Pattern Clustering via Ants Colony, Ward Method and Kohonen Maps

Rosangela Villwock1; Maria Teresinha Arns Steiner2; Paulo Henrique Siqueira3

1Universidade Estadual do Oeste do Paraná (UNIOESTE)
Rua Universitária, 2069, Bairro Universitário, Cep 85.819-110, Cascavel, PR
Universidade Federal do Paraná (UFPR): Graduate Programs in 1,2,3 Numerical Methods in Engineering (PPGMNE) and 2,3 Insdustrial Engineering (PPGEP); 3Graphical Expression Department;
2Pontifícia Universidade Católica do Paraná: Graduate Program in Industrial Engineering and Systems (PPGEPS); P.O. Box: 19081; Zip: 81531-990; Curitiba, PR

Summary
The collective and self-organization behaviors of social insects have inspired researchers to reproduce them. Methods inspired by ants are a great promise for clustering problems. In the clustering algorithm based on Ants, patterns are spread out on a grid and each ant has assigned a pattern. The ants are responsible for picking, transporting and dropping patterns on the grid. After the algorithm converges, the recovery of clusters is done using the patterns' positions on the grid. The purpose with this paper is to present changes and improvements to the Ant-based Clustering algorithm originally proposed by [2], hereinafter called proposed algorithm, evaluating its performance relatively to the Ward Method, the Kohonen Maps and the ACAM algorithm (Ant-based Clustering Algorithm Modified), proposed by [1]. The major changes in the proposed algorithm were: the introduction of a comparison between the probability of dropping a pattern at the position chosen randomly and the probability of dropping this pattern at its current position; the introduction of an evaluation of the probability of a neighboring position when the decision to drop a pattern is positive and the cell in which the pattern should be dropped is occupied; and the substitution of the pattern carried by an ant in case this pattern is not dropped within 100 consecutive iterations. To assess the performance of the proposed algorithm three real and public databases were used (ÍRIS, WINE and PIMA Indians Diabetes). The results showed superior performance of the proposed algorithm over the ACAM for two of the three databases and equality with other methods.

Keywords:
Data Mining; Metaheuristics; Ant-Based Clustering.

1. Introduction

Many researchers have focused their attention on a new class of algorithms called metaheuristic. According to [7], a metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide range of different problems.
A particularly promising metaheuristic was inspired by the behavior of real ants. Starting with the Ants System, several algorithmic approaches based on these ideas were developed and implemented with considerable success for a variety of combinatorial optimization problems, academic and real [7].
Ant colony optimization – ACO is a metaheuristic in which the artificial ants colony cooperates to find good solutions to difficult discrete optimization problems [7]. Dorigo, Caro and Gambardella [5] present an assessment of recent works about ant algorithms for discrete optimization and introduce the ACO metaheuristic. Dorigo and Blum [4] present a research about the theoretical results of the optimization of ant colonies.
According [1], many researchers have applied the Ants Colony Optimization mechanism based on the ideas of Dorigo and Stützle to many combinatorial optimization problems and then extended it to an entire class of optimization problems.
Socha and Dorigo [18] present an extension of the ACO for continuous domains. In this article, the authors show how the ACO, initially developed for combinatorial optimization, can be adapted to continuous optimization with no conceptual changes in its structure. The authors present the general idea, the implementation and results obtained, which were compared with other methods for continuous optimization.
According to Dorigo, Maniezzo and Colorni [8], in choosing a path an ant is influenced by the intensity of the pheromone trails. A higher level of pheromone gives an ant stronger stimulus and thus, higher likelihood to choose it. The result is that an ant will find a stronger trail on shorter paths. As a result, the number of ants that follow these paths will be higher. This will cause the amount of pheromone on the shortest path to grow faster than on the longest, so the probability with which any ant chooses a path to follow rapidly tends to the shorter. The end result is that very rapidly all ants will choose the shortest path.
The Traveling Salesman Problem – TSP is a problem that is quite studied in the literature. The TSP also has an important role in the research of ACO: the first ACO algorithm, called Ant System - AS was first tested in the TSP [7].

Manuscript received June 5, 2012
Manuscript revised June 20, 2012

The Ant Colony System – ACS described by Dorigo and Gambardella [6] differs from the AS in three main aspects. First, it explores more closely the search experience accumulated by ants. Second, the evaporation of pheromone and pheromone deposit only occur in arches that belong to the best way so far. Third, each time an ant uses an arch it removes some of its pheromone, thus increasing the exploration of alternative pathways [7]

The study of ant colonies has given outstanding contribution, not only in combinatorial optimization, but also offering new ideas for clustering techniques [1]. Ant-based Clustering was initially proposed by [2]. In contrast to the ACO, no artificial pheromone is used and the environment itself serves as a stigmergy variable [9].

Among the behaviors of social insects the most widely recognized is the ability of ants to work together in order to develop a task that could not be performed by a single agent. Also seen in human society, this ability of ants is a result of cooperative effects. These cooperative effects have recourse to the fact that the effect of two or more individuals or coordinated parts is higher than the total of their individual effects. The high number of individuals in ant colonies and the decentralized approach for coordinated tasks (performed simultaneously) mean that ant colonies show high levels of parallelism, self-organization and fault tolerance. These characteristics are desired in modern optimization techniques [1].

The Clustering algorithm based on Ant Colonies was chosen for this study, analysis and new proposals due to several factors. First, it is a relatively new metaheuristic and has received special attention, mainly because it still requires much investigation to improve its performance, stability and other key characteristics that would make such an algorithm a mature tool for data mining [1]. Moreover, this algorithm can automatically "find" the number of clusters within patterns.

The purpose with this paper is to present an algorithm, hereinafter called proposed algorithm, through changes and improvements to the Ant-based Clustering algorithm originally proposed by [2], evaluating its performance relative to the Ward Method, the Unidimensional Kohonen Maps and the ACAM algorithm (Ant-based Clustering Algorithm Modified) proposed by [1].

