# Adaptive clustering with artificial ants

**Diego Alejandro Ingaramo, Guillermo Leguizamón, Marcelo Errecalde**
Lab. de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)*
Universidad Nacional de San Luis - Ej. de los Andes 950 - (D5700HHW) San Luis, Argentina
{*daingara,legui,merreca*} *@unsl.edu.ar*

## Abstract

Clustering task aims at the unsupervised classification of patterns (e.g., observations, data, vectors, etc.) in different groups. Clustering problem has been approached from different disciplines during the last years. Although have been proposed different alternatives to cope with clustering, there also exists an interesting and novel field of research from which different bio-inspired algorithms have emerged, e.g., genetic algorithms and ant colony algorithms. In this article we propose an extension of the AntTree algorithm, an example of an algorithm recently proposed for a data mining task which is designed following the principle of self-assembling behavior observed in some species of real ants. The extension proposed called Adaptive-AntTree (AAT for short) represents a more flexible version of the original one. The ants in AAT are able of changing the assigned position in previous iterations in the tree under construction. As a consequence, this new algorithm builds an adaptive hierarchical cluster which changes over the run in order to improve the final result. The AAT performance is experimentally analyzed and compared against AntTree and $K$-means which is one of the more popular and referenced clustering algorithm.

**Keywords:** computational intelligence, bio-inspired algorithms, clustering, data mining.

## 1 Introduction

Clustering task organizes a collection of patterns usually represented as $n$-dimensional points in different groups according to their similarity. Intuitively, patterns belonging to the same cluster are more similar than those patterns in a different group. Human beings can easily solve 2-dimensional clustering problems, however, the more interesting real problems for clustering involve a large number of dimensions. In addition, the respective problem data are not usually “well” distributed. For that reason there exist plenty of algorithms for clustering which perform differently according to the data set distribution under consideration, i.e., it is not easy to find a general method to cope with the inherent difficulty of the clustering problem.

State-of-the-art clustering techniques are characterized by the data representation, metric used to assess the similarity (or distance) among data, and the way they group the data. The simplest method for clustering, $K$-mean algorithm, needs to set beforehand the number of groups (also called *centroids*) in the data. Each centroid defines a data group and the remaining data are associated to the closest centroid. This process is iteratively repeated by changing the centroids in order to minimize the sum of square errors (SSE) function. The algorithm finishes when no further improvements are observed in SSE. Unfortunately, $K$-means has some drawbacks, for example is mandatory to define in advance the number of clusters which is precisely the information we want to find out. In other words, the performance of this algorithm will strongly depend on the information we have about the data regarding the possible number of clusters.

Is is important to remark that exist plenty of field of application for clustering. Accordingly, it can be found an important number of different methods and techniques for solving this problem. More recently, novel bio-inspired approaches have been successfully applied as alternative methods for clustering, e.g., evolutionary algorithms (EAs) and ant colony optimization (ACO). In addition, metaheuristics as Simulated Annealing and Tabu Search are also being considered for clustering.

In this work we investigate the application of algorithms based on the behavior of real ants (BBA) for clustering problems. In this direction, different examples of this kind of algorithms can be found in recent literature, e.g., *Ant-Class* [1], *Ant-Tree*[2], and *Ant-Clust*[3]. The main objective in this paper is the proposal of a new version of the Ant-Tree algorithm called Adaptive Ant Tree or AAT for short. More specifically, AAT incorporates some dynamic components in the sense that ants are able to disconnect from the tree (a hierarchical clustering structure) and reconnect in a different place, i.e., to reallocate itself on a more similar group or subgroup. This is an important new feature of the approach since it produces an adaptive way of building the tree and eventually obtain a more accurate result regarding both, number of clusters discovered and similarity among elements belonging to the same cluster. The experimental study includes a comparison of AAT against the original Ant-Tree and $K$-means over an number of well known benchmarks. Advantages and disadvantages of ATT with respect to the other algorithms is also analyzed.

The article is organized as follows: in the next section is given a short overview of the bio-inspired techniques for clustering. Section 3 describes the Ant-Tree algorithm in which we based our proposal presented in section 4. In section 5 we describe the experimental study and the respective results. Finally, section 6 discusses strengths and weaknesses of the proposed approach including some possible future directions of research in this area.

## 2 Clustering through bio-inspired techniques

Clustering problems have been approached by different techniques, in particular bio-inspired approaches, e.g., evolutionary algorithms [4, 5], and other metaheuristics such as tabu search and simulated annealing [6]. In addition, ACO metaheuristic [7] has been adapted for clustering problems as described in [8].

