM Algorithm

The fuzzy set theory proposed by Zadeh [25] in 1965 gave an idea of membership uncertainty described by a membership function. Cluster analysis theory was proposed by Bellman, Kalaba, and Zadeh [26], and Ruspini [27] coined the concept of fuzzy partitioning—more specifically, the fuzzy clustering algorithm. These documents set the tone for research on fuzzy clustering. In 1973, Dunn [28] extended the meaning

of hard grouping to preliminary concepts of fuzzy means. Finally, in 1981, Bezdek [11] generalized Dunn's approach to obtaining an infinite family of Fuzzy C-Means algorithms. The basic idea is that $X = \{x_1, \ldots, x_n\}$ is the set of n objects to be partitioned by a similarity criterion, where $x_i \in \mathrm{R}^d$ for i = 1, ..., n, and c is the number of clusters where $2 \leq c < n$.

The fuzzy clustering problem was formulated as an optimization problem [27], minimizing a function, as shown in Equation (2):

$$P: \text{ minimize } J_m(U, V) = \sum_{i=1}^{n} \sum_{j=1}^{c} (\mu_{ij})^m D(x_i, v_j) \tag{2}$$

where $U = \mu_{ij}$ is the membership matrix of each object i to each cluster j; $V = \{v_1, \ldots, v_c\}$ is the set of centroids, where v_j is the centroid of cluster j; and m is the weighting exponent or fuzzy factor that indicates how much the clusters overlap, $m > 1$ y $D(x_i, v_j)$, indicating the Euclidean distance between the object x_i and the centroid v_j for $i = 1, \ldots, n$ and $j = 1, \ldots, c$.

Minimizing J_m, an estimated model of U and V is obtained as

$$u_{ij} = \frac{1}{\sum_{i=1}^{c} \left(\frac{D(x_i, v_j)}{D(x_i, v_k)} \right)^{2/(m-1)}} \quad 1 \leq j \leq n; \quad 1 \leq j \leq c \tag{3}$$

$$v_j = \frac{\sum_{i=1}^{n} (u_{ij})^m x_i}{\sum_{i=1}^{n} (u_{ij})^m} \quad 1 \leq j \leq c \tag{4}$$

where x_i and v_j are vectors that belong to a space R^d and are represented as

$$x_i = (x_1, x_2, \ldots, x_d), \quad 1 \leq i \leq n \tag{5}$$

$$v_j = (v_1, v_2, \ldots, v_d), \quad 1 \leq j \leq c \tag{6}$$

The constraints of diffuse clustering are formalized in Equations (7)–(9):

$$u_{ij} \in [0, 1], \quad 1 \leq j \leq c, \quad 1 \leq i \leq n \tag{7}$$

$$\sum_{j=1}^{c} u_{ij} = 1, \quad 1 \leq i \leq n \tag{8}$$

$$0 < \sum_{i=1}^{n} u_{ij} < n, \quad 1 \leq j \leq c \tag{9}$$

Equation (7) indicates that the degree of membership of an object i to a cluster j must be between 0 and 1. Equation (8) defines that the sum of the degrees of membership of an object i to different clusters must be equal to 1. Equation (9) indicates that the sum of all the degrees of membership in a cluster must be greater than 0 and less than n; that is, there must be no empty clusters and only one cluster.

The pseudocode of the standard FCM algorithm is shown in Algorithm 4 [11].

Algorithm 4: Standard FCM

Initialization:
  Assign the value for c y m;
  Determine the value of the threshold ε for convergence;
  t: = 0, $TMAX$: = 50;
  X: = $\{x_1, \ldots, x_n\}$;
  $U^{(t)}$: = $\{\mu_{11}, \ldots, \mu_{ij}\}$; is randomly generated
Calculate centroids:
  Calculate the centroid v_k;

9	**Classification:**
10	Calculate and update the membership matrix $U^{(t+1)}$: = $\{\mu_{ij}\}$
11	**Convergence:**
12	If max [abs($\mu_{ij}^{(t)} - \mu_{ij}^{(t+1)}$)] < ε or t $\leq$ $TMAX$:
13	Stop the algorithm;
14	Otherwise:
15	$U^{(t)}$: = $U^{(t+1)}$ y t: = t + 1;
16	Go to **Classification**
17	End of algorithm

4. Proposal for Improvement

The FCM algorithm is sensitive to the initial values of the membership matrix. Finding a good initial configuration is important since it accelerates the algorithm's convergence. To reduce the time complexity of the FCM algorithm, this section proposes a new solution approach that we call HOFCM. This approach results from observations of the dynamic behavior of the standard K-Means and FCM algorithms. Figure 1 shows Example 1, in which two alternatives are observed to select the initial centroids to execute the FCM algorithm. The first alternative is to randomly generate the centroids, Panel (a), and the second alternative runs the K-Means algorithm, Panel (c), whose final centroids are passed as initial centroids to FCM, Panel (d). In Panel (e), the final centroids of FCM are presented, considering the two alternatives. The dataset used is URBAN, which contains the longitude and latitude coordinates of traffic accidents in urban areas of Great Britain [29]; its values are n = 360,177, d = 2. For the example, a value of c = 8 is considered.

