AntClass: discovery of clusters in numeric data by an hybridization of an ant colony with the Kmeans algorithm

N. Monmarché, M.Slimane, G. Venturini

Laboratoire d'Informatique,
École d'Ingénieurs en Informatique pour l'Industrie (E3i),
Université de Tours,
64 av. Jean Portalis, 37200 Tours, France
monmarche,slimane,venturini@univ-tours.fr
Phone: +33-2-47-36-14-14
Fax: +33-2-47-36-14-22

Internal Repport No 213, E3i, January 1999

Abstract

We present in this paper a new hybrid algorithm for data clustering. This algorithm discovers automatically clusters in numerical data without prior knowledge of a possible number of classes, without any initial partition, and without complex parameter settings. It uses the stochastic and exploratory principles of an ant colony with the deterministic and heuristic principles of the Kmeans algorithm. Ants move on a 2D board and may load or drop objects. Dropping or picking up an object on an existing heap of objects depends on the similarity between this object and the heap. The Kmeans algorithm improves the convergence of the ant colony clustering. We repeat two stochastic/deterministic steps and introduce hierarchical clustering on heaps of objects and not just objects. We also use other refinements such as an heterogeneous population of ants to avoid complex parameter settings, and a local memory in each ant. We have applied this algorithm on standard databases and we get very good results compared to the Kmeans and ISODATA algorithms. We have also developed with success a real world application.

Keywords

ant algorithm, unsupervised clustering, numeric data.

AntClass : découverte de classes dans des données numériques grâce à l'hybridation d'une colonie de fourmis avec l'algorithme des centres mobiles

N. Monmarché, M.Slimane, G. Venturini

Laboratoire d'Informatique,
École d'Ingénieurs en Informatique pour l'Industrie (E3i),
Université de Tours,
64 av. Jean Portalis, 37200 Tours, France
monmarche,slimane,venturini@univ-tours.fr
Phone: +33-2-47-36-14-14
Fax: +33-2-47-36-14-22

Rapport Interne No. 213, E3i, Janvier 1999

Résumé

Nous présentons dans ce rapport un nouvel algorithme hybride pour la classification. Cet algorithme découvre automatiquement des classes dans des données numériques sans connaissance a priori du nombre possible de classes, sans partition initiale et sans nécessiter de paramétrage délicat. Il utilise les principes exploratoires stochastiques d'une colonie de fourmis avec les principes heuristiques et deterministes de l'algorithme des centre mobiles. Les fourmis se déplacent sur un tableau à deux dimensions et peuvent saisir et déposer les objets. La saisie et la dépose des objets sur un tas d'objets existant dépend de la similarité entre les objets et le tas. L'algorithme des centres mobiles est utilisé pour améliorer la convergence de la partition élaborée par les fourmis. Nous répétons plusieurs itérations stochastique/déterministe et introduisont une classification hiérarchique sur les tas d'objets et non plus seulement sur les objets seuls. Certains raffinements sont proposés comme l'utilisation d'une population de fourmis hétérogènes afin d'éviter le difficile choix des paramètres ainsi qu'une mémoire pour chaque fourmi. Cette méthode a été appliquée sur des données numériques classiques et nous obtenons des résultats encourageants par rapport aux algorithmes des centres mobiles seul ou par rapport à l'algorithme ISODATA. Une application du monde réel a aussi été traitée avec succés.

Mots clés

algorithme à base de fourmis, classification non supervisée, données numériques.

Contents

1 Introduction

Recently, several papers have highlighted the efficiency of stochastic approaches based on ant colonies for problem solving. This concerns for instance combinatorial optimization problems like the travelling salesman problem (Dorigo and Gambardella 1997), the quadratic assignment problem (Gambardella et al. 1997), routing problems (Di Caro and Dorigo 1997) (Bullnheimer et al. 1997), the bin packing problem (Bilchev 1996) or time tabling problems (Forsyth and Wren 1997). Numerical optimization problems have been tackled also with artificial ants such as in (Bilchev and Parmee 1996), as well as robotics (Deneubourg et al. 1990) (Goss and Deneubourg 1991).

Data clustering is also one of those problems in which real ants can suggest very interesting heuristics for computer scientists. One of the first studies related to this domain is due to (Deneubourg et al. 1990), where a population of ant-like agents randomly moving onto a 2D grid are allowed to move basic objects so as to cluster them. This method has been further developed by (Lumer and Faieta 1994), with simple objects, and in parallel by (Kuntz and Snyers 1994) (Kuntz et al. 1997) where a real clustering problem is studied in order to efficiently resolve an optimization problem.

Based on this existing work, we contribute in this paper to the study of clustering ants from the knowledge discovery point of view, with the aim of solving real world problems. In such problems, we consider that a domain expert has collected a data set and that he would like to be proposed a partitioning of this data set into relevant clusters. This requires to be able to discover the interesting clusters without any initial partition and without knowing how many clusters will be necessary. In addition, we need to handle missing values which occur very often in real world problems. Two other key points in this work are to propose understanble results since we deal with a knowledge discovery approach, and also to avoid complex parameter settings because, most of the time, domain experts are not computer scientists and generally do not know how to tune the parameters of a given classification method.

