Journal of Computer Science 3 (3): 162-167, 2007
ISSN 1549-3636
© 2007 Science Publications

# An Efficient Ant Algorithm for Swarm-Based Image Clustering

[1]Salima Ouadfel and [2]Mohamed Batouche
[1]Department of Computer Science, University of Batna, Algeria
[2]Computer Vision Group, LIRE Laboratory, University of Constantine, Algeria

**Abstract:** A collective approach to resolve the segmentation problem was proposed. AntClust is a new ant-based algorithm that uses the self-organizing and autonomous brood sorting behavior observed in real ants. Ants and pixels are scatted on a discrete array of cells represented the ants' environment. Using simple local rules and without any central control, ants form homogeneous clusters by moving pixels from the cells of the array according to a local similarity function. The initial knowledge of the number of clusters and initial partition were not needed during the clustering process. Experimental results conducted on synthetic and real images demonstrate that our algorithm AntClust was able to extract the correct number of clusters with good clustering quality compared to the results obtained from a classical clustering algorithm like Kmeans algorithm.

**Key words:** Image clustering, Swarm intelligence, Artificial ants, Kmeans

## INTRODUCTION

Image segmentation is a fundamental task in a vision system. Its purpose is to subdivide an image into meaningful non-overlapping regions[1-3]. Image segmentation can be viewed as a clustering problem, which aims to partition the image into clusters such that the pixels within a cluster are as homogenous as possible whereas the clusters among each other are as heterogeneous as possible with respect to a similarity measure.

Clustering algorithms can be divided into four main classes: partitioning methods, hierarchical methods, density-based clustering and grid-based clustering. An extensive survey of clustering techniques are described in[4]. Several partitioning methods are provided in the literature; they can be classified as Hard or Fuzzy algorithms. In Hard clustering algorithms, the pixel is assigned to one cluster. Fuzzy algorithms can assign pixels to multiple clusters. The degree of membership in the clusters depends on the closeness of the pixels data point to the cluster center. The drawback of the most portioning algorithms is, prior knowledge of the number of clusters in the pixels is required and they have significant sensitivity to cluster center initialization.

For many years now, several papers have highlighted the efficiency of approaches inspired from the nature[5]. In particular a variety of algorithms inspired from the swarm intelligence observed from the ants‘ behaviors such food hunting and nest building have been introduced for solving several combinatorial optimization problems. In this respect ant algorithms have been recently emerged. In this field we found the Ant Colony Optimization (ACO) inspired from the foraging behavior of ant colonies and the ant based clustering algorithms based on the cemetery organization and brood sorting of ants[6].

Cemetery building is obtained when ants clean their nests and form piles by collecting corpses found within the nest. Brood sorting is widespread in ant colonies. Ants gather their larvae together according to the larva's size. The basic principle of this sorting behavior is attraction between the items transported by the ant. Small clusters of similar items grow by attracting ants to deposit more items according to their type or their size. This positive feedback leads to formation of homogeneous clusters[2].

The first modelization of this ants' behavior has been done by Deneubourg *et al.*[6,7]. In their model a population of simple ants are used to cluster objects together using simple local rules and without any central control. From this basic model, Lumer and Faieta introduced further developments and extend its application to clustering data objects[8]. The LF algorithm's basic principles are straightforward: ants and data objects are scattered randomly on a rectangular grid, which represents the environment of the ants. Ants move on the grid and probably pick up and drop data objects using a measure of similarity of two data

**Corresponding Author:** Salima Ouadfel Department of Computer Science, University of Batna, Algeria

objects and density of data objects within the ants' local neighborhood. Both the Deneubourg and LF algorithms have become well-known models that have been extensively used in different applications like data mining[1,2,8], graph-partitioning[9] and text-mining[1-46,8,9,11].

From these basic models, some works have been done concerning the setting of the parameters like the size of the grid which influence the convergence of the algorithm and the overall quality of the clustering obtained based essentially of visual observation[1,2,4,6,8,9,11].

Based on the existing works, we propose in this study AntClust a new ant-based clustering algorithm for image segmentation. In AntClust, simple agents that randomly move on a discrete array model ants. Pixels that are scatted within the cells of the array can be moved from one cell to another to form clusters The pixels' movement are implemented indirectly through the ants' movement. Each ant can picks up or drop a pixel according to a similarity function which measures the pixel' similarity with other pixels in a cluster. In this way, ants dynamically cluster pixels into distinctive independent groups within which similar pixels are closely placed in the same cluster.