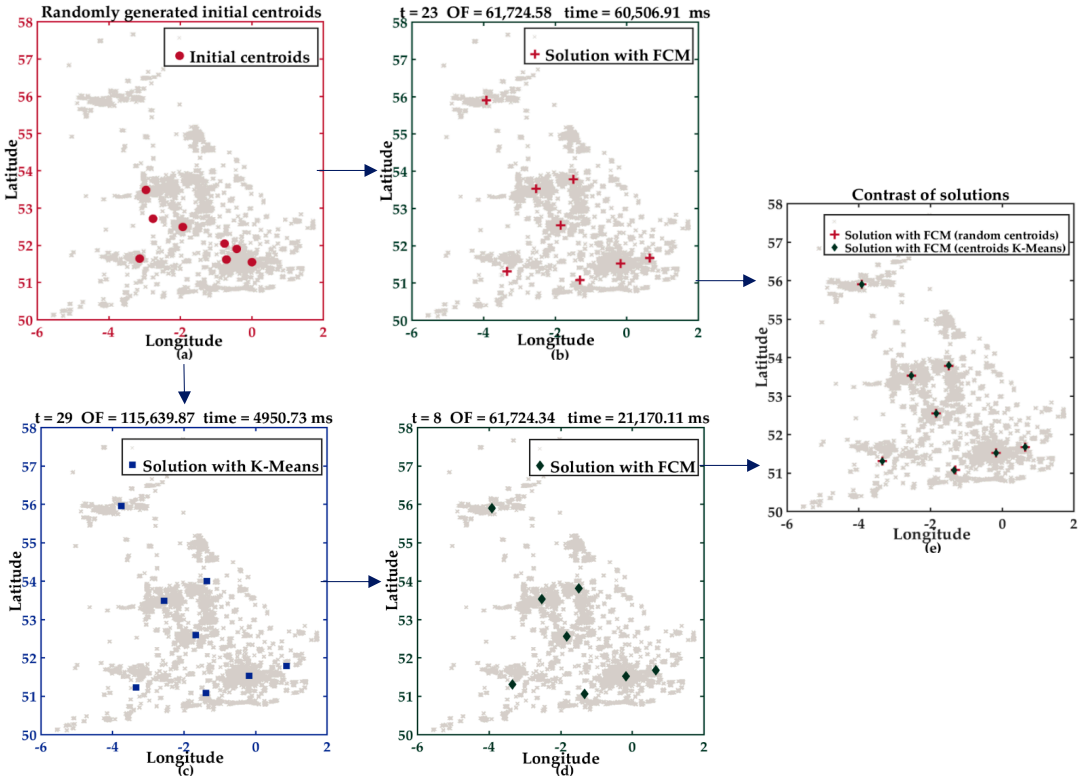

Figure 1. Initial and final centroids of the standard K-Means and FCM algorithms with the URBAN dataset in Example 1. Panel (**a**) shows the randomly generated initial centroids; (**b**) contains the final centroids with the solution of the standard FCM algorithm, considering the random initial centroids; (**c**) displays the final centroids of the K-Means algorithm receiving random centroids; (**d**) displays the final centroids of FCM with initial centroids generated after the convergence of K-Means; and (**e**) presents a comparison of the solutions of the FCM algorithm, whose initial centroids are random and generated by K-Means.

From Figure 1, the following observations emerge:

Observation 1: The positions of the final centroids for the standard K-Means and FCM algorithms, when run with randomly generated initial centroids and K-Means-generated centroids, are, in general, approximately the same, as shown in the sample in Panel (e).

Observation 2: The solution time of FCM, in the first alternative, is 60,506.91 ms (Panel (b)), and in the second, it is 26,120.84 ms (Panel (d)). This last value is the result of adding the execution times of the K-Means and FCM of Panels (c) and (d). This results in a time reduction of 56.83%.

Observation 3: The difference in the value of the objective function is negligible compared to the gain in the solution of the algorithm.

Employing the same idea as that in Figure 1, in Figure 2, Example 2 is presented, illustrating the dynamic behavior of the K-Means and FCM algorithms with different initial random centroids.

