Parallel Computing 37 (2011) 230–243

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Parallel c-means algorithm for image segmentation on a reconfigurable mesh computer

Omar Bouattane [a,*], Bouchaib Cherradi [b], Mohamed Youssfi [c], Mohamed O. Bensalah [c]

[a] E.N.S.E.T, Bd Hassan II, BP 159, Mohammedia, Morocco
[b] Faculté des Sciences et Technique, Bd Hassan II, Mohammedia, Morocco
[c] Faculté des Sciences, Université Mohamed V Agdal, Rabat, Morocco

ARTICLE INFO

Article history:
Received 10 April 2009
Received in revised form 24 August 2010
Accepted 2 March 2011
Available online 9 March 2011

Keywords:
Image segmentation
Classification
MRI image
Parallel algorithm
c-means

ABSTRACT

In this paper, we propose a parallel algorithm for data classification, and its application for Magnetic Resonance Images (MRI) segmentation. The studied classification method is the well-known c-means method. The use of the parallel architecture in the classification domain is introduced in order to improve the complexities of the corresponding algorithms, so that they will be considered as a pre-processing procedure. The proposed algorithm is assigned to be implemented on a parallel machine, which is the reconfigurable mesh computer (RMC). The image of size ($m \times n$) to be processed must be stored on the RMC of the same size, one pixel per processing element (PE).

1. Introduction

Image segmentation is a splitting process of images into a set of regions, classes or homogeneous sub-sets according to some criteria. Usually, gray levels, texture or shapes constitute the well-used segmenting criteria. Their choice is frequently based on the kind of images and the goals to be reached after processing. Image segmentation can be considered as an image processing problem or a pattern recognition one.

In the case of image processing consideration, we distinguish two essential approaches that are: region approach and contour approach. In the first approach, we look for homogeneous regions using some basic techniques such as thresholding, region growing, morphological mathematics and others [1]. Thresholding technique discriminates pixels using their gray levels. It supposes implicitly that intensity structures are sufficiently discriminative, and guarantee good separation [2]. In [3] the authors propose a multi-modal histogram thresholding method for gray-level images.

Region growing technique is based on progressive aggregations of pixels starting from a given point named "germ". In [4] the authors propose a region growing method for cervical 3-D MRI image segmentation. The proposed algorithm combines the automatic selection of germs with a growing regions procedure that is based on the "watershed" principle.

Morphological mathematics techniques based on erosion, dilation, closure and opening operations are also used in segmentation problems. It was used in [5] to isolate the cerebral overlap for MRI images. In [6] the authors propose an MR image segmentation algorithm for classifying brain tissues. Their method associates the adaptive histogram analysis, morphological operations and knowledge based rules to sort out various regions such as the brain matter and the cerebrospinal fluid, and detect if there are any abnormal regions.

* Corresponding author. Tel.: +212 523 32 22 20; mobile: +212 661 33 65 86; fax: +212 523 32 25 46.
E-mail address: o.bouattane@yahoo.fr (O. Bouattane).

0167-8191/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2011.03.001

In the contour approach case, some derivative operators are used at first to sort out discontinuities on image, on the other hand, some deformable models coming from dynamic contours methods are introduced in [7,8]. The latter have some advantages comparing to the first by the fact that, they deliver the closed contours and surfaces.

In the case of the pattern recognition point of view, the problem is to classify a set of elements defined by a set of features among which a set of classes can be previously known. In the MRI segmentation domain, the vector pattern X corresponds to the gray level of the studied point (pixel). From these approaches, one distinguishes the supervised methods where the class features are known a priori, and the unsupervised ones which use the features auto-learning. From this point of view, several algorithms have been proposed such as: c-means, fuzzy c-means (FCM) [9], adaptive c-means [10], modified fuzzy c-means [11] using illumination patterns and fuzzy c-means combined with neutrosophic set [12].

Segmentation is a very large problem; it requires several algorithmic techniques and different computational models, which can be sequential or parallel using processor elements, cellular automata or neural networks. Improvement and parallelism of c-means clustering algorithm was presented in [19] to demonstrate the effectiveness and how the complexity of the parallel algorithm can be reduced in the Single Instruction Multiple Data (SIMD) computational model.

In the literature, there are several parallel approaches to implement the clustering algorithms. The difference from one to another method is how to exploit the parallelism feasibilities and the huge programming field offered by various parallel architectures to achieve effective solutions and subsequently the reduced complexity algorithms. Notice that all the steps of the well known c-means clustering algorithm are generally identical. They are implemented differently by authors according to the parallel architecture used as a computational model. For example, in [20] authors have proposed a parallel approach using a hardware VLSI systolic architecture. In their proposed method, the c-means clustering problem is subdivided into several elementary operations that are organized in a pipeline structure. The resulted schemes of the obtained logic cells are also associated to design the processing modules of a hardware circuit for clustering problem. This solution was argued by a simulation experiments to evaluate the effectiveness of the proposed systolic architecture. Another strategy was proposed in [21] where the authors have started by presenting a set of basic data manipulation operations using reconfiguration properties of the reconfigurable mesh (RMESH). These basic operations are also used to elaborate some parallel data processing procedures in order to implement the parallel clustering algorithm. The problem of data clustering was studied and detailed in [21] for a general case of N vector patterns. The proposed complexity for their method is $O(Mk + k\log N)$ where k is the number of clusters, M is the size of the vector features of each data point and N is the size of the input data set which is the same as the RMESH size. In [22] the authors have proposed an optimal solution for parallel clustering on a reconfigurable array of processors. The proposed approach is based on the same strategy that divides the clustering problem into a set of elementary operations. Each of these elementary operations is implemented on wider communication bus architecture to reach an optimal run time complexity. Also, some basic operations are translated into shifting operation in the same wider bus network to achieve $O(1)$ time complexity in order to optimize the global clustering algorithm complexity.

In the same way, using the same strategy and the same computational model as in [21], we propose in this paper an $O(kM + \log(k(N/k)^k)$ *times* parallel algorithm for c-means clustering problem and its application to the MRI cerebral images. The presented algorithm is assigned to be implemented on a massively parallel reconfigurable mesh computer of the same size as the input image. The corresponding parallel program of the proposed algorithm is validated on a 2-D reconfigurable mesh emulator [23]. Some interesting obtained results and the complexity analysis and an effectiveness features study of the proposed method are also presented.

