computers

Article

A New Scalable, Distributed, Fuzzy C-Means Algorithm-Based Mobile Agents Scheme for HPC: SPMD Application

Fatéma Zahra Benchara *, Mohamed Youssfi, Omar Bouattane and Hassan Ouajji

Laboratory SSDIA, ENSET Mohammedia, Hassan II University of Casablanca, Mohammedia 28999, Morocco; med@youssfi.net (M.Y.); o.bouattane@gmail.com (O.B.); ouajji@enset-media.ac.ma (H.O.)
* Correspondence: benchara.fatemazahra@gmail.com; Tel.: +212-665966441

Academic Editor: José Luis Vázquez-Poletti
Received: 4 May 2016; Accepted: 1 July 2016; Published: 11 July 2016

Abstract: The aim of this paper is to present a mobile agents model for distributed classification of Big Data. The great challenge is to optimize the communication costs between the processing elements (PEs) in the parallel and distributed computational models by the way to ensure the scalability and the efficiency of this method. Additionally, the proposed distributed method integrates a new communication mechanism to ensure HPC (High Performance Computing) of parallel programs as distributed one, by means of cooperative mobile agents team that uses its asynchronous communication ability to achieve that. This mobile agents team implements the distributed method of the Fuzzy C-Means Algorithm (DFCM) and performs the Big Data classification in the distributed system. The paper shows the proposed scheme and its assigned DFCM algorithm and presents some experimental results that illustrate the scalability and the efficiency of this distributed method.

Keywords: Fuzzy C-Means algorithm; parallel and distributed computing; distributed classification; MRI Big Data analysis; Multi Agent System; mobile agent

1. Introduction

Computer science technologies have introduced several intensive data application-based complex tasks in different domains (Internet of Things (IoT), cloud computing, data mining, Big Data analysis), etc., in order to improve HPC (High Performance Computing).

Consider the large amount of data and the complex tasks that these applications have to process. Their scalability and efficiency depends on their abilities to manage these considerations. They also depend on the processing environment where they are deployed. For example, in the medical domain, performing an application for MRI (magnetic resonance imaging) image cerebral analysis-based clustering algorithms. It involves a wide number of data to be processed by this application and that requires great processing power to achieve HPC.

Clustering algorithms are widely used in the medical field to analyze, and diagnose and detect abnormal regions based on MRI image classification. However, these features require using high performance computational models that grant the efficiency and the flexibility with the most complex clustering algorithms such as the Fuzzy C-Means Algorithm. So, how can we implement these requirements in parallel and distributed computational model-based distributed system? Consider the great challenge of optimizing the communication cost in distributed computational models. We will present a cooperative computational processing model that achieves these computational requirements. This paper is organized as follows:

- We provide the model of parallel and distributed computing where the distributed DFCM method is assigned to be implemented (Section 3).

- We demonstrate that the DFCM method implementation-based mobile agents is promising for Big Data classification (Section 4); and by implementing single program, multiple data (SPMD) clustering applications (Section 5) as a distributed one, we ensure HPC.

2. Background

To highlight the aim of this paper, we start with a brief overview about the parallel and distributed computational models [1] and their ability to perform clustering tasks on a large image D(w,h) where w represents the width of the image and h the height of the image. We suppose that this image is split into a set of ($me \times ne$) elementary images Dj\{$j = 1, \ldots, (me \times ne)$\}. Performing a task T based on intensive data Dj using a single machine can be a difficulttask and unreachable. So, in the classification case the task T needed to be performed is split into two global tasks, one executed by the host node and the second executed by all of the slave nodes in the computing grid. The data Dj will be distributed to the nodes in order to perform a collaborative parallel and distributed classification. However, we need to take into account the integrated computational model that ensures the classification management and optimizes the communication cost between the nodes.

The multi-agent system (MAS) [2] is a distributed technology system composed of a set of distributed agents which live, and cooperate, between each other using their intelligence and skills indifferent environments in order to overcome complex challenges. There are several interesting proposed models and methods based on this technology such as: in [3] the authors proposed a method that parallelizes the procedure for detection of illegal consumers of electricity which can be improved by the use of a distributed architecture based on multi-agent systems; and in [4] the authors proposed a new model for automatic construction of business processes based also on this technology. Additionally, in [5], the authors proposed a platform to facilitate the provision of home-care services based on agent three layered architecture. In the HPC domain, the authors in [6] proposed the use of this technology to improve the management, the flexibility, and the reusability of grid-like parallel computing architecture, and in [7] the authors presented the improvement of the time efficiency of a medical reasoning system based on a multi-agent architecture. Thus, how can the mobile agents be both the promising solution for distributed clustering methods and ensure HPC?

Mobile agents have interesting skills, such as autonomy, mobility, and asynchronous communication ability. They can communicate by sending asynchronous ACL (agent communication language) messages between each other, which significantly reduces the communication cost in the computational model. Thus, the mobile agents grant efficient communication mechanisms for HPC.

3. Parallel and Distributed Computational Model

3.1. Model Overview

The cooperative computational model where the proposed (DFCM) method is assigned to be implemented is a parallel and distributed virtual machine based on mobile agents. This machine is built over a distributed computing grid of size ($me \times ne$) of mobile agents. In this grid (Figure 1), the mobile agents are arranged on a 2D matrix ($me \times ne$) as agent virtual processing elements (AVPEs) according to SPMD (Single Program Multiple Data) architecture. Each AVPE (i,j) is localized in row i and column j and has an identifier AID (agent identifier) defined by AID = $me \times i + j$. These AVPEs emulate the PEs (processing elements) in a parallel computing grid. In this model we consider that the asynchronous communication between the AVPEs by exchanging ACL messages have a great benefit on reducing the communication cost involved by the PEs.

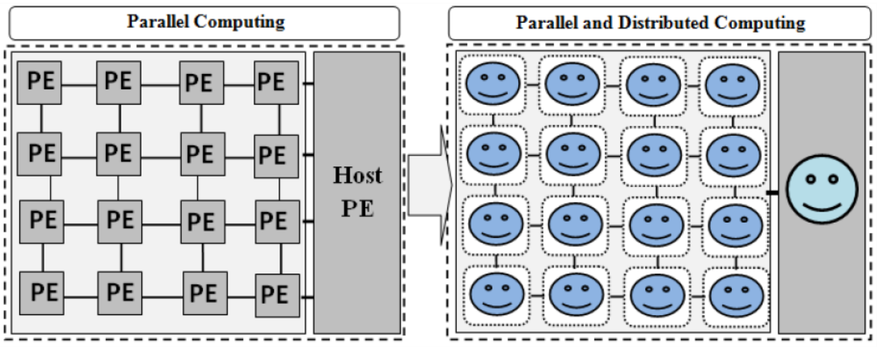


Figure 1. A cooperative computational grid of size 4 × 4.

3.2. Cooperative Mobile Agent Virtual Element (AVPE) Model

The distributed classification is performed by the implementation of the proposed DFCM method on a cooperative mobile agents team works model as illustrated in Figure 2. This model is composed of the Team Leader agent and the team worker agents (AVPEs). When the DFCM program is implemented in the model, the mobile Team Leader agent receives the MRI input image and splits it into (*me* × *ne*) elementary images, as shown in Figure 2a. Then it deploys a set of AVPEs of size (*me* × *ne*), and encapsulates the classification tasks and the elementary image per AVPE (Figure 2b). The AVPEs in Figure 2c perform the distributed classification on their assigned elementary image and send the results to their mobile Team Leader agent who, later in Figure 2d, computes the final results and assembles all elementary segmented images in order to display the output image results of the classification.