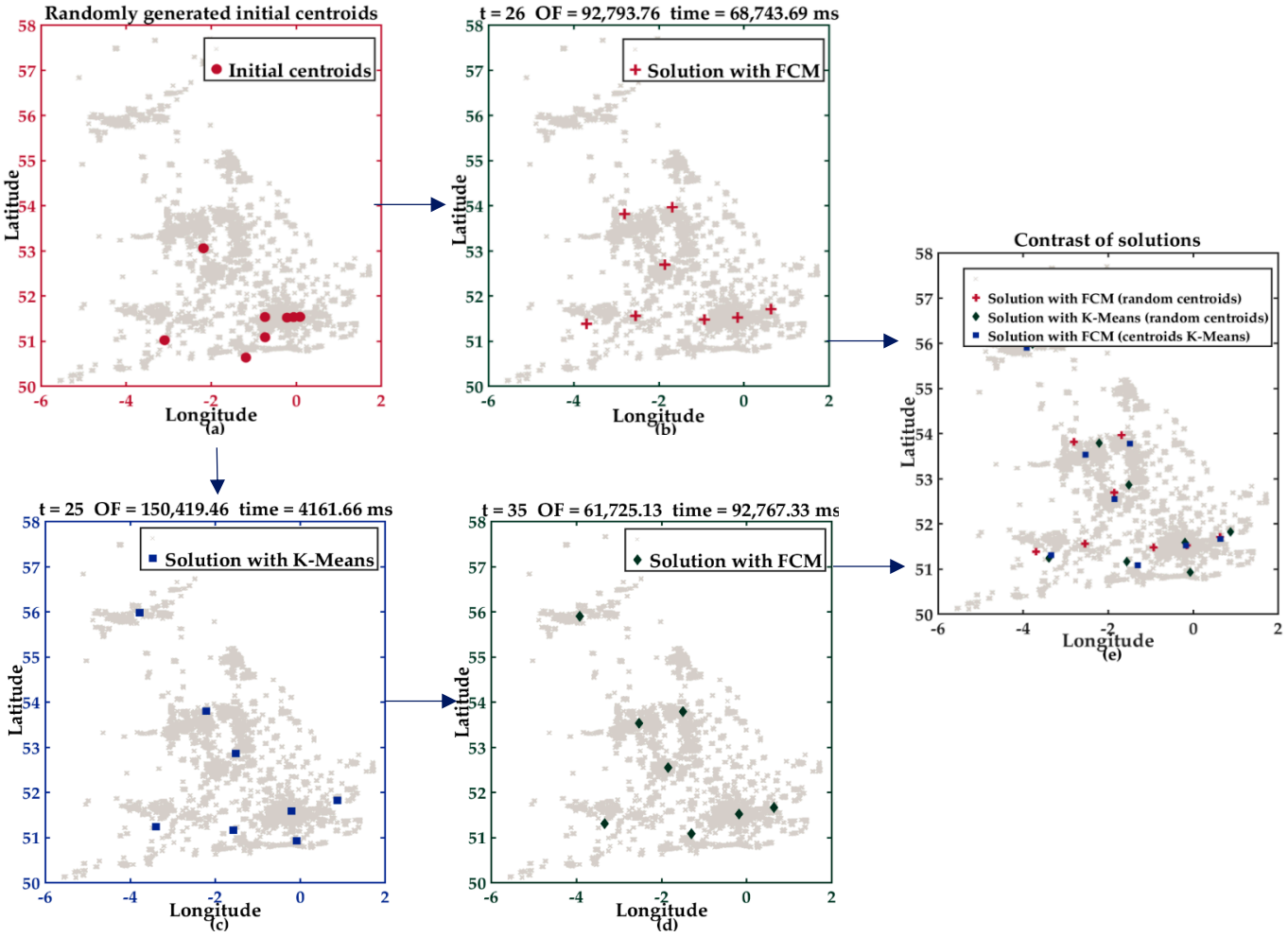

Figure 2. Initial and final centroids of the standard K-Means and FCM algorithms with the URBAN dataset in Example 2. Panel (**a**) shows the randomly generated initial centroids; (**b**) contains the final centroids with the solution of the standard FCM algorithm, considering the random initial centroids; (**c**) visualizes the final centroids with the solution of the K-Means algorithm that receives random centroids; (**d**) shows the final centroids of FCM with initial centroids generated after K-Means convergence; and (**e**) presents a comparison of solutions of the FCM and K-Means algorithms.

From Figure 2, the following observation can be deduced:

Observation 4: The final centroids of the K-Means algorithm, which are passed as an input parameter to FCM, do not always favor the reduction of the time complexity of FCM. In Panel (d), it is observed that the execution times, also considering the execution times of

the K-Means algorithm, increased by 41% compared to those in Panel (b). However, the quality of the solution was better by 33.5%.

It is important to mention that, in the different experiments carried out, the behavior of the K-Means algorithm was similar to that in Example 1. Most of the time, the final centroids are close to the neighborhood of the final centroids of FCM. However, sometimes, this is not the case. HOFCM takes a sample of 10 solutions from a dataset to solve this problem. Additionally, the promising heuristics of the K-Means algorithm are used. In Figure 3, the structure of the HOFCM algorithm is described.

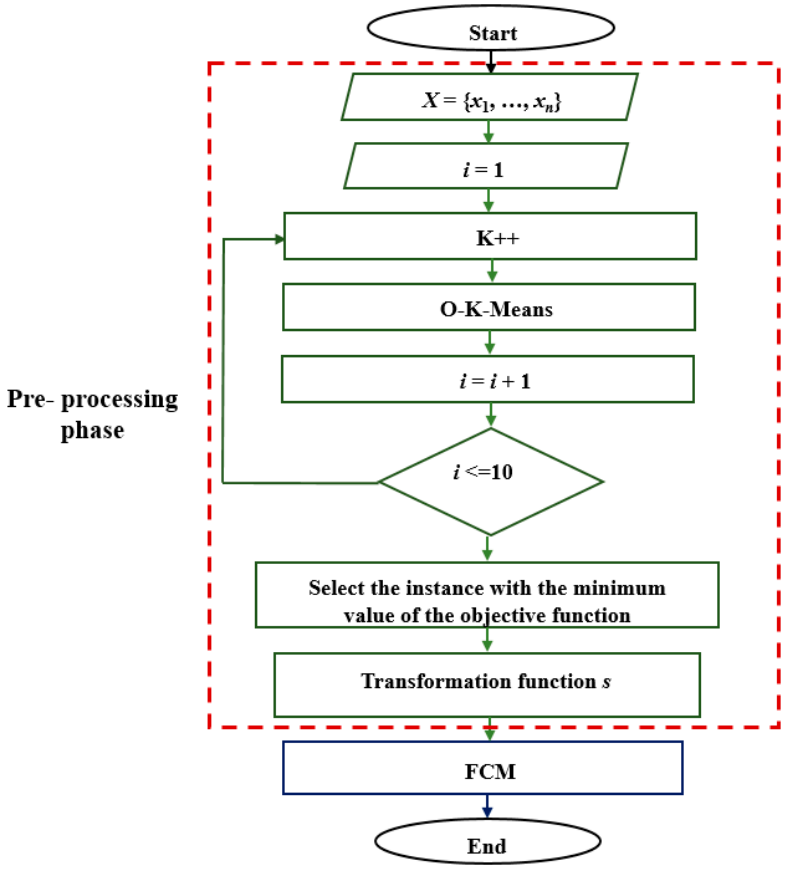

Figure 3. Structure of the HOFCM algorithm.

Figure 3 shows that, given dataset X, a pre-processing phase generates a set of 10 solutions for that dataset. For this, the K++ algorithm is executed, which returns initial centroids; the O-K-Means algorithm receives these. At the end of the convergence phase, such an algorithm returns optimized centroids. From the set of 10 solutions of dataset X, the value of i, in which the objective function obtained the minimum value, is identified, and its final centroids are selected. These final centroids are transformed into the initial values of the membership matrix through a transformation function s.

The components of the transformation function s are described in more detail in the following subsection.

4.1. Transformation Functions

The transformation function s has three elements: domain, codomain, and transformation rules [30]. The following mathematical notation corresponds to the description of the domain:

$X = \{x_1, \ldots, x_n\}$ the set of n objects to partition, where $x_i \in \mathrm{R}^d$ for $i = 1, \ldots, n$, and $d \geq 1$ is the number of dimensions.

$V = \{v_1, \ldots, v_c\}$ is the set of final centroids of a solution obtained with the O-K-Means algorithm, where v_j is the centroid of the cluster j.

x_i and v_j are the vectors described in Equations (5) and (6).

m = value of the weighting exponent, where $1 < m < \infty$.

The codomain is represented by μ_{ij}, where μ_{ij} is the membership matrix of each object x_i to each cluster j.

The transformation rules are defined in Equations (10) and (11):

$$u_{ij} = 1 \quad \text{if } x_i = v_j \tag{10}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{D(x_i, v_j)}{D(x_i, v_k)} \right)^{2/(m-1)}} \quad \text{if } x_i \neq v_j \tag{11}$$

The transformation function s is defined in Equation (12):

$$s(x_i, v_j, m) = u_{ij} \tag{12}$$

Carrying out this heuristic is reasonable because the O-K-Means algorithm has shown promising results in the experimental part. For this reason, the FCM algorithm would have less work in the clustering, and, therefore, the number of iterations would be reduced.