As we will see in the following, AntClust introduced new probabilistic rules for picking up or dropping pixels and also a local movement strategy is used to speed up the clustering convergence. Experimental results show that AntClust algorithm gives better clustering quality compared to those obtained from Kmeans algorithm.

**Ants algorithms:** Ant-based clustering algorithms are based upon the brood sorting behavior of ants. The pioneer of this work are Deneubourg *et al.*[7,12], which apply it for tasks in robotics. This basic work has been modified by Lumer and Faita to extend to numerical data analysis[1]. In this algorithm, the data is randomly dispersed onto a two dimensional grid. Each ant moves randomly around this grid picking and dropping the data items. The decision to pick up or drop an item is random but is influenced by the data items in the ant's immediate neighborhood, thus causing similar items to be more likely placed together on the grid. The probability of dropping an object increases with high densities of similar objects in the neighborhood. In contrast, the probability of picking an object increases with low-density neighborhoods and decreases with high similarity among objects in the surrounding area.

The probability of picking and dropping are given by:

$$P_p(i) = \left(\frac{k_1}{k_1 + f(i)}\right)^2 \tag{1}$$

$$P_d(i) = \begin{cases} 2f(i) & \text{if } f(i) < k_2 \\ 1 & \text{if } f(i) < k_2 s \end{cases} \tag{2}$$

with

$$f(i) = \begin{cases} \dfrac{1}{s^2} \sum_{j \in R\ (r(i))} 1 - \dfrac{d(i,j)}{\alpha} & \text{si} \quad f > 0 \\ 0 & \text{sinon} \end{cases} \tag{3}$$

*r(i)* is the position of the data item *i* on the grid and *f(i)* is a measure of the average dissimilarity of object *i* to the other objects *j* present in its neighborhood $R$ with the size $s*s$. $\alpha$ is the scale of the dissimilarity and its value is crucial to the successful execution of this algorithm.

Lumer and Faieta have introduced the notion of a short-term memory within each agent. Each ant remembers a small number of locations where it has successfully dropped an item. And so, when picking a new item this memory is consulted in order to bias the direction in which the ant will move. Thus, the ant tends to move towards the location it last dropped a similar item.

In[2] Momarché proposed AntClass which is a major extension of the LF algorithm. In AntClass, ants can load and drop more than an object in the same cell, forming heaps of objects. More, it is a hybridization of ant-based algorithm and the k-means algorithm.

**AntClust algorithm:** AntClust is a distributed algorithm that uses positive feedback to achieve the clustering of pixels. It is mainly based on the versions described in[8] and[2]. A number of slight modifications have been introduced that improve the quality of the clustering and to speed up of the convergence.

In both LF and AntClass algorithms, the ants evolve on the discrete 2D board. The size of this grid should be selected in dependence of the data collection, as it otherwise impairs convergence speed. If the size is too small, the ants carrying data items will make everlasting move since they will never find an empty case to drop down them. This will consume large amount of computational time. Otherwise, if the size of the grid is too large, ants will make idle movement when no carrying data items before encountering an item data.

As far as we know these is not theoretical guideline to determine automatically the size of the board. For this reason, we replace the rectangular grid by a discrete array of $N$ cells when $N$ is the total number of pixels to be clustered. All cells of the array are connected to the nest of the ants' colony in order to let the ants travel from one cell to another easily. During the algorithm ants are able to create, build or destroy existing clusters of pixels. A cluster is defined as a collection of two or more pixels. As in[2], a cluster is spatially located in a single cell, which makes the identification of clusters easily unlike in the LF algorithm where spatial patterns of objects car touch each other.

Initially the $N$ pixels $\{p_1,....,p_N\}$ to be clustered are placed on the array such that each array cell can only occupied by one pixel. Each ant $a_i$ from a colony of $K$ ants $\{a_1,....,a_K\}$ picks up a randomly chosen pixel from its cell and returns to his nest. After this initialization phase, the clustering phase starts. During an iterative process, one ant is selected randomly; it performs a number of movements between its nest and the array and decides with a probabilistic rule whether or not to drop its pixel. If the ant becomes free, it searches for a new pixel to pick up. The ant has knowledge of a list of the locations of all pixels not being carried by other ants. The ant randomly selects a pixel from this list of "free" pixels and probabilistically decides whether or not to pick up that pixel. This process is repeated for all ants. The stopping criterion of the algorithm is the number of iteration fixed by the user at the beginning.
The main AntClust algorithm is presented in Fig. 1.