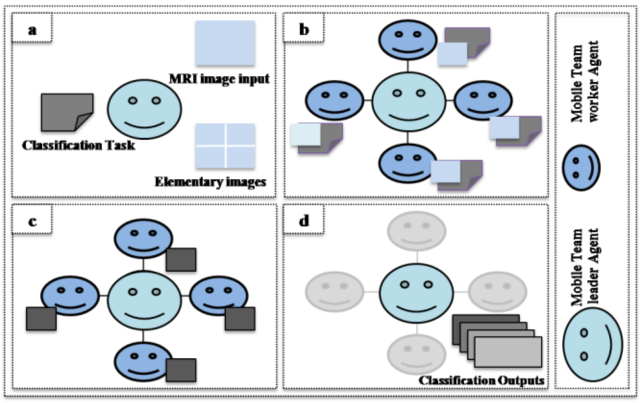


Figure 2. Computational model for DFCM classification overview.

4. Distributed Clustering Algorithm

4.1. Standard Fuzzy C-Means Algorithm

The well-known clustering algorithm named the fuzzy c-means (FCM) is proposed by Dunn [8] and extended by Bezdek [9]. It is a clustering method that allows one pixel of the segmented image to belong to two or more clusters, each one with a different membership degree between 0 and 1. The main goal of the FCM algorithm is to find the c-cluster centers (centroids) in the data set X = $\{x_1, x_2, \ldots, x_N\}$ that minimizes the objective function given by the following equation:

$$\mathrm{J}\left(\mathrm{U}, \mathrm{V}_1, \mathrm{V}_2, \ldots, \mathrm{V}_c\right) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\sum_{j=1}^{N} u_{ij}^{m} d\left(v_i, x_j\right) \tag{1}$$

The membership matrix U has the properties:

$$u_{ij} \in [0,1]\,,\ \forall i,j \tag{2a}$$

$$\sum_{i=1}^{c} u_{ij} = 1,\ \forall j = 1,\ldots,\ N \tag{2b}$$

$$0 < \sum_{j=1}^{N} u_{ij} < \mathrm{N},\ \forall i = 1,\ldots,c \tag{2c}$$

where

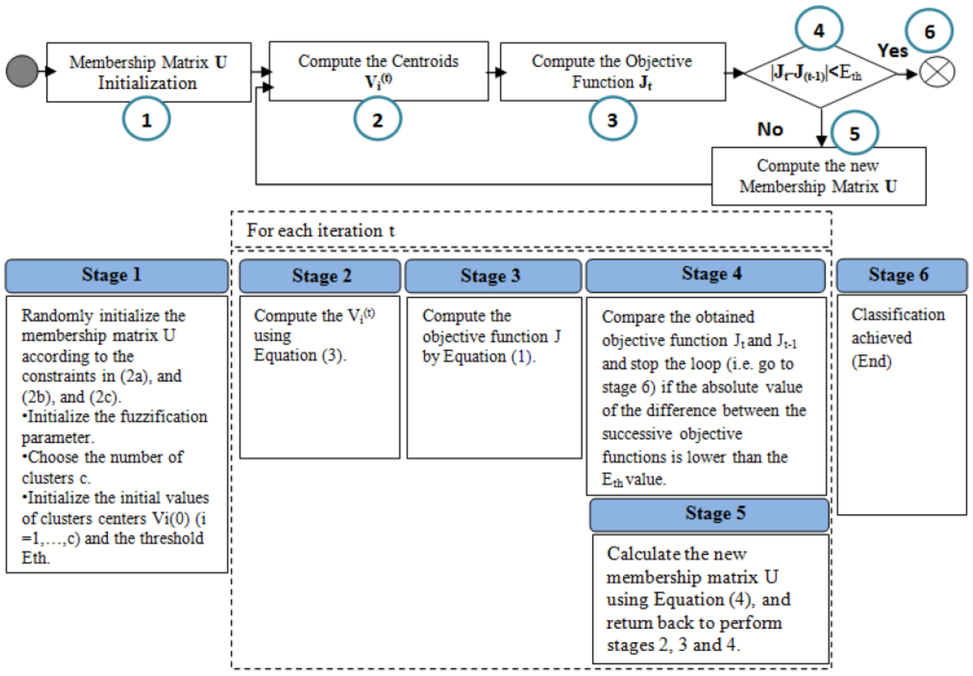
u_{ij}	Membership of data x_j in the cluster V_i.
V_i	Centroid of the cluster i.
$\mathrm{d}\left(v_i, x_j\right)$	Euclidian distance between centroid (V_i) and data point x_j.
$\mathrm{m} \in [1,\infty[$	Fuzzification parameter generally equals 2.
N	Number of data.
c	Number of clusters $2 \leqslant c < \mathrm{N}$.

To reach a minimum of dissimilarity function there are two conditions:

$$\mathrm{V}_i = \frac{\sum_{j=1}^{N} u_{ij}^{m} x_j}{\sum_{j=1}^{N} u_{ij}^{m}} \tag{3}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}} \tag{4}$$

The standard FCM classification is achieved according to the following algorithm stages, which are summarized in Figure 3.


Membership Matrix **U** Initialization
1
Compute the Centroids $\mathrm{V}_i^{(t)}$
2
Compute the Objective Function J_t
3
4
$|\mathrm{J}_t - \mathrm{J}_{(t-1)}| < \mathrm{E}_{th}$
Yes
6
No
5
Compute the new Membership Matrix **U**
For each iteration t
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6

Stage 1: Randomly initialize the membership matrix U according to the constraints in (2a), and (2b), and (2c).
•Initialize the fuzzification parameter.
•Choose the number of clusters c.
•Initialize the initial values of clusters centers Vi(0) (i =1,…,c) and the threshold Eth.

Stage 2: Compute the $\mathrm{V}_i^{(t)}$ using Equation (3).

Stage 3: Compute the objective function J by Equation (1).

Stage 4: Compare the obtained objective function J_t and J_{t-1} and stop the loop (i.e. go to stage 6) if the absolute value of the difference between the successive objective functions is lower than the E_{th} value.

Stage 5: Calculate the new membership matrix U using Equation (4), and return back to perform stages 2, 3 and 4.

Stage 6: Classification achieved (End)

Figure 3. Standard fuzzy c-means classification stages.