In Algorithm 5, the pseudocode that incorporates the HOFCM heuristics proposed in Figure 1 is shown.

Algorithm 5: HOFCM

Initialization:
  X: = $\{x_1, \ldots, x_n\}$;
  V: = $\{v_1, \ldots, v_c\}$;
  εok: = Threshold value for determining O-K-Means convergence;
  Assign the value for c;
  i: = 1;
  Repeat
    Function K++ (X, c):
      Return V';
    Function O-K-Means (X, V'', εok, c):
      Return V'';
    $i = i + 1$;
  While i <=10;
  Select V'' for the value of i at which the objective function obtained the minimum value;
  Transformation function s;
  Determine the value of the threshold ε to determine the convergence of the algorithm;
  Assign the value for m;
  t: = 1;
Calculate centroids:
  Calculate the centroid v_j;
Classification:
  Calculate and update the membership matrix $U^{(t+1)}$: = $\{\mu_{ij}\}$;
Convergence:
  If max [abs($\mu_{ij}{}^{(t)} - \mu_{ij}{}^{(t+1)}$)] < ε:
    Stop the algorithm;
  Otherwise:
    $U^{(t)}$: = $U^{(t+1)}$ y t: = $t + 1$;
  Go to **Classification**
End of algorithm

The HOFCM algorithm has four phases. The first is the initialization phase; in this phase, three functions are created: (1) The K++ function, which receives dataset X and the number of clusters as input parameters; this function returns a set of centroids identified

with variable V'. (2) The O-K-Means function, which receives dataset X, the set of centroids V' generated from K++, and the threshold value εok as input parameters to determine the convergence of the algorithm and the number of clusters; at the end of the phase of convergence, it returns optimized centroids defined with variable V''. (3) The transformation function s, which allows the final centroids of the O-K-Means algorithm to be transformed, taken from the lowest value of the objective function into the initial optimized values of the membership matrix; these are the input parameters for the FCM algorithm. The second, third, and fourth phases of the HOFCM algorithm, called the calculation of centroids, classification, and convergence, respectively, are typical of the standard FCM.

5. Results

5.1. Experiment Environment

To evaluate the performance of HOFCM, a set of four real datasets was selected, obtained from the UCI machine learning repository [29]. Additionally, a synthetic dataset randomly generated its attribute values with a uniform distribution between 0 and 1. To compare the results of HOFCM, the following algorithms were selected and implemented: K-Means, O-K-Means, K++, standard FCM, FCM++, NFCM, and HOFCM. All algorithms were coded in C language with the GCC 7.4.0 compiler. The computer equipment used had the following configuration: Intel® Core™ i9-10900 (Santa Clara, CA, USA) 2.80 GHz, 32 GB RAM, a 1 TB hard drive, and a Windows 10 Pro operating system.

For the design of the computational experiments and the analysis of our algorithms, we used the methodology proposed in [31].

The standard FCM, FCM++, NFCM, and HOFCM algorithms were executed with the following parameter values: $m = 2$, $\varepsilon = 0.01$. For the standard FCM, the initial values of the membership matrix were randomly generated; for the FCM++ algorithm, its centroids were generated with the K++ algorithm and a variant of it for the NFCM algorithm. In the case of the HOFCM algorithm, the optimized membership matrix, whose values are the results of the transformation function s, is described in Section 4.1.

For the K-Means algorithm, the centroids were randomly generated, and for the O-K-Means, they were generated with the K++ algorithm. Additionally, it is worth mentioning that the threshold's value determines the convergence of the O-K-Means $\varepsilon ok = 0.72$.

For all of the implemented algorithms, the value of $c = 2, 4, 6, 8, 10, 14, 18$, and 26. With this, eight test case configurations were considered for each dataset, one for each value of c.

The algorithms were run 10 times for each of the eight test case configurations of each dataset. The number of times the algorithm was selected to run was based on preliminary tests in which a sample size of 10 runs was found to provide good results.

Table 1 describes the set of datasets used. The table structure is as follows: column one contains the dataset identifier; column two contains the name; column three contains the data type; columns four and five contain the total objects and dimensions of the dataset; and column six contains the product of columns four and five. It is important to note that the SPAM, URBAN, and 1m2d datasets in this research are large datasets, considering an $n*d$ indicator greater than 200,000 objects.

Table 1. Datasets used in experiments.

Id	Name	Type	*n*	*d*	Size Indicator *n*d*
1	*WDBC*	Real	569	30	17,070
2	*ABALONE*	Real	4177	7	29,239
3	*SPAM*	Real	4601	57	262,257
4	*URBAN*	Real	360,177	2	720,354
5	*1m2d*	Synthetic	1,000,000	2	2,000,000

5.2. Description of Test Cases

In general, the design of the experiments focused on showing the quality of the solution and the efficiency of the HOFCM algorithm when solving large datasets such

as those used in Big Data. To perform the experimental evaluation of the HOFCM, two experiments were designed to compare the results obtained with respect to the standard FCM, FCM++, FCM-KMeans, and NFCM algorithms.

5.2.1. Description of Experiment I

The purpose of this experiment was to compare the results of the standard FCM and HOFCM algorithms.

Table 2 shows the average percentages of time reduction and the gain of the objective function, resulting from the comparison of the standard FCM and HOFCM algorithms, for all the datasets in Experiment I. The structure of Table 2 is as follows: column 1 contains the consecutive number of test cases; column two includes the value of each cluster; columns 3, 5, 7, 9, and 11 contain the average percentages of time of each dataset; and columns 4, 6, 8, 10, and 12 contain the average percentages of the value of the objective function for each dataset.

Table 2. Average percentages of time reduction and value of the objective function in Experiment I.