The use of the method from the Multivariate Statistics area (Ward Method) is justified because it is one of the most established methods in the literature [14]. On their turn, the One-dimensional Kohonen Maps were used because, as the Ant-based Clustering, they simultaneously perform the clustering and topographic mapping tasks. And finally, the ACAM because it is a method similar to the proposed algorithm, very recent.

This paper is structured as follows: section 2 presents a theoretical background to the Ant-based Clustering, describing the algorithm, how to carry out the cluster recovery and some possible measures to evaluate clusters. Section 3 presents the databases used, computational implementation details for the methods that were used, as well as the major contributions (modifications and improvements) for the Ant-based Clustering. Section 4 presents the results and discussions and finally, section 5 presents the final considerations.

2. Theoretical Foundation

In the Ant-based Clustering proposed by [2], ants were represented as simple agents that moved randomly on a square grid. The patterns were scattered within this grid and the agents (ants) could pick, transport and drop them. These operations are based on similarity and on the density of the patterns distributed within the local neighborhood of agents, isolated patterns - or those surrounded by dissimilar ones - are the most likely to be picked and then dropped in a neighborhood of similar ones.

The decisions to pick and drop patterns are made by the P_{pick} and P_{drop} probabilities given by equations (1) and (2), respectively.

$$P_{pick} = \left(\frac{k_p}{k_p + f(i)} \right)^2 \quad (1)$$

$$P_{drop} = \left(\frac{f(i)}{k_d + f(i)} \right)^2 \quad (2)$$

In these equations, *f(i)* is an estimate of the fraction of the patterns located in the neighborhood and that are similar to the current Ant's pattern, and k_p and k_d are real constants. In [2], the authors used $k_p = 0.1$ and $k_d = 0.3$. The authors obtained the estimate *f* through a short-term memory of each ant, in which the content of the last cell in the analyzed grid is stored. This choice of the neighborhood function *f(i)* was primarily motivated due to its ease of implementation with simple robots.

Lumer and Faieta (1994, *apud* [11]) introduced a number of modifications to the model, which allowed the manipulation of numeric data, and improved the quality of the solution as well as the algorithm's convergence time. The idea was to define a measure of similarity or dissimilarity between patterns, since in the algorithm initially proposed objects were similar if objects were identical and dissimilar if objects were not identical. In that work, topographic mapping has its first appearance. The general idea with this algorithm is to have similar data in the original *n*-dimensional space in neighboring regions of the grid, this is, data that are neighbors on the grid indicate similar patterns on the original space.

In this work, the decision of picking patterns is based on the P_{pick} probability given by equation (1) above, and the decision to drop patterns is based on the probability P_{drop}

given by equation (3) below, where *f(i)* is given by equation (4).

$$P_{drop} = \begin{cases} 2f, & if\ f(i) < k_d \\ 1, & otherwise \end{cases} \quad (3)$$

$$f(i) = \max\left\{0, \frac{1}{\sigma^2} \sum_{j \in L} \left[1 - \frac{d(i,j)}{\alpha}\right]\right\} \quad (4)$$

In equation (4), $d(i, j)$ is a function of dissimilarity between patterns i and j belonging to interval [0,1]; α is a scalar parameter that dependents on the data (patterns) and belongs to the interval [0,1]; L is the local neighborhood of size σ^2, where σ is the perception radius (or neighborhood). Lumer and Faieta (1994, *apud* [11]) used $k_p = 0.1$, $k_d = 0.15$ and $\alpha = 0.5$ in their work.

The Ant-based Clustering algorithms are mainly based on the versions proposed by [2] and Lumer and Faieta (1994, *apud* [11]). Several modifications were introduced to improve the quality of a cluster and, in particular, the spatial separation between the clusters on the grid [1].

Changes that improve the spatial separation of clusters and allow a more robust algorithm were introduced by [11]. One of them is the restriction on the *f(i)* function, given by equation (5) below, which serves to penalize high dissimilarities.

$$f^*(i) = \begin{cases} \frac{1}{\sigma^2} \sum_{j \in L} \left[1 - \frac{d(i,j)}{\alpha}\right], & if\ \forall j \left(1 - \frac{d(i,j)}{\alpha}\right) > 0 \\ 0, & otherwise \end{cases} \quad (5)$$

According [25], a difficulty in applying the Ants Clustering algorithm to complex problems is that in most cases they generate a number of clusters that is much larger than the real one. Moreover, usually these algorithms do not stabilize in a cluster solution, this is, they constantly construct and deconstruct clusters during the process. To overcome these difficulties and improve the quality of results, the authors proposed an Adaptive Ant Clustering Algorithm - A^2CA. A modification included in the present approach is a cooling program for the parameter that controls the probability of ants picking up objects on the grid.

2.1 Parameters of the Neighborhood Function

The clusters' spatial separation on the grid is crucial so that individual clusters are well defined, thus allowing their automatic recovery. Spatial proximity, when it occurs, may indicate the premature formation of the cluster [11].

Defining the parameters for the neighborhood function is a key factor in cluster quality. In the case of the σ perception radius it is more attractive to employ larger neighborhoods to improve the quality of clusters and their distribution on the grid. However, this procedure is computationally more expensive, once the number of cells to be considered for each action grows quadratically with the radius, while it also inhibits the rapid formation of clusters during the initial distribution phase. A radius of perception that gradually increases in time accelerates the dissolution of preliminary small clusters [11]. A progressive radius of perception was also used by [25].

Moreover, after the initial clustering phase, [11] replaced the scalar parameter $\frac{1}{\sigma^2}$ by $\frac{1}{N_{occ}}$ in equation (5), where N_{occ} is the number of grid cells occupied, observed within the local neighborhood. Thus, only similarity (not density) was taken into account. In his ACAM algorithm, [1] proposed to replace the scalar $\frac{1}{\sigma^2}$ in equation in (5) by the scalar $\frac{\sigma_0^2}{\sigma^2}$, in which σ_0 is the initial radius of perception.