Inside of the class of bio-inspired algorithms it is worth remarking that BBA algorithms have showed to be effective in comparison with other traditional techniques since BBA algorithms include probabilistic rules for clustering assignation which avoid local optima. Consequently, better results can be obtained without any knowledge or information about the data at hand. As examples of BBA algorithms is the Ant-Class algorithm [1] which applies explorative and stochastic principles from the ACO metaheuristic combined with deterministic and heuristic principles from $K$-means. The simulated environment is a 2-dimensional grid on which the ants leave/take objects on/from the ground according to their similarity. On the other hand, in the Ant-Clust [3] algorithm the ants proceed according to chemical properties and odors to recognized themselves as similar or not. Other interesting algorithm is the ACODF [4], a novel approach which represents a direct adaptation of the ACO metaheuristic for solving clustering problems.

Finally, we describe the AntTree algorithm which behaves following the self-assembling characteristics observed in some ant species. In this case, the ants build “living” structures with their bodies in order to solve different problems dealing with the survival of the whole colony, e.g., build a living bridge to let the colony follow a way from/to the nest to the food source. This principle of behavior is adapted for solving clustering problems in the following way: each ant (data) from de colony (the data set to analyze) tries to connect to another ant according to their similarity. Repeating this process, AntTree builds a tree structure representing a hierarchical partition of the original data set. AntTree has been previously compared against $K$-means and AntClass [2]. AntTree showed to be effective and promising approach for being considered for future research in developing new clustering techniques. In the following section we describe in detail the AntTree algorithm highlighting its main features, finally we present our proposal based on the AntTree, the AAT algorithm.

## 3 The AntTree algorithm

The design of the AntTree algorithm [2] is based on the self-assembly behavior observed in certain species of ants. As an example we can mention the Argentine ant *Linepithema humiles* and African ant *Oecophylla longinoda* where the living structures are used as bridges or auxiliary structures for building the nest. The structure is built in a incremental manner in which the ants joint a fixed support or another ant for assembling. The computational model implemented in the AntTree builds a tree structure representing a hierarchical data organization partitioning the whole data set. Each ant represents a single datum from the data set and it moves in the structure according to the similarity with the other ants already connected in the tree under construction (see [2] for a detailed discussion about the biological concepts used in the design of the AntTree).

In order to make a partition of the whole data set it is built a tree structure where each node represents a single ant and each ants represents a data single data. The clustering task in this context means to make the more appropriate decision when determining the place in which each ant will be connected, either to the main support (generates a new cluster) or to another ant (refines an existing cluster).

In order to apply AntTree, it is supposed that is possible to define a similarity function $Sim$ among any pair of data elements, i.e., if $N$ is the cardinality of the data set and $(d_i,\, d_j)$ is an arbitrary pair of data, $i \in [1, N]$, $j \in [1, N]$, the value of $Sim(i, j) \in [0, 1]$ could represent the degree of similarity between $d_i$ and $d_j$. When $d_i$ and $d_j$ are completely different, $Sim(d_i, d_j) = 0$. On the other hand, $Sim(d_i, d_j) = 1$ when $d_i = d_j$.

The general principles of the algorithm are the following: each ant represents a node to be connected to the tree, i.e, a data to be classified. Starting from an artificial support called $a_0$, all the ants incrementally will connect either to that support or to other ants already connected. This process continues until all ants were connected to the structure, i.e., all data where already clustered. The resulting organization of this structure will depends directly on $Sim$ definition and the local neighborhood of each moving ant.

To each ant $a_i$ will be associated the following terms:

1. The *ingoing links* of $a_i$, called $\mathcal{I}(a_i)$ represent the set of links toward ant $a_i$, that is, the children of node $a_i$.

2. An *outgoing link* of $a_i$, called $\mathcal{O}(a_i)$ which represents either the support or another ant, i.e., the parent node.

3. A data $d_i$ represented by $a_i$.

4. Two metrics called respectively *similarity threshold* ($T_{Sim}(a_i)$) and *dissimilarity threshold* ($T_{Dissim}(a_i)$) which will be locally updated during the process of building the tree structure.