This paper is organized as follows: Section 2 presents the computational model used to implement our parallel algorithm. The parallel segmentation c-means algorithm is described in more details in Section 3. The complexity analysis of the proposed parallel c-means algorithm is presented in the next section. Section 5 is devoted to the program code implementation and to the obtained results on an MRI image. Finally, the last section gives some concluding remarks on this work.

2. Parallel computational model

2.1. Presentation

A reconfigurable mesh computer (RMC) of size $n \times$ n, is a massively parallel machine having n^2 processing elements (PEs) arranged on a 2-D matrix as shown in Fig. 1. It is a Single Instruction Multiple Data (SIMD) structure, in which each PE(i,j) is localized in row i and column j and has an identifier defined by ID = $n \times i + j$. Each PE of the mesh is connected to its four neighbors (if they exist) by communication channels. It has a finite number of registers of size ($\log_2 n$) bits. The PEs can carry out arithmetic and logical operations. They can also carry out reconfiguration operations to exchange data over the mesh.

2.2. Basic operations of a PE

2.2.1. Arithmetic operations

Like any processor, each processing element (PE) of the RMC possesses an instruction set relating to the arithmetic and logical operations. The operands concerned can be the local data of a PE or the data arising on its communication channels after any data exchange operation between the PEs.

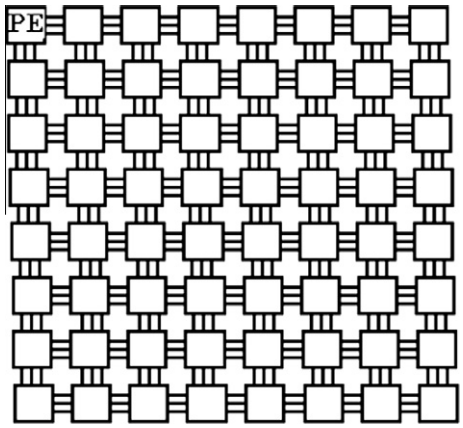

Fig. 1. A reconfigurable mesh computer of size 8×8.

2.2.2. Configuration operations

In order to facilitate data exchange between the PE's over the RMC, each PE possesses an instruction set relating to the reconfiguration and switching operations. Below, we enumerate all the possible reconfiguration operations that can carry out any PE of the RMC according to its usefulness in any stage of any algorithm.

– *Simple bridge (SB):*
A PE of the RMC is in a given state of SB if it establishes connections between two of its communication channels. This PE can connect itself to each bit of its channels, either in transmitting mode, or in receiving mode, as it can be isolated from some of its bits (i.e. neither transmitter, nor receiver). Various SB configurations of Fig. 2a) are described by the following formats:

$$\{EW, S, N\}, \{E, W, SN\}, \{ES, W, N\}, \{NW, S, E\}, \{E\ N, S, W\} \text{ and } \{WS, E, N\}$$

E, W, N and S indicate the East, West, North and South Ports of a PE, respectively.

– *Double bridge (DB):*
A PE is in a DB state when it carries out a configuration having two independent buses. In the same way as in the simple bridge configuration, each PE can connect itself to each bit of its channels, either in transmitting mode, or in receiving mode. Also, it can be isolated from some of its bits (i.e. neither transmitter, nor receiver). The various possible DB configurations of Fig. 2b are:

$$\{EW, NS\}, \{ES, NW\} \text{ and } \{EN, SW\}$$

– *Crossed bridge (CB):*
A PE is in CB state when it connects all its active communication channels in only one; each bit with its correspondent. This operation is generally used when we want to transmit information to a set of PEs at the same time.
The full CB of Fig. 2c.2 is defined by the configuration: {NESW}. But, the other CB configurations defined by the following formats {E,WNS}, {EWN,S}, {ENS,W}, {EWS,N} are considered as the partial CB configurations, because, in each case, one of the four communication channel of the PE is locked.

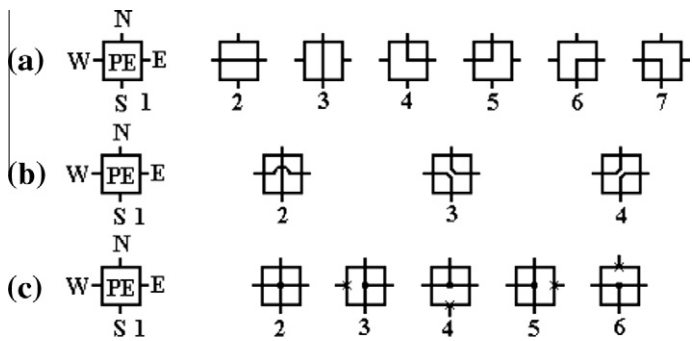

Fig. 2. Different configurations carried out by the PE's of the RMC. (a) Simple bridge (SB), (b) double bridge (DB) and (c) crossed bridge (CB).

3. Parallel segmentation algorithm

3.1. Standard c-means algorithm

Hard classification called k-means is also known by c-means classification. It consists on a partitioned groups of a set S of n attribute vectors into c classes (clusters C_i $i = 1,\ldots,c$). The goal of the classification algorithm is to find the class centers (centroïd) that minimize the objective function given by the following equation:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{k,x_k \in C_i} d(x_k, c_i) \tag{1}$$

c_i is the center of the ith class; $d(x_k, c_i)$ is the distance between ith center c_i and the kth data of S.

We use the Euclidean distance to define the objective function as follows:

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \left(\sum_{k,x_k \in C_i} \|x_k - c_i\|^2 \right) \tag{2}$$

The partitioned groups can be defined by a binary membership matrix $U(c,n)$, where each element u_{ij} is formulated by:

$$u_{ij} = \begin{cases} 1 & \text{if } \|x_j - c_i\|^2 \leqslant \|x_j - c_k\|^2, \quad \forall k \neq i, \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

($i = 1$ to c, $j = 1$ to n; n is the total number of points in S).

Since a data must belong to only one class, the membership matrix U has two properties which are given in the following equations:

$$\sum_{i=1}^{c} u_{ij} = 1, \quad \forall j = 1, \ldots, n \tag{4}$$

$$\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} = n \tag{5}$$

The value c_i of each class center is computed by the average of all its attribute vectors:

$$c_i = \frac{1}{|C_i|} \sum_{k,x_k \in C_i} x_k \tag{6}$$

$|C_i|$ is the size or the cardinal of C_i.