Standard FCM versus HOFCM											
		WDBC		***ABALONE***		***SPAM***		***URBAN***			
P	***c***	**%** ***time***	**%** J_m	**%** ***time***	**%** J_m	**%** ***time***	**%** J_m	**%** ***time***	**%** J_m	**%** ***time***	**%** J_m
1	2	80.88	0.00	64.14	0.0035	54.37	−0.001	46.12	4.12	74.95	0.03
2	4	87.12	0.02	56.15	−0.02	75.36	6.53	76.33	−0.001	87.81	0.03
3	6	67.83	**−2.19**	52.59	0.00	92.58	22.75	68.18	0.07	45.26	0.36
4	8	64.30	−1.17	86.60	0.08	**93.94**	36.90	74.60	1.42	85.56	0.22
5	10	57.55	3.03	75.61	0.09	86.22	50.69	70.98	1.85	67.75	−0.05
6	14	78.99	4.50	37.00	−0.11	77.20	49.30	70.38	4.26	89.65	−0.27
7	18	59.04	12.39	2.71	−0.60	79.80	56.24	68.45	7.81	84.07	0.05
8	26	64.99	27.31	36.78	−2.05	84.88	**68.31**	44.38	10.73	88.00	0.02

In Equations (13) and (14), it is shown how the percentages of time reduction and gain in the objective function, respectively, were obtained:

$$time = 100\left(1 - \frac{averageTime\_HOFCM}{averageTime\_standard\_FCM}\right) \tag{13}$$

$$Jm = 100\left(1 - \frac{JmAverage\_HOFCM}{JmAverage\_standard\_FCM}\right) \tag{14}$$

Figure 4 shows the millisecond time averages of each of the eight test cases of the five datasets in Experiment 1.

The analysis of the results in Table 2 and Figure 4 is described in Section 5.3.1.

5.2.2. Description of Experiment II

The purpose of this experiment was to compare the results of HOFCM and the other improvements of the standard FCM algorithm, which are called FCM++ [12], FCM-KMeans [13], and NFCM [14].

Tables 3–5, show the average percentages of time reduction and the gain of the objective function, resulting from the HOFCM versus FCM++, HOFCM versus FCM-KMeans, and HOFCM versus NFCM comparisons, respectively, for all the datasets and test cases in Experiment II. The structure of the tables is similar to that of Table 2. Equations (13) and (14) are used to obtain the time reduction and gain percentages in the objective function, respectively.

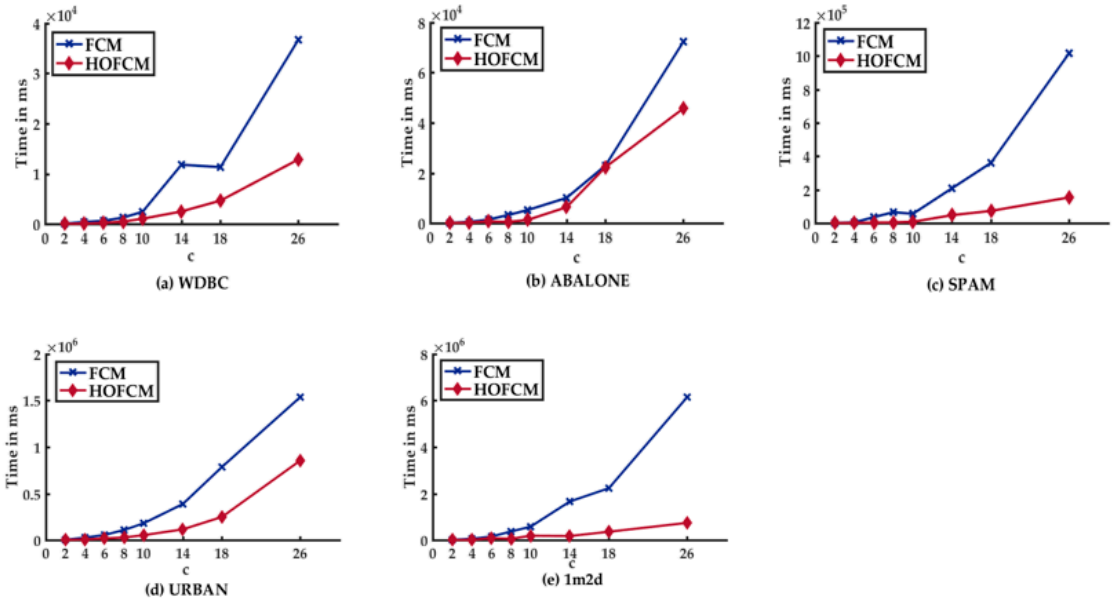

Figure 4. Results of the clustering in Experiment I.

Table 3. Average percentages of time reduction and value of the objective function in Experiment II; HOFCM versus FCM++.

HOFCM versus FCM++											
		WDBC		*ABALONE*		*SPAM*		*URBAN*		*1m2d*	
P	*c*	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m
1	2	66.87	0.00	59.38	0.00	41.67	0.00	18.19	0.00	32.08	0.03
2	4	78.59	0.02	36.47	−0.02	53.08	0.01	52.36	0.00	73.61	0.03
3	6	69.42	−0.52	54.03	0.04	54.04	**5.48**	54.83	1.86	37.78	−0.01
4	8	57.86	−0.20	80.64	0.05	60.42	1.97	70.42	2.20	84.36	0.02
5	10	57.29	1.76	68.80	0.09	71.62	**−2.33**	63.27	1.70	62.20	−0.07
6	14	45.97	0.43	39.00	0.04	25.56	2.77	41.40	2.10	86.60	−0.19
7	18	37.01	0.18	−7.22	−0.06	32.72	0.94	40.44	3.57	80.08	−0.03
8	26	27.58	2.92	38.48	−0.71	42.08	2.24	13.33	0.62	82.99	0.09

Table 4. Average percentages of time reduction and value of the objective function in Experiment II; HOFCM versus FCM-KMeans.