4.2. Distributed Fuzzy C-Means Algorithm

The distributed algorithm described in Figure 4 is implemented in the proposed model based on a multi-agent system. The fuzzy c-means program is implemented according to SPMD architecture over a 2D Mesh of size ($me \times ne$). In this model each AVPE(a) ((a = 1 to NA), where NA = ($me \times ne$) is the number of AVPEs), is asked to perform the fuzzy c-means program using its assigned elementary image and return its elementary results to the mobile Team Leader agent. This later computes the current global class centers and newly distributes them. This process is repeated until the convergence of the distributed algorithm. This DFCM program is performed according to the three global distributed method steps:

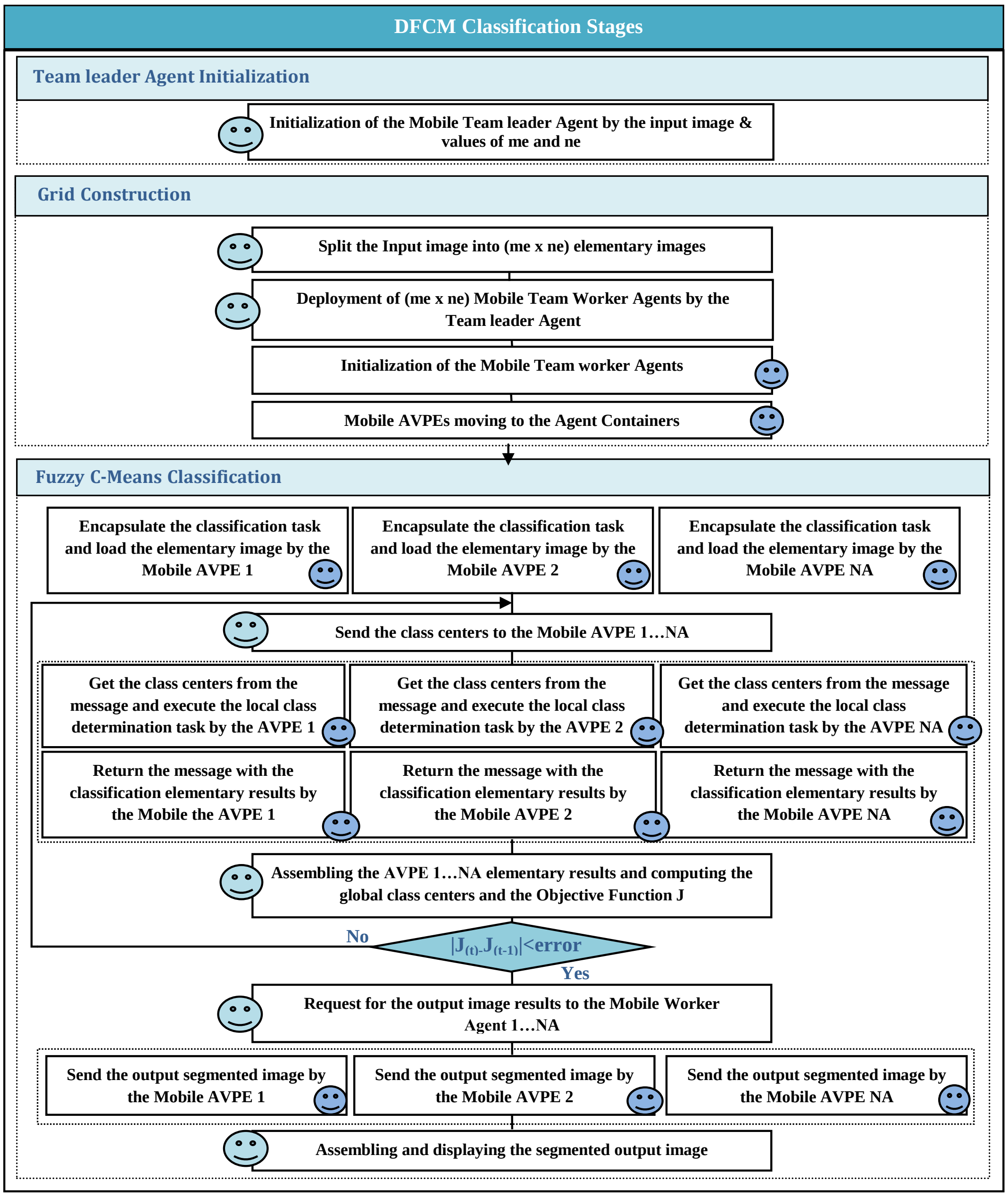

DFCM Classification Stages
Team leader Agent Initialization
Initialization of the Mobile Team leader Agent by the input image & values of me and ne
Grid Construction
Split the Input image into (me x ne) elementary images
Deployment of (me x ne) Mobile Team Worker Agents by the Team leader Agent
Initialization of the Mobile Team worker Agents
Mobile AVPEs moving to the Agent Containers
Fuzzy C-Means Classification
Encapsulate the classification task and load the elementary image by the Mobile AVPE 1
Encapsulate the classification task and load the elementary image by the Mobile AVPE 2
Encapsulate the classification task and load the elementary image by the Mobile AVPE NA
Send the class centers to the Mobile AVPE 1…NA
Get the class centers from the message and execute the local class determination task by the AVPE 1
Get the class centers from the message and execute the local class determination task by the AVPE 2
Get the class centers from the message and execute the local class determination task by the AVPE NA
Return the message with the classification elementary results by the Mobile the AVPE 1
Return the message with the classification elementary results by the Mobile AVPE 2
Return the message with the classification elementary results by the Mobile AVPE NA
Assembling the AVPE 1…NA elementary results and computing the global class centers and the Objective Function J
No
$|J_{(t)}-J_{(t-1)}|<error$
Yes
Request for the output image results to the Mobile Worker Agent 1…NA
Send the output segmented image by the Mobile AVPE 1
Send the output segmented image by the Mobile AVPE 2
Send the output segmented image by the Mobile AVPE NA
Assembling and displaying the segmented output image

Figure 4. Distributed FCM algorithm organization chart.

Step 1. Mobile Team Leader Agent Initialization

In this step the Team Leader agent is initialized by the input MRI image and the values of *me* and *ne*, to define the AVPE grid size (*me*, *ne*).

Step 2. Grid Construction

The Team leader agent splits the input image into (*me* × *ne*) elementary images and deploys (*me* × *ne*) AVPEs. Then, each AVPE(*a*) is initialized and migrates to its appropriate node to perform the classification task on its elementary image.

Step 3. Fuzzy C-Means Classification

- Each AVPE(*a*) encapsulates the task and load its elementary image.
- For each iteration *t*

{ 1. The Team Leader agent sends the class centers to all the AVPEs.

2. Each AVPE(*a*) gets the class centers from the message and executes the local class determination task.
3. Each AVPE(*a*) returns its classification elementary results, which consist of the following terms: TE1(*a*,*i*), TE2(*a*,*i*), TE3(*a*), and Cardinal(*a*,*i*).

   where:

   ○ ***TE1(a,i)*** contains the result of the sum of (U^m × data) computed for each class center *i*. This term is computed by:

   $$\boldsymbol{TE}1\,(\boldsymbol{a},\boldsymbol{i}) = \sum_{j=1}^{pi} \boldsymbol{u}_{ij}^{m}\boldsymbol{x}_j \tag{5}$$

   ○ ***TE2(a,i)*** contains the result of the sum of (U^m) computed for each class center *i* computed by:

   $$\boldsymbol{TE}2\,(\boldsymbol{a},\boldsymbol{i}) = \sum_{j=1}^{pi} \boldsymbol{u}_{ij}^{m} \tag{6}$$

   ○ ***TE3(a)*** contains the result of the sum of (U^m × distance^2) computed for all classes. This term is computed by:

   $$\boldsymbol{TE}3\,(\boldsymbol{a}) = \sum_{j=1}^{pi}\sum_{i=1}^{c} \boldsymbol{u}_{ij}^{m}\boldsymbol{d}(\boldsymbol{v}_i,\boldsymbol{x}_j) \tag{7}$$

   ○ ***Cardinal(a,i)*** contains the result of the sum of pixel membership for each class center *i*. This term is computed by:

   $$\boldsymbol{Cardinal}\,(\boldsymbol{a},\boldsymbol{i}) = \sum_{j=1}^{pi} \boldsymbol{u}_{ij} \tag{8}$$

   ○ ***pi:*** Number of pixels of the elementary image of the AVPE(*a*).

