A multi-agent-based approach for fuzzy clustering of large image data

Nashwa M. Abdelghaffar, Hewayda M. S. Lotfy & Soheir M. Khamis

Journal of Real-Time Image Processing

ISSN 1861-8200

J Real-Time Image Proc
DOI 10.1007/s11554-014-0473-3

J Real-Time Image Proc
DOI 10.1007/s11554-014-0473-3

ORIGINAL RESEARCH PAPER

A multi-agent-based approach for fuzzy clustering of large image data

Nashwa M. Abdelghaffar · Hewayda M. S. Lotfy · Soheir M. Khamis

Received: 18 March 2014 / Accepted: 17 November 2014

Abstract Data clustering usually requires extensive computations of similarity measures between dataset members and cluster centers, especially for large datasets. Image clustering can be an intermediate process in image retrieval or segmentation, where a fast process is critically required for large image databases. This paper introduces a new approach of multi-agents for fuzzy image clustering (MAFIC) to improve the time cost of the sequential fuzzy *c*-means algorithm (FCM). The approach has the distinguished feature of distributing the computation of cluster centers and membership function among several parallel agents, where each agent works independently on a different sub-image of an image. Based on the Java Agent Development Framework platform, an implementation of MAFIC is tested on 24-bit large size images. The experimental results show that the time performance of MAFIC outperforms that of the sequential FCM algorithm by at least four times, and thus reduces the time needed for the clustering process.

Keywords Image clustering · Fuzzy *c*-means · Multi-agent system

N. M. Abdelghaffar (✉) · H. M. S. Lotfy · S. M. Khamis
Mathematics Department, Faculty of Science, Ain Shams University, Cairo 11566, Egypt
e-mail: nashwa.abdelghaffar@sci.asu.edu.eg

H. M. S. Lotfy
e-mail: hewayda@hotmail.com

S. M. Khamis
e-mail: soheir_khamis@hotmail.com

1 Introduction

Data clustering is a key topic in Computer Science and a common technique for data analysis that is used in a variety of fields, such as data mining, pattern recognition, and machine learning. Clustering is the process of partitioning or grouping a given dataset into a number of clusters such that the items in the same cluster are as similar as possible, and items in different clusters are as dissimilar as possible [37]. Clustering methods can be classified according to whether the clusters are hard or fuzzy. Fuzzy clustering methods allow the items to belong to several clusters with different degrees of membership. In real systems, it is very often the case that no sharp boundaries between clusters exist, so that fuzzy clustering is more natural than hard clustering. Various methods of fuzzy clustering are given in [38].

One of the best-known and most widely used fuzzy clustering methods is the fuzzy *c*-means algorithm (FCM), which is developed based on iterative minimization of an objective function [6, 15, 34]. It was introduced in 1973 by Dunn and generalized in 1981 by Bezdek, as mentioned in [38]. In addition to using FCM in a variety of applications, such as medical diagnosis [18, 35], remote sensing [14, 16], and image segmentation [10, 25, 39, 40], FCM is also utilized in content-based image retrieval (CBIR) [26, 32]. In particular, a higher retrieval performance can be achieved when the clustering process for CBIR is used [8, 23], and the efficiency of utilizing the clustering algorithms for CBIR systems has already been shown. For instance, the FCM algorithm for an image clustering based on the color and texture features is presented in [26], where the color and texture features are extracted from an image pixel, then clustered into regions that exhibit similar features. Finally, regions are described by a set of features that are indexed for the retrieval process. Another CBIR system based on the FCM and *k*-means algorithms to improve the accuracy of the retrieval process is presented in [29].

2 Related works

Many solutions have been proposed to overcome FCM's high time complexity by reducing the number of computations in each iteration. For instance, Eschrich et al. [12] introduced a means of reducing the dataset that will be clustered through combining similar feature vectors. Solutions other than Eschrich et al.'s that follow the same approach are given in [2, 17]. Another approach is to make use of distributed or parallel processing technology, such that several processors can be used in parallel to speedup FCM. This approach is presented in [22, 24, 28], where the main concept is to distribute the iterative process to find the cluster centers among many processors. Usually, there is one master processor that equally divides the dataset and sends the results to the other processors to start the clustering process on the local data. The centralized control at the master process leads to lack of robustness.

A different approach commonly used within distributed artificial intelligence is offered by multi-agent systems (MASs), which are cooperative sets of agents acting together to solve a problem. An agent-based system is one of the most important paradigms, since it represents a new way of analyzing, designing, and implementing complex software systems. MASs are ideally suited to modeling a wide range of complex problems utilizing the massive power of distributed parallel systems. An advantage of employing MAS is "modularity", where each agent is specialized in solving a particular problem that enables efficient reusability of agents. Another advantage is "efficiency", which is due to the inherent concurrency of agents that are working at the same time in different independent or dependent sub-problems of the main problem. A third advantage is "reliability", which avoids a single point of failure in centralized systems [36].

Some work has been done in the clustering problem using MAS. A multi-agent approach to continuous online clustering of streaming data in a dynamic and distributed environment is proposed in [21]. A distributed clustering based on MAS for large dataset of documents is introduced in [30], to improve accuracy and relevancy in information retrieval. More on utilizing MAS in data clustering is given in [1, 7, 33].

Based on advantages of MAS, work has been done in image segmentation as well [9, 20, 27]. In addition, the MAS paradigm has also been utilized in CBIR. As an example, an agent-based CBIR system is proposed in [31], which is an interactive retrieval system that focus on using relevance feedback from the user, by means of using a particular user's image similarity preferences. Another multi-agent architecture for CBIR is presented in [11]. Each agent works independently and different agents work in parallel, to assess the similarity of the query image to each candidate image based on a specific similarity criterion. Finally, the partial results of different agents are integrated based on the user's selected voting scheme [11].

In this paper, we propose a multi-agent-based approach, multi-agents for fuzzy image clustering (MAFIC), which has the distinguished feature of distributing the computation of cluster centers and membership of the dataset among several independent parallel agents on different parts of the dataset. Consequently, the approach solves the problem of iterating computations in a much faster way. The rest of the paper is organized as follows. Section 3 introduces the FCM algorithm. The proposed MAFIC approach is presented in Sect. 4, and its implementation is elaborated on and explained in Sect. 5. A comparison of the results of implementing the proposed agent-based approach's performance against the sequential FCM is described and evaluated in Sect. 6. The MAFIC approach is contrasted with the parallel FCM approach in Sect. 7. Finally, a conclusion is presented in Sect. 8.

3 Sequential fuzzy *c*-means image clustering

FCM is based on the concept of fuzzy *c*-partition, so that it assigns different degrees of membership to each data point, creating fuzzy boundaries.

Let $X = \{x_1, x_2, \ldots, x_n\}$ be an image of n pixels, where $x_k (k = 1, 2, \ldots, n)$ is a pixel in the p-dimensional vector space $\mathbb{R}^p$, and c be the number of clusters, such that $2 \le c \le n$. Then, a fuzzy c-partition of the image X is represented by a real $c \times n$ matrix $U = [u_{ik}]$, where u_{ik} is the degree of membership of x_k in the ith cluster that satisfies the following conditions.

$$\left.\begin{array}{ll} u_{ik} \in [0,1], & \forall i, k, \\ 0 < \sum_{k=1}^{n} u_{ik} < n, & \forall i \in \{1, 2, \ldots, c\}, \\ \sum_{i=1}^{c} u_{ik} = 1, & \forall k \in \{1, 2, \ldots, n\}. \end{array}\right\} \tag{1}$$

FCM aims to find the optimal fuzzy c-partition that minimizes the objective function, $J(U, V; X)$, given by:

$$J(U, V; X) = \sum_{k=1}^{n} \sum_{i=1}^{c} u_{ik}^{m} \quad d^2(x_k, v_i), \tag{2}$$

where $V = \{v_1, \ldots, v_c\}$ is a set of cluster centers with $v_i \in \mathbb{R}^p$ indicating the center of cluster i (prototype), m is a weighting exponent on each fuzzy membership such that $m \in [1, \infty)$, $d^2(x_k, v_i) = ||x_k - v_i||^2$ is a distance measure between data point x_k and cluster center v_i, and $||.||$ is the

Euclidean norm. The objective function $J(U,V;X)$ is minimized via an iterative process, in which the degrees of membership u_{ik} and the cluster centers v_i are updated using the following equations:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left(\frac{d(x_k,v_i)}{d(x_k,v_j)}\right)^{\frac{2}{m-1}}}, \tag{3}$$

$$v_i = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m}. \tag{4}$$

FCM updates the membership matrix U and the cluster centers V by iteratively looping between Eqs. (3) and (4) until it converges (which consumes much of the time). The convergence occurs when the difference between two successive values of $J(U,V;X)$ is less than or equal a predefined threshold value (ξ), or a predefined number of iterations is reached. A multi-agent-based approach to improve the performance of FCM is proposed in this paper, by means of dividing an image to equally sized partitions (sub-images) and distributing the computation of u_{ik} and v_i among several agents that work simultaneously on a part of the image (sub-image).