According to [11], α determines the percentage of patterns on the grid that are rated as similar. The choice of a very small value for α prevents the formation of clusters on the grid. On the other hand, choosing too large a value for α results in cluster merging.

Determining parameter α is not simple and its choice is highly dependent on the data set's structure. An inadequate value is reflected by an excessive or extremely low activity on the grid. The amount of activity is reflected by the frequency of successful operations of an ant in picking and dropping. Based on these analyses, [11] proposed an automatic adaptation of α. Boryczka [1] proposed a new scheme for adapting the value of α.

Tan, Ting and Teng [20] examined the scalar parameter of dissimilarity in Ant Colonies approaches for data clustering. The authors show that there is no need to use an automatic adaptation of α. They propose a method to calculate a fixed α for each database. The value of α is calculated regardlessly of the clustering process.

To measure the similarity between patterns, different metrics are used. Handl, Knowles and Dorigo [11] use Euclidean distance for synthetic data and cosine for real data. Boryczka [1] tested different dissimilarity measures: Euclidean, Cosine and Gower measures.

2.2. The Basic Algorithm Proposed by [2]

In an initial phase of the basic algorithm proposed by [2] all patterns are randomly scattered on the grid. Then, each ant randomly chooses a pattern to pick and is placed at a random position on the grid.

In the next phase, called the distribution phase, in a simple loop each ant is randomly selected. This ant travels the grid running steps of length L in a direction randomly determined. According to [11], using a large step size

speeds up the clustering process. The ant, then, probabilistically decides if it drops its pattern at that position.
If the decision to drop the pattern is negative another ant is randomly chosen and the process starts over. If the decision is positive, the ant drops the pattern at its current position on the grid, if it is free. If this grid cell is occupied by another pattern, the pattern must be dropped at an immediate free neighboring cell through a random search.
The ant then seeks for a new pattern to pick. Among the free patterns on the grid, this is, patterns that are not being carried by any ant, the ant randomly selects one, goes to its position on the grid, evaluates the neighborhood function and probabilistically decides if it picks this pattern. This choosing process of a free pattern on the grid runs until the ant finds a pattern that should be picked. Only then this phase is resumed, choosing another ant, until a stop criterion is satisfied.

2.3. Cluster Recovery

Hierarchical clustering methods include techniques that hierarchically seek clusters and therefore, admit obtaining multiple clustering levels. The hierarchical methods can be subdivided into agglomerative and divisive. The hierarchical agglomerative method considers, in principle, each pattern as a cluster and iteratively groups the pair of clusters with greater similarity into a new cluster until it forms a single cluster containing all patterns. The divisive hierarchical method, in contrast, begins with a single cluster and performs a process of successive subdivisions [3].
This work will address only the agglomerative hierarchical method. The process begins with each pattern forming a cluster. After calculating the distances between all clusters, the two clusters with smaller distance should be connected. The most common types of connections are: Simple Link, Complete Link, Average Link and the Ward Method [14].
The distances between clusters are defined in terms of their distances on the grid. Each pattern is now composed of only two attributes that position it on the two-dimensional grid. The distance between any two patterns is then the Euclidean distance between two grid points. This process repeats until a stop criterion is satisfied.
When patterns around the clusters' edges are isolated, [11] introduced a weight that encourages the fusion of these patterns with the clusters.

2.4 Clustering Evaluation

In the evaluation of clusters different aspects can be observed: determine the clustering trend of a set of data, compare results of a clusters analysis with results externally known, assessment of how well the results of a clusters analysis fit the data without reference to external information, compare the results of two different sets of clusters analysis to determine which one is better, or even determine the correct number of clusters [19].
According to [19], the numerical measures applied to assess different aspects of cluster evaluation are classified into three types: external ones that are used to measure the extent to which cluster labels correspond to class labels provided externally, the internal one that is used to measure how good the clustering structure is unrelated to external information and the relative, which is used to compare two different clusters.
Boryczka [1] used two internal indices (the Intra-Cluster Variance and the Dunn Index) and two external indices (Measure *F* and the Random Index). These last two measures are described here because they were used in this paper as well.
Measure *F* uses the idea of accuracy and memory of information retrieval. Each class i is a set of n_i desired patterns; each cluster *j* (generated by the algorithm) is a set of n_j patterns; n_{ij} is the number of patterns in class i belonging to cluster *j*. For each class *i* and cluster *j*, precision *p* and memory *r* are defined as $p(i,j)=\frac{n_{ij}}{n_j}$ and $r(i,j)=\frac{n_{ij}}{n_i}$, respectively. The value of measure *F* is given by equation (6).

$$F=\sum_i \frac{n_i}{n}\max_j\{F(i,j)\} \tag{6}$$

where:

$$F(i,j)=\frac{(b^2+1).p(i,j).r(i,j)}{b^2.p(i,j)+r(i,j)}$$

The value of *b* should be "1" to give equal weight to precision *p* and memory *r*. In equation (6), *n* is the size of the dataset. Measure *F* is limited to the interval [0, 1] and should be maximized.
In its turn, the Random Index (*R*) is given by equation (7), where *a*, *b*, *c* and *d* are calculated for all possible pairs of *i* and *j* patterns and their respective clusters *U* (correct classification - $c_U(i)$ and $c_U(j)$) and *V* (solution generated by the clustering algorithm - $c_V(i)$ and $c_V(j)$). Measure *R* is limited to the interval [0, 1] and should be maximized.

$$R=\frac{a+d}{a+b+c+d} \tag{7}$$

where:

$$a = \left|\left\{i, j \mid c_U(i) = c_U(i) \wedge c_V(i) = c_V(i)\right\}\right|$$
$$b = \left|\left\{i, j \mid c_U(i) = c_U(i) \wedge c_V(i) \neq c_V(i)\right\}\right|$$
$$c = \left|\left\{i, j \mid c_U(i) \neq c_U(i) \wedge c_V(i) = c_V(i)\right\}\right|$$
$$d = \left|\left\{i, j \mid c_U(i) \neq c_U(i) \wedge c_V(i) \neq c_V(i)\right\}\right|$$

2.5 Other Clustering Methods Used

In this work, as already mentioned, three methods were selected for comparison with the algorithm here proposed: Ward Method (classical statistical method); One-dimensional Kohonen Neural Networks (performs topographic mapping and clustering simultaneously) and the ACAM (analogous to the method proposed here). All three methods are briefly described below.