As will be seen in the following, to achieve this goal we have developed a new ant-based algorithm for clustering, namely AntClass. This algorithm uses more robust "ant-like heuristics" than in the previous approaches. AntClass also introduces hierarchical clustering in a population of artificial ants which will be able to carry heaps of objects. Furthermore, AntClass encompasses an hybridization with the Kmeans algorithm in order to solve various problems inherent to ant-based clustering, like assigning "free" objects or getting a faster and more accurate convergence by removing quickly "obvious" errors. AntClass thus consists of four different steps: ant-based clustering of objects to create an initial relevant partition, followed by the Kmeans clustering, and then the same ant-based step but using hierarchical clustering on heaps of objects previously created rather than single objects, and finally the Kmeans once more. The motivations for this new approach will be given all along the paper.

The remaining of this paper is organized as follows: section 2 describes the ant-based approaches to clustering problems. Section 3 presents the ant colony based heuristics of AntClass. Sections 4 and 5 present respectively the hybrid part of AntClass, i.e. the use of the Kmeans algorithm, and the hiearchical clustering technique. Section 6 describes all the experiments which have been performed with AntClass on artificial and real world data sets. Section 7 concludes on future work and extensions to AntClass.

2 Current artificial ant based approaches to clustering

2.1 Real ants

One basic behavior of real ants consists in carrying various kind of objects like food, eggs or dead bodies, for solving different problems in their real environment.

This may correspond to an individual behavior for carying a single object, like for instance the solitary forraging strategy of *Pachycondyla apicalis* (Fresneau 1985) where ants hunt alone and bring their prey back to the nest.

However, very often "transportation" of objects in real ants may induce much more complex behavoirs which involve distributed cooperation cooperation. For instance, a single object can be too heavy to carry for just one ant. In this case, the ant may recruit other ants for carrying a large prey, and several ants may cooperate together to bring this object back, like for instance the *Ectatomma ruidum* ant (Schaltz et al. 1997).

In the two previous examples, a single object has to be carried only. But real may solve even more complex problems where many objects of different kind have to be carried and clustered together.

2.2 Pionneering work

As far as we know, the initial work in the context of objects clustering with artificial ants has been done by Deneubourg and his colleagues (Deneubourg et al. 1990). In this work, a population of robots are able to cluster objects together by using ant-like heuristics and without any central control.

2.3 Further developments

This seminal work has been used as a basis in all studies using artificial ants to cluster objects.

One first important real world application has been developed in the context of VLSI technology (Kuntz and Snyers 1994) (Kuntz et al. 1997). The problem here is to find a relevant partitionning of a graph. For this purpose, this partionning problem is turned into a clustering problem which is solved by artificial ants. Ants try to pick up/drop objects on a 2D board according to a local density measure of similar objects. Results can be evaluated for instance with a spatial entropy measure.

The second application of clustering ant is the closest to the work presented here. It consists in letting the ants to cluster together a data set (Lumer and Faieta 1994). The objects are initially scattered randomly on a 2D board, and the ants are also using a local density measure of similar objects to take decisions. This work is important because it introduces the concepts of clustering ants in data analysis problems, and AntClass can be viewed as a major extension of it. Among the improvements that we present in this paper, one can notice for instance the following points: introducing more robust ant-like heuristics, dealing with "unassigned objects", speeding up convergence with the Kmeans algorithm, using hierarchical clustering on heaps of objects, testing the resulting algorithm on several real world data sets and providing a successful comparison with the Kmeans and ISODATA algorithms.

Objects	A_1	...	A_k
O_1	1.32	...	3.67
...	...	...	...
O_n	?	...	2.75

Figure 1: The format of the data sets that AntClass will deal with.

3 Basic notations and heuristics of AntClass

3.1 Motivation

We would like here to remind the reader about our motivations for using an ant-based algorithm in a clustering problem. In data clustering, many algorithms (like Kmeans and ISODATA used in the following) require that an initial partition is given as input before the data can be processed. This is one major drawback for these methods, and it is important to notice that ant-based approaches to clustering do not require such an initial partionning. In addition, the same remark can be done with the number of classes. Further more, if this initial information is not consistent with the data, then the algorithm will not find a relevant partition. So one first motivation for using clustering ants is that they do not require initial information about the future classification of the data.

One should also notice that many methods for clustering are entirely based on heuristics, and may run fastly but are very often locally optimal. One way to improve those methods is for instance to introduce a stochastic search rather than a deterministic one, which can be done for instance with a GA (Jones and Beltrano 1991) (Cucchiara 1993) and also with an ant colony. In addition, those algorithms can be easily parallelized.

Finally, one could argue that using artificial ants for clustering, instead of GAs for instance, is sensible because this is one of the problems that real ants have to solve. So in some way, the artificial ant model for clustering is certainly "closer" to the clustering problem than the genetic model, and we are thus expecting it to work better and faster. However, we do not provide in this paper a comparison of AntClass with a genetic approach but rather with Kmeans and ISODATA.

3.2 The objects and their metrics

We assume that a set $E = \{O_1, ..., O_n\}$ of n data or objects has been collected by the domain expert, where each object is a vector of k numerical values $v_1, ..., v_k$ which correspond to k numerical attributes $A_1, ..., A_k$ (see figure 1). We consider in addition the possibility that some attribute values can be missing. These values are thus represented as "?" in the database.