On the array, each ant $a_i$ may possibly picks up a pixel $p_i$ from a cell $c_k$ or drop it in the cell $c_k$ according to the similarity function $f$, which represents the average distance between the pixel $p_i$ and others pixels $p_j$ in the cell $c_k$. It is used to determine when should pixel $p_i$ leave from others pixels. The similarity function is defined by:

$$f(p_i,c_k)=\frac{1}{n_k}\sum_{p_j\in c_k}\frac{\alpha^2}{\alpha^2+d(p_i,p_j)^2} \quad (4)$$

where $d(p_i,p_j)$ is determined by the contrast between two pixels $p_i$ and $p_i$ in terms of gray level and is defined as:

$$d(p_i,p_j)=\frac{|ng_i-ng_j|}{NG} \quad (5)$$

$NG$ is the number of the gray levels in the image. $\alpha$ represents the mean distance between all pixels and is defined by

```
/* Initialization phase*/
For each pixel $p_i$ do
    Place $p_i$ in a cell of the array
End For

For each ant $a_i$ do
     $a_i$ pick up a randomly chosen pixel
     $a_i$ return to the nest
End For

/* Main loop*/
For $t = 1$ to $t_{max}$ do
  For all ants do
   $a_i$= randomly selected from all ants
   If ($a_i$ carrying pixel $p_i$) then
      Select a cell $c_k$
      Compute $f(p_i, c_k)$ and $p_{drop}(p_i, c_k)$
      Select random reel number $R$ between 0 and 1
      If ( $R \le p_{drop}(p_i, c_k)$ ) then
        Move $a_i$ to the cell $c_k$ and drop pixel $p_i$
     End if

 Else
   If ($a_i$ is free ) then
      $p_i$ = randomly selected free pixel from the array
      $c_k$ = the cell of the pixel $p_i$
      Compute $f(p_i, c_k)$ and $p_{pick}(p_i, c_k)$
       Select random reel number $R$ between 0 and 1
       If ( $R \le p_{pick}(p_i, c_k)$ ) then
         Move $a_i$ to the cell $c_k$ and pick up pixel $p_i$
       End if
   End if
 End if

    Move $a_i$ to the nest
  End For
End For
```

Fig. 1: The AntClust Algorithm

$$\alpha=\frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{j=1}^{N}d(p_i,p_j) \quad (6)$$

The value of $\alpha$ can be calculated before the clustering process.

*The function* $f(.)$ gives its maximum response when $d(p_i,p_j)$ tend to 0.

In the following, we will explain in detail the heuristic and the exact mechanism for picking and dropping a pixel.

**Picking up a pixel:** When the ant is not carrying any pixel, it searches for a possible pixel to pick up. This search is guide by an index table that contains all free pixels (not transported by the ants). This index table is function of the similarity between the gray level of the

pixel and the center of the cluster where it is located, such that the most dissimilar pixels, which are the most farthest from the center of theirs clusters are in the top of the index table. Three cases have to be considered: if the considered pixel $p_i$ is alone in its cell $c_k$ , if it has one pixel with it in the same cell and if there is some others pixels with it in the cell. In the first case, the ant picks up it automatically. In the second case, we have an invalid cluster with only two pixels; the ant will destroy this cluster by picking up the considered pixel with a probability *q*. In the third case, the ant has a high probability to pick up the pixel if its similarity with all the pixels in the cluster is too low (tend to 0).
For picking decision the following probability $p_{pickup}(p_i, c_k)$ is used:

$$p_{pick}(p_i, c_k) = \begin{cases} 1 & \text{if } |c_k| = 1 \\ q & \text{if } |c_k| = 2 \\ \cos^2\left(\frac{\pi}{2} f(p_i, c_k)\right) & \text{otherwise} \end{cases} \quad (7)$$

where *q* is a fixed parameter in [0,1].