HOFCM versus FCM-KMeans											
		WDBC		*ABALONE*		*SPAM*		*URBAN*		*1m2d*	
P	*c*	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m
1	2	20.05	0.00	15.52	0.00	33.95	0.00	30.15	0.00	55.79	0.03
2	4	23.74	0.00	9.72	0.01	58.11	6.53	28.55	0.00	47.73	0.00
3	6	32.29	−2.51	−10.98	0.00	90.88	22.75	68.57	0.97	−13.94	0.42
4	8	47.87	−1.28	50.98	−0.02	72.58	14.66	58.99	0.14	53.89	−0.02
5	10	66.05	3.09	60.49	0.12	41.19	33.65	49.89	3.56	−63.13	0.05
6	14	84.13	4.52	46.83	0.69	57.67	50.13	63.45	2.17	78.87	0.06
7	18	84.21	11.99	24.45	−0.06	73.88	57.61	42.56	9.40	41.81	0.02
8	26	80.98	29.01	41.00	−0.81	83.03	66.86	5.55	3.99	45.48	−0.01

Table 5. Average percentages of time reduction and value of the objective function in Experiment II; HOFCM versus NFCM.

HOFCM versus NFCM											
		WDBC		*ABALONE*		*SPAM*		*URBAN*		*1m2d*	
P	*c*	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m	% *time*	% J_m
1	2	74.65	0.01	61.84	0.00	47.58	0.00	18.43	0.00	19.79	0.03
2	4	76.70	3.74	44.02	−0.02	55.53	4.66	60.83	0.00	82.07	0.03
3	6	61.78	−1.09	51.07	0.02	75.69	4.77	54.85	2.34	30.90	0.13
4	8	55.55	−0.30	82.67	0.76	81.25	9.07	72.68	1.81	87.45	0.01
5	10	48.54	3.21	71.65	0.46	55.00	20.87	56.78	3.77	59.43	0.16
6	14	35.87	1.15	53.76	0.17	19.88	12.98	58.73	2.38	86.76	−0.15
7	18	1.49	2.63	−4.36	−0.40	34.42	1.35	44.02	3.29	76.42	0.12
8	26	20.09	4.98	26.10	−2.02	47.18	5.33	22.79	3.47	84.93	0.14

Figure 5 shows the behavior of the set of datasets for each of the test cases in Experiment II.

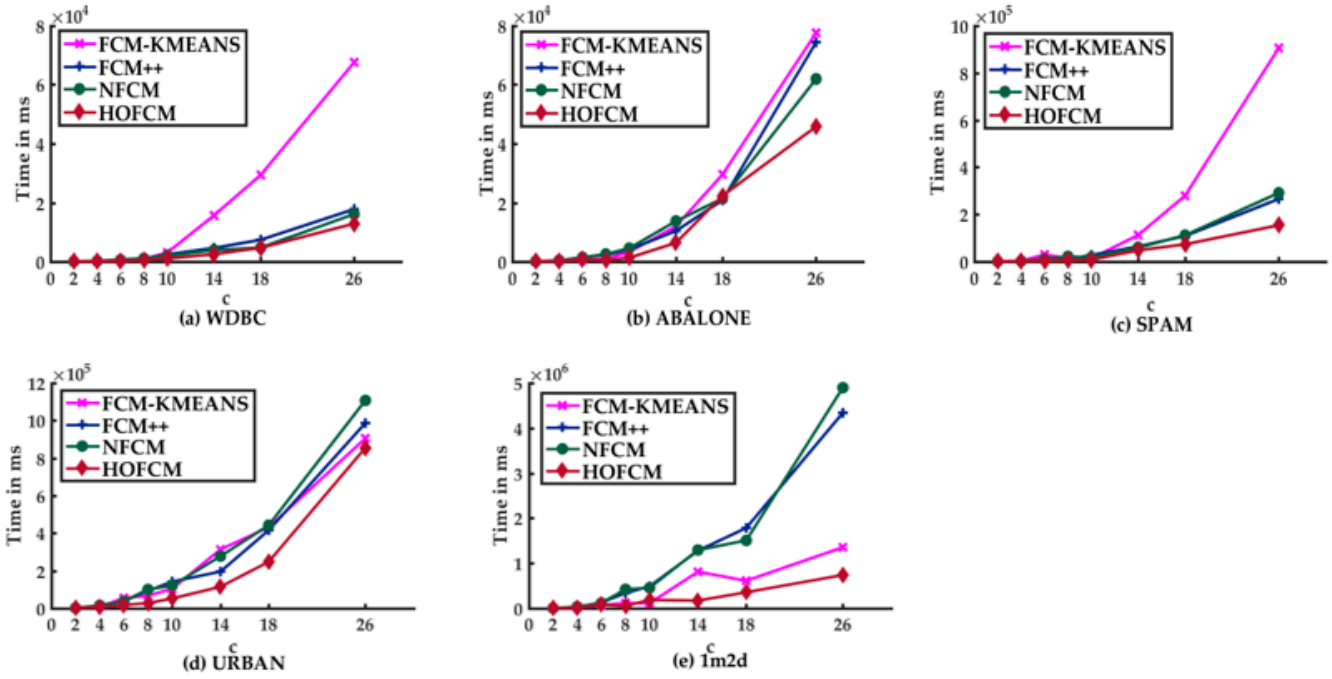

Figure 5. The behavior of datasets in Experiment II.

The analysis of the results in Tables 3–5 and Figure 5 is described in Section 5.3.2.

5.3. Analysis of Experiments

This section analyzes the results obtained after solving the problems described in Experiments I and II.

5.3.1. Analysis of the Results of Experiment I

Based on the experimental results shown in Table 2 and Figure 4, where the results of HOFCM and the standard FCM are shown, the following was observed:

1. In the case of HOFCM, for all datasets, time was reduced in all test cases. In the best case, it was reduced by up to 93.94%. This percentage is highlighted in bold in column seven. Notably, this percentage was achieved with the SPAM dataset, which has high dimensionality.

2. HOFCM improved the quality of the solution in large datasets in 20 of 24 cases. In the best case, the quality of the solution was enhanced by 68.31%. This percentage is highlighted in bold in column eight. In the worst case, a quality loss of 2.19% was identified, as can be seen in column four. Regarding the small datasets, a solution quality of 62.5% was obtained in all cases, that is, in 10 of the 16 test cases.
3. In general, based on the results obtained, it is possible to affirm that the HOFCM proposal performs better than the standard FCM algorithm.