2.5.1 Ward Method

According to [14], the Ward Method used in this work connects two clusters based on "information loss". The sum of the square error (SSE) is considered the criterion for "information loss". For each cluster *i*, the cluster's mean (or centroid) is calculated, as well as the sum of the square error of cluster *i* (SSE_i), which is the sum of the square error of each pattern in the cluster in relation to the mean value. For *k* clusters there are SSE_1, SSE_2, ..., SSE_k, where SSE is defined by equation (8).

$$SSE = SSE_1 + SSE_2 + ... + SSE_k \quad (8)$$

For each pair of clusters *m* and *n*, the mean (or centroid) is first calculated for the cluster formed (cluster *mn*). Next, the sum of square error for cluster *mn* (SSE_{mn}) is calculated according to equation (9). The *m* and *n* clusters that show the smallest increase in the sum of square error (*SSE*) (smallest "loss of information") will be merged.

$$SSE = SSE_1 + SSE_2 + ... + SSE_k - SSE_m - SSE_n + SSE_{mn} \quad (9)$$

2.5.2 One-dimensional Kohonen Neural Networks

According to [10], in 1982 Teuva Kohonen developed the method of self-organizing maps that makes use of a topological structure to cluster units (patterns). Self Organizing Maps - SOM, also known as Kohonen Neural Networks, form a class of neural networks in which learning is unsupervised.

According to [12] the main purpose with the Kohonen Neural Networks is transform input patterns of arbitrary dimension into a discrete map. The neurons are placed at the nodes of a grid, which can have any number of dimensions. Usually two-dimensional grids are used (called 2D-SOMs). There are also the 1D-SOMs (used here) and 3D-SOMs, which use grids (or maps) of one and three dimensions, respectively.

The learning process of a Kohonen Neural Network is based on competitive learning in which the grid's output neurons compete to be activated. The output neuron that wins the competition is called the winning neuron. All neurons on the grid should be exposed to a sufficient number of input patterns to ensure proper ripening of the self-organization process [12].

According to [12], besides the competition process to form the map, the cooperation and adaptation processes are also essential. In the cooperation process, the winning neuron locates the center of a topological neighborhood of cooperative neurons. For the self-organization process to occur the excited neurons have their synaptic weights set in the adaptation process. The adjustment made is such that the winning neuron's response to the application of a similar input pattern is enhanced.

According to [17], several error measures can be used to determine the quality of a map. In his work, the author uses the quantization error, which represents the average error that corresponds to the difference between the patterns and the weights of the winning neurons, the topological error, which represents the percentage of winning neurons that lack the second winner in a neighborhood of unitary radius centered on the winning neuron and the square mean error.

There are several approaches to variants of Kohonen Neural Networks. The algorithms, inspired by the original, modify some aspects as, for instance, neighborhood criterion, how to choose the winning neuron, the use of hierarchical maps and accelerated learning, among others [15].

2.5.3 ACAM Method

Boryczka [1] presented a modification of the clustering algorithm proposed by Lumer and Faieta. To increase the robustness of the clustering based on ants, the author has incorporated two major changes compared to the classical approach: 1. an adaptive perception scheme occurred in the density function and 2. a cooling scheme of α-adaptation, this is, a cooling scheme for the adaptation of parameter α, modifications already mentioned in section 2.1.

3. Material and Methods

The databases used in this paper, which were meant to compare the methods mentioned here, were: Iris, Wine and Pima Indians Diabetes, available at http://mlearn.ics.uci.edu/databases. Table 1, at the end, shows the number of patterns, the number of attributes and the number of clusters for each one of these databases. The data were standardized before the clustering methods were

applied. Standardization was done by dimension.

3.1 Ward Method

The Ward Method [14] was applied to the three databases with the aid of the computer software *MatLab*2008 [16]. The number of clusters (last column of Table 1) was provided for the evaluation of this method. The dissimilarity measurement used was the Euclidean distance because it is the best known of the dissimilarity measures and because it has been employed in previous works for all methods used here.

3.2 Kohonen Maps

Clustering with One-dimensional Kohonen Maps applied to the databases was implemented in computer software *MatLab*2008 [16] and ran 10 times for each database. Implementation details can be obtained in [21].

In this method it is necessary to set the number of neurons, which was defined here as being equal to the number of clusters (k). The parameters for this method were defined as follows: the initial learning rate = 0.5, the minimum learning rate = 0.05, initial neighborhood radius = max [1, ¼ k], initial synaptic weights $\in$[0, 1], maximum number of iterations N = 500. At each iteration all patterns are presented to the network at random. The stopping criterion was defined as the maximum number of iterations. In the implemented algorithm, two phases (initial and final) were defined in which the parameter settings are modified. The initial phase was defined as $t_{inicial}$ = 0.2 N. In the final phase, the initial neighborhood radius is equal to the radius of the neighborhood at the end of the first phase.