4 MAFIC: multi-agents for fuzzy image clustering

In MAFIC, the dataset is a set of image pixels that is equally partitioned among the agents. Let the number of clustering agents be $b \times b$, so that the image is divided horizontally into b sub-images and vertically as well. Each agent handles part of the image different from other parts handled by other agents. The fuzzy c-partition using Eq. (3) and the cluster centers using Eq. (4) are computationally distributed among the agents, so that each agent calculates them only for its local data.

The suggested working procedure for the MAFIC approach consists of five different types of agents: a *Load* agent, a *Manager* agent, a *User* agent, a *Master Clustering* agent, and b^2 *Clustering* agents. Henceforth, these five types of agents are denoted as *Lagent*, *Magent*, *Uagent*, *MCagent*, and *Cagents*, respectively. The conceptual architecture of the multi-agent approach illustrating the proposed types of agents and their interactions is shown in Fig. 1. More on this issue is elaborated by explaining a usage scenario below, after a sequence that describes how the main ideas of the MAFIC approach generally work is listed first.

4.1 The MAFIC parallel clustering method

1. The *Uagent* initiates the interaction between a user and the MAS and gets as inputs the image's name and the clustering parameters: the number of clusters (c), the fuzziness value (m), a small positive threshold constant (ξ), and the maximum number of iterations, then sends a request to the *Magent*.
2. The *Magent* sends two requests: one to the *Lagent* to show the image and the other to the *MCagent* to start the process of clustering.
3. The *MCagent* divides the image into $b \times b$ sub-images. Then, it creates b^2 *Cagents* and sends requests to them.
4. *Cagent*$_l$ starts the clustering process on sub-image$_l$, where $l = 1,\ldots,b^2$. Each *Cagent*$_l$ then performs as follows.

   4.1. Set iteration number $t = 0$.

   4.2. Initialize randomly the fuzzy c-partition $U_l^{(0)}$ that satisfies the constraints in Eq. (1).

   4.3. Calculate cluster center $v_{il}^{(t)}$ on its sub-image$_l$ using Eq. (4).[1] Let the numerator and denominator of Eq. (4) for *Cagent*$_l$ be denoted by $N_{il}^{(t)} = \sum_{k=1}^{n_l} u_{ik}^m x_k$ and $D_{il}^{(t)} = \sum_{k=1}^{n_l} u_{ik}$, respectively, where n_l gives the number of pixels for sub-image$_l$. Thereafter, *Cagent*$_l$ sends a request to the *MCagent* to compute the global cluster centers V of the whole image, by means of collecting and combining the above-mentioned values using $\frac{\sum_{l=1}^{b^2} N_{il}^{(t)}}{\sum_{l=1}^{b^2} D_{il}^{(t)}}$, where $i = 1,\ldots,c$. Then, *MCagent* sends V to all *Cagent*$_l$, so that each *Cagent* $_l$ has the same cluster centers.

   4.4. Compute local objective function $J_l^{(t)}$ as in Eq. (2) using the values of V that are computed in step 4.3. Then, $J_l^{(t)}(U_l^{(t)}, V_l^{(t)}; X_l)$ values are collected and combined by the *MCagent* to compute the global objective function $J = \sum_{l=1}^{b^2} J_l^{(t)}$.

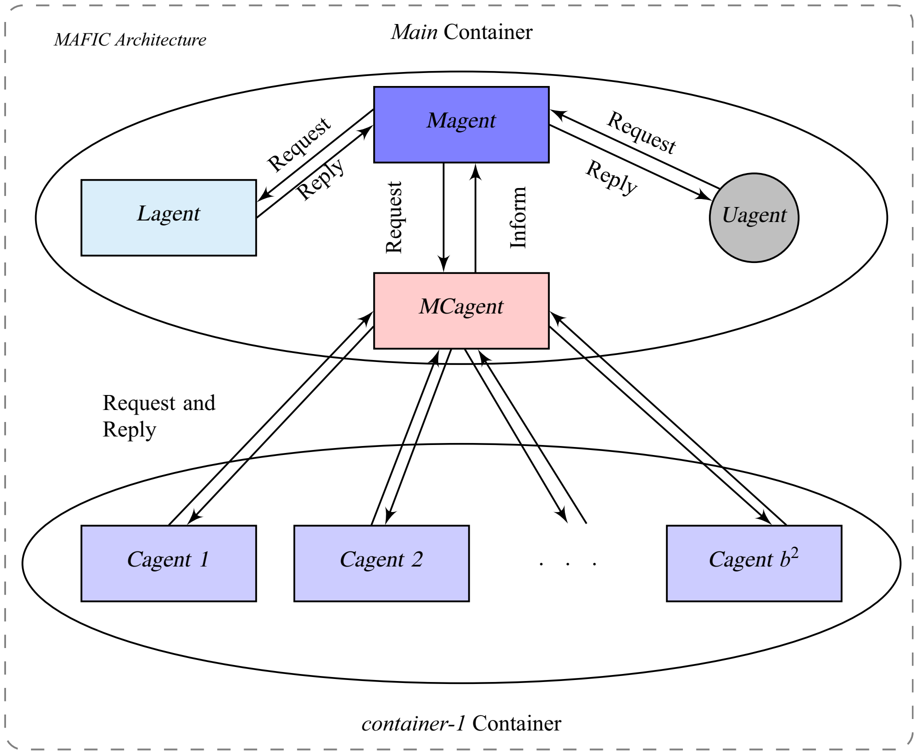
   4.5. Stop if the difference between two successive values of J is less than or equal a threshold (ξ), $|J^{(t+1)} - J^{(t)}| \leq \xi$, or the maximum number of iterations is reached. Otherwise, set $t = t + 1$ and compute a new fuzzy c-partition $U_l^{(t)}$ using the global cluster centers V and go to step 4.3.

   4.6. Use the membership function U_l of sub-image$_l$ and the global cluster centers V to get clustered sub-image$_l$.

   4.7. Send clustered sub-image$_l$ to the *MCagent*. Then, *Cagent*$_l$ is destructed after finishing its task.

[1] Note that superscript (t) indicates computations in iteration t.

Fig. 1 The conceptual architecture of MAFIC



5. The *MCagent* combines all clustered sub-images to get the total one. Then, it sends the final clustered image to the *Magent*.
6. The *Magent* sends a request to the *Lagent* for displaying the clustered image.
7. The *Magent* notifies the *Uagent* that the process of clustering is finished. The *Uagent* consequently, notifies the user.