4. The Team Leader agent performs these three sub tasks: assembling the elementary results, computing the new class centers, and computing the objective function J_t.

   ○ Assembling the elementary resultsThe Team Leader agent receives the elementary results (TE1(*a*,*i*), TE2(*a*,*i*), TE3(*a*), Cardinal(*a*,*i*)) from each AVPE(*a*) and assembles them in order to compute the global values (GTE1(i), GTE2(i), GTE3(i), GC(i)), respectively, by the given equations:

$$\mathbf{GTE1}\,(\boldsymbol{i}) = \sum_{\mathbf{a}=1}^{\mathbf{NA}} \mathbf{TE1}\,(\boldsymbol{a}, \boldsymbol{i}) \tag{9}$$

$$\mathbf{GTE2}\,(\boldsymbol{i}) = \sum_{\mathbf{a}=1}^{\mathbf{NA}} \mathbf{TE2}\,(\boldsymbol{a}, \boldsymbol{i}) \tag{10}$$

$$\mathbf{GTE3}\,(\boldsymbol{i}) = \sum_{\mathbf{a}=1}^{\mathbf{NA}} \mathbf{TE3}\,(\boldsymbol{a}) \tag{11}$$

$$\mathbf{GC}\,(\boldsymbol{i}) = \sum_{\mathbf{a}=1}^{\mathbf{NA}} \mathbf{Cardinal}\,(\boldsymbol{a}, \boldsymbol{i}) \tag{12}$$

where:

- **GTE1(*i*)** is the global value of TE1(a,i) over the AVPEs of the grid.
- **GTE2(*i*)** is the global value of TE2(a,i) over the AVPEs of the grid.
- **GTE3(*i*)** is the global value of TE3(a) over the AVPEs of the grid.
- **GC(*i*)** is the global value of Cardinal(a,i) over the AVPEs of the grid.

- Computing the global class centersThe Team Leader agent gets the computed global values (GTE1(i), GTE2(i)) to compute the new class centers V_i by the following equation:

$$\mathbf{V}\boldsymbol{i} = \frac{\mathbf{GTE1}\,(\boldsymbol{i})}{\mathbf{GTE2}\,(\boldsymbol{i})} \tag{13}$$

- Computing the objective function J_t:The Team leader agentgets the global value of GTE3(i) to compute the objective function given by the following equation:

$$\mathbf{J}\boldsymbol{t} = \sum_{i=1}^{c} \mathbf{GTE3}\,(\boldsymbol{i}) \tag{14}$$

5. The Team Leader agent tests the condition of the algorithm convergence ($|J_t - J_{(t-1)}| < E_{\text{th}}$).

} // End of iteration t

- The Team Leader agent requests to each AVPE(a) the segmented elementary image.
- Each AVPE(a) sends the segmented elementary image result to the Team Leader agent.
- The Team Leader agent assembles the segmented elementary images and displays the segmented output image.

5. Distributed Environment and Results

5.1. Distributed Environment Communication Mechanisms

A distributed computing environment, as illustrated in Figure 5, presents how the proposed method can perform the distributed programs thanks to the several mobile agent skills: mobility, autonomy, adaptability, and asynchronous communication. To illustrate the main idea of this contribution we present in Figure 6 an example of 2D mesh of size (4 × 4). Each AVPE in the grid computing has an agent message queue. When the Team Leader agent sends the computing data to their AVPEs team, the messages are stored in their queues. Thus, the AVPEs and the Team Leader agent can perform their assigned tasks and communicate between each other without the need of acknowledgments. Assume that each AVPE(*a*) needs to check the data in its queue and manage the computing time and the communication time. In [10], the authors implemented a distributed c-means method DCM in a distributed environment that includes an excellent agent communication management mechanism. This mechanism ensures asynchronous communication between the agents, which significantly reduces the communication cost and improves HPC. Thus, this mechanism is integrated in the proposed DFCM model to implement a scalable and efficient DFCM method.

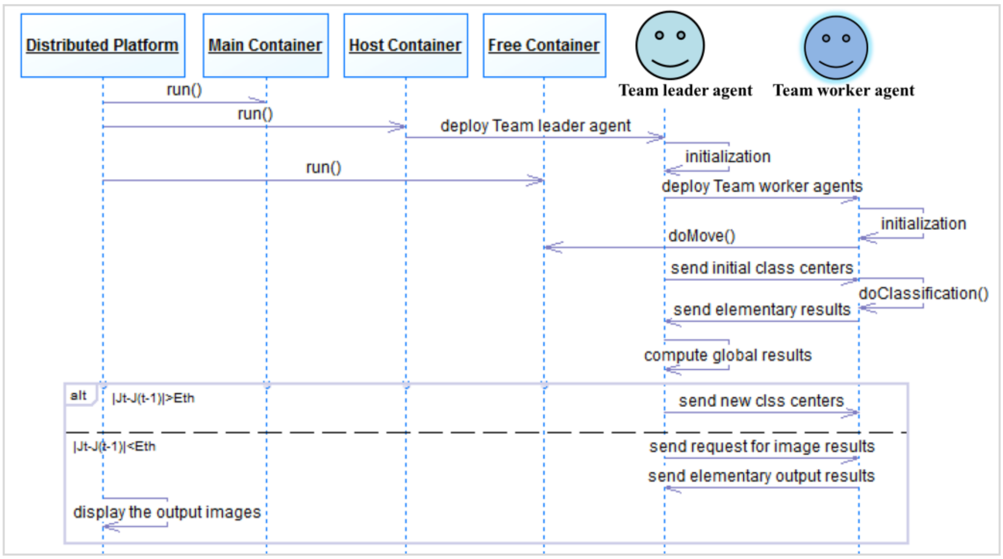


Figure 5. Sequence diagram for DFCM on a cooperative multi-agent model.

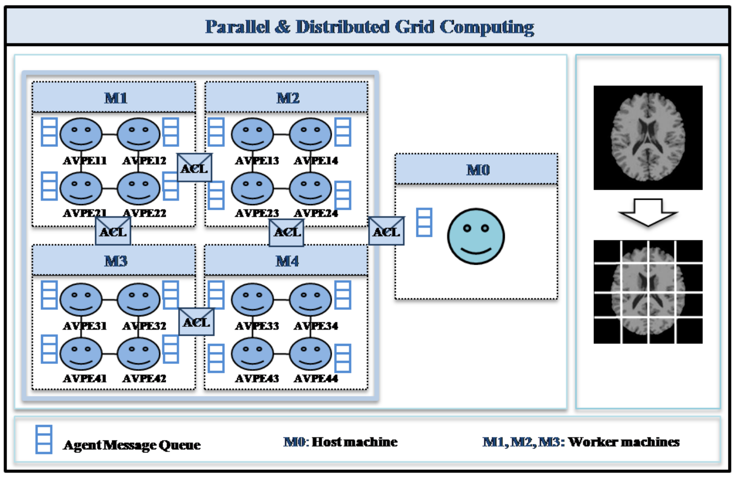


Figure 6. Agent asynchronous communication mechanisms in a 2D mesh (4 × 4).