The training involves all competitive, cooperative and adaptive phases in which each pattern must be presented to the network. In the competitive stage, the distances of the pattern to all neurons (Euclidean distance) are computed and the winning neuron is determined. In the cooperative phase, winning neuron's neighbors are located and in the adaptive phase, the synaptic weights of neurons neighboring the winning neuron are updated. The updating of synaptic weights was made according to equation (10) with neighborhood function defined by equation (11). This update takes into consideration the distance from the neighbor to the winning neuron and the learning rate.

$$\underline{w_j}(n+1) = \underline{w_j}(n) + \eta(n).h_{ji(x)}(n) \,.\, (x - w_j(n)) \quad (10)$$

$$h_{ji(x)}(n) = e^{\left(\frac{d^2_{ji}}{2\sigma(n)^2}\right)} \quad (11)$$

Then, the learning rate and the neighborhood radius should be updated according to equations (12) and (13), respectively.

$$\eta(n) = \eta_0 e^{\frac{n}{\tau_2}} \quad (12)$$

$$\sigma(n) = \sigma_0 e^{\frac{n}{\tau_1}} \quad (13)$$

In these equations, $\tau_1 = \dfrac{N}{\log(\sigma_0)}$ and $\tau_2 = N$, where N is the maximum number of iterations and σ_0 is the neighborhood's initial radius. These values were defined based on the values used by [12], where τ_2 = 1000.

3.3 ACAM Method

This method, unlike the others (Ward, Kohonen and proposed), was not implemented. The comparison was made directly to the results presented in [1].

3.4 Proposed Algorithm

The proposed algorithm, based on the basic algorithm by [2] and presented in section 2.1, was implemented with computer software *MatLab*2008 [16]. For this, the LCPAD computational grid resources were used: Central Laboratory for High Performance Processing/UFPR, partially financed by FINEP project CT-INFRA/UFPR/Modeling and Scientific Computing.

The implemented algorithm used the number of iterations as stopping criterion and the algorithm was run 10 times. Here, n is the number of patterns and m is the number of attributes, the number of iterations N_{max} was set to N_{max} = 500.$n.m$. Several tests were performed to set the maximum number of iterations. In the implemented algorithm two phases (initial and final) were defined in which the parameter settings are modified. The initial phase was defined as $t_{inicial} = 0.2.N_{max}$.

In defining the size of the grid, the number of cells was chosen as 10 times the number of patterns and 10 ants were used (p=10), as in [11]. It was observed that changing these values is not essential for the clustering process and, for this reason, the same values were used. A square neighborhood was used when searching for pattern's neighbors.

As in [11], the initial neighborhood radius was defined as being equal to "1", with the use of value increment during the initial phase. Since an equation for the increase of this value was not explicitly found in other studies, this was done according to equation (14), where t is the current iteration of the initial phase. During the final phase, this value decreases in 0.05 per 100 substitutions of the pattern carried by an ant (a suggested modification that is detailed below). The value of the neighborhood radius is always the integer value smaller than or equal to the one defined in any of the phases. This automatic adjustment during the final phase aims to "relax" the neighborhood size when ants are not being able to drop the patterns they carry.

$$\sigma = 4^{\frac{t}{t_{inicial}}} \quad (14)$$

In defining the neighborhood to calculate the probability of dropping a pattern in its current position and to calculate the probability of picking a pattern, the neighborhood's radius was always considered equal to "1". In seeking a new position, the direction of the step is random. Set the direction, the maximum possible size of the step is calculated. A random number belonging to interval [0, 1] was used to determine this size, multiplying this number by the maximum step size.

The used probabilities of picking (p_{pick}) and dropping (p_{drop}) are described by equations (1) and (2), respectively, where $k_p = 0.1$ and $k_d = 0.3$, the same values used by [2]. A pattern is picked if probability p_p is greater than a minimum picking value ($pick_{min}$). A pattern is dropped if probability p_d is greater than a minimum dropping value ($drop_{min}$).

The values of $drop_{\min}$ and $pick_{\min}$ were set to 0.13397 during the initial phase. This value was defined by making the picking probability (p_{pick}) equal to the dropping probability (p_{drop}). Figure 1, at the end, shows the graph of the picking and dropping probabilities. The definition of a random value greater than 0.13397 during the final phase was made to restrain the change in position during this phase without "immobilizing" the process, however. The definition of a value that increases in time in the long term would prevent the ants from moving the patterns.

To calculate function f, function f^* was used, defined by equation (5) already presented and proposed by [11], substituting the scaling parameter $\frac{1}{\sigma^2}$ by $\frac{1}{N_{occ}}$, where N_{occ} is the number of grid cells occupied within the local neighborhood, as shown in (15).

$$f*(i) = \begin{cases} \frac{1}{N_{occ}} \sum_{j \in L} \left[1 - \frac{d(i,j)}{\alpha} \right], \; if \; \forall j \left(1 - \frac{d(i,j)}{\alpha} \right) > 0 \\ 0, \qquad otherwise \end{cases} \quad (15)$$

Parameter α_0, after some preliminary tests, was set to 0.8. Its updating, during the initial phase, was defined according to equation (16); for the final phase, this value decreases to 0.001 every 100 substitutions of the pattern carried by an ant. This decrease during the final phase was made to prevent ants to fail to move their patterns.

$$\alpha = \alpha_0 + \frac{2t}{p.t_{inicial}} - 0,01 \quad (16)$$

It is noted that any change in the values of k_p, k_d and α directly influences the clustering process. It was decided to keep the values of k_p and k_d and use only an adaptation for α. If the values of k_p and k_d are changed, the adaptation for α, as well as the values $pick_{min}$ and $drop_{min}$, should be reviewed.

When a pattern is dropped onto the grid a new pattern should be picked. The search for this pattern is random, but each free pattern is evaluated only once, until all of them has been evaluated. If no pattern shows probability p_{pick} greater than $pick_{\min}$, the pattern that has the highest p_{pick} probability is picked.

When a pattern has no neighbors, function f is set to zero. This makes probability p_{d} equal to "0", this is, the pattern should not be dropped at that position, and probability p_{p} equal to "1", this is, the datum must be picked and later leave this position.