As presented in [9], the c-means classification is achieved using the following algorithm stages:

Stage 1: Initialize the class centers C_i, ($i = 1,\ldots,c$). This is carried out by selecting randomly c points in the gray level scale $[0 \cdots 255]$.
For each iteration i:
Stage 2: Determine the membership matrix using Eq. 3.
Stage 3: Compute the objective function J by Eq. 2.
Stage 4: Compare the obtained objective function J to the one computed at iteration $i - 1$ and stop the loop (i.e. go to stage 6) if the absolute value of the difference between the successive objectives functions is lower than an arbitrary defined threshold (S_{th}).
Stage 5: Calculate the new class centers using Eq. 6, and return back to perform stages 2, 3 and 4.
Stage 6: End.

3.2. Parallel c-means algorithm

In this part, we propose a parallel implementation of the c-means algorithm on a reconfigurable mesh. The input data of the algorithm is an $m \times n$ MRI cerebral image of 256 gray levels, stored in a reconfigurable mesh of the same size one pixel per PE.

In the used computational model, each PE must be equipped by a set of internal registers. These registers will be used during the different stages of the algorithm. They are defined as in the following Table 1.

Our algorithm is composed of the set of following procedures.

3.2.1. Initialization procedure

This procedure consists on initializing the set of registers of each PE of the mesh. We have:

- $R_i = i$, $R_j = j$ and $R_g = Ng$.

Table 1
The different registers required by a PE to perform the parallel c-means algorithm.

Register name	Variable content	Description
R_i	I	Line index of the PE in the mesh
R_j	j	Column index of the PE in the mesh
Rg	Ng	Gray level of the pixel (i,j) stored in the PE(i,j)
Rc_m ($m = 1$ to c)	$U_{m,n}$	Membership class register: if $U_{m,n} = 1$, then m is the index of the class to which the PE belongs during iteration n. The content of this register can be changed at each iteration in the set {0,1}
Rx_m ($m = 1$ to c)	$X_{m,n}$	Class center register: each Rx_m contains the value $X_{m,n}$ of the class center m during iteration n. The content of this register can be changed at each iteration in the range [0···255]

This means that these registers are initialized by the values of the coordinates i, j of the PE and the gray level Ng of its associated pixel.

- $Rc_m = C_{00} = 0$ (i.e. for $m = 0$ at iteration $n = 0$).
  This means that, initially, each PE does not know the index of the class to which it belongs.
- $Rx_m = X_{m,n}$.
  The initial value of X_{m0} (at iteration $n = 0$) is randomly the one of the m gray levels (n_g) that are randomly chosen among the 256 gray levels of the image.

3.2.2. Class determination procedure

This procedure consists of six essential stages which are:

1. Data broadcasting.
2. Distance computation.
3. Membership decision.
4. Objective function computation.
5. Loop stop test.
6. New class center determination.

These various stages are included in a loop as follows:
<Beginning of iteration (n)>

1. *Data broadcasting*
   For ($m = 1$ to c)
   {
   - All the PEs of the mesh configure themselves in cross bridge (CB) state.
   - The representative PE of the class C_m broadcasts its value $X_{m,n}$ through the mesh.
   - All the PEs of the mesh store in their Rx_m the value $X_{m,n}$ received through the mesh.

   }
   It should be noted that the representative PE of a class is the PE having the smallest identifier ID in its class. The search for this smallest identifier calls a procedure of the Min-search in a group of PEs as in [13,14]. The used procedure of min-search has a complexity of $O(1)$ *time*. Thus, for one pass data broadcasting stage the complexity is $O(c)$ *times* for c-means problem.
2. *Distance computation*
   At each iteration n:
   - Each PE computes the distances $d(Ng,X_{m,n})$ between its gray level Ng and the values of the c class centers. These values are stored in its c registers $(X_{1,n},X_{2,n},\ldots,X_{c,n})$. The distance computation at each PE has a complexity of $O(c)$ *times*.
   - Each PE determines the smallest distance d_{min} among those calculated previously

   $$d_{min} = Min(d(Ng, X_{m,n})), \quad m = 1, \ldots, c \tag{7}$$

   As stated in the d_{min} equation, the d_{min} of each PE is achieved after $O(\log_2 c)$ *times* using a hierarchical Min-Search procedure.

3. *Membership decision*
   Each PE determines the new index m of the class C_m to which it belongs. Subsequently, it updates its Rc_m registers by assigning the value 1 to the variable $U_{m,n}$ and 0 to its other Rc registers. It should be noted that the index value m corresponds to m_{min} for which we have:

   $$d(Ng, Xm_{min}, n) = d_{min} \tag{8}$$

The membership decision is a stage of bit assignment. It is completed in $O(1)$ *time*.

4. *Objective function computation*

   For (m = 1 to c)

   {

   - All the PEs that are not belonging to the class C_m configure themselves in the full cross bridge (CB) of Fig. 2c.2) and remain in standby.
   - All PEs belonging to the class C_m participate in the parallel summation to compute local objective function (J_m) using their selected minimal distances d_{min}. This summation procedure was detailed in [15], it has a complexity of $O(\log_2(\text{Card}\ (C_m)))$ iterations; $\text{Card}(C_m)$ is the cardinality of the class C_m.
   - The representative PE (RPE) of the class C_m retains the result of the summation (J_m).

   }

   In order to compute the global cost function J, we must prepare at each PE its local function J_m. So, for a given pass of the algorithm this stage requires $O(\sum_{m=1}^{c}\log_2(\text{Card}(C_m)))$ *times*.

   Thus, the c representative PEs of c classes carry out a parallel summation on their results (J_m) by using the same procedure of sum on tree used in [15], this summation is completed in $O(\log_2 c)$ *times*.

   The final result J of this summation which represents the cost function will be stored in the representative PE of these representative PEs noted RRPE.

5. *Loop stopping test*

   The RRPE calculates the absolute value $|J_n - J_{n-1}|$ and compares it with an arbitrary threshold S_{th}.

   - If $|J_n - J_{n-1}| < S_{\text{th}}$ then go to the end of procedure.
   - Else, return back to the procedure of new class centers computation.

This stage corresponds to a simple comparison operation, it is achieved in $O(1)$ *time*.

6. *Class center computation*

   For (m = 1 to c)

   {

   - All the PEs that are not belonging to the class m, go to the full crossed bridge (CB) state of Fig. 2c.2) and remain in standby.
   - All the PEs belonging to the class m, carry out a parallel summation to sort out the cardinal of the class C_m, ($\text{Card}(C_m)$), we propose the method used in [16] that is achieved in $O(1)$ *time*.
   - All the PEs belonging to the class m must compute the value S_m using a parallel summation procedure on their Ng_m gray levels. The summation procedure used has a complexity of $O(\log_2(\text{Card}(C_m)))$ *iterations*. It is the same procedure as in [15].
   - The RPE of the class m carries out the ratio: $X = (S_m/\text{Card}(C_m))$ which corresponds to the new center of its class. It will then broadcast this result to all the PEs of the mesh to update their Rx_m registers.