One crucial point inherent to clustering algorithms is how to measure the similarity between two objects. Here we will use by default the euclidean distance between two vectors, denoted by D in the following, but one should notice that AntClass can use any other kind of distance measures, like Minkowski's or Hamming distances. In the following, D_{max} will

denote the maximum distance between two objects of E, i.e.:

$$D_{max} = \max_{O_i, O_j \in E} D(O_i, O_j)$$

Initially all the objects will be scattered randomly on the 2D board described in the next section. Then ants will be able to load or drop objects in order to create heaps of objects, i.e. clusters in our classification task.

3.3 The 2D board

As in previous applications of ant-based algorithms to clustering, we will assume that the ants evolve on a discrete 2D board. This board can be considered as a 2D matrix C of $m \times m$ cells. This matrix is also considered as being toroidal in order to let the ants travel from one side to another easily.

One important point is how to determine the size of the board automatically, especially because we have mentioned in the introduction that we would like to avoid complex parameter settings. The number of possible cells in the matrix has to be at least greater than the number of objects ($m^2 \geq n$). But if the board is too large, the ants will waste a lot of time and will need many moves before encountering an object. So it is reasonnable to say that the size of the board has to be a function of the number of objects in order to scale automatically to the problem size. Since, as far as we know, no experimental or theoretical guidelines exist to determine m as a function of n, we have chosen the following relation between m and n:

$$m^2 = n \times 4$$

It means that, whatever the number of objects, the probability that a randomly moving ant finds an object is indepedent of n. In our case it equals initially 50% because ants can look at the eight cells in their neighborhood. We have also observed experimentally that this method gives good results.

3.4 Heaps of objects

As we will seen in the next section, ants will be able to create, build or destroy heaps of objects. A heap H is considered to be a collection of at least two objects. A heap is located on a given single cell, and is not a spatial pattern as explained in figure 2. The major advantage of this improvement compared to (Lumer and Faieta 1994) is that a heap or cluster can be easily identified, while in previous work spatial patterns of objects may "touch" each others, thus making the identification of clusters difficult. The reader may refer to figure 6 in (Lumer and Faieta 1994) for an explicit example of this phenomenon.

Another advantage of this technique is that it allows us to define more accurate heuristics for dropping or removing objects from a heap, as described extensively in the next section. For instance, ants will be able to remove the most dissimilar object from a heap, or to add a carried object to a heap provided that this object is sufficiently similar to the other objects of the heap.

For this purpose, we have to define the following notations, for a given heap H of n_H objects:

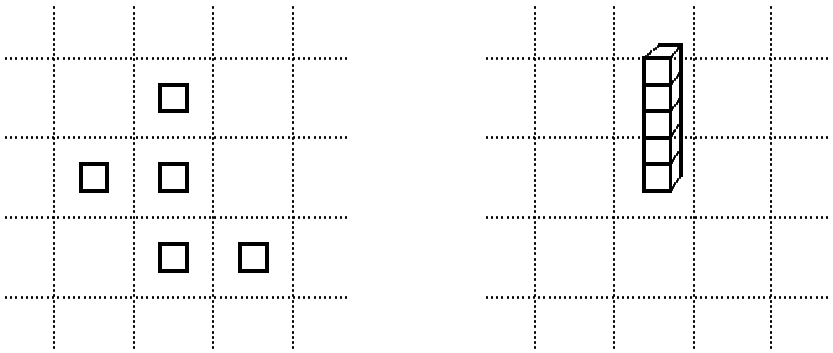

Figure 2: The definition of a cluster has been improved in AntClass. On the left is represented a cluster as in (Lumer and Faieta 1994). A cluster is thus a spatial pattern, but two different clusters can be in contact and may thus be difficult to identify. On the right is given the new representation of a cluster in AntClass which solves the previous problem and allows us to define more robust heuristics.

- $D_{max}(H)$ is the maximum distance between two objects of H:

$$D_{max}(H) = \max_{O_i, O_j \in H} D(O_i, O_j)$$

- $O_{center}(H)$ is the center of mass of all objects in H:

$$O_{center}(H) = \frac{1}{n_H} \sum_{O_i \in H} O_i$$

  Objects in this case are considered as vectors of k numerical values. One should notice also that $O_{center}(H)$ does not correspond to a real object in general,

- $O_{dissim}(H)$ is the most dissimilar object in H, i.e. which maximizes $D(., O_{center}(H))$,

- $D_{mean}(H)$ is the mean distance between the objects of H and the center of mass $O_{center}(H)$:

$$D_{mean}(H) = \frac{1}{n_H} \sum_{O_i \in H} D(O_i, O_{center}(H))$$

3.5 The ants

3.5.1 The colony

The colony consists of p ants $ant_1, ..., ant_p$. Each ant is located on one cell of the board. Initially this position is generated randomly and uniformly. Of course, there is absolutely no central control of the colony.