5.3.2. Analysis of the Results of Experiment II

Considering the results of Tables 3–5 and Figure 5, the following was observed:

1. HOFCM outperformed the FCM++ algorithm in terms of solution time in all test cases with large datasets. In the small datasets, only in one case was it not higher. Regarding the quality of the solution in large datasets, in the best case, an average gain of 5.48% was obtained, and in the worst case, there was a loss of 2.33%. Both percentages are in bold in column eight in Table 3. It can be stated that HOFCM was higher in terms of solution quality in 75% of all cases.
2. HOFCM outperformed the FCM-KMeans algorithm in solution quality in 82.5% of all test cases. Regarding the solution time, HOFCM was better in the large and real datasets.
3. HOFCM outperformed the NFCM algorithm in solution quality in 85% of all test cases. Regarding the solution time, HOFCM was better in the large datasets.
4. In all three comparisons, HOFCM obtained the greatest time reductions and the greatest gains in solution quality when dealing with large and real datasets.
5. In general, based on the results obtained, it is possible to affirm that the HOFCM proposal performs better than the FCM++, FCM-KMeans, and NFCM algorithms.

As an example, Table 6 shows the sum of the solution times in milliseconds for the eight test cases of each dataset and each algorithm implemented in the two experiments carried out in this research. The structure of Table 6 is as follows: column one shows the name of the dataset, and columns two–six show the names of each algorithm.

Table 6. Sum of solution times in milliseconds of Experiments I and II.

Algorithm Dataset Name	Standard FCM	FCM-KMeans	FCM++	NFCM	HOFCM
WDBC	64,516.85	116,784.57	34,037.94	28,395.16	21,689.84
ABALONE	115,702.20	123,930.39	113,680.29	105,730.07	76,996.63
SPAM	1,746,301.34	1,355,655.16	482,874.71	513,205.42	289,610.79
URBAN	3,080,009.70	1,886,664.03	1,885,697.60	2,104,360.89	1,321,799.74
1m2d	11,181,675.94	3,091,877.12	8,385,299.13	8,729,514.14	1,592,132.11
The total sum of solution time	**16,188,206.03**	**6,574,911.27**	**10,901,589.67**	**11,481,205.68**	**3,302,229.11 ***
The number of times by which the HOFCM algorithm is faster	**4.90**	**1.99**	**3.30**	**3.48**	

* Best result obtained.

As can be observed in Table 6, the standard FCM algorithm is the one that generates more time in the convergence of the eight test cases for all the datasets. In general, it can be stated that the HOFCM algorithm is 4.90, 1.99, 3.30, and 3.40 times faster than the standard FCM, FCM-KMeans, FCM++, and NFCM algorithms, respectively.

6. Discussion

In this research, HOFCM was compared not only with the standard FCM algorithm but also with other improved algorithms in the literature, such as FCM++ [12], FCM-KMeans [13], and NFCM [14], which are the closest to HOFCM. It is worth mentioning that, among these three improved algorithms, it has not been visualized analytically whether

the solution of the K-Means algorithm is on the path of convergence with FCM most of the time. Due to the above, in this research, it was proposed to obtain a set of ten solutions for each of the datasets used with the K++ and O-K-Means algorithms, which have shown promising solutions. The purpose of having ten solutions is to obtain the best set of final centroids by selecting the best value of the objective function, which allows, through a transformation process, one to obtain the optimized membership matrix to accelerate the algorithm's convergence.

The results obtained in the experimental part show that the HOFCM algorithm outperformed the standard FCM, FCM++, FCM-KMeans, and NFCM algorithms in all cases. Evidence for this is given in Table 6. Based on the above, it is highlighted that, for the initialization of the standard FCM, it is not enough to implement the K++ algorithm, as in the case of FCM++; a variant of K++, as in the case of NFCM; or the K-Means algorithm, as in the case of FCM-KMeans.

Another important point to note is that, in the results presented in Table 6, the FCM++ and NFCM algorithms show a minimum difference of approximately 5%. This can be corroborated by the results presented in [14]. Additionally, it can be stated that, in general, the FCM-KMeans algorithm obtained better results than those mentioned above.

7. Conclusions

This paper shows that it is feasible to reduce the time complexity of the FCM algorithm by optimizing the initial membership matrix. To validate the proposal, which we call HOFCM, a set of experiments composed of real and synthetic datasets was designed. It is noteworthy that the sizes of some of the datasets were larger than those of the datasets reported in the specialized literature. To compare the results of the algorithms, all datasets were solved using HOFCM and the standard FCM, FCM++, FCM-KMeans, and NFCM algorithms.

Based on the results, it was observed that HOFCM obtained a reduction in solution time in all large datasets compared to the standard FCM algorithm. When comparing the results of HOFCM with the other algorithms, it was observed that it was 2.0, 3.3, and 3.5 times faster than the FCM-KMeans, FCM++, and NFCM algorithms, respectively. In particular, it was observed that the HOFCM algorithm showed a good performance in solving large datasets. For example, the SPAM dataset, which has high dimensionality, reduced the time by 93.94% compared to the standard FCM. Regarding the quality of the solution, it was observed that, with large datasets, a gain of up to 68.12% was obtained in the best case, and in the worst case, the quality was reduced by 2.51%.

To continue this research, we suggest two lines of investigation: the first is to implement HOFCM under the parallel and distributed programming paradigm, and the second is to determine in detail the type of datasets where the FCM algorithm is dominant.

Finally, we consider that the principles used in our approach could be used in other variants that improve the FCM algorithm.