5.2. Cooperative Multi-Agent Middleware

JADE (Java Agent DEvelopment) [1] is a middleware for developing the distributed multi-agent system (MAS), which is based on JAVA. It supports the agent asynchronous communication mechanisms by using the ACL messages according to the FIPA-ACL message specifications [1]. The cooperative DFCM model is implemented on a parallel and distributed virtual machine based on mobile agents according to the architecture in Figure 7.

The classification of the MRI image is performed on this platform by applying this middleware. It creates the main components of this model, which are:

(1) The host container: this is the second container which is started in the platform after the main container, where the mobile team leader agent is deployed in order to perform its tasks in the grid.
(2) The agent containers: these are the containers that are started in the platform, where the mobile team worker agents will move to perform their tasks.

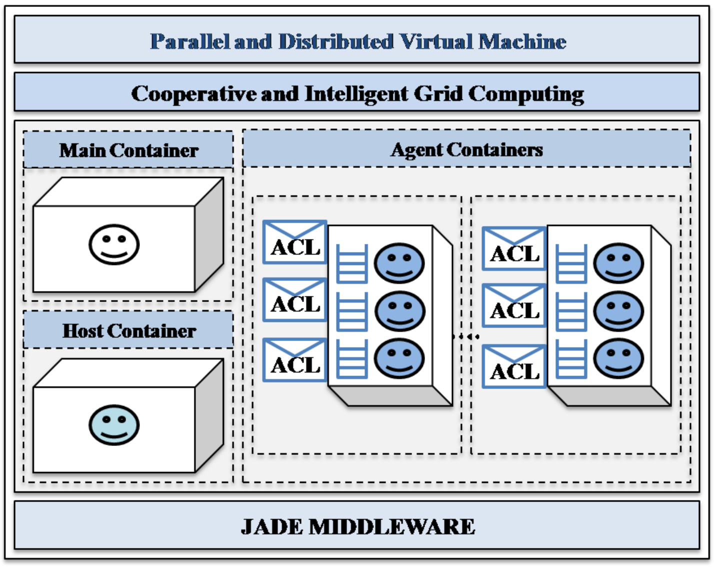

Parallel and Distributed Virtual Machine
Cooperative and Intelligent Grid Computing
Main Container
Agent Containers
Host Container
ACL
JADE MIDDLEWARE

Figure 7. Multi-agent middleware overview.

5.3. Implementation and Results

The proposed DFCM algorithm is implemented in this model for MRI medical image analysis. To do so, we choose two cerebral MRI images: brain MRI image (Img1) in Figure 8 and an abnormal brain MRI image (Img2) in Figure 9. Each image in Figure 8a is encapsulated on the team leader agent as an input image in order to be split into elementary images, as in Figure 8b. At the end of the classification process these images will be segmented into c output images (Figure 8c–e) that correspond, respectively, to the grey matter, the cerebrospinal fluid, and the white matter. The same algorithm is performed for the second image of Figure 9a to detect the abnormal region.

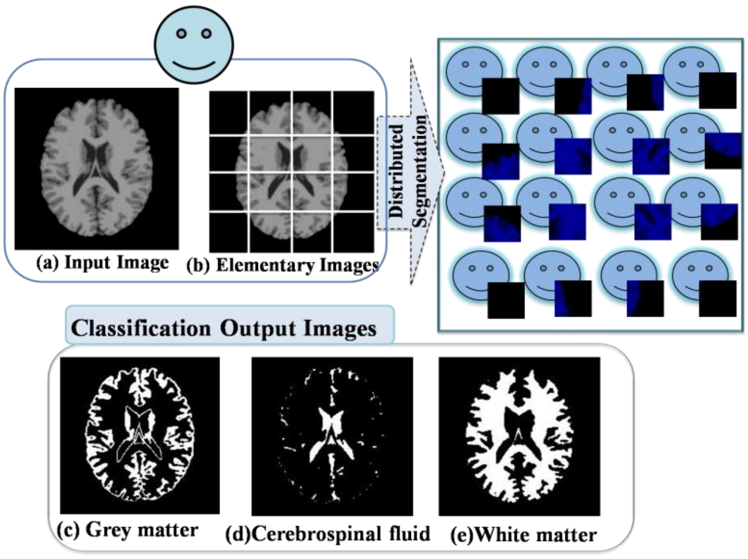


Figure 8. Classification results by the elaborated DFCM model implementation for Img1.

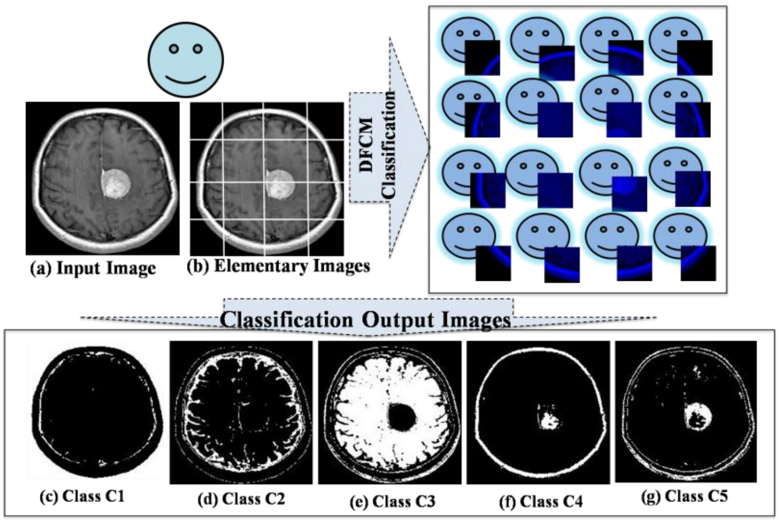


Figure 9. Classification results by the elaborated DFCM Model implementation for Img2.

To illustrate the effectiveness features of the implementation of the FCM program in this model, we present the proposed five cases studies:

(1) Dynamic convergence of this program for the MRI image (Img1) with two different class center initializations:

(a) (*c1*, *c2*, *c3*) = (*1.1*, *2.5*, *3.8*): in Table 1 and in Figure 10, where we see clearly the dynamic convergence of the algorithm to the final class centers (c1, c2, c3) = (1.100, 97.667, 146.569). The convergence is achieved after 13 iterations.

(b) (*c1*, *c2*, *c3*) = (*140.5*, *149.5*, *150.5*): the algorithm converges to the same final class centers (c1, c2, c3) = (1.100, 97.661, 146.566) after 20 iterations, as illustrated in Table 1 and Figure 11.

(2) Dynamic convergence of this program for the MRI image (Img2) with two different class center initializations:

(a) (*c1*, *c2*, *c3*, *c4*, *c5*) = (*1.5*, *2.2*, *3.8*, *5.2*, *8.6*): in Table 2 and in Figure 12 we see clearly the dynamic convergence of the algorithm to the final class centers (c1, c2, c3, c4, c5) = (1.742, 67.587, 101.709, 238.983, 170.040) after 35 iterations.

(b) (*c1*, *c2*, *c3*, *c4*, *c5*) = (*140.5*, *149.5*, *150.5*, *220.2*, *250.5*): where the algorithm converges to the same final class centers (c1, c2, c3, c4, c5) = (1.764, 67.967, 101.858, 239.140, 170.560) after 26 iterations, as shown in Table 2 and Figure 13.