   }

Notice that, for each class C_m, the complexity of the different steps of this stage depends essentially on the cardinality of C_m class. This stage is then achieved in $O(c) + O(\sum_{m=1}^{c}\log_2(\text{Card}(C_m)))$ *times*.

– *Return to the stage 2 of the distance computation*

<End of iteration (n)>.

It is clear that, for each of the six stages of the class determination procedure, the reconfigurable mesh allows us to choose the rapid and safe parallel procedures. We will report the global complexity for one pass of the proposed algorithm to make comparison with the previous works on the similar computational model. Also, we will make a rigorous complexity analysis for one pass and sort out other aspects related to the dynamic evolution of the class cardinality during each pass. This evolution is pursued until the convergence of the algorithm. This aspect will be studied by the proposed parallel classification algorithm for MRI images. Indeed, the MRI images represent interesting data sets which require real time algorithms and high performance computational models such as the massively reconfigurable mesh computers.

4. Complexity analysis

In order to evaluate the complexity of our parallel algorithm, it is useful to report the complexities of all its stages. They are summarized in Table 2.

The resulted complexity is:

$$O\left(c + \log_2 c + \sum_{m=1}^{c}\log_2(\text{Card}(C_m))\right)\ \textit{times} \tag{9}$$

This complexity depends on the unknown cardinalities of the C_m classes. So, it must be studied according to the c variable to look for its maximum in order to compare it with the well known previous works.

Table 2
The complexities of each stage of the proposed parallel algorithm.

Stage	Time complexity
1. Data broadcasting	$O(1)+O(c)$
2. Distance computation	$O(c)+O(\log_2 c)$
3. Membership decision	$O(1)$
4. Objective function computation	$O(\log_2 c)+O(\sum_{m=1}^{c}\log_2(\mathrm{Card}(C_m)))$
5. Loop stop test	$O(1)$
6. New class center determination	$O(c)+O(\sum_{m=1}^{c}\log_2(\mathrm{Card}(C_m)))$
The complexity of the proposed algorithm	$O(c)+O(\log_2 c)+O(\sum_{m=1}^{c}\log_2(\mathrm{Card}(C_m)))$

For a given c, to reach the maximum of 9 it is necessary to solve the following problem:

$$\text{Maximize} \quad \left(E=\sum_{m=1}^{c}\log_2(\mathrm{Card}(C_m))\right) \tag{10}$$

$$\text{Subject to} \quad \sum_{m=1}^{c}\mathrm{Card}(C_m)=N \tag{10'}$$

$$\text{and} \quad \bigcap_{m=1}^{c}C_m=\Phi \tag{10''}$$

where N is the total number of points of the input image.

Let $X_m = \mathrm{Card}(C_m)$; the problem (10) is viewed as:

$$\text{Maximize} \quad \left(F=\prod_{m=1}^{c}X_m\right) \tag{11}$$

$$\text{Subject to} \quad \sum_{m=1}^{c}X_m=N \tag{11'}$$

$$\text{and} \quad \bigcap_{m=1}^{c}C_m=\Phi \tag{11''}$$

Combining (11) and (11′) leads to:

$$F=\left(N-\sum_{m=1}^{c-1}X_m\right)\cdot\prod_{m=1}^{c-1}X_m=N\cdot\prod_{m=1}^{c-1}X_m-\left(\prod_{m=1}^{c-1}X_m\right)\cdot\left(\sum_{m=1}^{c-1}X_m\right) \tag{12}$$

The maximum of F is obtained when

$$\frac{\partial F}{\partial X_1}=\frac{\partial F}{\partial X_2}=\cdots=\frac{\partial F}{\partial X_c}=0 \tag{13}$$

This leads to: $X_1 = X_2 = \cdots = X_c = N/c$.

In this condition (10) becomes: $E_{max}=\sum_{m=1}^{c}\log_2(X_m)=\sum_{m=1}^{c}\log_2(N/c)=c\cdot\log_2(N/c)$.

Then, the maximum value of the complexity expressed in 9 is:

$$\mathrm{O}(\mathrm{c}+\log_2\mathrm{c}+c.\log_2(N/c))times. \tag{14}$$

In order to compare our complexity with those of the previous works, we use the comparison Table 3 as stated in [22]. In this table the authors were studied the clustering problem by taking into account that each data point must have a number of M features. The number M is used in all the steps of their algorithms. Furthermore, some authors proposed some enhanced solutions by extending the sizes of their computational models. In this paper, the input data set of our algorithm is a gray leveled image, where each point has $M = 1$ feature (its gray level). This algorithm can be extended easily for any ($M > 1$) features. In this case the complexities of the two first stages of the algorithm (data broadcasting and the first step of the distance computation stage) are altered. They are multiplied by M. Hence, the maximum value of our algorithm complexity becomes:

$$O(Mc+\log_2 c+c\log_2(N/c))\ times=O(Mc+\log_2(c(N/c)^c)\ times \tag{15}$$

Table 3
Results of the complexities comparison for parallel clustering.

Algorithm	Architecture	Bus width (bit)	Processors	Time complexity
Li and Fang [24]	Hypercube	$O(\log N)$	MN	$O(k \log MN)$
Hwang et al. [25]	OMP	$O(\log N)$	p	$O(kM + kMN/p)$
Ranka and Sahni [26]	Hypercube	$O(\log N)$	MN	$O(k + \log MN)$
			kMN	$O(\log kMN)$
Jenq and Sahni [21]	RMESH	$O(\log N)$	N	$O(kM + k \log N)$
			MN	$O(k \log MN)$
			kMN	$O(M + \log kMN)$
Tsai and Horng [22]	RAPWBN	$N^{1/c}$	$N^{1+1/c}$	$O(kM)$
			$MN^{1+1/c}$	$O(k)$
			$kMN^{1+1/c}$	$O(1)$
This paper	RMESH	$O(\log N)$	N	$O(kM + \log(k(N/k)^k)$

Notice that, in the results comparison Table 3, the variable k is used instead of c to represent the same thing. It represents the number of clusters.