The dissimilarity measure used was the Euclidean distance. The distance matrix was calculated according to equation (17) and then was standardized. In this equation, the weight refers to the attribute and is calculated by dividing the standard deviation by the average, calculated for each attribute of the data matrix already standardized (Q).

$$d(i,j) = \sum_{a=1}^{m} \left[\left(Q(a,i) - Q(a,j) \right).weight(a,1) \right]^2 \quad (17)$$

For cluster recovery the Ward Method was used and a maximum number of clusters was defined. It is noteworthy that in [22] other methods have been tested and the Ward Method showed better results.

As for the evaluation of results, two external indexes were used (Measure F and Random Index), as shown in Section 2.4, as well as the misclassification percentage.

3.4.1. Proposed Changes to the Ant-based Clustering

During the study of the Ant-based Clustering, it was observed that many of the changes in position of patterns occur unnecessarily. It is considered an unnecessary change when a pattern is among similar ones on the grid and, in this case, there is no need to change this pattern to another position. Aiming to avoid these unnecessary changes, was introduced a comparison of the probability of dropping a pattern in the position chosen randomly with the probability of dropping this pattern at its current position. The pattern is only dropped at the position chosen randomly if this probability is greater than the probability of dropping this pattern at its current position.

The occurrence of fusion of close clusters on the grid was also observed. When a decision to drop a pattern is positive and the cell where that pattern should be dropped is occupied, a free random position close to this one is searched for. However, this new position may also be close to another pattern cluster on the grid. This may be one reason for the merger of close clusters. As an alternative to prevent the merger of close clusters on the grid, in this paper was proposed an assessment of the probability for the new position. The pattern is only dropped at the position chosen randomly if this probability

is greater than the probability of dropping this pattern at its current position. All free neighboring positions are evaluated. If at no free neighboring position the probability of dropping the pattern is higher than the probability of dropping the pattern at its current location, the pattern is not dropped and the process starts again by choosing another ant.

Another issue observed in the Ant-based Clustering is that an ant can carry a pattern that is among similar ones on the grid. An ant only carries a pattern when it is not among similar ones on the grid. However, since the ant carries a pattern until it is drawn to attempt to drop the pattern, changes occur in this neighborhood and then can it leave it among the similar ones. Therefore, this ant is inactive because the operation of dropping the pattern is not performed. In this case, it was proposed to replace the pattern picked by an ant, if this pattern is not dropped in 100 consecutive iterations. The new pattern was chosen by lot, but it was only picked by the ant if the probability of carrying this pattern is greater than 0.13397. The value 0.13397 was defined by making the pick probability (p_{pick}) equal to the drop probability (p_{drop}). If there is no pattern with a picking probability higher than 0.13397, the ant picks the last pattern drawn. This could also be a stopping criterion.

3.4.2 Pseudo-code

Initial phase

Patterns are randomly scattered on the grid.

Each ant randomly chooses a pattern to pick and is placed at a random position on the grid.

Distribution phase

Each ant is selected randomly.

This ant moves randomly on the grid and evaluates its neighborhood function *f (i)* (equation 15).

The ant decides probabilistically if it drops its pattern at this position (equation 2). The pattern is only dropped at the position chosen randomly if this probability is greater than the probability of dropping this pattern at its current position.

If the decision is negative, another ant is selected at random and the distribution phase starts over again.

If the decision is positive, the ant drops the pattern at its current position on the grid, if it is free.

If this grid cell is occupied, the pattern must be dropped at a free neighboring cell through a random search. The evaluation of probability of dropping the pattern at the new position is made and the pattern is only dropped at that neighboring cell if the probability of dropping the pattern at this position is still higher than the probability of dropping this pattern at its current position. If at no free neighboring position the probability of dropping the pattern is higher than the probability of dropping the pattern at its current location, the pattern is not dropped and the process starts again by choosing another ant.

The ant randomly searches for a new pattern to pick (among the free patterns), goes to its position on the grid, evaluates the neighborhood function (equation 15) and decides probabilistically whether it picks this pattern or not (equation 1).

This choosing process of a free pattern on the grid runs until the ant finds a pattern that should be picked.

The pattern an ant picks will be replaced if this pattern is not dropped after 100 consecutive iterations. Another pattern is randomly chosen, but the ant only picks it if the probability of picking this pattern is higher than 0.13397, a figure previously discussed in section 3.4. If there is no pattern with a picking probability higher than 0.13397, the ant picks the last pattern drawn.

Cluster recovery phase

The process begins with each pattern forming a cluster.

After calculating the distances between all clusters, the two clusters with smaller "loss of information" should be merged (equation 9).

4. Results

The proposed Clustering algorithm based on Ants was applied to three real and public databases listed in Table 1. Because it is a metaheuristic, this method was applied 10 times to each database, as already mentioned. Preliminary results for the Iris and Wine databases have been published in [23] and [24].

4.1. Results of the Application of the Proposed Algorithm to the Databases

Table 2, at the end, presents the average and the standard deviation of the evaluation measurements for the databases, using the proposed algorithm, in addition to measurements to evaluate the clustering for the best result.

As can be seen, the results were quite satisfactory for databases IRIS and WINE (11.9% and 12.7%, on average, of wrong ratings). As for the PIMA database, the results were not as good; below it is shown that the other methods also showed no satisfactory results for this database.

Figures 2 and 3 show the grid for the best result (whose evaluation measurements are presented in Table 2) for databases IRIS and WINE, respectively. In these figures, the patterns in red belong to cluster 1, patterns in black belong to cluster 2 and patterns in blue belong to cluster 3. It is worthy to point out that cluster 1 contains all the patterns assigned to it.

Table 3 (confusion matrix) shows the pattern distribution for the IRIS database, where one can observe the patterns correctly assigned to clusters and patterns erroneously assigned to clusters. In this database there are only nine patterns (*) in wrong clusters from a total of 150 patterns. Cluster 1 contains all the patterns assigned to it.

Similarly, Table 4 shows the pattern distribution for the WINE database. In this database there are only 18 patterns (*) in wrong clusters from a total of 178 patterns.