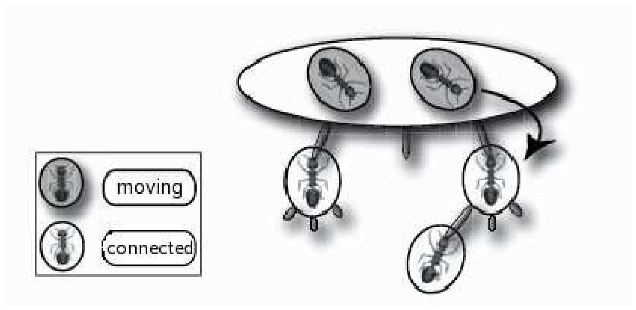

Figure 1: General outline for building the tree structure by self-assembling artificial ants (adapted from [2])

Figure 1 shows a general outline of self-assembling process involving artificial ants. It can be observed that each ant $a_i$ is either of the two following situations:

1. *Moving on the tree*: a walking ant $a_i$ (gray highlighted in figure 1) can be either on the support ($a_0$) or another ant ($a_{pos}$). In both cases, $a_i$ is not connected to the structure. Consequently, it will be free of moving to the closest neighbors connected to either $a_0$ or $a_{pos}$. In figure 2 is showed the neighborhood corresponding to an arbitrary ant $a_{pos}$.

2. *Connected to the tree*: in this case $a_i$ has already assigned a value for $\mathcal{O}(a_i)$, therefore, it can not move anymore. Additionally, an ant is not able to have more than $L_{max}$ ingoing links $(|\rangle(a_i)| \leq L_{max})$. The objective is to bound the maximum number of incoming links, i.e., the maximum number of clusters.

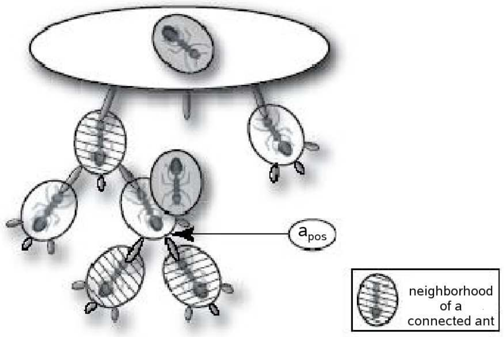

Figure 2: Neighborhood corresponding to an arbitrary ant $a_{pos}$ (adapted from [2])

In order to understand completely the AntTree algorithm, the next section presents a detailed description of this algorithm.

### 3.1 Detailed description of the AntTree algorithm

The main loop implemented in the AntTree algorithm is showed in figure 3. The very first step involves the allocation of all ants on the tree support and their respective thresholds of similarity and dissimilarity are accordingly initialized. In this stage the whole collection of ants are represented by a list $\mathcal{L}$ of ants waiting to be connected in further steps. List $\mathcal{L}$ can be sorted in different ways, later on this article we make some considerations concerning different possibilities of arranging $L$. During the process of construction each selected ant $a_i$ can be either connected to the support (or another ant) or moved on the tree looking for a correct place to connect itself. Simulation will continue while exist ants moving on the tree until all of them found the more adequate assembling place; either on the support (figure 4) or on another ant (figure 5).

```
Let $\mathcal{L}$ be a list (possibly sorted) of ants to be connected
    Initialize: Allocate all ants on the support.
    $T_{Sim}(a_j) \leftarrow 1$ and $T_{Dissim}(a_j) \leftarrow 0$, for all ant $a_j$
    Repeat
        1. Select an ant $a_i$ from list $\mathcal{L}$
        2. If $a_i$ is on the support ($a_0$)
                then support case (see figure 4)
                else ant case (see figure 5)
    Until all the ants are connected to the tree
```

Figure 3: Main loop of the AntTree

When $a_i$ is on the support and $a_i$ is the first ant considered, the situation is easier since this ant is connected directly to the support. Otherwise, $a_i$ is compared against $a^+$, the more similar ant to $a_i$ among all ants connected directly to the support. If these ants are similar enough, then $a_i$ will move to the subtree corresponding to $a^+$. In case that $a_i$ and $a^+$ are dissimilar enough (according to a dissimilarity threshold), $a_i$ is connected directly to the support. This last action generates a new subtree (i.e., a new cluster) due to the incoming ant is different enough from the other ants connected directly to the support. Finally, if $a_i$ is neither similar or dissimilar enough, the respective thresholds (similarity and dissimilarity) are updated in the following way:

$$T_{Sim}(a_i) \longleftarrow T_{Sim}(a_i) * 0.\ 9$$
$$T_{Dissim}(a_i) \longleftarrow T_{Dissim}(a_i) + 0.\ 01$$