4.2 The MAFIC usage scenario

The following items explain in more detail the exchange of messages between agents in a generic MAFIC usage scenario, and the flowchart in Fig. 2 fully describes the scenario from its start until it finishes:

- Once the MAS starts, the *Uagent*, *Magent*, *MCagent*, and *Lagent* are created.
- The *Uagent* connects a user to the MAS, by means of getting inputs and showing results from/to the user. At first, the *Uagent* checks that there is no empty value for any of the inputs. Then, it waits for the response from the *Magent*, which indicates that the process of clustering is successfully finished.
- The *Magent* manages the MAS. The received message from the *Uagent* must satisfy specific constraints, such as including the same content language[2] and ontology[3] of the *Magent*, to communicate effectively. These constraints determine the type of messages that the *Magent* can handle. At first, the *Magent* checks whether or not there is an image having that name.[4] If the *Magent* could not find the image, then it sends a "refuse message" to the *Uagent* indicating the image is not found. Otherwise, the *Magent* sends two requests, as shown in Fig. 2 and step 2 of the above method. Thereafter, it waits for the response from the *MCagent* (i.e., the result of the clustering process). When an "agree message" is received from the *MCagent*, the *Magent* immediately sends its own "agree message" to the *Uagent*. The *Magent* has to wait for an "inform message" that indicates that the clustered image is received. Meanwhile, the *Magent* receives an "inform message" from the *Lagent* indicating the image is successfully displayed to the user.
- The *MCagent* is responsible for managing the process of clustering. The *MCagent* starts dividing the image, as described in step 3 of the above method, by means of determining the x- and y-coordinates, and the size

[2] The content language indicates the syntax used to express the content.

[3] The ontology indicates the vocabulary of the symbols used in the content.

[4] Images are stored in a physical storage such as hard disk.

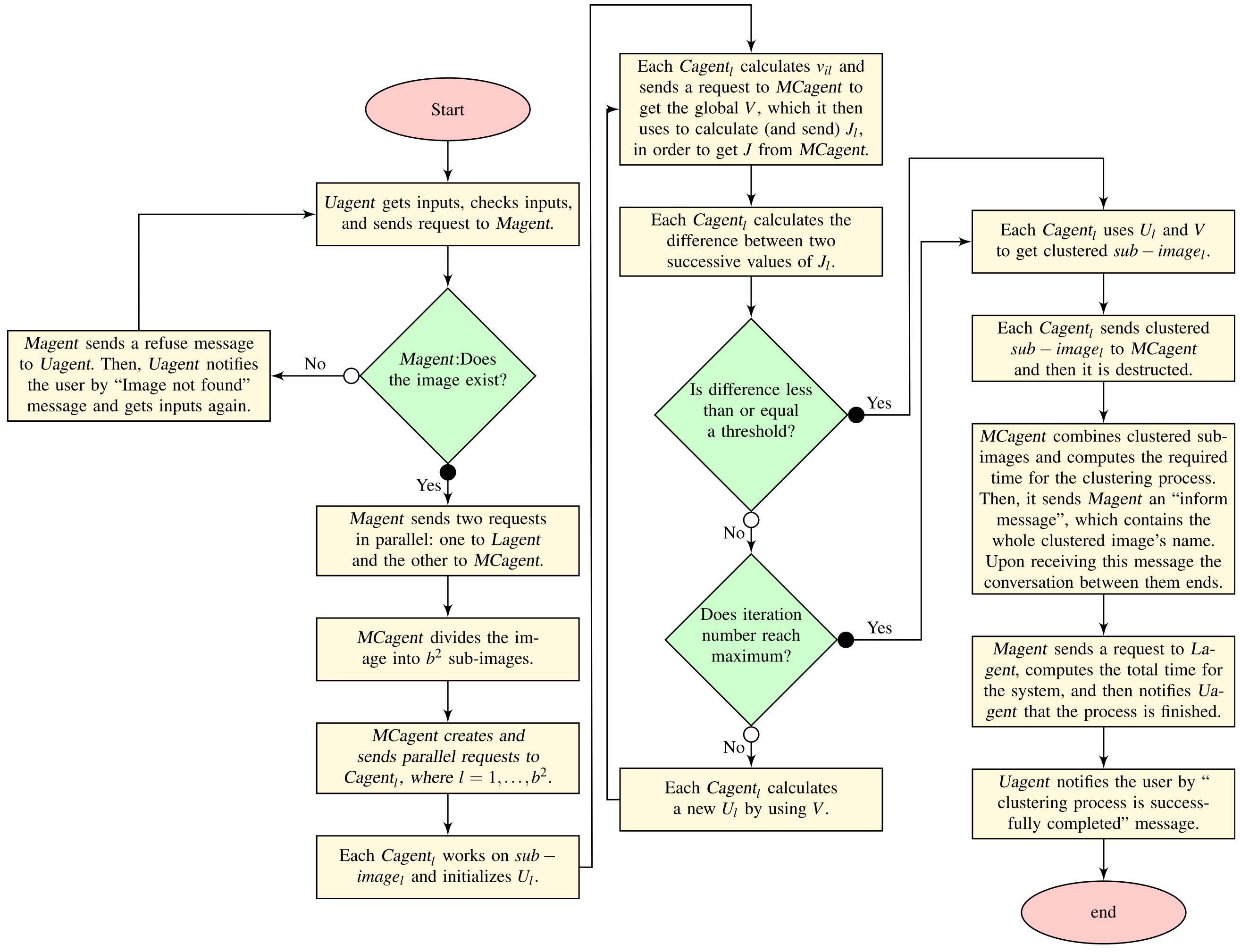

Start
Uagent gets inputs, checks inputs, and sends request to *Magent*.
Magent:Does the image exist?
No
Magent sends a refuse message to *Uagent*. Then, *Uagent* notifies the user by "Image not found" message and gets inputs again.
Yes
Magent sends two requests in parallel: one to *Lagent* and the other to *MCagent*.
MCagent divides the image into b^2 sub-images.
MCagent creates and sends parallel requests to $Cagent_l$, where $l = 1,\ldots,b^2$.
Each $Cagent_l$ works on $sub-image_l$ and initializes U_l.
Each $Cagent_l$ calculates v_{il} and sends a request to *MCagent* to get the global V, which it then uses to calculate (and send) J_l, in order to get J from *MCagent*.
Each $Cagent_l$ calculates the difference between two successive values of J_l.
Is difference less than or equal a threshold?
Yes
No
Does iteration number reach maximum?
Yes
No
Each $Cagent_l$ calculates a new U_l by using V.
Each $Cagent_l$ uses U_l and V to get clustered $sub-image_l$.
Each $Cagent_l$ sends clustered $sub-image_l$ to *MCagent* and then it is destructed.
MCagent combines clustered sub-images and computes the required time for the clustering process. Then, it sends *Magent* an "inform message", which contains the whole clustered image's name. Upon receiving this message the conversation between them ends.
Magent sends a request to *Lagent*, computes the total time for the system, and then notifies *Uagent* that the process is finished.
Uagent notifies the user by " clustering process is successfully completed" message.
end

Fig. 2 A generic usage scenario of the MAFIC approach

(i.e., width and height) of each sub-image. In case that the image's size is divisible by b, the sizes of sub-images are equal. Otherwise, the sub-images' sizes are not equal due to distributing the excess pixels among sub-images.

- Each $Cagent_l$ communicates with the *MCagent* in three cases, as clarified in steps 4.3., 4.4., and 4.7. of the above method (and Fig. 2).

5 JADE: a platform to implement MAFIC

An implementation of the proposed MAFIC approach has been made within the Java Agent DEvelopment Framework[5] (JADE) agent platform. JADE is one of the best modern agent environments that is completely written in Java, open source, and FIPA[6]-complaint [4]. The JADE framework simplifies the development of agent-based applications through the run-time environment (in which the agents can live); a library of classes that can be used to develop agents; and graphical tools that facilitate monitoring and debugging the activities of agents. For more detailed descriptions of the JADE architecture and developing agents with JADE, see [3]. The rest of this section only elaborates on the JADE characteristics that are necessary for the reader to follow on later discussions.

The JADE platform is composed of agent containers that can be distributed over a network. Agents live in containers, which are the Java processes that provide the JADE run-time and all the services needed for hosting and executing agents. There is a special container called the main container and every platform contains only one main container. It is the first container to be launched and all

[5] The online documentation is available at http://jade.tilab.com.

[6] FIPA: the Foundation for Intelligent Physical Agents.

other containers must join to a main container by registering with it. JADE provides a number of additional agents that are included in the main container by default when loading JADE: the agent management system agent responsible for managing and controlling the life cycle of other agents in the platform, and the directory facilitator agent that provides the yellow pages service to allow agents to register their capabilities and to allow other agents to identify appropriate agents they need.