4.2. Evaluation of the Proposed Algorithm in relation to the Ward Methods, the One-dimensional Kohonen and the ACAM

Table 5, at the end, shows the comparisons of the average evaluation measurements for the three methods (Ward, Kohonen, ACAM and proposed algorithm) for the IRIS, WINE and PIMA databases. The best results are in bold.
In the IRIS database, the Ward Method was the best for the three assessment measures (about 3% of errors); in the WINE database, the proposed algorithm was better for two of the three assessment measures (about 12% of errors) and with the PIMA database, the Uni-dimensional Kohonen Networks technique was the best for two of the three assessment measures (about 34% of errors).
It is worthy to point out that [11] claim that no algorithm dominates the others forever. According to [13], according to the "*NO-FREE-LUNCH*" theorem, if there is no prior assumption about the optimization problem one is trying to solve, it is expected that no strategy has better performance than others when tested on a large set of databases with different characteristics.
When comparing the measures averages for evaluating clustering for the proposed algorithm and for the ACAM algorithm, the results show that the proposed algorithm is better than the ACAM for two of three databases (IRIS and WINE). The best results are marked with (+). As already mentioned, for the PIMA database the results show that none of the methods showed satisfactory results.

5. Conclusions

The proposed clustering algorithm based on Ants was applied to three databases, and to assess its performance it was compared with three other methods: Ward, One-dimensional Kohonen and ACAM.
Comparing the proposed algorithm to the Ward and Kohonen methods (columns 2, 3 and 4 of Table 5), the results show no superiority of any of them. On the other hand, when comparing the mean clustering evaluation measurements through the proposed algorithm and the *ACAM* algorithm (columns 4 and 5 of Table 5), the results show that the proposed algorithm showed a better performance for two of the three databases.
Therefore, although the proposed algorithm has shown no superiority in relation to the already established Ward and Kohonen methods, it showed improvements in relation to one of the latest approaches involving the ant colonies (ACAM, by [1]), thus certifying the importance of this paper.
It is intended to continue this work, using additional databases for testing and the use of additional indexes for clustering evaluation so that then, additional improvements to the algorithm here proposed may be introduced.

Acknowledgements

To FINEP for the financial support to the research project CT – INFRA / UFPR / Modeling and for Scientific Computing and to CAPES for the scholarship awarded to first author.

References

[1] Boryczka, U. (2009). Finding groups in data: Cluster analysis with ants. *Applied Soft Computing*, **9**, 61-70.
[2] Deneubourg, J.-L.; Goss, S.; Franks, N.; Sendova-Franks, A.; Detran, C.; Chretien L. (1991) The dynamics of collective sorting: Robot-like ants and ant-like robots. In: *Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats*, Cambridge, MA: MIT Press, 356–365.
[3] Diniz, C. A. R.; Louzada Neto, F. (2000). *Data Mining: uma Introdução*. São Paulo: ABE.
[4] Dorigo, M.; Blum, C. (2005). Ant colony optimization theory: A survey. *Theoritical Computer Science*, **344**, 243-278.
[5] Dorigo, M.; Caro, G.D.; Gambardella L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, **5**, 137-172.
[6] Dorigo, M.; Gambardella, L. M. (1997). Ant colonies for the traveling salesman problem. *BioSystems*, **43**, 2, 73–81.
[7] Dorigo, M.; Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.
[8] Dorigo, M.; Maniezzo, V.; Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, **26**, 1, 1–26.
[9] Dorigo, M.; Bonabeau, E.; Theraulaz, G. (2000). Ant algorithms and stigmery. *Future Generation Computer Systems*, **16**, 8, 851–871.
[10] Fausett, L. (1994). *Fundamentals of Neural Networks – Architectures, Algorithms, and Applications*. New Jersey: Prentice Hall.
[11] Handl, J.; Knowles, J.; Dorigo, M. (2006). Ant-Based Clustering and Topographic Mapping. *Artificial Life*, **12**, 1, 35-61.
[12] Haykin, S. (2001). *Redes neurais: princípios e prática.* Tradução: Paulo Martins Engel. Porto Alegre: Bookman.
[13] Ho, Y.C.; Pepune, D.L. (2002). Simple Explanation of the No-Free-Lunch Theorem and Its Implications. *Journal of Optimization Theory and Applications*, **115**, 3, 549-570.
[14] Johnson, R.A.; Wichern, D.W. (1998). *Applied Multivariate Statistical Analysis*. New Jersey: Prentice Hall.
[15] Kohonen, T. (1995). *Self-Organizing Map*. Berlin: Springer-Verlag.
[16] *MATLAB* R2008b – The MatWorks, *MATLAB* (R2008b), The MathWorks Inc., Natick, 2008.

[17] Siqueira, P.H.; Scheer, S.; Steiner, M.T.A. (2007). *A new approach to solve the traveling Salesman problem. Neurocomputing* (Amsterdam), **70**, 1013-1021.
[18] Socha, K.; Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research*, **185**, 1155-1173.
[19] Tan, P.N.; Steinbach, M.; Kumar, V. (2005). *Introduction to Data Mining*. Inc. Boston, MA, USA: Addison-Wesley Longman Publishing Co.
[20] Tan, S.C.; Ting, K.M.; Teng, S.W. (2007). Examining Dissimilarity Scaling in Ant Colony Approaches to Data Clustering, *ACAL 2007*, Springer-Verlag.
[21] Villwock, R. (2009). *Técnicas de Agrupamento e de Hierarquização no Contexto de KDD: Aplicação a Dados Temporais de Instrumentação Geotécnica-Estrutural da Usina Hidrelétrica de Itaipu.* 125 f. Tese (Doutorado em Métodos Numéricos em Engenharia), UFPR, Curitiba, PR, Brasil.
[22] Villwock, R.; Steiner, M.T.A. Agrupamento baseado em Colônia de Formigas: Estudo Comparativo de Algoritmos para Recuperação dos Grupos. *Anais do XII Encontro Regional de Matemática Aplicada e Computacional*, Foz do Iguaçú, PR, Brasil, 2008.
[23] Villwock, R.; Steiner, M.T.A. (2009a). Análise do Desempenho do Algoritmo de Agrupamento Baseado em Colônia de Formigas Modificado. *Anais do XXXII Congresso Nacional de Matemática Aplicada e Computacional*, Cuiabá, MT, Brasil.
[24] Villwock, R.; Steiner, M.T.A. (2009b). *Análise do Desempenho de um Algoritmo de Agrupamento Modificado Baseado em Colônia de Formigas*. Anais do XLI Simpósio Brasileiro de Pesquisa Operacional, Porto Seguro, Ba, Brasil.
[25] Vizine, A.L.; Castro, L.N.; Hruschka, E.R.; Gudwin, R.R. (2005). Towards improving clustering ants: an adaptive ant clustering algorithm. *Informatica*, ***29***, 143–154.