The above updating rules let ant $a_i$ be more "tolerant" when considered in a further iteration, i.e., the algorithm increases the probability of connecting this ant in a future time.

```
If no ant is connected to the support then connect $a_i$ to $a_0$
  else
    Let $a^+$ be the more similar ant to $a_i$
    where $a^+$ is connected to $a_0$
    (a) If $Sim(a_i, a^+) \geq T_{Sim}(a_i)$ then move $a_i$ toward $a^+$
    (b)   else
            i. Si $Sim(a_i, a^+) < T_{Dissim}(a_i)$ then
                 /* $a_i$ is dissimilar enough to $a^+$ */
                 connect $a_i$ to $a_0$ (in case there is no
                 more links available in $a_0$, then move
                 $a_i$ toward $a^+$ and decrease $T_{Sim}(a_i)$)
            ii. else decrease $T_{Sim}(a_i)$ and
                 increase $T_{Dissim}(a_i)$
            /* $a_i$ is more tolerant */
```

Figure 4: Support case

The second case to be considered is when $a_i$ is on another ant, e.g., $a_{pos}$ in figure 5. If $a_i$ is *a*) similar enough to $a_{pos}$, *b*) dissimilar enough to the ants connected to $a_{pos}$, and *c*) there exists an available incoming link ($|\mathcal{I}(a_i)| < L_{max}$), then $a_i$ is connected to $a_{pos}$, i.e., $a_i$ will be the root of a new subtree connected to $a_{pos}$. The main difference between $a_i$ and the other ants connected to $a_{pos}$ is that $a_i$ represents a new subcluster similar to the cluster represented by $a_{pos}$ but dissimilar to the other subclusters hanging from $a_{pos}$. In case the above conditions are no met, $a_i$ is moved randomly on any neighbor of $a_{pos}$. At the same time, its respective thresholds are updated accordingly as explained before. The algorithm finishes when all ants have been connected to the tree structure.

```
Let $a_{pos}$ be the ant on which $a_i$ positioned.
Let $a_k$ be a random neighbor of $a_{pos}$
1. If $Sim(a_i, a_{pos}) \geq T_{Sim}(a_i)$ then
     Let $a^+$ be more similar to $a_i$ connected to $a_{pos}$
     (a) If $Sim(a_i, a^+) \leq T_{Dissim}(a_i)$ then connect $a_i$
           to $a_{pos}$ /* In case no more incoming links
           are available, move randomly $a_i$ toward $a_k$ */
     (b) else decrease $T_{Dissim}(a_i)$, increase $T_{Sim}(a_i)$
,            and move $a_i$ toward $a_k$
2. else move $a_i$ toward $a_k$
```

Figure 5: Ant case

Before finishing the description of the algorithm, it is important to highlight the arrangement of the ant on list $\mathcal{L}$ (the initial step). Since the algorithm proceeds iteratively taking the ants from list $L$, the features of the first ants on the list will significantly influence the final result. The strategy that showed the best results [2] is by making an increasing sorting of list $\mathcal{L}$ according to the average similarity with the remaining ants. In this way, the first ant connected to the tree will be the closest to its own group and most dissimilar to the remaining ants.

We should emphasize that the tree obtained following the above procedure is different from the popular dendrograms (the result of applying some traditional clustering techniques). In our case. each node corresponds to a single data from the whole data set. Therefore, the resulting tree can be interpreted (see figure 6) as a data partition (considering each ant connected to $a_0$ as a different group) as well as a dendrogram where the ants in the inner nodes could move to the leaves following the more similar nodes to them. Thus, it is possible a comparisson between Antree against other hierarchical clustering method by using the last procedure.

Finally, it is worth remarking that the AntTree algorithm includes many interesting features as avoiding local minima due to its probabilistic behavior. In addition, it produces very accurate results without using any previous information of the possible distribution of the data set (i.e., there

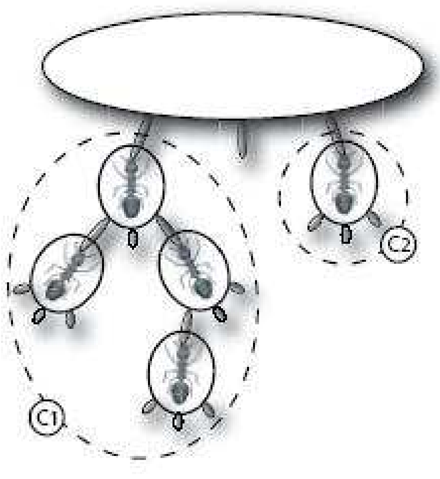

Figure 6: A resulting tree interpreted as a data partition without any hierarchy (adapted from [2])

is no need of having information of the existing groups in the data set).