Author Contributions: Conceptualization, J.P.-O., S.S.R.-A. and N.N.A.-O.; methodology, J.P.-O. and S.S.R.-A.; software, S.S.R.-A. and N.N.A.-O.; validation, J.P.-O., S.S.R.-A., C.Z.-D., Y.H. and J.F.S.; formal analysis, J.P.-O., J.F.S. and C.Z.-D.; investigation, J.P.-O., S.S.R.-A. and N.N.A.-O.; resources, S.S.R.-A., N.N.A.-O. and V.L.-N.; data curation, S.S.R.-A., N.N.A.-O. and V.L.-N.; writing—original draft preparation, J.P.-O. and S.S.R.-A.; writing—review and editing, J.P.-O., S.S.R.-A., N.N.A.-O., J.F.S., C.Z.-D., Y.H. and V.L.-N.; visualization, S.S.R.-A.; supervision, J.P.-O.; project administration, S.S.R.-A.; funding acquisition, J.P.-O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Tecnológico Nacional de México, grant number 13869.22-P, grant number 13541.22-P, and PRODEP.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The real datasets used were obtained from the UCI machine learning repository https://archive.ics.uci.edu/ml/index.php (accessed on 26 January 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yang, M.S. A survey of fuzzy clustering. *Math. Comput. Model.* **1993**, *18*, 1–16. [CrossRef]
2. Nayak, J.; Naik, B.; Behera, H.S. Fuzzy C-Means (FCM) Clustering Algorithm: A Decade Review from 2000 to 2014. In Proceedings of the Comput Intell Data Mining, Odisha, India, 20–21 December 2014.
3. Shirkhorshidi, A.S.; Aghabozorgi, S.; Wah, T.Y.; Herawan, T. Big Data Clustering: A Review. In Proceedings of the International Conference on Computational Science and Its Applications—ICCSA 2014, Guimaraes, Portugal, 30 June–3 July 2014.
4. Ajin, V.W.; Kumar, L.D. Big data and clustering algorithms. In Proceedings of the 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), Bangalore, India, 6–7 April 2016.
5. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symp Math Statis and Probability, Berkeley, CA, USA, 21 June–18 July 1965.
6. Ruspini, E.H.; Bezdek, J.C.; Keller, J.M. Fuzzy Clustering: A Historical Perspective. *IEEE Comput. Intell. Mag.* **2019**, *14*, 45–55. [CrossRef]
7. Lee, G.M.; Gao, X. A Hybrid Approach Combining Fuzzy c-Means-Based Genetic Algorithm and Machine Learning for Predicting Job Cycle Times for Semiconductor Manufacturing. *Appl. Sci.* **2021**, *11*, 7428. [CrossRef]
8. Lee, S.J.; Song, D.H.; Kim, K.B.; Park, H.J. Efficient Fuzzy Image Stretching for Automatic Ganglion Cyst Extraction Using Fuzzy C-Means Quantization. *Appl. Sci.* **2021**, *11*, 12094. [CrossRef]
9. Ghosh, S.; Kumar, S. Comparative Analysis of K-Means and Fuzzy C-Means Algorithms. *Int. J. Adv. Comput. Sci. Appl.* **2013**, *4*, 35–39. [CrossRef]
10. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman & Co.: New York, NY, USA, 1979; pp. 109–120.
11. Bezdek, J.C. *Pattern Recognition with Fuzzy Objective Function Algorithms*; Plenum Press: New York, NY, USA, 1981; pp. 43–93.
12. Stetco, A.; Zeng, X.J.; Keane, J. Fuzzy C-means++: Fuzzy C-means with effective seeding initialization. *Expert Syst. Appl.* **2015**, *42*, 7541–7548. [CrossRef]
13. Wu, Z.; Chen, G.; Yao, J. The Stock Classification Based on Entropy Weight Method and Improved Fuzzy C-means Algorithm. In Proceedings of the 2019 4th International Conference on Big Data and Computing, Guangzhou, China, 10–12 May 2019.
14. Liu, Q.; Liu, J.; Li, M.; Zhou, Y. Approximation algorithms for fuzzy C-means problem based on seeding method. *Theor. Comput. Sci.* **2021**, *885*, 146–158. [CrossRef]
15. Arthur, D.; Vassilvitskii, S. k-means++: The Advantages of Careful Seeding. In Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 7–9 January 2007.
16. Cai, W.; Chen, S.; Zhang, D. Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation. *Pattern Recognit.* **2007**, *40*, 825–838. [CrossRef]
17. Al-Ayyoub, M.; Al-andoli, M.; Jararweh, Y.; Smadi, M.; Gupta, B.B. Improving fuzzy C-mean-based community detection in social networks using dynamic parallelism. *Comput Elect. Eng.* **2018**, *74*, 533–546. [CrossRef]
18. Hashemzadeh, M.; Oskouei, A.G.; Farajzadeh, N. New fuzzy C-means clustering method based on feature-weight and cluster-weight learning. *Appl. Soft. Comput.* **2019**, *78*, 324–345. [CrossRef]
19. Khang, T.D.; Vuong, N.D.; Tran, M.-K.; Fowler, M. Fuzzy C-Means Clustering Algorithm with Multiple Fuzzification Coefficients. *Algorithms* **2020**, *13*, 158. [CrossRef]
20. Khang, T.D.; Tran, M.-K.; Fowler, M. A Novel Semi-Supervised Fuzzy C-Means Clustering Algorithm Using Multiple Fuzzification Coefficients. *Algorithms* **2021**, *14*, 258. [CrossRef]
21. Naldi, M.C.; Campello, R.J.G.B. Comparison of distributed evolutionary k-means clustering algorithms. *Neurocomputing* **2015**, *163*, 78–93. [CrossRef]
22. Pérez, J.; Almanza, N.N.; Romero, D. Balancing effort and benefit of K-means clustering algorithms in Big Data realms. *PLoS ONE.* **2018**, *13*, e0201874.
23. Selim, S.Z.; Ismail, M.A. K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. *IEEE Trans. Pattern Anal. Mach. Intell.* **1984**, *PAMI-6*, 81–87. [CrossRef]
24. Jancey, R.C. Multidimensional group analysis. *Aust. J. Bot.* **1966**, *14*, 127–130. [CrossRef]
25. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
26. Bellman, R.; Kalaba, R.; Zadeh, L.A. Abstraction and pattern classification. *J. Math. Anal. Appl.* **1966**, *13*, 1–7. [CrossRef]
27. Ruspini, E.H. A new approach to clustering. *Inf. Control* **1969**, *15*, 22–32. [CrossRef]
28. Dunn, J.C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *J. Cybern.* **1974**, *3*, 32–57. [CrossRef]
29. UCI Machine Learning Repository, University of California. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 26 January 2022).
30. Rosen, K.H. *Discrete Mathematics and Its Applications*; McGraw-Hill Education: New York, NY, USA, 2018; pp. 90–98.
31. McGeoch, C.C. *A Guide to Experimental Algorithmics*; Cambridge University Press: Cambridge, UK, 2012; pp. 17–114.