Table 3 shows the different time complexities for the same clustering problem, using different architectures of the computational models. In this table N represents the size of the input data set also it corresponds to size of the parallel architecture used. For a reduced size computational model, N remains the size of the data set and P is the size of parallel architecture used. M is the vector features size of the data point.

For a rigorous comparison, we must compare our results with those obtained by authors of [21], because they used the same computational model RMESH of the same size and the same Bus width. Thus, we can easily show that the obtained complexity $O(kM + \log(k(N/k)^k)$ is less than $O(kM + k \log N)$ obtained in [21].

5. Implementation and results

The parallel algorithm, described in Section 3, is implemented in our emulating framework [23] using its parallel programming language. The program code presented in Fig. 3 is performed using the MRI cerebral image as data input. Before its execution, we define in the initialization phase the number of classes by $c = 3$. This means that the classes looked for in the image are the white matter, the gray matter and the cerebrospinal fluid. The background of the image is not considered by the algorithm.

Parallel c-means program code:

```
<prog>

<!-
  Load data and initialization phase
->
  <loadImage file="cerveau3.jpg" reg="0" codage="8"/>
    <hote>
      <loadValue reg="0" value="140"/><!- reg[0]contains the initial value of C1 center->
      <loadValue reg="1" value="149"/> <!- reg[1]contains the initial value of C2 center->
      <loadValue reg="2" value="150"/><!- reg[2]contains the initial value of C3center->
      <loadValue reg="3" value="0"/><!- reg[3]contains the initial value of Jn-1->
      <loadValue reg="4" value="0"/><!- reg[4]contains the initial value of Jn ->
      <loadValue reg="5" value="0.1"/><!- reg[5]contains the initial value of Sth->
    </hote>
<!-
  loop test with reg[30] having the absolute value of (Jn - Jn-1) to be compared with Sth
->
  <doWhile test="Math.abs (reg[30])>reg[5]" target="hote">
    <for-eachPE test="reg[0]!=0">
      <mark type="true"/>
<!-
  All the PE's load the three class centers from the hote
->
      <loadValue reg="8" value="hote.reg[0]"/>
      <loadValue reg="9" value="hote.reg[1]"/>
      <loadValue reg="10" value="hote.reg[2]"/>
```