1. Initialize randomly the ants positions,
2. Repeat
3. For each ant ant_i Do
   (a) Move ant_i,
   (b) If ant_i does not carry any object Then look at the 8 cells in the neighborhood of ant_i location and possibly pick up an object (see algorithm in figure 4),
   (c) Else (ant_i is already carrying an object O) look at the 8 cells around ant_i and possibly drop O (see the algorithm in figure 5),
4. Until Stopping criterion.

Figure 3: The core of the AntClass algorithm.

3.5.2 Main ant-based algorithm in AntClass

The algorithm at the heart of AntClass is represented in figure 3. Initially, the ants are located randomly on the 2D board. Then each ant ant_i performs a move and possibly drop or pick up an object.

The move is not totally random. Initially, ant_i selects a random direction among the 8 possible ones. Then, ant_i has a probability $P_{direction}$ to further continue in this direction when moving next, else it generates randomly a new direction. Each ant also has a *speed* parameter which tells of how many steps it will move in the selected direction before stopping on again.

Once it has moved, the ant may possibly pick up or drop an object as described in the next two sections.

The stopping criterion of this algorithm is simply the number of iterations.

3.5.3 Picking up an object

When the ant is not carrying any object, it looks for a possible object to pick up by considering the 8 cells around its current position. As soon as one object or heap is found, then three cases have to be considered: one object alone, a heap of two objects, a heap of more than two objects. In the first case, the ant has a fixed probability to pick up the object. In the second case, since there are two objects only in the heap, we have the following specific property: $D(O_{dissim}(H), O_{center}(H)) = D_{mean}(H)$. So there is no real heuristic based on distance D to be applied here. This is why we have simply given to the ant a probability $P_{destroy}$ to pick up any of the two objects which results in destroying the heap. In the third case, the ant picks up the most dissimilar object in the heap provided that its "dissimilarity" is above a given threshold T_{remove}. The formula for evaluating the dissimilarity considers the ratio between the distance to center and the mean distance of all objects to the center. It is a simple heuristic but which is powerful because it makes the heap more homogeneous.

1. Label the 8 cells around ant_i as "unexplored",
2. Repeat
   (a) Consider the next unexplored cell c around ant_i with the following order: cell 1 = in front of ant_i, cell 2 = to the left, cell 3 = to the right, etc,
   (b) If c is not empty Then do one of the following action only:
      i. If c contains a single object O, Then load O with a probability P_{load}, Else
      ii. If c contains a heap of two objects, Then remove one of the two objects only with a probability $P_{destroy}$, Else
      iii. If c contains a heap H of more than 2 objects, Then remove the most dissimilar object $O_{dissim}(H)$ from H but provided that:

$$\frac{D(O_{dissim}(H), O_{center}(H))}{D_{mean}(H)} > T_{remove}$$

   (c) Label c as "explored",
3. Until all the 8 cells have been explored or one object has been loaded.

Figure 4: Algorithm for picking up an object.

3.5.4 Dropping an object

When the ant is carrying an object, then it looks also at the 8 cells around its current location. Three cases have to be considered on again: the cell is empty, the cell contains one object only, the cell contains a heap. In the first case, the ant has simply a constant probability P_{drop} to drop the object. In the second case, the ant will drop the object and will thus create a heap of two objects but provided that the carried object is sufficiently similar to the one already in the cell. In the third case, the ant will add its carried object to the heap provided that it is closer to H's center than the most dissimilar object of H.

In order to avoid that an ant carries an object for a too long time, in the case for instance of a very dissimilar object compared to the others, the ant will drop this object automatically after Max_{carry} iterations on the first empty cell it encounters.

3.5.5 Ants local memory

Since real ants have the possibility to memorize several sites in their environment (see for instance (Fresneau 1985)), we have added a memory to each ant in order to speed up the classification, in a similar way to (Lumer and Faieta 1994). Once an ant has encountered any heap H, it stores in its memory the location of H on the board, as well as $O_{center}(H)$ and $D(O_{dissim}(H), O_{center}(H))$. Then, the algorithm of table 5 for dropping an object and also the algorithm that defines the moves performed by the ant are modified as follows: when the ant is carrying an object, it searches in its memory for a heap H on which it could drop the object. If it finds one, then the memory of this heap is activated and the ant will go

1. Label the 8 cells around ant_i as "unexplored",
2. Repeat
   (a) Consider the next unexplored cell c around ant_i with the following order: cell 1 = in front of ant_i, cell 2 = to the left, cell 3 = to the right, etc, and possibly perform one of the following:
      i. If c is empty Then drop O on this cell with a probability P_{drop}, Else
      ii. If c contains a single object O' Then drop O on O' to create a heap but provided that:
      $$\frac{D(O,O')}{D_{max}} < T_{create}$$
      Else
      iii. If c contains a heap H Then drop O on H but provided that:
      $$D(O, O_{center}(H)) < D(O_{dissim}(H), O_{center}(H))$$
   (b) Label c as "explored",
3. Until all the 8 cells have been explored or the carried object has been dropped.

Figure 5: Algorithm for dropping an object. In addition, an ant drops its object on an empty cell if this object has been carried for more than Max_{carry} iterations.