In the suggested implementation, a JADE platform consists of two containers, which are running on one machine. The *Uagent*, the *Magent*, the *Lagent*, and the *MCagent* live on the *main container*. *Cagents* are created and live on the second container, which is called *container-1*. Both these containers and the agents that live on them are depicted in Fig. 1, in which the parallel arrows having opposite directions (depicted as single-headed arrows) indicate interactions between agents.

The implementation of the MAFIC approach here is based on several Java classes organized in the following categories:

- *Agent classes:* these classes are used for describing agent types. The MAFIC implementation utilizes the classes *UAgent*, *MAgent*, *LAgent*, *MCAgent*, and *CAgent* that implement *Uagent*, *Magent*, *Lagent*, *MCagent*, and *Cagents*, respectively. An agent is implemented in JADE by extending the provided Agent base class and overriding the default implementation of the methods `setup()` and `takeDown()`. These methods are automatically invoked by the platform during the agent life cycle. The `setup()` method is intended to include agent initialization, for instance starting the initial behaviors. The `takeDown()` method is invoked to do clean-up operations before the agent is destroyed.
- *Agent activity classes (behaviors):* the actual job (or jobs) an agent has to do is carried out within behaviors. A behavior is an abstraction that represents a task performed by an agent. MAFIC uses local classes for defining behaviors that describe the agent responses to FIPA messages, such as REQUEST and INFORM. A behavior is implemented in JADE by extending the provided Behavior abstract base class. The class Behavior is the root of a class hierarchy abstracting various agent behavior types. There is another type of behavior that is called a CompositeBehaviour, which is itself a behavior that embeds a number of child sub-behaviors. It can be considered a simple and clear approach to implement complex tasks in JADE. Three types of composite behaviors are provided: a SequentialBehaviour; an FSMBehaviour; and a ParallelBehaviour. The SequentialBehaviour schedules its children according to a very simple sequential policy. It starts with the first child and when this is finished, it moves to the second one and so on. The FSMBehaviour schedules its children according to a finite state machine whose states correspond to the FSMBehaviour children. Finally, the ParallelBehaviour schedules its children in parallel.MAFIC utilizes the FSMBehaviour to implement the clustering process in *Cagents*. The ParallelBehaviour has been used in *Magent*, when it sends a request to the *Lagent* to show the clustered image and computes the finish time of MAFIC in parallel.
- *Ontology classes:* these classes are necessary for implementing the agent communication semantics using concepts and relations. In other words, if agents need to communicate in a non-meaningless way, they have to share the same language and know exactly the meaning of vocabularies that are used in that language. This means that an ontology is required. An ontology describes the elements that can be used as content of an agent's messages. The ontology is composed of two parts, a vocabulary that describes the terminology of concepts used by agents in their space of communication and the terminology of relationships between these concepts. These parts describe messages' semantic and structure. In MAFIC, two simple ontologies are used: the *ImageOntology*, and the *clusteringOntology*. The ImageOntology defines the concept of the image and its related actions, such as the *LoadImage* action, which is used by the *Magent* to the *Lagent*, to show the image. The ClusteringOntology defines the concept of clustering parameters, which are the number of clusters, the fuzziness, the maximum number of iterations, and the threshold. In addition, some related actions are included, such as *ApplyClustering* by the *Magent* to the *MCagent*, to start the clustering process.

Agents need to communicate with each other, to perform their tasks. The communication paradigm in JADE is based on asynchronous message passing. The structure of messages in JADE is complaint with the structure of messages in FIPA-Agent Communication Language (ACL), which is one of the popular languages that have been developed in MASs [13]. The major aim of FIPA-ACL is to provide the standard communication language that can be used by autonomous agents. The FIPA-ACL provides an outer language, which defines the structure of the message that is totally separate from content of message. In the outer language, FIPA-ACL provides a set of performatives (or communicative acts), to differentiate between the different types of messages that can be passed between agents.In addition, formal semantics is supported which should eliminate ambiguity and confusion.

6 Experimental results and discussion

The performance of the MAFIC implementation discussed above is compared with that of the sequential FCM, in terms of time cost. Both approaches were implemented on a single machine with 4 GBytes main memory, 3 MBytes cache memory, and core i5 2.67 GHz CPU with four (active) cores. The images used in the evaluation are two sets of 24-bit RGB images of different sizes. Each image is divided horizontally into four sub-images and vertically as well. Consequently, the number of clustering agents (b^2) is 16. The following parameters are kept fixed: the maximum number of iterations is 100; the exponent weight (m) is 2; and the threshold value (ξ) is 0.001.

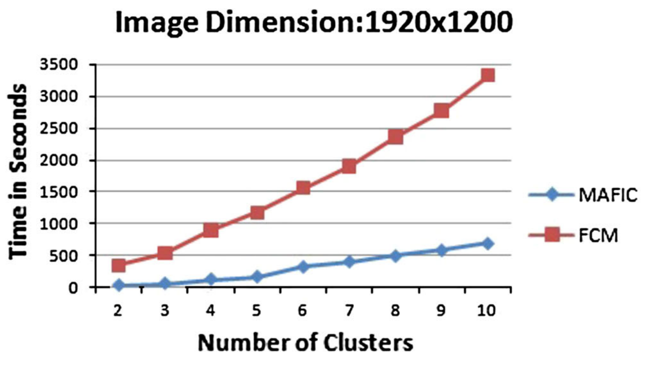
The MAFIC and sequential FCM (simply FCM) performances were measured for each image by increasing the number of clusters and computing the time cost. MAFIC and FCM were run with each image in 5 trials and the computation times of MAFIC and FCM were measured in terms of the mean of the computing time.

Figure 3 gives comparisons between performance times required to cluster the images of the first set using both FCM and MAFIC. Figure 4 gives comparisons between the speedup percentage of MAFIC for the four images.

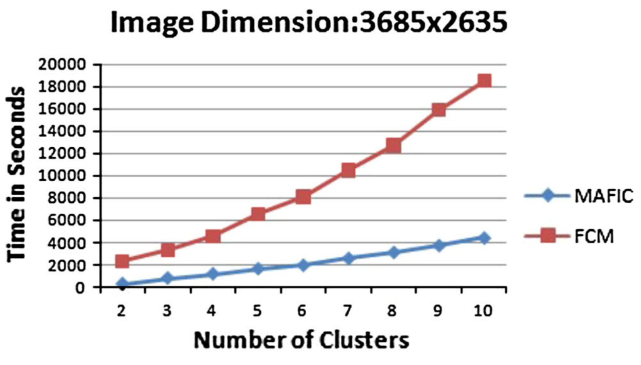
The first image is a synthetic image with size 1,920 × 1,200, and its computation times using MAFIC and FCM are shown in Fig. 3a. It can be seen that the best performance of MAFIC for this image is achieved when the number of clusters is 2, where the performance of MAFIC is 12 times faster than the performance of FCM. Furthermore, the best convergence rate is achieved when the average number of iterations is 28. For 3 clusters, the average number of iterations is 39 and the performance of MAFIC is 9 times better than FCM's performance. In case of 4 and 5 clusters the performance of MAFIC is 7 times better than the performance of FCM and the average number of iterations is 62. The convergence rate is 100 for clusters more than 5.

The second image is a satellite image with size 3,685 × 2,635, and the comparison of its computation times is given in Fig. 3b. The best convergence rate using MAFIC is 52, which is achieved in case of 2 clusters. Furthermore, the performance of MAFIC is eight times faster than the performance of FCM.