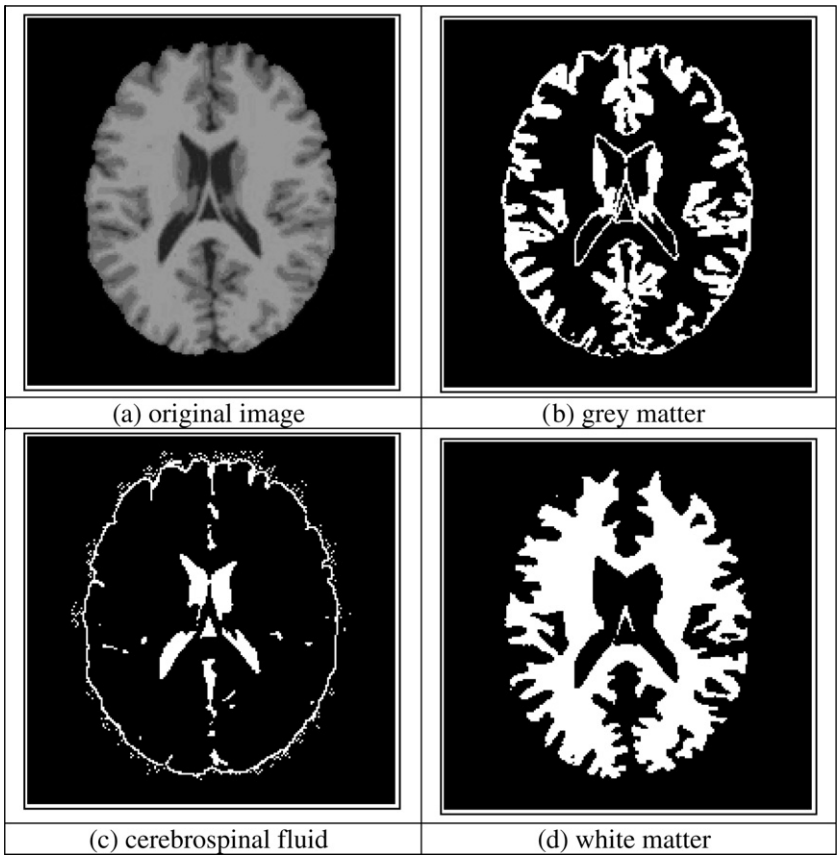

Fig. 3. Segmentation results by the elaborated parallel program.

```
<!--
  Each PE computes the three distances separating it from the three class centers
-->
     <doOperation expression="reg[11]=(reg[0]-reg[8])*(reg[0]-reg[8])"/>
     <doOperation expression="reg[12]=(reg[0]-reg[9])*(reg[0]-reg[9])"/>
     <doOperation expression="reg[13]=(reg[0]-reg[10])*(reg[0]-reg[10])"/>
<!-
  Membership decision:Each PE decides to belong to its nearest neighbor class
->
     <loadValue reg="7" value="reg[10]"/>
     <loadValue reg="16" value="0"/>
     <loadValue reg="17" value="0"/>
     <loadValue reg="18" value="1"/>
     <loadValue reg="15" value="reg[13]"/>
     <if test="reg[15]>reg[12]">
       <loadValue reg="15" value="reg[12]"/>
       <loadValue reg="7" value="reg[9]"/>
       <loadValue reg="16" value="0"/>
       <loadValue reg="17" value="1"/>
       <loadValue reg="18" value="0"/>
     </if>
     <if test="reg[15]>reg[11]">
       <loadValue reg="15" value="reg[11]"/>
       <loadValue reg="7" value="reg[8]"/>
       <loadValue reg="16" value="1"/>
       <loadValue reg="17" value="0"/>
```

```
      <loadValue reg="18" value="0"/>
    </if>
    <mark type="false"/>
  </for-eachPE>
<!-
All the PEs of the first class are marked to participate to the hierarchical sum. The result of this sum
is stored in the hote.
Hregs[20]contains the gray levels sum of all the points of C1 class.
Hregs[21]contains the distances sum from all the points of C1 class to its center.
Hregs[22]contains the cardinality of C1 class.
->
  <for-eachPE test="(reg[7]==reg[8]) and (reg[0]!=0)">
    <mark type="true"/>
    <defineRepresentativePE/>
    <doHSum PEregs="0,11,16" Hregs="20,21,22"/>
    <mark type="false"/>
  </for-eachPE>
  <inialiseRepresentativePE/>
<!-
All the PEs of the second class are marked to participate to the hierarchical sum. The result of this
sum is stored in the hote.
Hregs[23]contains the gray levels sum of all the points of C2 class.
Hregs[24]contains the distances sum from all the points of C2 class to its center.
Hregs[25]contains the cardinality of C2 class.
->
  <for-eachPE test="(reg[7]==reg[9]) and (reg[0]!=0)">
    <mark type="true"/>
    <defineRepresentativePE/>
    <doHSum PEregs="0,12,17" Hregs="23,24,25"/>
    <mark type="false"/>
  </for-eachPE>
  <inialiseRepresentativePE/>
<!-
All the PEs of the third class are marked to participate to the hierarchical sum. The result of this sum
is stored in the hote.
Hregs[26]contains the gray levels sum of all the points of C3 class.
Hregs[27]contains the distances sum from all the points of C3 class to its center.
Hregs[28]contains the cardinality of C3 class.
->
  <for-eachPE test="(reg[7]==reg[10]) and (reg[0]!=0)">
    <mark type="true"/>
    <defineRepresentativePE/>
    <doHSum PEregs="0,13,18" Hregs="26,27,28"/>
    <mark type="false"/>
  </for-eachPE>
  <inialiseRepresentativePE/>
<!-
The hote computes the global cost function J, the three new class centers and sets up the iteration
counter.
reg[21]contains the value of J1
reg[24]contains the value of J2
reg[27]contains the value of J3
->
  <hote>
    <doOperation expression="reg[3]=reg[4]"/>
    <doOperation expression="reg[4]=reg[21]+reg[24]+reg[27]"/>
    <doOperation expression="reg[0]=reg[20]/reg[22]"/><!—New class center of C1 —>
    <doOperation expression="reg[1]=reg[23]/reg[25]"/><!—New class center of C2 —>
    <doOperation expression="reg[2]=reg[26]/reg[28]"/><!—New class center of C3 —>
    <doOperation expression="reg[30]=reg[4]-reg[3]"/>
    <doOperation expression="reg[31]=reg[31]+1"/>
```

```
    </hote>
<!-
    End loop do while
->
    </doWhile>
<!-
Labelling the different image components:assigning to each PE the index of the class to which it
belongs
->
  <for-eachPE test="reg[0]!=0">
    <mark type="true"/>
    <if test="reg[7]==reg[8]">
      <loadValue reg="1" value="hote.reg[0]"/>
    </if>
    <if test="reg[7]==reg[9]">
      <loadValue reg="2" value="hote.reg[1]"/>
    </if>
    <if test="reg[7]==reg[10]">
  <loadValue reg="3" value="hote.reg[2]"/>
    </if>
  </for-eachPE>
<!-
  End of the program
->
</prog>
```

Program results:

After performing the presented parallel program, we obtain the following results: the image of Fig. 3a corresponds to a human brain cut, it is the original input image of the program. Figs. 3b–d represent the three matters of the brain. They are named respectively the gray matter, cerebrospinal fluid and white matter.

After performing the proposed parallel algorithm, we complete its effectiveness features by the following study which is focused on its dynamic convergence analysis. To do so, we present three cases of study. For each case we use the same input MRI image, but the initial class centers are changed. The results are presented by tables and figures. In the first case, the initial class centers are arbitrarily chosen as: $(c_1, c_2, c_3) = (1, 2, 3)$.

In Table 4, the class centers do not change after 14 iterations and the algorithm converges to the final class centers $(c_1, c_2, c_3) = (28.6, 101.6, 146.03)$.

Notice that, for any iteration, the total number of the image points N is constant.

$N = \mathrm{Card}(C_1 \cup C_2 \cup C_3)$. In our image case, $N = NC_1 + NC_2 + NC_3 = 19{,}273$ pixels.

Fig. 4 shows the curves of the different data of Table 4. Theses curves represent the dynamic changes of each class center value and the cardinality of its corresponding class. In Fig. 4a we see clearly the convergence of the class centers. Also, the cardinality of each class is presented in Fig. 4b and the absolute value of the objective function error $|J_n - J_{n-1}|$ in Fig. 4c.

In the second case, the initial class centers are arbitrarily chosen as: $(c_1, c_2, c_3) = (1, 30, 255)$.

Table 4
Different states of the classification algorithm starting from class centers $(c_1, c_2, c_3) = (1, 2, 3)$.

Iteration	Value of each class center			Number of points in each class			Absolute value of the error
	c_1	c_2	c_3	NC_1	NC_2	NC_3	$\lvert J_n - J_{n-1}\rvert$
1	1.00	2.00	3.00	367	47	18,859	2.84E+08
2	1.00	2.00	120.14	367	1922	16,984	2.70E+08
3	1.00	33.86	129.57	810	2188	16,275	4.96E+06
4	4.38	51.91	132.04	906	2491	15,876	1.33E+06
5	6.15	58.40	133.23	946	2534	15,793	1.50E+05
6	7.14	60.05	133.43	946	2952	15,375	5.36E+04
7	7.14	65.14	134.44	1460	2517	15,296	2.34E+05
8	16.70	72.44	134.64	1801	4713	12,759	9.02E+05
9	21.49	90.47	141.16	2019	4912	12,342	2.76E+06
10	24.68	93.37	142.41	2289	4644	12,340	1.24E+05
11	28.60	95.44	142.42	2289	4921	12,063	7.92E+04
12	28.60	96.71	142.98	2289	6074	10,910	1.02E+05
13	28.60	100.94	145.51	2289	6292	10,692	2.41E+05
14	28.60	101.60	146.03	2289	6292	10,692	5.64E+03
15	28.60	101.60	146.03	2289	6292	10,692	0.00E+00

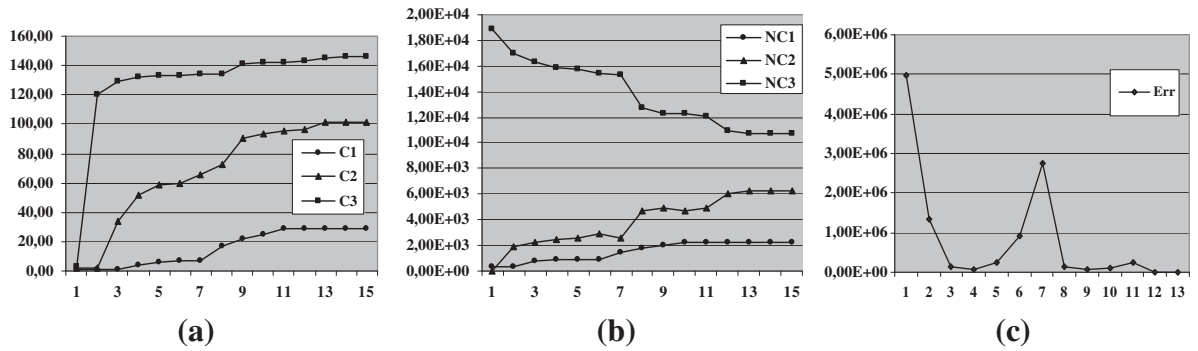

Fig. 4. Dynamic changes of the different classification parameters starting from class centers $(c_1, c_2, c_3) = (1, 2, 3)$. (a) Class centers, (b) cardinality of each class and (c) absolute value of error of the objective function.