**Dropping a pixel:** Once an ant has picks up a pixel $p_i$, it returns with it to the nest then it looks for an interesting new cell where it will drop it. For this purpose a modified version of the short-term memory introduced in[8] and[2] is used. Ants remember a small number of pixels that it has pick up and the cells where it has successfully dropped them. And so, when the ant is carrying a pixel, its memory is consulted in order to select a cell on which it could drop the pixel. It then evaluates locally the suitability of each of the memorized cells as dropping cell for the current carried pixel. For this, the similarity function *f* defined in Eq.3 is applied to each cell and the best interesting cell $c_k$ is the one, at which the similarity function yields the higher value. Instead of just biasing the ants' random walk as in[8] and[2], the ant directly moves from the nest to this cell. The decision whether to drop the pixel is taken probabilistically. The probability $p_{drop}(p_i, c_k)$ for dropping the pixel is given in equation Eq.8

$$p_{drop}(p_i, c_k) = 1 - \cos^2\left(\frac{\pi}{2} f(p_i, c_k)\right) \quad (8)$$

## EXPERIMENTAL RESULTS

The AntClust algorithm was tested on synthetic and real images presented in Fig. 2 and 3. Synthetic images are used because the correct classification and the exact number of classes are known in advance. We can so evaluate the ability of the ant-clustering algorithm to find the correct number of clusters. Moreover, we can evaluate the partition computed by AntClust with the known correct partition.

Table 1 resume the characteristic of each image like the number of clusters (*K*), the number of the gray levels (*NG*). The optimal range of the number of clusters for real images are taken from[13].

Table2 summarized all ants setting used in our experimentations for all test images. Theses setting were selected because they gave good results in preliminary tests.

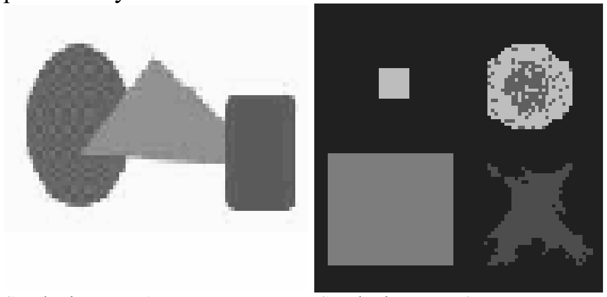
Synthetic Image1 Synthetic Image 2

Fig. 2: Synthetic test images

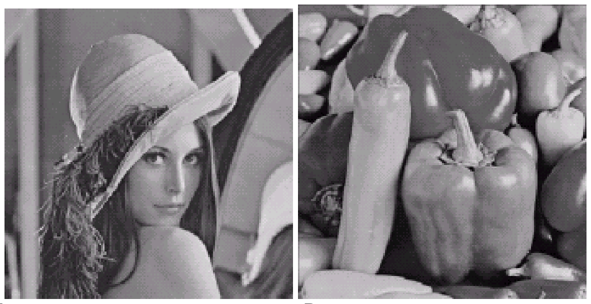
Lenna Peppers

Fig. 3: Real test images

Table 1: Characteristics of test images

| Images | *K* | *NG* |
|---|---|---|
| Synthetic Image 1 | 3 | 46 |
| Synthetic Image 2 | 6 | 117 |
| Lenna | [5,10] | 52 |
| Peppers | [6,10] | 54 |

Table 2: Values of the parameters of AntClust

| Parameter | Value |
|---|---|
| Number of ants *K* | 20 |
| *q* | 0.7 |

The quality of the clustering obtained by AntClust is compared to a classical clustering algorithm; the well-known Kmeans Algorithm. For this purpose, we initialize kmeans with *N* ( number of pixels), 20 and 100 clusters. Starting from a random partitioning, the algorithm repeatedly computed the centers of the clusters and reassigns each pixel to the cluster, which the center is closer to it in term of gray levels.

In order to formally evaluate the quality of the clustering obtained from the two algorithms AntClust and Kmeans on the test images., we used the Rand index[6] and the Rosenberger's measure[14]. The first of the two measures make use of the correct clustering which is known only for synthetic images. They are defined as follows:

**The rand index R:** It determines the degree of pixels (in term of pairwise co-assignments) that are correctly classified according to a segmented image *Seg* and the reference image *Re.* It is defined as:

$$R=\frac{a+d}{a+b+c+d} \tag{9}$$

where *a*,*b*,*c* and *d* are parameters computed for each couples of pixels $p_i$ and $p_j$ as following: if $c_{ref}(i)$, $c_{ref}(j)$, $c_{seg}(i)$ and $c_{seg}(j)$ are the labels of clusters of $p_i$ and $p_j$ in reference image and the segmented one, we have

$$a = \left| \left\{ i, j \backslash c_{ref}(i) = c_{ref}(j) \wedge c_{seg}(i) = c_{seg}(j) \right\} \right|$$

$$b = \left| \left\{ i, j \backslash c_{ref}(i) = c_{ref}(j) \wedge c_{seg}(i) \neq c_{seg}(j) \right\} \right|$$

$$c = \left| \left\{ i, j \backslash c_{ref}(i) \neq c_{ref}(j) \wedge c_{seg}(i) = c_{seg}(j) \right\} \right|$$

$$d = \left| \left\{ i, j \backslash c_{ref}(i) \neq c_{ref}(j) \wedge c_{seg}(i) \neq c_{seg}(j) \right\} \right| \tag{10}$$