Parameter	Role	Value (or Range)
Speed	# amplitude of moves	$[1, 10]$
$P_{direction}$	prob. to move in the same dir.	$[0.5, 1[$
Max_{carry}	object max. carrying time	$[20, 200]$
P_{load}	prob. to pick up a single object	$[0.4, 0.8]$
$P_{destroy}$	prob. to destroy a heap of 2 objects	$[0, 0.6]$
T_{remove}	min. dissimilarity necessary for removing an object from a heap	$[0.1, 0.2]$
T_{create}	max. dissimilarity permitted for creating a heap of two objects	$[0.05, 0.2]$

Figure 6: The ants parameters in AntClass. All parameters which are crucial for the results are generated randomly within the indicated bounds.

to H location. If it has not dropped the object on its way to H, then the ant will drop the object on H provided that H is still valid, i.e. it has not been destroyed or too much modified by the other ants. Ants have only four slots in their memory. So when a new heap is encountered and if the memory is full, the "oldest heap" in memory is removed and replaced by the new one. Ants may thus forget about a heap.

As a result of this memorization, the ant will increase the proportion of objects which are assigned to a heap, and will also speed up this assigment.

3.5.6 Ants heterogeneous parameters

As mentioned previously, we must absolutely avoid complex parameter settings in order to simplify the use of AntClass by domain experts. This is even more necessary because, as shown in table 6, the parameters that control the ants only are already numerous, without mentionning the other parameters dealing with the Kmeans and also with hierarchical clustering that will be presented in sections 4 and 5.

So the way to solve this problem is on again to get inspired from real ants and to have an heterogeneous population of ants with different behaviors. Setting the same parameters for all ants has two major drawbacks: (1) it is difficult to find the optimal parameters, especially because these depend on the data set being processed and on the properties of the clustering algorithm, and (2) if the parameters are not the optimal ones, then the results have a chance to be poor. This is exactly what happens initially when using ISODATA for instance (see section 6.4).

Initially, the ants parameters will be generated randomly within the bounds represented in figure 6. These values will then be used in this paper for all the tested data sets. These bounds have been found by several trials on the artificial data sets.

Lumer and Faieta have also suggested to use different and non homogeneous parameters. We have generalized this idea to more ants parameters.

1. Take as input the partition P of the data set found by the ants in the form of k heaps: $H_1, ..., H_k$,
2. Repeat
   (a) Compute $O_{center}(H_1), ..., O_{center}(H_k)$,
   (b) Remove all objects from all heaps,
   (c) For each object $O_i \in E$:
      i. Let H_j, $j \in [1, k]$ be the heap which center is the closest to O_i,
      ii. Assign O_i to H_j,
   (d) Compute the resulting new partition $P = H_1, ..., H_{k'}$ by removing all empty clusters,
3. Until Stopping criterion.

Figure 7: The Kmeans algorithm.

3.5.7 Missing values

We have used a very simple and standard method for dealing with missing values. We have replaced the "?" in the database by the mean value of the corresponding attribute.

4 Hybridization

4.1 Motivations

The previous algorithm based on ants only has the major advantage of providing a relevant partition of the data without any initial information about the future classification. However, two important problems remain. The first one is due to the fact that some objects are not assigned to any heaps when the ant algorithm stops, what we call in this paper "free" objects. This corresponds for instance to objects which are still carried by the ants or to objects which are alone on the board. The second problem is that if an object has been assigned to a wrong heap then it can take a long time until the object is transported to the right cluster.

So the solution that we propose is to combine two complementary algorithms, i.e. ant based clustering and the Kmeans algorithm. The first one uses stochastic exploratory principles during the search for relevant clusters, without prior knowledge. The second one uses deterministic/heuristic principles and require an initial partition that is close to the real one. The ant based approach is powerful because it avoids local optima but its convergence is long and difficult to achieve. It is proven mathematically that the Kmeans algorithm converges. We hope that the Kmeans algorithm will remove quickly obvious errors and will provide a fast and efficient heuristic for assigning "free" objects.

4.2 Integrating Kmeans in AntClass

The Kmeans is an iterative algorithm which requires an initial partition of the data. Here, the initial partition is provided by the ants. Then, the Kmeans algorithm proceeds as follows (see figure 7): it computes the center of each cluster, then it computes a new partition by assigning every object to the heap which center is the closest to that object. This cycle is repeated during a given number of iterations or until the assignment has not changed during one cycle.

The experiments described in section 6.2 show that this hybridization is valuable because it effectively removes many misclassification errors. However, the Kmeans algorithm is not optimal and is thus not able to approximate very well the right number of classes. This is the reason why we have introduce the hierarchical clustering described the next section.

5 Hierarchical clustering

5.1 Motivations

We have noticed experimentally that the two previous steps of AntClass (ants + Kmeans) usually give good results in terms of misclassification. However the number of classes is always over estimated. We have observed that these two first steps of AntClass generate many small heaps but which are very homogeneous. So the idea which is presented in the next section consists in considering those small and homogeneous heaps as objects themselves or "building blocks", and to perform another ant-based step but on those newly defined objects.

Since hierarchical clustering is a very standard technique in classification, another motivation was to introduce this technique in order to test if it can be used efficiently with ants.

5.2 Clustering heaps of objects with ants

Let us consider now that the ants+Kmeans hybrid algorithm has lead to the creation of k heaps of objects.