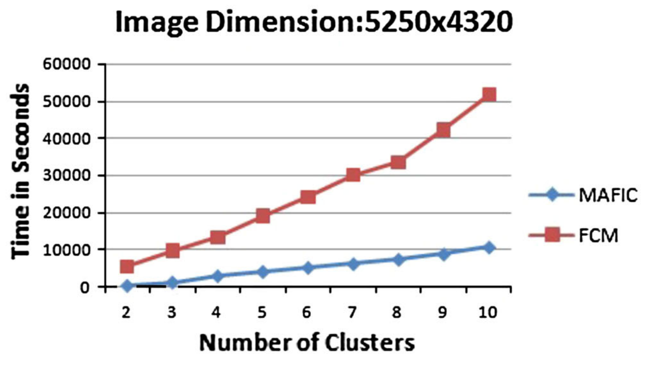
The performance time comparisons for the third image, which is a satellite image with size 5,250 × 4,320, are given in Fig. 3c. The best convergence rate for this image using MAFIC is achieved for 2 and 3 clusters, with average



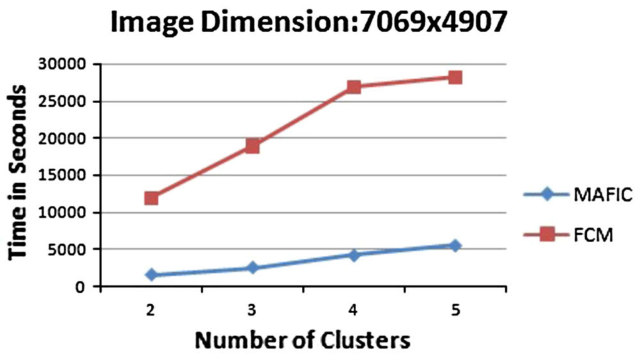
(a) Image dimension 1920 × 1200



(b) Image dimension 3685 × 2635



(c) Image dimension 5250 × 4320



(d) Image dimension 7069 × 4907

Fig. 3 The times comparison for FCM and MAFIC

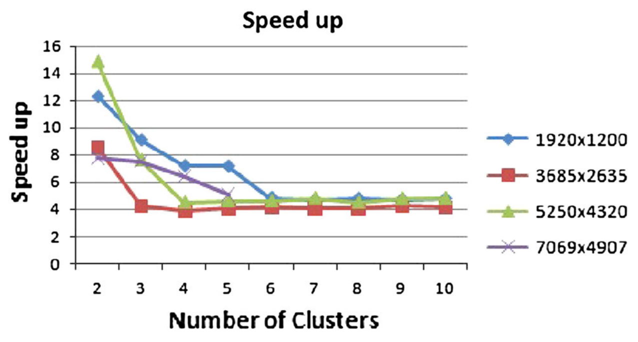


Fig. 4 The relation between MAFIC speedup of execution time and number of clusters

numbers of iterations 32 and 68, respectively. The performance of MAFIC in case of 2 and 3 clusters are 14 and 7 times, respectively, better than the performance of FCM.

The fourth image is a satellite image with size 7,069 × 4,907, and has been tested only using 2–5 clusters. Comparisons based on time using MAFIC and FCM are presented in Fig. 3d. The average numbers of iterations using MAFIC are 77 and 82, for 2 and 3 clusters, respectively. The performance of MAFIC for two and three clusters is similar and is at least seven times better than the performance of FCM. The performance of MAFIC in case of four and five clusters are six and five times, respectively, better than the performance of FCM. Figure 5 shows the visual results of image clustering that can be obtained using MAFIC.

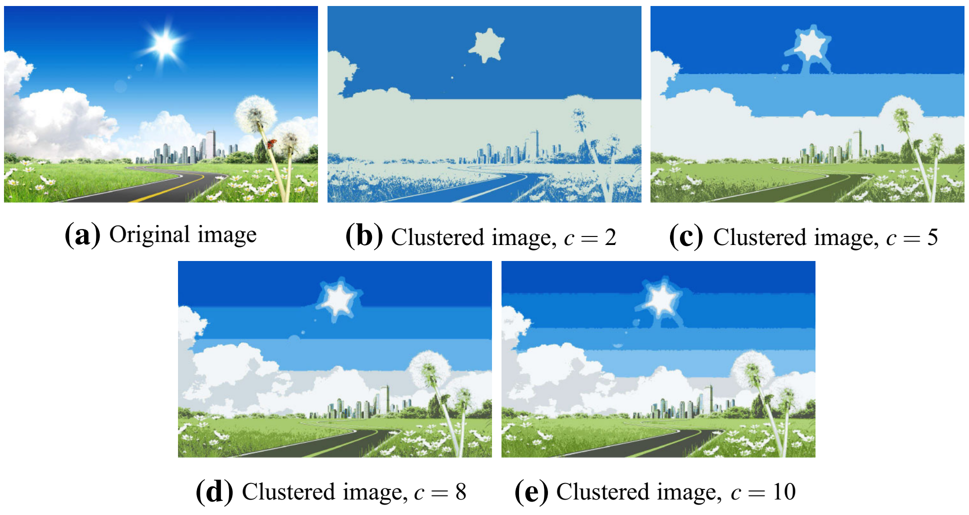
Table 1 presents percentages that reflect the speeding up of MAFIC (over FCM) for three of the four tested images. As can be seen, the best speedup percentage of MAFIC is always achieved when the number of clusters is <4. For two clusters in particular, MAFIC always has a convergence rate that is less than the maximum number of iterations for the four images. For clusters more than 4, speedup percentages of MAFIC are very close to each other. As also can be seen from Table 1, the performance of MAFIC is at least four times better than the performance of FCM for clusters more than 4. The MAFIC approach, therefore, outperforms the sequential FCM under the given settings.

Another set of (synthetic) images of different sizes were used for the sake of measuring the scalability of the MAFIC approach with respect to changing the number of used cores. The performances of MAFIC and FCM were measured for each image in this set by increasing the number of clusters while varying the number of cores from 1 to 4, then computing the time cost.

Figure 6 gives comparisons between performance times required to cluster the second set of tested images using both MAFIC and FCM. Figure 7 gives comparisons between the speedup percentages of MAFIC using different numbers of cores. Each of the following parameters is kept fixed for Figs. 6 and 7: the maximum number of iterations is 100; m is 2; c is 8; and ξ is 0.001. Generally, the performance of MAFIC is improved by increasing the number of cores. It can be seen in Fig. 6 that the best performance of MAFIC for this new set of images is achieved when the number of cores is 4. In contrast to MAFIC, varying the number of cores does not seem to considerably affect the performance of FCM. In addition, by comparing Figs. 4 and 7, it can be seen that the performance of MAFIC for larger images is better than its performance for smaller images using the same number of cores, which is 4 in our case.

Figure 8 gives comparisons between the processing times of MAFIC for the second set of images in case of eight clusters. It can be observed that the processing time of MAFIC is proportional to the size of images, and that

Fig. 5 Clustering obtained using MAFIC


(a) Original image
(b) Clustered image, $c = 2$
(c) Clustered image, $c = 5$
(d) Clustered image, $c = 8$
(e) Clustered image, $c = 10$

the approach generally reduces the time cost for the tested images with the different sizes. The speedup percentages of MAFIC for images of larger sizes, in particular, are better. Figure 9 compares the processing times of MAFIC when applied to an image with size 640 × 374, while varying the number c of clusters to 2, 3, 4, 5, 6, 7, 8, 9, and 10. It can be observed that the processing time of MAFIC is proportional to the number of clusters. Figures 8 and 9 both show that the MAFIC approach seems to have the same performance rate with respect to the size of images and the number of clusters using different numbers of cores that range from 1 to 4.

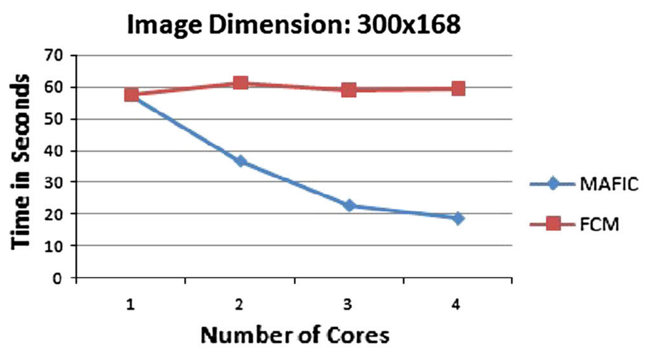
The discussions make it evident that the performance of MAFIC provides a strong speedup that is at least four times better than the sequential FCM. This is true even when the number of clusters exceeds 4 and in cases when the convergence rate reaches the maximum number of iterations (which is 100). Furthermore, it is worth noting that the experimental results of MAFIC as a multi-agent approach are in agreement with results given by Kelash et al. [19], for instance, who also proposed to use the distribution of agents as a cluster computing paradigm for multi-agent applications that employ the interactions among computing entities [19]. Finally, from the preliminary results, it is concluded that MAFIC can speedup the overall computation time for fuzzy image clustering.