(3) The DFCM classification time according to the number of agents involved in the classification for the initial class centers (c1, c2, c3, c4, c5) = (1.5, 2.2, 3.8, 5.2, 8.6) for Img1, and (c1, c2, c3, c4, c5) = (1.5, 2.2, 3.8, 5.2, 8.6) for Img2. In Figure 14 we see clearly that from 16 agents the classification time of the two images achieves minimum values of 108 ms for Img1 and of 278 ms for Img2. Thus, it is considered as the appropriate number of agents needed to classify these images.

A detailed comparison between the FCM and the DFCM methods is made in Table 3 for each image. Both methods converge to the same values of the class centers. We see clearly that the classification time of the DFCM method corresponds to the FCM for one agent and by adding the number of agents the DFCM method performs a reduced classification time which achieves its minimum values from 16 AVPEs for both images.

(4) The DFCM classification time according to the number of nodes in the grid computing by considering 16 AVPEs for the two images (Img1) and (Img2). In Figure 15, we see clearly that for both images the classification time achieves a gain of time of about 78% using eight nodes, compared to using just one. The corresponding classification data size for each node is illustrated in Figure 16.

(5) The speedup S(DFCM), its relative speedup S_R(DFCM), and the efficiency of the DFCM classification method are presented, respectively, in Figures 17 and 18, compared to the sequential FCM method. We perform interesting maximum relative speedups of 86.760% for Img1, which corresponds to again of 7.55, and of 82.372% for Img2, which corresponds to again of 5.67, by using 32 AVPEs. The speedup S(DFCM) and the relative speedup S_R(DFCM) are illustrated in Table 4 and computed, respectively, by the following equations:

$$S\,(\mathrm{DFCM})_{\mathrm{NA}} = \frac{T\,(\mathrm{FCM})}{T\,(\mathrm{DFCM})_{\mathrm{NA}}} \tag{15}$$

$$S_R\,(\mathrm{DFCM})_{\mathrm{NA}} = \frac{T\,(\mathrm{FCM}) - T\,(\mathrm{DFCM})_{\mathrm{NA}}}{T\,(\mathrm{FCM})} \times 100 \tag{16}$$

where:

- T(FCM)is the classification time of the FCM method which corresponds to one agent; and
- T(DFCM) is the classification time of the DFCM method which corresponds to the number NA of agents.

Table 1. Different states of the distributed fuzzy c-means (DFCM) algorithm forImg1 classification starting from different class centers initialization.

Case	**Initial Class Centers**			**Final Class Centers**			**Number of Iteration**
	C1	**C2**	**C3**	**C1**	**C2**	**C3**	
CASE 1	1.1	2.5	3.8	1.100	97.667	146.569	13
CASE 2	140.1	149.5	150.8	1.100	97.661	146.566	20

Table 2. Different states of the distributed fuzzy c-means (DFCM) algorithm for Img2 classification starting from class centers initialization.

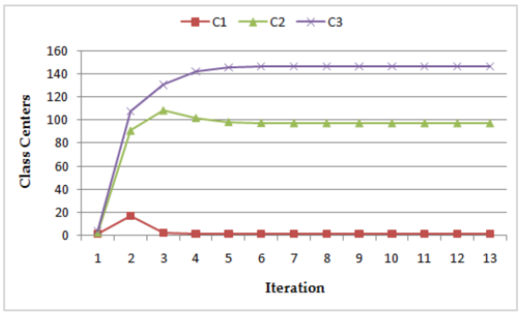
	Initial Class Centers					**Final Class Centers**					**Number of Iteration**
Case	**C1**	**C2**	**C3**	**C4**	**C5**	**C1**	**C2**	**C3**	**C4**	**C5**	
CASE 1	1.5	2.2	3.8	5.2	8.6	1.742	67.587	101.709	238.983	170.040	35
CASE 2	140.5	149.5	150.5	220.5	250.5	1.764	67.967	101.858	239.140	170.560	26

Table 3. FCM and DFCM method comparison for classification of two images (Img1and Img2).

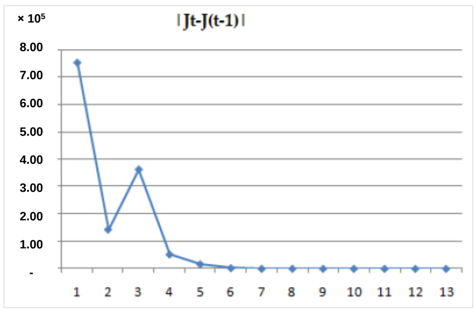
FCM Method		**DFCM Method**		
Classification Time (Img1) (ms)	**Classification Time (Img2) (ms)**	**Number of Agents**	**Classification Time (Img1) (ms)**	**Classification Time (Img2) (ms)**
778	1509	1	778	1509
-	-	2	334	860
-	-	4	200	516
-	-	8	144	371
-	-	16	108	278
-	-	32	103	266

Table 4. Speedup of the distributed fuzzy c-means (DFCM) method for Img1 and Img2.

Number of Agents	S_R(DFCM) (Img1) (%)	S(DFCM) (Img1)	S_R(DFCM) (Img2) (%)	S(DFCM) (Img2)
1	0	1	0	1
2	57.069	2.32	43.008	1.75
4	74.293	3.89	65.805	2.92
8	81.491	5.4	75.414	4.06
16	86.118	7.2	81.577	5.42
32	86.76	7.55	82.372	5.67

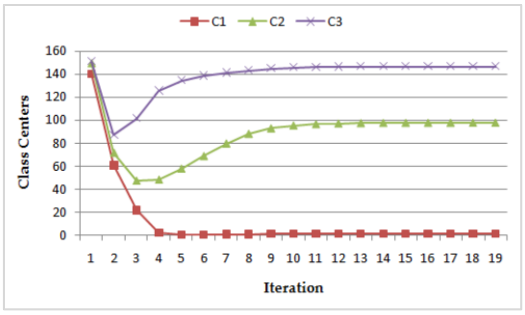


(**a**)

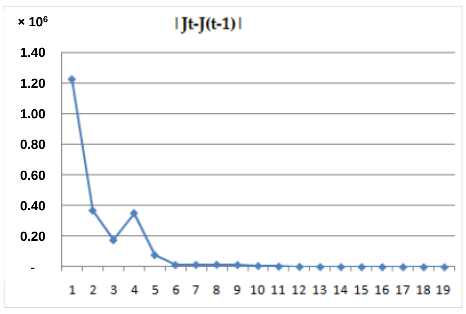


(**b**)

Figure 10. Dynamic convergence of the class centers starting from class centers (c1, c2, c3) = (1.1, 2.5, 3.8). (**a**) Class centers; and (**b**) Error of the objective function.

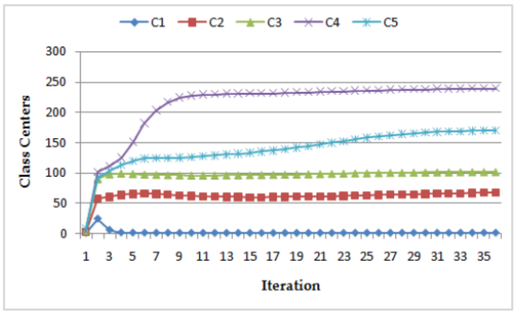


(**a**)

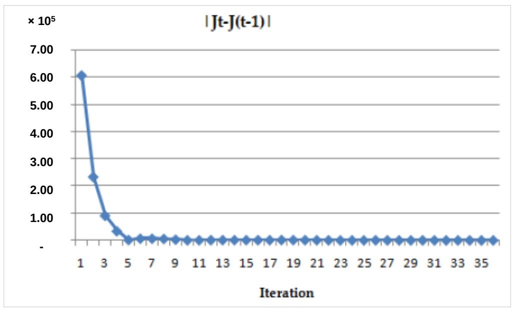


(**b**)

Figure 11. Dynamic convergence of the class centers starting from class centers (c1, c2, c3) = (140.5, 149.5, 150.5). (**a**) Class centers; and (**b**) Error of the objective function.