Table 5
Different states of the classification algorithm starting from class centers $(c_1, c_2, c_3) = (1, 30, 255)$.

Iteration	Value of each class center			Number of points in each class			Error value
	c_1	c_2	c_3	NC_1	NC_2	NC_3	$(J_n - J_{n-1})$
1	1.00	30.00	255.00	806	10,933	7534	1.47E+08
2	4.32	102.35	151.79	2019	6703	10,551	−1.43E+08
3	24.68	100.37	146.29	2289	6292	10,692	−1.42E+06
4	28.60	101.60	146.03	2289	6292	10,692	−4.53E+04
5	28.60	101.60	146.03	2289	6292	10,692	0.00E+00

Table 6
Different states of the classification algorithm starting from class centers $(c_1, c_2, c_3) = (140, 149, 150)$.

Iteration	Value of each class center			Number of points in each class			Error value
	c_1	c_2	c_3	NC_1	NC_2	NC_3	$(J_n - J_{n-1})$
1	140.00	149.00	150.00	11,739	409	7125	4.01E+07
2	95.62	148.99	151.95	8581	5783	4909	−2.65E+07
3	82.13	140.26	152.83	6931	4808	7534	−3.72E+06
4	73.36	127.72	151.79	3977	7762	7534	−1.81E+06
5	51.98	117.99	151.79	3066	8372	7835	−3.41E+06
6	40.07	114.55	151.15	2667	6944	9662	−7.80E+05
7	34.22	107.44	147.88	2615	6107	10,551	−6.90E+05
8	33.37	104.04	146.29	2615	5966	10,692	−1.21E+05
9	33.37	103.50	146.03	2615	5966	10,692	−2.45E+03
10	33.37	103.50	146.03	2615	5966	10,692	0.00E+00

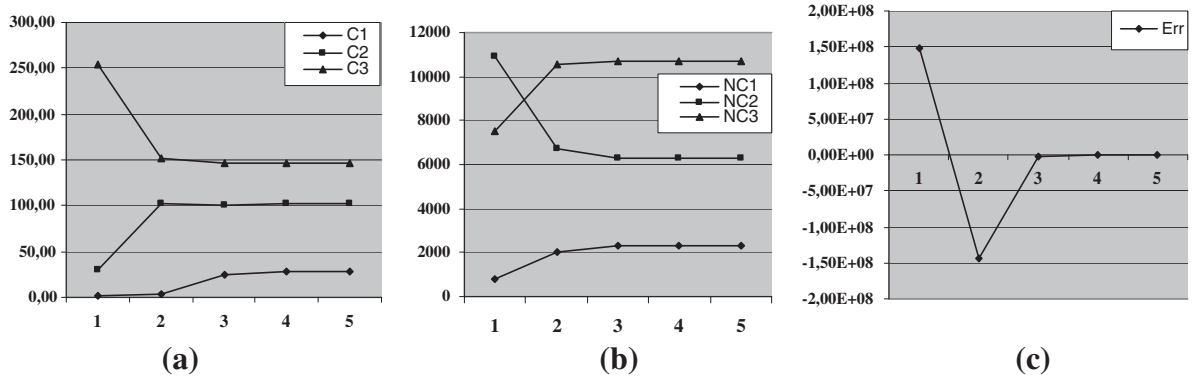

Fig. 5. Dynamic changes of the different classification parameters starting from the class centers $(c_1, c_2, c_3) = (1, 30, 255)$. (a) Class centers, (b) cardinality of each class and (c) error of objective function.

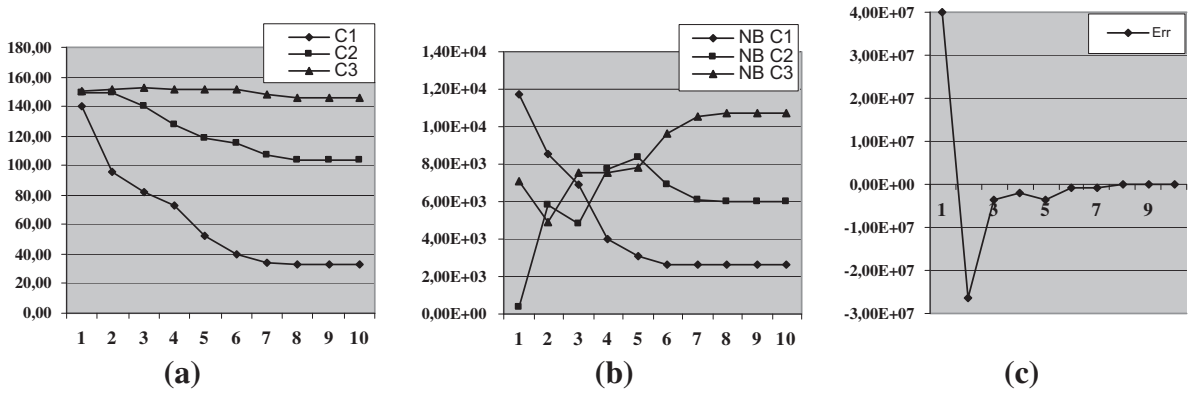

Fig. 6. Dynamic changes of the different classification parameters starting from class centers $(c_1, c_2, c_3) = (140, 149, 150)$. (a) Class centers, (b) cardinality of each class and (c) error of objective function.

In the third case, the initial class centers are arbitrarily chosen as: $(c_1, c_2, c_3) = (140, 149, 150)$.