Table 1 The speedup of MAFIC

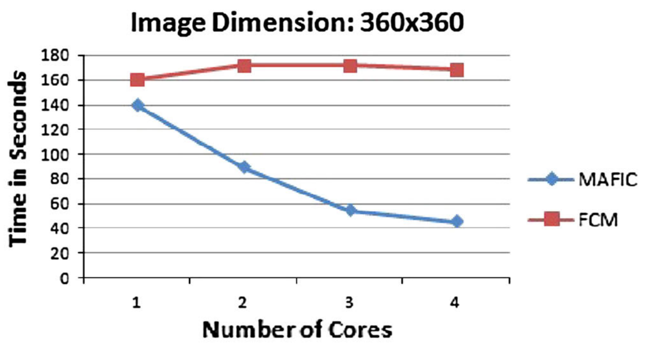
No. of clusters	Image size		
	1,920 × 1,200	3,685 × 2,635	5,250 × 4,320
2	12.35	8.54	14.86
3	9.11	4.22	7.63
4	7.19	3.86	4.53
5	7.17	4.01	4.64
6	4.86	4.17	4.64
7	4.71	4.09	4.86
8	4.79	4.08	4.55
9	4.73	4.26	4.82
10	4.79	4.17	4.85

7 Comparison with parallel FCM approaches

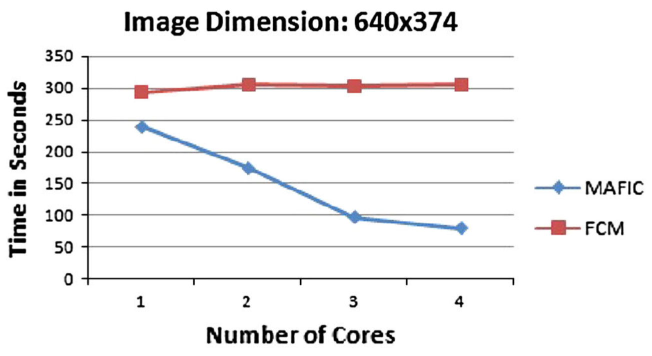
The previous section compares the results of implementing the proposed approach with the classical version of FCM from a statistical analysis perspective. This provides an initial proof of concept that our approach is more feasible



(a) Image dimension 300 × 168



(b) Image dimension 360 × 360



(c) Image dimension 640 × 374

Fig. 6 The times comparison for FCM and MAFIC using different numbers of cores

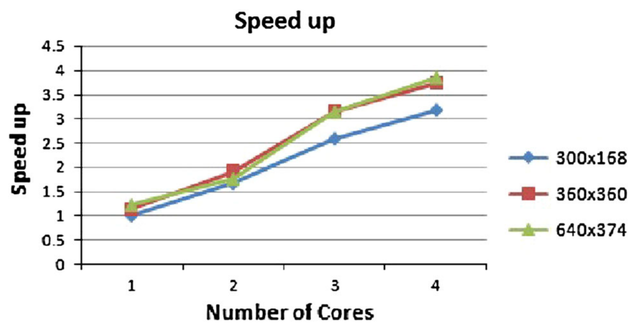


Fig. 7 The relation between MAFIC speedup and number of cores

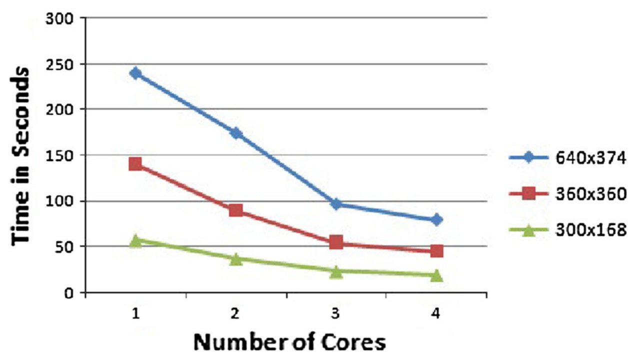


Fig. 8 Time measurements for the second set of images with respect to the number of cores

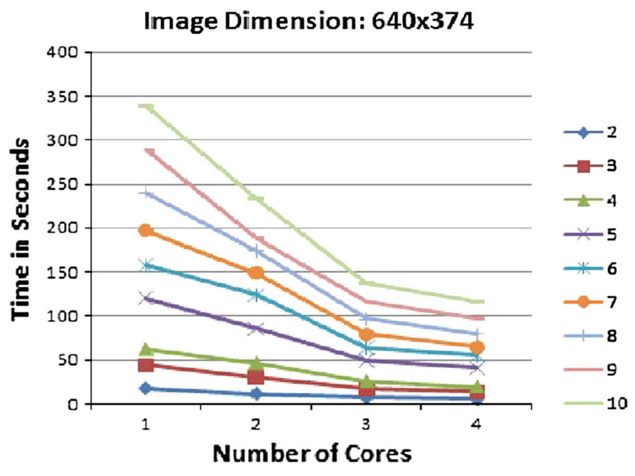


Fig. 9 Time measurements for different numbers of clusters

by evaluating aspects of experimental performance. The current section contrasts the MAFIC approach with the parallel approach of FCM, which uses parallel processors to reduce the computational load. In addition, it suggests possibilities of adopting the MAFIC approach to multiple distributed parallel processors. The section thus highlights aspects in which the MAFIC approach is different from other approaches. It further discusses and evaluates the MAFIC's performance in comparison with the parallel FCM approach. The latter approach is undertaken by several researchers in the literature, but we restrict the detailed discussions in this section to [22, 24, 28].

The parallel version of FCM given in [22] is designed to run on parallel computers of the single program multiple data (SPMD) model incorporating message passing. The proposed parallel algorithm is implemented on an AlphaServer computing cluster with a total of 128 processors and 64 GBytes of main memory. There are three tests that were executed to evaluate the performance of this parallel FCM. The evaluation has been done on five different sizes of a dataset, which are $n = 2^9$, 2^{11}, 2^{13}, 2^{15}, and 2^{17}. The first test measures the speedup percentages and compares them to the *ideal speedup* (cf. [22, pp. 371]), where the results show that the speedup percentages for large values of n are almost ideal. For small values of n, speedup percentages deteriorate. The second test measures the performance with respect to the number of clusters. The dataset with size $n = 2^{17}$ is used while the number of clusters is changed with $c = 2$, 4, 8, 16, and 32. The results show that the speedup behavior of the parallel FCM algorithm in [22] is not sensitive to the number of desired clusters. Since speedup percentages are almost ideal for small values of c, and close to ideal for large values of c. The third test measures the performance when dealing with larger datasets that use more processors. The results show that an increase in the size of the dataset can almost always be balanced by a proportional increase in the number of processors used.

In contrast to [22], at least the following aspects differentiate the MAFIC approach. The different dataset sizes that were used are larger. Regarding the ideal speedup in MAFIC, Figs. 4 and 7 already show that datasets with large sizes get better speedup percentages than datasets with small sizes. It can be seen from Fig. 4 that MAFIC achieves a speedup percentage of more than 4 for clusters more than 4 using a quad-core processor. For a small size dataset, a speedup percentage of more than 3 is achieved for clusters more than 6 using a quad-core processor. Regarding the performance of MAFIC with different numbers of clusters, a speedup percentage of more than 8 for small values of c is achieved. For large values of c, a speedup percentage of more than 4 is achieved. Both speedup percentages are on a quad-core processor using large size dataset.