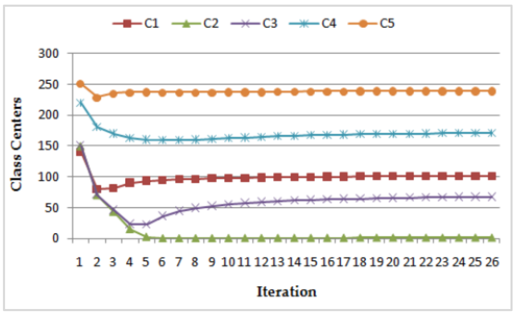


(**a**)

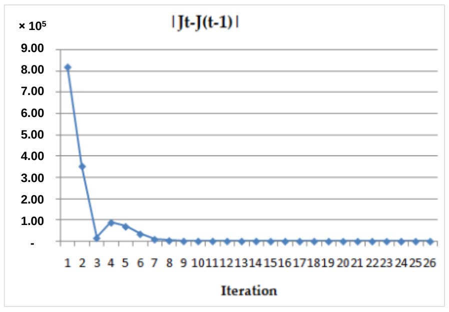


(**b**)

Figure 12. Dynamic convergence of the class centers starting from class centers (c1, c2, c3, c4, c5) = (1.5, 2.2, 3.8, 5.2, 8.6). (**a**) Class centers; and (**b**) Error of the objective function.



(**a**)



(**b**)

Figure 13. Dynamic convergence of the class centers starting from class centers (c1, c2, c3, c4, c5) = (140.5, 149.5, 150.5, 220.2, 250.5). (**a**) Class centers, and (**b**) Error of the objective function.

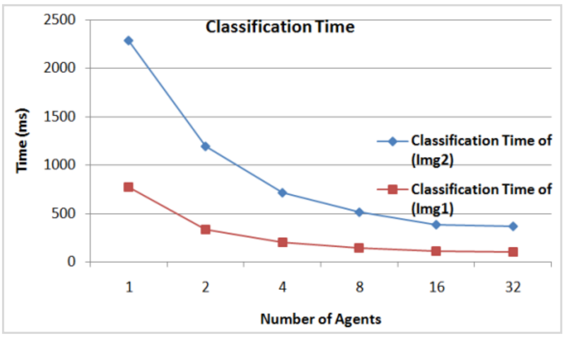


Figure 14. Time of DFCM classification for each MRI image depending on the number of agents with initial class centers (c1, c2, c3) = (1.1, 2.5, 3.8) for Img1, and (c1, c2, c3, c4, c5) = (1.5, 2.2, 3.8, 5.2, 8.6) for Img2.

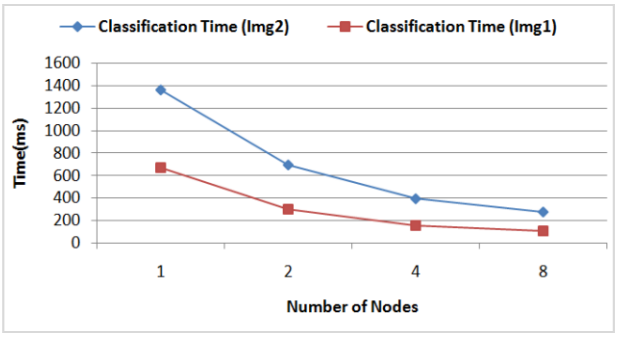
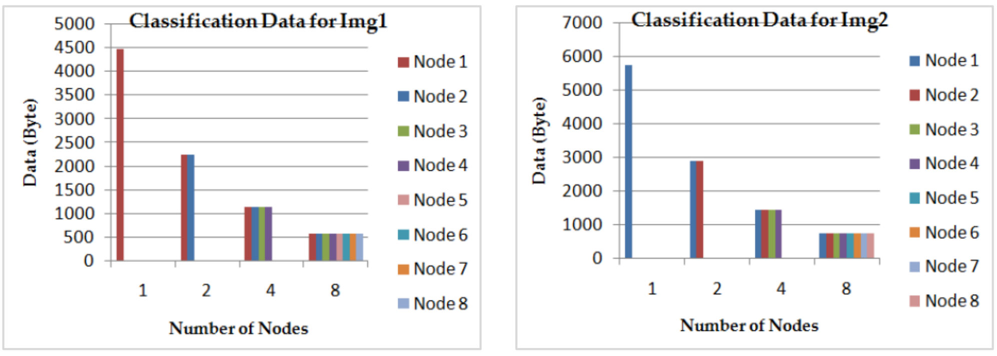


Figure 15. DFCM Classification time depending on the number of nodes in the grid using 16 AVPEs for Img1 with initial class centers (c1, c2, c3) = (1.1, 2.5, 3.8), and with initial class centers (c1, c2, c3, c4, c5) = (1.5, 2.2, 3.8, 5.2, 8.6) for the Img2.



(**a**) (**b**)

Figure 16. DFCM classification data depending on the number of nodes in the grid using 16 AVPEs (**a**) for Img1; (**b**) for Img2.

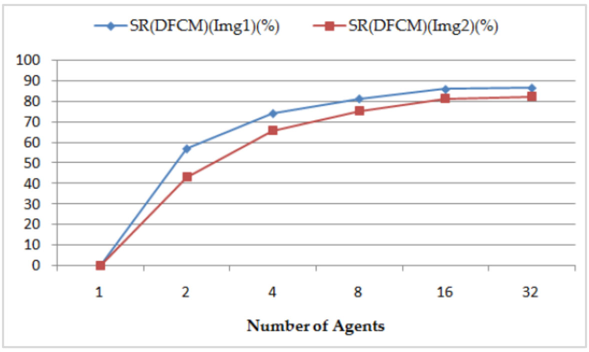


Figure 17. Relative Speedup of DFCM Classification depending on the number of AVPEs and with initial class centers (c1, c2, c3) = (1.1, 2.5, 3.8) for Img1, and (c1, c2, c3, c4, c5) = (1.5, 2.2, 3.8, 5.2, 8.6) for Img2.

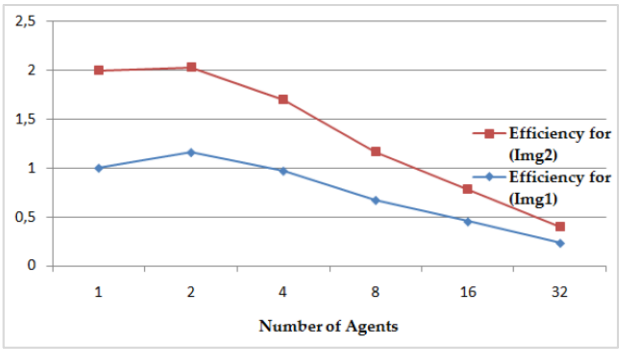


Figure 18. Efficiency of DFCM classification depending on the number of AVPEs and with initial class centers (c1, c2, c3) = (1.1, 2.5, 3.8) for Img1, and (c1, c2, c3, c4, c5) = (1.5, 2.2, 3.8, 5.2, 8.6) for Img2.