Through the obtained results of the three cases of study of Tables 4–6 and represented in Figs. 4–6, we can conclude that the complexity of the presented algorithm is low in term of iteration number, but it depends on the initial class centers. It was discussed in the literature that the c-means algorithm complexity for sequential and parallel strategies can be reduced using some additional preprocessing phases to start from some appropriate class centers. Among the mentioned preprocessing phase, we find the histogramming procedure that orients the class centers towards the histogram modes of the image. In this paper we do not introduce any preprocessing phase, because our goal is at first, the parallelization of the c-means algorithm and its implementation on a reconfigurable mesh computer emulator to validate the corresponding parallel procedures. In the second time, by the obtained results, we will focus our further studies on the dynamic evolution of the class cardinality from one pass to another until the convergence of the algorithm. Parallel histogram computation algorithms can be easily implemented on the used emulator. The proposed algorithms in [17,18] are well suited to our computational model.

6. Conclusion

In this paper, we have presented a method for parallelizing the c-means classification algorithm and its implementation on a massively parallel reconfigurable mesh computer. An application of this algorithm to the MRI images segmentation was considered. The elaborated program was performed on the reconfigurable mesh computer emulator. The obtained results show that, in little number of iterations, the algorithm converges to the final class centers. Hence, the parallel computation method is proposed essentially to reduce the complexity of the classification and clustering algorithms. To enhance the effectiveness of this work, it is useful to improve the complexity of this algorithm by avoiding random initializations of the class centers. Moreover, the extension of this parallel technique to other classification methods is very possible. It can be easily oriented to parallelize the well known algorithms such as: the fuzzy c-means, the adaptive fuzzy c-means and others based on neural networks.

References

[1] H.D. Heng, X.H. Jiang, Y. Sun, J. Wang, Color image segmentation: advances and prosoects, Pattern Recognition 34 (2001) 2259–2281.
[2] A. Zijdenbos, B.M. Dawant, Brain segmentation and white matter lesion detection in MR images, Critical Reviews in Biomedical Engineering 22 (5/6) (1994) 401–465.
[3] J.H. Chang, K.C. Fan, Y.L. Chang, Multi-modal gray-level histogram modeling and decomposition, Image and Vision Computing 20 (2002) 203–216.
[4] J.P. Thiran et al, A queue-based region growing algorithm for accurate segmentation of multi-dimensional digital images, Signal Processing 60 (1997) 1–10.
[5] J.-F. Mangin, Robust brain segmentation using histogram scale-space analysis and mathematical morphology, in: MICCAI'98 First International Conference on Medical Image Computing and Computer Assisted Intervention, 1998, pp. 1230–1241.
[6] C. Tsai, B.S. Manjunath, R. Jagadeesan, Automated segmentation of brain MR images, Pattern Recognition 28 (12) (1995) 1825–1837.
[7] D. Terzopoulos et al, Snakes: active contour models, International Journal of Computer Vision (1988) 321–331.
[8] D. Terzopoulos et al, Deformable models in medical image analysis: a survey, Medical Image Analysis 1 (2) (1996) 91–108.
[9] J.S.R. Jang, C. Sun, T.E. Mizutani, Neuro-Fuzzy and Soft Computing, Prentice Hall, 1997. pp. 426–427.
[10] L. Dzung Pham, L.P. Jerry, An adaptative fuzzy c means algorithm for image segmentation in the presence of intensity inhomogeneities, Pattern Recognition Letters 20 (1999) 57–68.
[11] L. Ma, R.C. Staunton, A modified fuzzy c-means image segmentation algorithm for use with uneven illumination patterns, Pattern Recognition 40 (2007) 3005–3011.
[12] Y. Guo, H.D. Cheng, W. Zaho, Y. Zhang, A novel image segmentation algorithm based on fuzzy c-means algorithm and neutrosophic set, in: Proceeding of the 11th Joint Conference on Information Sciences, Atlantis Press, 2008.
[13] H. Li, M. Maresca, Polymorphic torus network, IEEE Transaction on Computer C-38 (9) (1989) 1345–1351.
[14] H. Li, M. Maresca, Polymorphic torus architecture for computer vision, IEEE Transaction on PAMI 11 (3) (1989) 233–242.

[15] J. Elmesbahi, O. Bouattane, M. Sabri, M. Chaibi, A fast algorithm for ranking and perimeter computation on a reconfigurable mesh computer, in: Proceedings of the IEEE International Conference on Systems Man and Cybernetics, Texas, October 2–5, 1994, pp. 1898–1902.
[16] Y. Ben-Asher et al, The power of reconfiguration, Journal of Parallel and Distributed Computing 13 (1991) 139–153.
[17] M. Eshaghian-Wilner Mary, R. Miller, The systolic reconfigurable mesh, Parallel Processing Letters 14 (3–4) (2004) 337–350. ISSN 0129-6264.
[18] J. Jang, H. Park, V.K. Prasanna, A fast algorithm for computing histograms on a reconfigurable mesh, in: Fourth Symposium on the Frontiers of Massively Parallel Computation, 19–21 October 1992, pp. 244–251.
[19] J. Tian et al, Improvement and parallelism of *k*-means clustering algorithm, Tsinghua Science and Technology 10 (3) (2005) 277–281. ISSN 1007-0214, 01/21.
[20] L.M. Ni, A.K. Jain, A VLSI systolic architecture for pattern clustering, IEEE Transaction on Pattern Analysis and Machine Intelligence (1985) 79–89.
[21] J.F. Jenq, S. Sahni, Reconfigurable mesh algorithms for image shrinking, expanding, clustering and template matching, International Parallel Processing Symposium (1991) 208–215.
[22] H.R. Tsai, S.J. Horng, Optimal parallel clustering algorithms on a reconfigurable array of processors with wider bus networks, Image and Vision Computing (1999) 925–936.
[23] M. Youssfi, O. Bouattane, M.O. Bensalah, A massively parallel re-configurable mesh computer emulator: design, modeling and realization, Journal of Software Engineering and Applications 3 (2010) 11–26.
[24] X. Li, Z. Fang, Parallel clustering algorithms, Parallel Computing 11 (1989) 275–290.
[25] K. Hwang, H.M. Alnuweiri, V.K.P. Kumar, D. Kim, Orthogonal multiprocessor sharing memory with an enhanced mesh for integrated image understanding, CVGIP Image Understanding 53 (1991) 31–45.
[26] S. Ranka, S. Sahni, Clustering on a hypercube multicomputer, IEEE Transaction on Parallel and Distributed Systems 2 (2) (1991) 129–137.