*R* takes its value in the interval [0,1] and is to be maximized.

**The Rosenberger's measure[14]:** It combined inter-cluster and inter-cluster disparities. It is defined by the following equation

$$\mathrm{Rosenberg(seg)} = \frac{\overline{D}(seg) + 1 - \underline{\underline{D}}(seg)}{2} \tag{11}$$

The global intra-region disparity $\overline{D}$ quantifies the homogeneity of each class in the segmented image *seg* is defined as:

$$\overline{D}(seg) = \frac{1}{NC}\sum_{k=1}^{NC}\frac{|C_k|}{|seg|}\overline{D}(C_k) \tag{12}$$

And

$$\overline{D}(C_k) = \frac{2}{255}\sqrt{\frac{1}{|C_k|}\sum_{p_i \in C_k} ng(p_i) - \frac{1}{|C_k|^2}\left(\sum_{p_i \in C_k} ng(p_i)\right)^2} \tag{13}$$

where *NC* is the number of clusters, *ng(p<sub></sub>)* is the gray level of the pixel $p_i$

Similarly, the global inter-classes disparity $\underrightarrow{D}$ measures the disparity between the classes. It is given by:

$$\underline{D}(C_k) = \frac{1}{q^{(k)}}\sum_{j=1}^{q^{(k)}}\underline{D}(C_k, C_j) \tag{14}$$

where $q^k$ is the number of clusters $c_j$ that are neighborhood of the cluster $c_k$. The disparity between two clusters is defined by the following equation:

$$\underline{D}(C_i, C_j) = \frac{\left|\overline{C_i} - \overline{C_j}\right|}{NG} \tag{15}$$

where $\overline{C_i}$ is the centroid of cluster $C_i$ and *NG* is the total gray levels present in the image to be segmented. A good clustering minimized the Rosenberger's measure.

Tables 3 gives the means and standard deviation over 50 runs obtained for each of the two measures from the two clustering algorithms on synthetic and real images. Additionally, it gives the average number of clusters. The AntClust algorithm was simulated during 4500 iterations and the number of iterations of the Kmeans algorithm was set to 10.

As can be seen, AntClust algorithm outperforms the three versions of the Kmeans algorithm; both in terms of correct number of clusters, of Rand index and of Rosenberg's measure. In general the kmeans algorithm over estimated the number of clusters which tends to be equal to the number of the gray levels present in the image to be segmented when the number of clusters gives to the algorithm is two high.

Table 3: Results for k-means and AntClust algorithms on synthetic and real images. The table shows means and standard deviations (in brackets) for 50 independent runs

| Synthetic Image 1 | *N*-Kmeans | 20-Kmeans | 100-Kmeans | AntClust |
|---|---|---|---|---|
| #clusters | 46 | 6 | 22 | 3.08 |
| | (0) | (0) | (0) | (0.284) |
| *R* | 0,7342 | 0,9580 | 0,7754 | 0,9897 |
| | (0) | (0) | (0) | (0,433) |
| *Ros* | 0,4888 | 0,3400 | 0,4663 | 0,05082 |
| | (0) | (0) | (0) | (1.729) |
| Synthetic Image 2 | *N*-Kmeans | 20-Kmeans | 100-Kmeans | AntClust |
| #clusters | 117 | 10 | 99 | 6 |
| | (0) | (0) | (0) | (1.05) |
| *R* | 0,6199 | 0,6843 | 0,6266 | 0,8984 |
| | (0) | (0) | (0) | (1.323) |
| *Ros* | 0,4981 | 0,4864 | 0,4979 | 0,04500 |
| | (0) | (0) | (0) | (1.06) |
| Lenna | *N*-Kmeans | 20-Kmeans | 100-Kmeans | AntClust |
| #clusters | 52 | 19 | 52 | 10.09 |
| | (0) | (0) | 0) | (1.789) |
| *Ros* | 0,4882 | 0,4652 | 0,4882 | 0,3313 |
| | (0) | (0) | (0) | (0.456) |
| Peppers | *N*-Kmeans | 20-Kmeans | 100-Kmeans | AntClust |
| #clusters | 54 | 14 | 54 | 7.46 |
| | (0) | (0) | (0) | (1.458) |
| *Ros* | 0,4887 | 0,4334 | 0,4887 | 0,2462 |
| | (0) | (0) | (0) | (0.433) |