6. Related Work

There are several inspiring parallel methods of the clustering algorithm on massively parallel computational models which have demonstrated interesting clustering results. In [1], the authors proposed a parallel FCM implementation on a parallel architecture, based on dividing the computation among the processors for image segmentation analysis. Their proposed method presents a great improvement of the performance and the efficiency of the image segmentation as compared to the sequential implementation. In [11], the authors proposed a parallel FCM implementation for clustering large datasets, which is designed to run on a parallel SPMD architecture using MPI tools.

Their proposed implementation achieves ideal speedups as compared to an existing parallel c-means implementation. The classification becomes the main core of wide number of researchers in different fields: in [12], for decision tree induction, fuzzy rule-based classifiers [13,14], neural networks [15,16], and clustering [1,17].

The different parallel methods differ from each other by the computational models which are assigned to be implemented. Their implementations depend on the parallel computing strategies [18] used and that present some challenges: high cost of the machines, limitation on the test and validation of new algorithms, and they also involve a great number of computational resources that increases the communication cost between the processors in grid computing. The proposed distributed method within this contribution grants a scalable and efficient implementation on computational model-based mobile agents. In [10] the authors improved the parallel c-means method [19] by the use of the distributed one.

So, thanks to these several interesting works, the proposed DFCM method is implemented on a scalable and efficient mobile agents model. The JADE middleware-based mobile agents model is the easiest and the most suitable solution to implement this distributed method.

7. Conclusions

In this paper, we presented a distributed fuzzy c-means method (DFCM) and its application on MRI image classification. This method is implemented on an SPMD model based on cooperative mobile agents grid computing. This model implements the asynchronous communication mechanism, which is based on exchanging ACL messages between the AVPEs. The results obtained by implementing this program, related to the class centers convergence, the classification time, the speedup of the method, and the efficiency, demonstrate that the proposed DFCM method can reduce the complexity of the fuzzy clustering algorithms and, especially, the communication cost between the PEs. The mobile agents abilities ensure the distributed performance keys that ensure HPC.

Author Contributions: Fatéma Zahra Benchara proposed, implemented the distributed model, performed the experimental results and wrote the paper. Mohamed Youssfi proposed the idea, implemented the sequential model, analyzed and validated the experimental results. Omar Bouattane and Hassan Ouajji performed the review of the paper.

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

References

1. Rahimi, S.; Zargham, M.; Thakre, A.; Chillar, D. A Parralel Fuzzy C-Mean Algorithm for Image Segmentation. In Proceedings of the IEEE Annual Meeting of the Fuzzy Information, Banff, AB, Canada, 27–30 June 2004; pp. 234–237.
2. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; Wiley: West Sussex, UK, 2007.
3. Depuru, S.S.S.R.; Wang, L.; Devabhaktuni, V.; Green, R.C. High performance computing for detection of electricity theft. *ELSEVIER Electr. Power Energy Syst.* **2013**, *147*, 21–30. [CrossRef]
4. Coria, J.A.G.; Castellanos-Garzón, J.A.; Corchado, J.M. Intelligent business processes composition based on multi-agent systems. *ELSEVIER Expert Syst. Appl.* **2014**, *41*, 1189–1205. [CrossRef]
5. Isern, D.; Moreno, A.; Sánchez, D.; Hajnal, A.; Pedone, G.; Varga, L.Z. Agent-based execution of personalised home care treatments. *Appl. Intell.* **2011**, *34*, 155–180. [CrossRef]
6. Sánchez, D.; Isern, D.; Rodríguez-Rozas, A.; Moreno, A. Agent-based platform to support the execution of parallel tasks. *ELSEVIER Expert Syst. Appl.* **2011**, *38*, 6644–6656. [CrossRef]
7. Rodríguez-González, A.; Torres-Niño, J.; Hernández-Chan, G.; Jiménez-Domingo, E.; Alvarez-Rodríguez, J.M. Using agents to parallelize a medical reasoning system based on ontologies and description logics as an application case. *ELSEVIER Expert Syst. Appl.* **2012**, *39*, 13085–13092. [CrossRef]
8. Dunn, J.C. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *J. Cybern.* **1973**, *3*, 32–57. [CrossRef]

9. Bezdek, J.C. *Pattern Recognition with Fuzzy Objective Function Algorithms*; Plenum Press: New York, NY, USA, 1981.
10. Benchara, F.Z.; Youssfi, M.; Bouattane, O.; Ouajji, H.; Bensalah, M.O. Distributed C-Means Algorithm for Big Data Image Segmentation on a Massively Parallel and Distributed Virtual Machine Based on Cooperative Mobile Agents. *JSEA* **2015**, *8*, 103–113. [CrossRef]
11. Kwok, T.; Smith, K.; Lozano, S.; Taniar, D. Parallel Fuzzy C-means Clustering for Large Data Sets. In *Euro-Par 2002 Parallel Processing*, Proceedings of the 8th International Euro-Par Conference on Parallel Processing, Paderborn, Germany, 27–30 August 2002; Monien, B., Feldman, R., Eds.; Lecture Notes in Computer Science; Springer Verlag: Heidelberg, Germany, 2002; Volume 2400, pp. 365–374.
12. Kubota, K.; Nakase, A.; Sakai, H.; Oyanagi, S. Parallelization of decision tree algorithm and its performance evaluation. In Proceedings of the Fourth International Conference on High Performance Computing in the Asia-Pacific Region, Beijing, China, 14–17 May 2000; Volume 2, pp. 574–579.
13. Kim, M.W.; Lee, J.G.; Min, C. Efficient fuzzy rule generation based on fuzzy decision tree for data mining. In Proceedings of the IEEE International Fuzzy Systems Conference FUZZ-IEEE ’99, Seoul, Korea, 22–25 August 1999; pp. 1223–1228.
14. Evsukoff, A.; Costa, M.C.A.; Ebecken, N.F.F. Parallel Implementation of Fuzzy Rule Based Classifier. In Proceedings of the VECPAR’2004, Valencia, Spain, 28–30 June 2004; Volume 2, pp. 443–452.
15. Phua, P.K.H.; Ming, D. Parallel nonlinear optimization techniques for training neural networks. *IEEE Trans. Neural Netw.* **2003**, *14*, 1460–1468. [CrossRef] [PubMed]
16. Costa, M.C.A.; Ebecken, N.F.F. A Neural Network Implementation for Data Mining High Performance Computing. In Proceedings of the V Brazilian Conference on Neural Networks, Granada, Spain, 13–15 June 2001; pp. 139–142.
17. Boutsinas, B.; Gnardellis, T. On distributing the clustering process. *Pattern Recognit. Lett.* **2002**, *23*, 999–1008. [CrossRef]
18. El-Rewini, H.; Abd-El-Barr, M. *Advanced Computer Architecture and Parallel Processing*; Wiley: Hoboken, NJ, USA, 2005.
19. Bouattane, O.; Cherradi, B.; Youssfi, M.; Bensalah, M.O. Parallel c-means algorithm for image segmentation on a reconfigurable mesh computer. *ELSEVIER Parallel Comput.* **2011**, *37*, 230–243. [CrossRef]