In [24], an implementation of a parallel FCM cluster analysis tool calculates not only clusters' centers but also the optimal number of clusters for a given dataset. There are several types of tests that were executed, and applied on two machines: the PC Cluster Mercury that has 16 dual pentium 3, 1GHz processor, and 8 GBytes of total memory; and the SGI Altix 350 machine that has 14 Intel Itanium 2 CPUs with 28 GBytes of RAM and 360 GBytes of disk storage [24]. The given dataset consists of a collection of records and variables, where a record can be seen to correspond to a pixel $x_k \in \mathbb{R}^p$ of an image of n pixels, $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^p$, and variables correspond to the dimension p. The various tests changed both the number of records and the number of variables for a given dataset. But since pixels in MAFIC's sense correspond to records with a fixed number of variables, only two tests may be related to fuzzy clustering of images.

One of the tests measures the processing times for different datasets when the number of records increases. The results show that the processing time grows when the number of records grows. Another test measures the speedup percentages of datasets with different numbers of records (and fixed number of variables) for two clusters. The results show that the dataset with smaller number of records get smaller speedup percentages using different numbers of processors varying from 1 to 6.

The MAFIC approach is not concerned with measuring the performance on a range of clusters to determine the best partition for a given dataset. Furthermore, if one lets the number of records correspond to the number of pixels and the number of variables to the pixels' dimension, some of the used dataset sizes in MAFIC are larger than the ones used in the parallel algorithm of [24]. The larger the size of the dataset, the larger the speedup percentages that results by MAFIC. By comparing the performances of both MAFIC and the parallel approach for two clusters, it can be observed that speedup percentages with values between 1.94 and 14.86 result when we use all the tested datasets (images) of MAFIC on a quad-core processor. By comparing the relative sizes of datasets in both MAFIC and [24], it can also be found that the performance of MAFIC is better than the performance of the parallel approach for two clusters. The processing time of MAFIC is analyzed with respect to the number of used cores. The analysis is in an agreement with the parallel algorithm, where the processing time is proportional to the size of a dataset. That is, MAFIC has a fixed rate for all different sizes, as shown in Fig. 8. In addition, MAFIC reduces the processing time for all datasets using a quad-core processor.

The parallel FCM algorithm in [28] is designed for an Open Source Cluster Application Resource cluster with nine nodes using SPMD model and MPI. Three images of different sizes are used to measure the performance of the parallel FCM algorithm. All images are segmented to a black and a white cluster. The size of images has been reduced by a factor of three, where RGB is converted to gray scale. The first image is 24-bit RGB image with size 270×200. The speedup percentage for two processors was 1.1, and for seven processors the speedup percentage has been increased by no more than 2.2. By increasing the number of processors the speedup percentage decreases again. The size of the second image is 800×600. The parallel FCM implementation took about 1.55 s with nine processors which gives a speedup percentage of 8. The third image has size 2,580 $\times$ 1,720. The parallel FCM took 12.90 s which achieves a speedup percentage of 8.97.

By comparing the two implementations of MAFIC and [28], one notices the following. MAFIC has been tested on images of large sizes, whereas the parallel FCM has been tested on images with small sizes. The experimental results in [28] did not measure how well the parallel FCM performs in speedup when the number of clusters increases. The results are restricted to a black and a white cluster. The scalability of the number of clusters has been tested in MAFIC (see Figs. 4, 9). MAFIC achieved a better speedup percentage than the parallel FCM for large image sizes in case of two clusters using a quad-core processor.

By contrasting the MAFIC approach with the parallel approaches of FCM, one may conclude that the MAFIC approach shares an essential similarity with the SPMD parallel model implemented on a multi-core processor. The b^2 *Cagents* do exactly the same task on different parts of an image. Thus, a parallel implementation of the proposed method can be adopted to multiple distributed parallel processors.

Two of the basic forms of multiprocessor systems are parallel systems and distributed systems which can be used with some overlap. A parallel system is usually one in which the processors are closely connected, whereas a distributed system is one in which the processors are independent of each other [5]. The MAFIC approach can be adopted to a multiple distributed processors-based parallel system. One possibility would be to distribute b^2 *Cagents* among b distributed parallel processors connected via a network. Each one of the b processors can be a multi-core processor and handles b *Cagents*. The communications between the b^2 *Cagents* can be done through the *MCagent*, as described earlier in the MAFIC method (see Sect. 4), where the *MCagent* will be on one of the b processors. In this case, b^2 *Cagents* communicate by accessing only one processor on which the *MCagent* resides. As the main concern in distributed systems is the communications between multiple processors, the communications between multiple distributed parallel processors can be improved by dividing the communications between b processors into pairs. In this case, one processor of each pair collects and combines its partner's values. Each one of the b processors has b *Cagents* and one of them acts as the *MCagent* in the MAFIC method. In other words, b*Cagents* communicate with each other through one of them that is called the master. The latter does the tasks of the *MCagent* as described in steps 4.3., 4.4., and 5 of the MAFIC method (see Sect. 4).

An alternative solution would be to distribute the b^2 *Cagents* among the b^2 distributed parallel processors, where each *Cagent* resides on one processor. The communications between the b^2 *Cagents* can be done through the *MCagent*, as described in the MAFIC method. In such case, the *MCagent* resides on a processor which is different from the other b^2 processors. The communications can also be done in pairs. In this solution, one needs to pay more attention to the communications between multiple distributed processors. The more *Cagents* we have, the more processors we need. This can lead to more communications, which is not a desired property. The cost of communications exceeds the cost of computation itself. Thus, the number of *Cagents* should be limited. One of the benefits of using multiple distributed processors is that each processor has an independent memory. This increases the capacity of the used main memory and reduces the access to a secondary storage. Therefore, datasets with very large sizes can easily be tested.

8 Conclusion

In this paper, MAFIC has been introduced as a new multi-agent-based fuzzy *c*-means image clustering approach. An experimental perspective is undertaken to thoroughly discuss preliminary analyses and evaluations of the approach. One implemented feature of MAFIC is distributing the computation of membership function and cluster centers among several interacting agents. Thus, it improves the performance of the sequential FCM algorithm, which is one of the most widely used fuzzy clustering methods.An implementation of MAFIC has been tested on two sets of 24-bit RGB images, and the experimental results show that it outperforms FCM in terms of the time needed for the clustering process. MAFIC is at least four times faster than FCM for clusters more than or equal 4, in particular, for the largest four images that have been tested. In addition, it is obvious that the multi-agent-based approach utilizes desirable features for the MAFIC approach, such as efficiency in computation, modularity, scalability, reusability and distributed computation.

The proposed approach can be applied to the hard clustering algorithm (*k*-means), so as to speedup the computation time. As indicated in [8, 23, 26], FCM can be used to achieve a higher retrieval performance for CBIR systems. The MAFIC algorithm can be efficiently used for applications, such as image segmentation and CBIR. The process of fast archiving of clusters' features of images is an important step for CBIR, and needs to be further studied in future work.

Acknowledgments The authors would like to thank the anonymous reviewers for their precious comments and suggestions which greatly helped in improving the presentation quality of the manuscript.