## 4 The Adaptive AntTree algorithm

In this section, a new method for clustering called *Adaptive AntTree* (or *AAT* for short) is presented. AAT extends the AntTree algorithm by allowing a more flexible behaviour of the ants. The ants in AAT are able of changing a previous assigned location in the tree structure and consequently they can be reallocated in a other, more adequate group. Another improvement introduced by AAT is the capability of exploring alternative solutions and selecting the one is the best fit to the data set.

AAT generates a model by means of a process consisting of three main stages: **1) AntTree execution** (with modified parameters), **2) reallocation of ants** and **3) joining groups**. Each of these stages is outlined in figure 7.

```
1. Execute the AntTree algorithm (with
   modified parameters).
2. Repeat until all ants belong to an adequate group
     a. Selection of ants to be disconnected from the tree
     b. Sorting of the ants list (in decreasing order)
     c. Reallocate the ants (AntTree with a branch
        of tree asigned in advance)
3. Repeat $L_{max}$ times or until no unions are possible
     a. Selection of two groups to be combined
     b. Join process
     c. Model evaluation
```

Figure 7: A general overview of AAT algorithm

### 4.1 AntTree execution

The first stage of the AAT algorithm is addressed to achieve a considerable number of small groups which should have the following two characteristics: a) the elements in the group exhibit a very high cohesion and b) the groups are well-separated. These two characteristics were achieved by running the AntTree with a small modification in the updating formula corresponding to the dissimilarity threshold of each ant[1]:

$$T_{Dissim}(a_i) \longleftarrow T_{Dissim}(a_i) + 0.\ 2$$

In addition, the branching factor of the tree was increased to $L_{max} = 20$, so that a greater number of clusters could be generated.

### 4.2 Reallocation of ants

This iterative process considers each ant in turns and attempt to find out a more adequate group where the ant should be reallocated. The process stops when all of ants have been analyzed. As can be observed in figure 7, stage 2 consists of three elemental steps: a) *selection* of ants to be disconnected from the tree, b) *sorting* of the ants list (in decreasing order), and c) *reallocation* of ants.

In step a), each ant in the tree structure is evaluated in order to deciding if it should be disconnected from the tree or not. Such decision is based on the *Silhouette* function [9] which estimates the membership degree of an arbitrary ant respect to the group under consideration. The *Silhouette* function is defined as follows:

$$s(i) = (b(i) - a(i))/max(a(i), b(i)), \qquad (1)$$

where $a(i)$ is the distance between ant $a_i$ and the mean value of the group that $a_i$ belongs, $b(i)$ is the minimum distance between ant $a_i$ and the mean values of the remaining groups, and $-1 \leq s(i) \leq 1$. A threshold $t = t_0$ must be defined to detect the assignation of an ant $a_i$ to a wrong group, i.e., when $s(i) < t$. This means that a "better" [2] group has been detected for ant $a_i$ and consequently this ant should be selected to be disconnected from the tree. If we assume that $\mathcal{L}$ represents the list of ants selected in the previous step to be desconnected, an important issue is to take into account the decreasing *sorting* of list $\mathcal{L}$ according to the similarity with the remaining ants (step b). Note that in this case, the criterion adopted for sorting the list is diametrically oppossed to that used by the AntTree

[1]The simmilarity threshold $T_{Sim}$ was not modified.

[2]A group which have a media value closer to ant $a_i$ than the media value of the group which $a_i$ belongs at the present moment.

algorithm. Therefore, the first place on the list will be occupied by the ant which is *more similar* to the remaining ones. In order to understand the reasons behind this criterion we need to consider some situations arising when the previous selection process is carried out. When some the ants of a group are removed, this group can eventually becomes empty. Therefore, it should not be considered in the future and those ants that were just assigned to the group recently eliminated have to be reallocated to another group[3]. In that sense, if we use a decreasing sorting of the list, the most similar ant on the list will have an important influence on the media value of the group that it was assigned to and those ants which are similar to this ant also will be assigned to the same group.

With respect to the **reallocation of ants** (step c), it is important to observe that this step involves two different processes. The first one is the **verification of pre-assigned groups**, i.e., verify if some group has become empty. In this case, a new existing group is assigned. The second one is the **reallocation of the ant** to its new group. This process is carried out by a simple movement of the ant toward the corresponding new group, i.e., toward the first ant connected to the support.