In order to let the ants deal with heaps of objects rather than objects, we have simply adapted the algorithms described previously: ants will be able to carry an entire heap of objects. The algorithm for picking up a heap is globally the same as for objects. Ants will pick up a heap with the same probability P_{load}. We have however added another mechanism in order to avoid that ants carry all heaps at the same time, because as seen in the next paragraph, the number of heaps will only decrease over time. So if only a few clusters remain on the board, it is important not to carry them all the time because in this case ants would not be able to further cluster them if needed. So once a heap has been dropped, it is marked with a kind of pheromone that prevent other ants from picking up this heap for a given time (500 iterations).

Ants drop a heap H_1 onto another heap H_2 provided that:

$$\frac{D(O_{center}(H_1), O_{center}(H_2))}{D_{max}} \leq T_{create}$$

Database	# of instances	# of numeric attributes	% of miss. val.	# of classes
Artif. 1	80	2	0	4
Artif. 2	270	2	0	9
Artif. 3	200	2	0	4
Artif. 4	150	10	0	3
Iris	178	4	0	3
Wine	178	13	0	3
Glass	214	9	0	2-6
Soybean	47	21	0	4
Thyroid	215	5	0	3
Breast cancers	699	9	0.25	2
Soybean 1 %	47	21	1	4
Soybean 5 %	47	21	5	4

Table 1: The 12 databases which are presented in this paper. Soybean 1% means that we have added randomly 1% of missing values in the total database.

When H_1 and H_2 are clustered together, they form only one heap H_3. It means that when two heaps have been stacked onto each others, they cannot be separated any more. This makes the convergence faster.

5.3 AntClass final hybrid and hierarchical algorithm

We have noticed experimentally that the previous step (ants that cluster heaps of objects) approximate well the number of classes but may introduce small misclassification errors. In order to remove these errors, we use once more the Kmeans algorithm on objects as in figure 7.

So AntClass consists mainly in four steps: (1) ant-based algorithm for clustering objects, followed by (2) the Kmeans algorithm using the initial partition provided by the ants, and then (3) ant-based clustering but on heaps previously found, and finally (4) the Kmeans algorithm once more.

Finally, we should add that all values in the data set are normalized in order to avoid any problems of scaling between the different attributes. This is done by considering the values of a given attribute A_i and by mapping these values linearly onto $[0, 1]$.

6 Experimental results

6.1 Experimental settings and databases

We have applied AntClass to several numerical databases, including artificial ones similar to the one used by Lumer and Faieta, real world databases from the Machine Learning repository which are often used as benchmark, and to a real world problem in cooperation with the industry. We will not describe this last real world application because it is confidential,

but we can just mention that the results obtained by AntClass are consistent with those obtained by SAS, which is a very positive results.

The general information about the databases is represented in table 1. As will be seen in the following, "Artif. 1" to "Artif. 4" have been used to evaluate AntClass on databases with known properties. In these artificial databases, examples are generated according to gaussian laws.

In order to evaluate the resulting partition obtained by AntClass (or the Kmeans and ISODATA algorithms in the comparative study of section 6.4), we have set up the following method. We have used in fact databases for supervised learning, i.e. databases where the true classes are known thanks to a "Class" attribute. But when AntClass is used to partition the data, this "Class" attribute is not given to it. It is only a posteriori that the class information is used to evaluate the results.

We have defined two performances measures to evaluate how close is the obtained partition to the real one. The first measure is a classification error rate. It is computed as follows: for a given cluster H obtained by AntClass, consider the majorative class among H according to the "Class" attribute. All objects of H that do not belong to this class are considered as being missclassified. The error rate is simply the ratio between the total number of misclassified objects for all heaps and the total number of objects in E.

The second performance measure is the clusters number.

All runs have been performed on a very standard PC (Pentium 166). One run ends in a few tens of seconds (usually from 10 to 20 seconds). All presented results have been averaged over 50 runs.

Finally, one very important point is that the ants parameters in AntClass were always the same for all the databases, i.e. all generated randomly within the same bounds (see figure 6). Ants were simulated during 2000 iterations when clustering objects, and 50000 iterations when clustering heaps. The number of iterations of the Kmeans algorithm in AntClass was set to 10.

6.2 Results on artificial data bases

These data sets are useful to show experimentally the efficiency of AntClass on data with known properties and difficulty. They also give an empirical motivation for using our hybrid and hierarchical clustering approach.

The results are mentioned in table 2. We have displayed the intermediary results after each of the four steps of AntClass to show the progression in the obtained classification.

The first Ant-based algorithm that clusters objects only finds an initial partition but with classification errors and really too many clusters. The convergence of the stochastic approach would be very long to achieve. One also has to deal with objects that have not been assigned to any heap yet, i.e. objects left alone on the 2D board or object which are still carried by the ants.

At the end of the second step of AntClass, i.e. the use of the Kmeans on the initial partition found in the previous step (see table 2 line 2), the classification errors are reduced but the number of clusters is still really too high. This is due to the fact that the Kmeans algorithm is really sensitive to the initial partition. If this initial partition contains too many clusters, then the final partition is unlikely to be the optimal one. In this step, all objects are assigned to a heap.