References

1. Agogino, A., Tumer, K.: Efficient agent-based cluster ensembles. In: Proceedings of 5th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'06), pp. 1079–1086. ACM (2006)
2. Al-Zoubi, M.B., Hudaib, A., Al-Shboul, B.: A fast fuzzy clustering algorithm. In: Proceedings of 6th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'07), vol. 6, pp. 28–32. World Scientific and Engineering Academy and Society (WSEAS) (2007)
3. Bellifemine, F., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley, New York (2007)
4. Bellifemine, F., Poggi, A., Rimassa, G.: JADE—a FIPA-compliant agent framework. Tech. rep., Telecom Italia Internal (1999). http://jade.cselt.it/papers/PAAM.pdf (2014). Accessed 28 Nov 2014
5. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice-Hall, Inc., USA (1989)
6. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: the fuzzy *c*-means clustering algorithm. Comput. Geosci. **10**(2–3), 191–203 (1984)
7. Chaimontree, S., Atkinson, K., Conenen, F.: A Multi-Agent Based Approach to Clustering: Harnessing the Power of Agents. Agents and Data Mining Interactions. Lecture Notes in Computer Science, vol. 7103, pp. 16–29. Springer, Berlin (2012)
8. Chen, Y., Wang, J.Z., Krovetz, R.: CLUE: cluster-based retrieval of images by unsupervised learning. IEEE Trans. Image Process. **14**(8), 1187–1201 (2005)
9. Chitsaz, M., Seng, W.: Medical image segmentation using a multi-agent system approach. Int. Arab J. Inf. Technol. **10**(3), 222–229 (2013)
10. Chuang, K.S., Tzeng, H.L., Chen, S., Wu, J., Chen, T.J.: Fuzzy *c*-means clustering with spatial information for image segmentation. Comput. Med. Imaging Graph. **30**, 9–16 (2006)
11. Dimitriadis, S., Marias, K., Orphanoudakis, S.C.: A multiagent platform for content based image retrieval. Multimed. Tools Appl. **33**(1), 57–72 (2007)
12. Eschrich, S., Ke, J., Hall, L.O., Goldgof, D.B.: Fast accurate fuzzy clustering through data reduction. IEEE Trans. Fuzzy Syst. **11**(2), 262–270 (2003)
13. FIPA: FIPA. The Foundation for Intelligent Physical Agents (1999). http://www.fipa.org (2014). Accessed 28 Nov 2014
14. Gen-yuan, D., Fang, M., Sheng-li, T., Xi-rong, G.: Remote sensing image sequence segmentation based on the modified fuzzy *c*-means. J. Softw. **5**(1), 28–35 (2010)
15. Ghosh, S., Dubey, S.K.: Comparative analysis of *k*-means and fuzzy *c*-means algorithms. Int. J. Adv. Comput. Sci. Appl. **4**(4), 35–39 (2013)
16. HongLei, Y., JunHuan1, P., BaiRu, X., DingXuan1, Z.: Remote sensing classification using fuzzy *c*-means clustering with spatial constraints based on markov random field. Eur. J. Remote Sens. **46**, 305–316 (2013)
17. Hung, M.C., Yang, D.L.: An Efficient fuzzy *c*-means clustering algorithm. In: Proceedings of IEEE International Conference on Data Mining (ICDM'01), pp. 225–232. IEEE Computer Society (2001)
18. Imianvan, A.A., Obi, J.C.: Fuzzy cluster means expert system for the diagnosis of tuberculosis. Glob. J. Comput. Sci. Technol. **11**(6), 41–48 (2011)

19. Kelash, H., El_Dein, M.Z.G., Kamel, N.: Agent distribution based systems for parallel image processing. In: Proceedings of 1st International Conference on Graphics Vision and Image Processing (GVIP'05), vol. 05, pp. 59–64. ICGST, Cairo (2005)
20. Keshtkar, F., Gueaieb, W., White, A.: An agent-based model for image segmentation. In: Proceedings of 13th Multi-disciplinary Iranian Researchers Conference in Europe. Leeds, UK (2005)
21. Kiselev, I., Alhajj, R.: Self-organizing multi-agent system for adaptive continuous unsupervised learning in complex uncertain environments. In: Proceedings of 23rd National Conference on Artificial intelligence (AAAI'08), vol. 3, pp. 1808–1809. AAAI Press (2008)
22. Kwok, T., Smith, K., Lazano, S., Taniar, D.: Parallel fuzzy *c*-means clustering for large data sets. In: Euro-Par Proceedings of 8th International Conference on Parallel Processing. Lecture Notes in Computer cience, vol. 2400, pp. 365–374. Springer, New York (2002)
23. Lotfy, H.M., Elmaghraby, A.S.: A novel cluster-based image retrieval. In: Proceedings of the 4th IEEE International Symposium on Signal Processing and Information Technology, pp. 338–341 (2004)
24. Modenesi, M.V., Costa, M.C.A., Evsukoff, A.G., Ebecken, N.F.F.: Parallel fuzzy *c*-means cluster analysis. In: VECPAR Proceedings of 7th International Conference on High Performance Computing for Computional Science. Lecture Notes in Computer Science, vol. 4395, pp. 52–65. Springer, New York (2006)
25. Mohamed, N.A., Ahmed, M.N., Farag, A.: Modified fuzzy *c*-mean in medical image segmentation. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 6, pp. 3429–3432 (1999)
26. Ooi, W.S., Lim, C.P.: A fuzzy clustering approach to content-based image retrieval. In: Proceedings of Post ICONIP'09 Workshop on Advances in Intelligent Computing, pp. 11–16, Kuala Lumpur (2009)
27. Pooja, Srivastava, N., Shukla, K.K., Singhal, A.: Agent based image segmentation methods: a review. Int. J. Comput. Technol. Appl. **2**(3), 704–708 (2011)
28. Rahimi, S., Zargham, M., Thakre, A., Chhillar, D.: A parallel fuzzy *c*-mean algorithm for image segmentation. In: IEEE Annual Meeting of the Fuzzy Information Processing Society (NAFIPS'04), vol. 1, pp. 234–237. IEEE (2004)
29. Reddi, K.: Integrating fuzzy *c*-means clustering technique with *k*-means clustering technique for CBIR. Int. J. Comput. Distrib. Syst. **3**(3), 1–7 (2013)
30. Reed, J.W., Potok, T.E., Patton, R.M.: A multi-agent system for distributed cluster analysis. In: Proceedings of 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'04), W16L Workshop-26th International Conference on Software Engineering, pp. 152–155 (2004)
31. Saha, S., Sen, S.: Agent based framework for content based image retrieval. In: Proceedings of AAAI Spring Symposium on Interaction Between Humans and Autonomous Systems over Extended Operation. Stanford University, USA (2004)
32. Shambharkar, S., Tirpude, S.: Fuzzy *c*-means clustering for content based image retrieval system. In: International Conference on Advancements in Information Technology with Workshop of ICBMG'11, International Proceedings of Computer Science and Information Technology IPCSIT, vol. 20, pp. 148–152 (2011)
33. da Silva, J.C., Klusch, M., Lodi, S., Moro, G.: Privacy-preserving agent-based distributed data clustering. Web Intell Agent Syst **4**(2), 221–238 (2006)
34. Suganya, R., Shanthi, R.: Fuzzy *c*-means algorithm—a review. Int. J. Sci. Res. Publ. **2**(11) (2012)
35. Wang, X.Y., Garibaldi, J., Ozen, T.: Application of the fuzzy *c*-means clustering method on the analysis of non pre-processed FTIR data for cancer diagnosis. In: Proceedings of 8th Australian and New Zealand Conference on Intelligent Information Systems, pp. 233–238 (2003)
36. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, USA (1999)
37. Yang, J., Watada, J.: Fuzzy clustering analysis of data mining: application to an accident mining system. Int. J. Innov. Comput. Inf. Control **8**(8), 5715–5724 (2012)
38. Yang, M.S.: A survey of fuzzy clustering. Math. Comput. Model. **18**(11), 1–16 (1993)
39. Yang, Y., Huang, S.: Image segmentation by fuzzy *c*-means clustering algorithm with a novel penalty term. Comput. Inform. **26**, 17–31 (2007)
40. Yang, Y., Zheng, C., Lin, P.: Fuzzy *c*-means clustering algorithm with a novel penalty term for image segmentation. Opto-Electron. Rev. **13**(4), 309–315 (2005)

Nashwa M. Abdelghaffar is a research and teaching assistant at Ain Shams University, Faculty of Science, Mathematics Department, Computer Science Subdivision. She received her B.Sc. in Pure Mathematics and Computer Science from Ain Shams University and her research interests include artificial intelligence, multi-agent systems, and image processing.

Hewayda M. S. Lotfy is a Lecturer in Ain Shams University, Faculty of Science, Mathematics Department, Computer Science Subdivision. She earned her PhD in Image Retrieval from the USA and her research interests are Multi-Agent Systems, Data and Web Mining and Image Processing.

Soheir M. Khamis is a Professor in Ain Shams University, Faculty of Science, Mathematics Department, Computer Science Subdivision. She earned her PhD in Enumerative Combinatorics from Ain Shams University and her research interests are Combinatorial algorithms, Graph theory and Combinatorics.