### 4.3 Joining groups

The goal of this stage is find out the real groups present in the data set. The general process consists in iteratively combine the most similar groups and evaluate each new model obtained. The result of this process will be the best adapted model to the data under consideration.

Pair of groups to to be combined are chosen according to their similarity (in mean values). The join process is very similar to the second stage explained above. The ants belonging to one of two selected groups are incorporated into the other one. It is worth remarking that it is not necessary any sorting of these ants because all of them have been assigned before to the same group.

For each iteration, the resulting model is evaluated and compared against the best model obtained to this moment. If the new model is more accurate than the best model so far, the former is saved to be compared in subsequent iterations. The number of iterations depends on the number of groups analyzed. However, this number will not be greater than $L_{max}$ for any size and/or type of data set.

Finally (last step in stage 3) it is necessary to asses each new model found with respect to the best model so far. An interesting function defined for this end, is that based on the *Davies-Boulding index* [10] which is defined tacking into account a similarity value $R_{ij}$ between two groups $C_i$ y $C_j$. The $R_{ij}$ value is defined considering the deviation estimate inside each group with respect to a similarity factor between both groups. $R_{ij}$ is restricted to be positive and symetric [4]. For example, a simple equation satisfying these conditions is the following:

$$R_{ij} = (s_i + s_j)/d_{ij} \tag{2}$$

where the deviation estimate $s_i$ is defined as the maximum distance between the mean value of $C_i$ and $d_m \in C_i$; $d_{ij}$ is the distance between the mean values of $C_i$ and $C_j$, respectively. Finally, if $n_c$ is the number of clusters, the Davies-Boulding (DB) index is defined as:

$$DB_{n_c} = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i \tag{3}$$

where $R_i = \max_{j=1\ldots k,\ i\neq j} R_{ij}$. Intuitively, we can say that this function estimates the average similarity between each group and its most similar. Thus, an ideal clustering algorihthm should cluster the data set in a way that groups are as different as possible, i.e., the lower values of $DB_{n_c}$ the better is the model achieved.

## 5 Experimental results

The AntTree and AAT algorithms used in the experiments were implemented in JAVA. The set of tools for clustering tasks provided by the WEKA [11] package also was used for experimenting with the $K$-means method. Additionallly, AntTree and AAT were included as alternative data mining tools in the WEKA package.

As similarity function, it was used the *Gower's function*:

$$Sim(i,j) = \frac{\sum_{k=1}^{p} w_{ijk}s_{ijk}}{\sum_{k=1}^{p} w_{ijk}} \tag{4}$$

where $s_{ijk}$ is the similarity between an element $i$ and other element $j$ respect to the $k$-th attribute. When all attributes are categorics, they are compared by equality ($s_{ijk} = 1$ if the elements are equal and 0 in other cases). When continuous data have to be considered, $s_{ijk}$ is defined as $s_{ijk} = 1 - |x_{ik} - x_{jk}|/R_k$ where $R_k$ is the difference beetween the maximum and minimum observed value corresponding to $k^{th}$ attribute. The $w_{ijk}$ factor will be 1 if the values for attribute $k$

[3]The reallocation process simply allocates each ant to the group which have the closest media value to the ant considered.

[4]The following restrictions have to be fulfiled: a) $R_{ij} \geq 0$, b) $R_{ij} = R_{ji}$ and c) if $s_i = 0$ y $s_j = 0$ then $R_{ij} = 0$

are comparable, otherwise, it will be 0. For example, if the value of atribute $k$ is missing either for element $i$ or $j$, $w_{ijk} = 0$.

The algorithms were compared over four real data bases: *breast-cancer-wisconsin* (bcw), *eucalyptus* (euca), *heart-disease* (heart), and *thyroid-disease* (thyroid) taken from the UCI repository [5]. Also, it were generated three artificial data bases: *art1*, *art2* and *art3*. These data bases differ with respect to the extent of proximity between the groups inside them: *art1* includes well-separated groups, *art2* has also a clear separation between groups but it is smaller than in *art1*. Finally, in *art3* the existing groups are very close to each other.