Dataset and perf.	1: Ant colony on objects	2: Kmeans on objects	3: Ant colony on heaps	4: Kmeans on heaps
Artif. 1: miscl. err.	11.58 %	0.21 %	0.42 %	0.00 %
# of cla.	8.15	7.76	4.24	4
Artif. 2: miscl. err.	17.24 %	0.52 %	2.22 %	0.00 %
# of cla.	22.30	17.07	10.46	9.02
Artif. 3: miscl. err.	20.35 %	6.32 %	6.93 %	4.66 %
# of cla.	15.06	14.98	5.42	4.42
Artif. 4: miscl. err.	22.23 %	3.32 %	2.68 %	1.33 %
# of cla.	5.22	5.18	2.94	2.96

Table 2: Intermediary and final results obtained on each of the four steps of AntClass for the four artificial databases. "Miscl. err." stands for "misclassifcation error rate" and "# of cla." for "number of classes".

Once the third step has been performed, the ants converge very closely to the right number of classes by working on heaps of objects rather than objects themselves. One can notice that some classification errors remain.

At the end of the fourth step, using the Kmeans once again decreases the classification errors. But this time, since the number of classes and an almost optimal partition have been well determined in the previous step, the Kmeans really finds optimal or near optimal results.

6.3 Results on real world databases

These results are presented in table 3. The good results found on artificial databases are confirmed on real world databases, except maybe for Fisher's Iris database where the class number is correct but not the error rate. One of the three classes, the Setosa class, is completely distinguishable from the two others. The two other classes are known to be more difficult to separate, and this is what increases the misclassification error rate.

On the other databases, all misclassification error rates are very good (see next section for the comparative study). Furthermore, AntClass approximates quite well the real number of classes. For the Glass database, one can define from 2 to 6 classes. AntClass obviously approximate the 6 natural classes, which shows a real regularity in the data set, i.e. several clusters of "isolated" points.

6.4 Comparative study

We describe in this section the results obtained with the Kmeans algorithm and with ISODATA (Ball and Hall 1965). Each algorithm is initialized with 10 classes, and all objects are initially assigned randomly to these classes. The Kmeans algorithm is used with 10 iterations, and has been described previously in section 4.2. ISODATA is an improved version of the Kmeans which can:

- delete classes with less than Min_{obj} objects,

Data set	# of cla. (real)	# of cla. (average)	# of cla. (most freq.)	miscl. err.
Artif. 1	4	4	4	0.00 %
Artif. 2	9	9.02	9	0.00 %
Artif. 3	4	4.42	4	4.66 %
Artif. 4	3	2.96	3	1.33 %
Iris	3	3.02	3	15.4 %
Wine	3	3.06	3	5.38 %
Glass	2-6	7.7	8	4.48 %
Soybean	4	4.82	5	0.13 %
Thyroid	3	3.28	3	6.38 %
Breast cancers	2	4.6	4	3.50 %
Soybean 1 %	4	4.84	5	0.00 %
Soybean 5 %	4	4.44	5	3.02 %

Table 3: Results obtained by AntClass with artificial and real world databases. "Soybean 1%" and "Soybean 5 %" denotes the Soybean database to which we have randomly added respectively 1% and 5 % of missing values.

Data set	# of cla. (real)	# of cla. (average)	miscl. err.
Artif. 1	4	5.63	2.15 %
Artif. 2	9	9.73	12.78 %
Artif. 3	4	7.26	7.30 %
Artif. 4	3	9.60	0.00 %
Iris	3	6.95	4.63 %
Wine	3	8.98	8.57 %
Glass	2-6	7.06	50.16 %
Soybean	4	7.93	3.89 %
Thyroid	3	8.77	8.26 %
Breast cancers	2	9.39	3.68 %

Table 4: Results obtained by the Kmeans algorithm. Here, missing values have been handled with the same method as in AntClass.

Data set	# of cla. (real)	# of cla. (average)	miscl. err.
Artif. 1	4	5.59	1.45 %
Artif. 2	9	6.46	28.22 %
Artif. 3	4	6.70	7.89 %
Artif. 4	3	9.56	0.00 %
Iris	3	6.52	6.93 %
Wine	3	9.36	5.42 %
Glass	2-6	9.39	42.39 %
Soybean	4	8.73	2.72 %
Thyroid	3	7.24	8.23 %
Breast cancers	2	10.06	3.13 %

Table 5: Results obtained by ISODATA.

- split one class into two when the deviation in this class is above Max_{dev},
- cluster together two classes when their distance to each others is less than Min_{dist}.

In the following, we have used $Min_{obj} = 1$, $Max_{dev} = 1.027$ and $Min_{dist} = 0.117$. These last two values have been determined with the data and the knowledge of the "Class" attribute, which is really in favor to ISODATA because this represents an important initial information which has not been given to AntClass. To compute Max_{dev}, the data have been normalized linearly between 0 and 1, and we have computed for each class the deviation of the objects around the center of their real classes. These values have been averaged on all the datasets in order to compute Max_{dev}. Min_{dist} has been computed in the same way by considering the average over each database of the minimum distance between two classes centers.

ISODATA is also run during 10 iterations. We have tried larger numbers of iterations for both algorithms but the results were similar.