## CONCLUSION

In this study, a new warm-based algorithm for image segmentation is presented by simulating the behavior of brood sorting of ants. In AntClust, each pixel is placed in a cell on a discrete array, which represent the environment of the ants. Each ant may pickup a pixel or drops a pixel according to a similarity function that measures the degree of the similarity of a pixel with others pixels in the cluster. In AntClust, we proposed new probabilistic rules for picking and dropping pixels moving and a local moving strategy that have salient effect in fastening the clustering process. Experimental results on synthetic and real images demonstrated the ability of AntClust to extract the correct number of clusters and to give better clustering quality compared to those obtained from a classical clustering algorithm like Kmeans.

## REFERENCES

1. Monga, O. and B. Wrobel, 1987. Segmentation d'images: vers une méthodologie. Traitement du Signal, 4: 169-193.
2. Monmarché, N., 1999. On Data Clustering with Artificial Ants. In: Freitas AA, (ed.), Data Mining with Evolutionary Algorithms: Research Directions – Papers from the AAAI Workshop, 23-26. AAAI Press.
3. Pal, N.R. and S.K. Pal, 1993. A review on image segmentation techniques. Pattern Recognition, 9: 1277–1294.
4. Jain, A., R. Duin and J. Mao, 2000. Statistical pattern recognition: A review. IEEE Trans. on Pattern Analysis and Machine Intelligence, 22: 4-37.
5. Bonabeau, E. and E. Theraulaz, 2000. L'intelligence en essaim. Pour la Science, 282: 66-73.
6. Handl, J., 2003. Ant_based methods for tasks of clustering and topograpgoc mapping: Extensions, analysis and comparison with alternative methods. Master Thesis Germany.
7. Deneubourg, J.L., S. Goss, N. Franks, A. Sendova-Franks, C. Detrain and L. Chretien, 1991. The Dynamic of Collective Sorting Robot-like Ants and Ant-like Robots. In: J.A. Meyer and S.W. Wilson (Eds.), SAB 90-1st Conf. on Simulation of Adaptive Behavior: From Animals to Animats, MIT Press.
8. Lumer, E.D. and B. Faieta, 1994. Diversity and Adaptation in Populations of Clustering Ants. In: D. Cli®, P. Husbands, J. Meyer and S. Wilson (Eds.), From Animals to Animats 3, Proceedings of the 3rd International Conference on the Simulation of Adaptive Behavior, The MIT press/Bradford Books.
9. Langham, A.E. and P.W. Grant, 1999. Using Competing Ant Colonies to Solve k-way Partitioning Problems with Foraging and Raiding Strategies. In: D. Floreano et al. (Eds.), Advances in Artificial Life, Lecture Notes in Computer Science, 1674, Springer, pp: 621–625.
10. Kuntz, P., P. Layzell and D. Snyers, 1997. A Colony of Ant- like Agents for Partitioning in Vlsi Technology. In: P. Husbands et I. Harvey, (Eds.), Proc. of the Fourth European Conf. on Artifcial Life, pp: 417-424. MIT Press.
11. Ramos, V. and J.J. Merelo, 2002. Self-Organized Stigmergic Document Maps: Environment as a Mechanism for Context Learning. In E. Alba, F. Herrera, J.J. Merelo et al. Eds., AEB 2002, First Spanish Conference on Evolutionary and BioInspired Algorithms, pp: 284-293.
12. Deneubourg J.-L., S. Aron, S. Goss and J.M. Pasteels, 1990. The self-organizing exploratory pattern of the argentine ant. Dans J. Insect Behaviour, 3: 159-168.
13. Turi, R.H., 2001. Clustering-based color image segmentation. Ph.D Thesis. Monash University, Australia.
14. Rosenberger, C., 1999. Mise en oeuvre d'un système adaptatif de segmentation d'images Thèse de Doctorat de l'université de Rennes I .