The main features of the data bases (*Name*) are summarized in table 1. In each case is specified the number of instances (*NI*), number of attributes (*NA*), type of attributes (*TA*, categoric or numerical), number of records with missing values (*MV*), number of classes (*K*), and distribution of instances over each class(*DI*).

| Name | *NI* | *NA* | *TA* | *MV* | *K* | *DI* |
|---|---|---|---|---|---|---|
| bcw | 699 | 9 | cat | 16 | 2 | $\langle 458, 241\rangle$ |
| euca | 736 | 19 | cat num | 141 | 5 | $\langle 180, 107, 130, 214, 105\rangle$ |
| heart | 303 | 13 | cat num | 0 | 5 | $\langle 164, 55, 36, 35, 13\rangle$ |
| thyroid | 1505 | 5 | num | 0 | 3 | $\langle 1050, 245, 210\rangle$ |
| $art_i$, $i = 1, 2, 3$ | 1002 | 31 | num | 0 | 3 | $\langle 334, 334, 334\rangle$ |

Table 1: Main features of experimental data bases

For each instance is known in advance the cluster it belongs to [6]. Therefore, it is possible evaluate the algorithms with respect to the classification error measure *Ec* proposed in [2]. In this case, $k_i$ represents the true class of the object $d_i$, $k'_i$ is the class found it by a particular method for $d_i$. $K$ corresponds to the real number of classes and $K'$ is the number of classes found by an arbitrary method. Now we can define *Ec* as:

$$Ec = \frac{2}{N(N-1)} \sum_{(i,j)\varepsilon\{1,.......N\}^2, i<j} \epsilon_{ij} \qquad (5)$$

wherw:

$$\epsilon_{ij} = \begin{cases} 0 & \text{if } (k_i = k_j \wedge k'_i = k'_j) \vee (k_i \neq k_j \wedge k'_i \neq k'_j), \\ 1 & \text{otherwise.} \end{cases}$$

It should be noted that the real number of clusters ($K$) has to be specified as an additional parameter when the $K$-means method is used. The remaining methods do not know this value, hence, they can not take any advantage of this valuable information.

In table 2 comparative results obtained with $K$-means, AntTree and AAT are showed. The table includes information about the classification error measure $Ec$ (equation 5), number of clusters found $K'$, running time (in seconds) of the algorithm, and the value from Davies-Boulding (DB) function (equation 3). It is also showed the *average global Silhouette* based on equation 1 and defined as: $GS = (\sum_{i=1}^{N} s(i))/N$. In all cases, the value $K'$ of $K$-means corresponds to the value $K$ given as parameter.

| Name | Alg. | Ec | $K'$ | Time | DB | *GS* |
|---|---|---|---|---|---|---|
| bcw | $K$-means | 0.04 | 2 | 0.38 | 1.89 | 0.58 |
| | AntTree | 0.21 | 10 | 0.94 | 4.85 | 0.4 |
| | AAT | 0.11 | 2 | 15.5 | 1.56 | 0.47 |
| euca | $K$-means | 0.31 | 5 | 0.59 | 2.38 | 0.27 |
| | AntTree | 0.78 | 1 | 2.03 | 0 | 0 |
| | AAT | 0.27 | 13 | 2.44 | 1.02 | 0.64 |
| heart | $K$-means | 0.33 | 5 | 0.34 | 2.38 | 0.24 |
| | AntTree | 0.65 | 1 | 0.47 | 0 | 0 |
| | AAT | 0.37 | 2 | 0.61 | 1.87 | 0.29 |
| thy-roid | $K$-means | 0 | 3 | 0.45 | 1.85 | 0.62 |
| | AntTree | 0.47 | 1 | 3.39 | 0 | 0 |
| | AAT | 0.31 | 2 | 2.83 | 1.63 | 0.68 |
| $art_1$ | $K$-means | 0. 28 | 3 | 1.41 | 3.37 | 0.41 |
| | AntTree | 0. 22 | 2 | 4.8 | 0.48 | 0.72 |
| | AAT | 0 | 3 | 4.7 | 0. 31 | 0.87 |
| $art_2$ | $K$-means | 0 | 3 | 0.61 | 0.42 | 0.83 |
| | AntTree | 0. 67 | 1 | 4.75 | 0 | 0 |
| | AAT | 0 | 3 | 4.61 | 0.42 | 0.83 |
| $art_3$ | $K$-means | 0.28 | 3 | 0.89 | 4.07 | 0.1 |
| | AntTree | 0. 67 | 1 | 4.67 | 0 | 0 |
| | AAT | 0 | 3 | 4.7 | 0. 38 | 0.83 |

Table 2: Results for different data bases

In the following a descriptive analysis of the obtained results is presented:

- With respect to the classification error $E_c$, it can be observed that AAT performed better than AntTree for all data sets tested. In addition, the performance of AAT is not degraded as can be observed for the AntTree when the clusters are not well-separated in the data sets (e.g., euca, heart, $art2$, and $art3$). In general, AAT obtained similar results to those obtained by $K$-means, however, AAT is superior for some data sets: euca, $art1$, and $art3$.