Results obtained by the Kmeans and ISODATA are represented respectively in tables 4 and 5. As can be seen, AntClass outperforms the two other algorithms, both in terms of misclassification errors and of correct number of classes. The only exception is for Ficher's Iris database where AntClass finds three classes but makes some classification errors.

In general, Kmeans and ISODATA over estimate the number of classes. But one would expect in this case the misclassification error to be low, but it is not the case.

7 Conclusion

We have presented in this paper a new hybrid and ant-based algorithm named AntClass for data clustering in a knowledge discovery context. The main features of this algorithm are the following ones. AntClass deals with numerical databases. It does not require any initial information about the future classification, such as a an initial partition or an initial number of classes. AntClass introduces new heuristics for the ant colony, and also an hybridization with the Kmeans algorithm in order to improve the convergence. We have also introduced in

AntClass hierarchical clustering where ants may carry heaps of objects and not just objects. Furthermore, AntClass uses an heterogeneous population of ants in order to avoid complex parameter settings to be performed by the domain expert. Finally, AntClass has been tested with success on several databases, including real world ones, and compares favorably with two standard algorithms, namely Kmeans and ISODATA. It has also been applied with success in comparison with SAS in a real world but confidential application.

Future work consists in testing how this model scales with larger databases of several thousands of examples, as in a second real world application that we are actually dealing with. We are also considering other sources of inspiration from real ants for the clustering problem. For instance, ants that meet on the board could exchange objects.

References

Ball G.H. and Hall D.J., ISODATA, a novel method of data analysis and pattern classification, Technical report, Stanford Research Institute, 1965.

Bilchev G. and Parmee I. (1996), Constrained Optimisation with an Ant Colony Search Model, Proceedings of ACEDC'96, Adaptive Computing in Engineering, Design and Control, Second Conference March 1996, Plymouth UK.

Bilchev G. (1996), Evolutionary Metaphors for the Bin Packing Problem, Proceedings of the Fifth Annual Conference on Evolutionary Programming.

Bullnheimer B., Hartl R.F. and Strauss C. (1997), Applying the ant system to the vehicle routing problem, Third International Conference on Metaheuristics.

Cucchiara R. (1993), Analysis and comparison of different genetic models for the clustering problem in image analysis, Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, R.F. Albrecht, C.R. Reeves, et N.C. Steele (Eds), Springer-Verlag, pp 423-427.

Deneubourg J.-L., Goss S., Franks N., Sendova-Franks A., Detrain C. and Chretien L. (1990), The dynamic of collective sorting robot-like ants and ant-like robots, Proceedings of the first Conference on Simulation of Adaptive Behavior 1990, J.A. Meyer et S.W. Wilson (Eds), MIT Press/Bradford Books, pp 356-363.

Di Caro G. and Dorigo M. (1997), AntNet: A mobile agents approach for adaptive routing, Technical Report, IRIDIA 97-12.

Dorigo M. and Gambardella L.M. (1997), Ant colony system: a cooperative learning approach to the travelling salesman problem, *IEEE Transactions on Evolutionary Computation*, 1, 1, pp 53-66.

Forsyth P. and Wren A. (1997), An Ant System for Bus Driver Scheduling, Presented at the 7th International Workshop on Computer-Aided Scheduling of public Transport, Boston.

Fresneau D. (1985), Individual foraging and path fidelity in a ponerine ant, *Insectes Sociaux, Paris*, 1985, Volume 32, n 2, pp 109-116.

Gambardella L.M., Taillard E.D. and Dorigo M. (1997), Ant Colonies for the QAP, Tech. Rep. No. IDSIA 97-4, IDSIA, Lugano, Switzerland.

Goss S. and Deneubourg J.-L. (1991), Harvesting by a group of robots, Proceedings of the first European Conference on Artificial Life 1991, F.J. Varela et P. Bourgine (Eds), MIT press/Bradford Books, pp 195-204.

Jones D.R. et Beltrano M.A. (1991), Solving partitioning problems with genetic algorithms, Proceedings of the fourth International Conference on Genetic Algorithms, 1991, R.K. Belew and L.B. Booker (Eds), Morgan Kaufmann, pp 442-449.

Kuntz P. et Snyers D. (1994), Emergent colonization and graph partitioning, Proceedings of the third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3 (SAB94), D. Cliff, P. Husbands, J.A. Meyer, S.W. Wilson (Eds), MIT-Press, pp 494-500.

Kuntz P., Layzell P. et Snyers D. (1997), A colony of Ant-like agents for partioning in VLSI technology, Proceedings of the fourth European Conference on Artificial Life 19941, P. Husbands et I. Harvey (Eds), MIT press, pp 417-424.

Lumer E.D. et Faieta B.(1994), Diversity and Adaptation in Populations of Clustering Ants, Proceedings of the third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3 (SAB94), D. Cliff, P. Husbands, J.A. Meyer, S.W. Wilson (Eds), MIT-Press, pp 501-508.

Schatz B., Lachaud J.-P. and Beugnon G. (1997), Graded recruitment and hunting strategies linked to prey weight and size in the ponerine ant *Ectatomma ruidum*, Behav. Ecol. Sociobiol. (1997) 40:337-349, Springer Verlag.