- When the groups in data sets are not well-separated (e.g., artificial data bases), the AAT algorithm finds the optimal solutions.

- AAT tends to find a number of clusters $K'$ closer to the real number of cluster $K$ when compared to AntTree. For instance, for four

[5]Machine Learning repository, University of California, Irvine (UCI) [12].

[6]Obviously, this information is not available to the algorithms.

out of seven data bases considered, the $K'$ found by AAT was equal to $K$.

- The running times of the proposed approach AAT are not significantly larger than those corresponding to the AntTree algorithm.

- The metrics $DB$ (the lower the better) and $GS$ (the greater the better) for assessing the quality of the clustering achieved show that AAT has obtained the best models with respect to the other algorithms considered (i.e., AntTree and $K$-means) over almost all data bases. More specifically, the index $DB$ for AAT is better than the respective index for $K$-means in all cases. The same situation is observed when AntTree yielded an index $DB \neq 0$. On the other hand, it can be observed that in terms of $GS$ the performance of AAT is superior to AntTree and $K$-means, except for data base bcw, for which $K$-means obtains the best result.

- Although AAT and AntTree do not need the number of cluster to proceed with the clustering task, the results obtained by these algorithms are similar or even better than $K$-means for some of the metrics considered here.

## 6 Conclusions

This paper presented a new clustering algorithm based on the AntTree algorithm, an approach inspired on the self-assembling behavior observed in some species of real ants. The proposed algorithm uses (with some parameters modified) the original AntTree algorithm as the first stage. Then, a re-assignation of ants to different clusters and a combination of those clusters is carried out to give a better clustering model.

The remarkable characteristic of AAT is given by its capacity of disconnecting ants from the tree under construction without increasing significantly the running time. Since the resulting clustering from AAT is structurally the same as that obtained by the AntTree, it is possible to visualize it as either a partition clustering or hierarchical clustering. Finally, it is important to note that AAT is robust approach in view of different data set distributions since it proceeds starting from small groups toward bigger ones by joining them based on their similarity.

## References

[1] N. Slimane, N. Monmarché, and G. Venturini. Antclass: discovery of clusters in numeric data by an hybridization of an ant colony with kmeans algorithm. Rapport interne 213, Laboratoire d ' Informatique de l ' Universit e de Tours, E3i Tours,, 1999.

[2] H. Azzag, N. Monmarche, M. Slimane, G. Venturini, and C. Guinot. Anttree: A new model for clustering with artificial ants. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 2642–2647, Canberra, 8-12 December 2003. IEEE Press.

[3] N. Labroche, N. Monmarché, and G. Venturini. AntClust: Ant Clustering and Web Usage Mining. In *Genetic and Evolutionary Computation Conference*, pages 25–36, Chicago, 2003.

[4] Cheng-Fa Tsai, Chun-Wei Tsai, Han-Chang Wu, and Tzer Yang. Acodf: a novel data clustering approach for data mining in large databases. *J. Syst. Softw.*, 73(1):133–145, 2004.

[5] Alex A. Freitas. A survey of evolutionary algorithms for data mining and knowledge discovery. In A Ghosh and S Tsutsui, editors, *Advances in Evolutionary Computation*, pages 819–845. Springer-Verlag, August 2002.

[6] Hussein Abbass, Charles Newton, and Ruhul Sarker. *Data Mining: A Heuristic Approach.* Idea Group Publishing, Hershey, PA, USA, 2002.

[7] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evolutionary Computation*, 1(1):53–66, 1997.

[8] V.K. Jayaraman P.S. Shelokar and B.D. Kulkarni. An ant colony approach for clustering. Technical report, Chemical Engineering and Process Division, National Chemical Laboratory, India, 2003.

[9] F. Azuaje N. Bolshakova. Improving expression data mining through cluster validation. 2003.

[10] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. Clustering validity checking methods: Part II. *SIGMOD Record*, 31(3):19–27, 2002.

[11] Ian H. Witten and Eibe Frank. *Data mining: practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[12] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. *University of California, Irvine, Dept. of Information and Computer Sciences*, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.