Rosangela Villwock completed her undergraduate studies in Mathematics at Estadual University of Paraná West, at Francisco Beltrão city, State of Paraná, in Brazil, in 1999. She got her Master's degree in this same university in 2003 and her Ph.D.'degree in Numerical Methods in Engineering at Federal University of Paraná, at Curitiba city, State of Paraná, in Brazil, in 2009. She is an Adjunct Professor at the Mathematics Department at Estadual University of Paraná West, since 2000. She lectures on Engineering Undergraduate Programs; on Numerical Methods in Engineering Graduate Program and on Engineering Production Graduate Program. Her recent interest includes Neural Networks, Metaheuristics, in special, Ant Colonies, and Mathematical Programming.

Maria Teresinha Arns Steiner completed her undergraduate studies in Mathematics and in Civil Engineering at Federal University of Paraná, at Curitiba city, State of Paraná, in Brazil, in 1978 and 1981, respectively. She got her Master's and Ph.D.'s degrees in Production Engineering, on Operations Research area, at Federal University of Santa Catarina, at Florianópolis city, State of Santa Catarina in 1988 and 1995, and her Pos-Doc on Operations Research area, at Technological Institute of Aeronautics, São Paulo, in 2005. She is an Associate Professor at Federal University of Paraná since 1978 and a Visitor Professor at Catholic University of Paraná since 2011. She lectures on Engineering Undergraduate Programs; on Numerical Methods in Engineering Graduate Program and on Engineering Production and Systems Graduate Program. Prof. Arns Steiner is the author of over hundred papers and other technical publications in the Operations Research, Data Mining, Neural Networks and Metaheuristics areas.

Paulo Henrique Siqueira completed his undergraduate studies in Mathematics at Federal University of Paraná, at Curitiba city, State of Paraná, in Brazil, in 1997. He got his Master's and Ph.D.'s degrees in this same university in 1999 and 2005, respectively, in Numerical Methods in Engineering. He is an Adjunct Professor at the Drawing Department at this university since 1998. He lectures on Engineering Undergraduate Programs; on Numerical Methods in Engineering Graduate Program and on Engineering Production Graduate Program. His recent interests include Neural Networks, Metaheuristics, Mathematical Programming, Computer Graphics and Geometry.

Table 1. Databases used to evaluate the proposed algorithm

Database	Patterns	Attributes	Clusters
Íris	150	4	3
Wine	178	13	3
Pima Indians Diabetes	768	8	2

Table 2. Results of applying the proposed algorithm, run averages for 10 times, for real databases (IRIS, WINE and PIMA).

Results		R	F	Misclassification (%)
Iris	**Average:**	0.871	0.877	11.9
	Standard deviation	0.039	0.050	4.6
	A) Result VH-RT3c.	0.927	0.940	6.0

Wine	**Average:**	0.843	0.871	12.7
	Standard deviation	0.019	0.021	1.9
	A) Result VH-RT3c.	0.871	0.899	10.1
Pima	**Average:**	0.510	0.583	43.6
	Standard deviation	0.010	0.022	4.0
	A) Result VH-RT3c.	0.531	0.623	37.5

Table 3. Confusion matrix showing the Pattern distribution for the IRIS database – best result

Iris	**Generated Solution**		
Correct Clustering	**Cluster 1**	**Cluster 2**	**Cluster 3**
Class 1	50	0	0
Class 2	0	48	2*
Class 3	0	7*	43

Table 4. Confusion matrix showing the Pattern distribution for the WINE database – best result

WINE	**Generated Solution**		
Correct Clustering	**Cluster 1**	**Cluster 2**	**Cluster 3**
Class 1	55	4*	0
Class 2	4*	64	3*
Class 3	2*	5*	41

Table 5. Comparison of average results from the application of clustering methods: Ward, One-dimensional Kohonen Networks, ACAM and Proposed Algorithm, for the IRIS, WINE and PIMA databases.

Database		**Ward**	**1D-SOM**	**Ants**	
				Algorithm	***ACAM***
Iris	**R** (higher is better)	**0.96**	0.86	0.87^{+}	0.82
	F (higher is better)	**0.97**	0.86	0.88^{+}	0.81
	Misclassification (%) (smaller is better)	**3.33**	12.8	11.9^{+}	18.7
WINE	**R** (higher is better)	0.82	0.76	0.843	$\mathbf{0.85}^{+}$
	F (higher is better)	0.85	0.76	$\mathbf{0.87}^{+}$	0.87
	Misclassification (%) (smaller is better)	15.17	22.42	$\mathbf{12.7}^{+}$	13.9
PIMA	**R** (higher is better)	0.53	**0.55**	0.51	0.52^{+}
	F (higher is better)	0.62	**0.66**	0.58^{+}	0.57
	Misclassification (%) (smaller is better)	37.37	34.57	43.6	$\mathbf{33.7}^{+}$

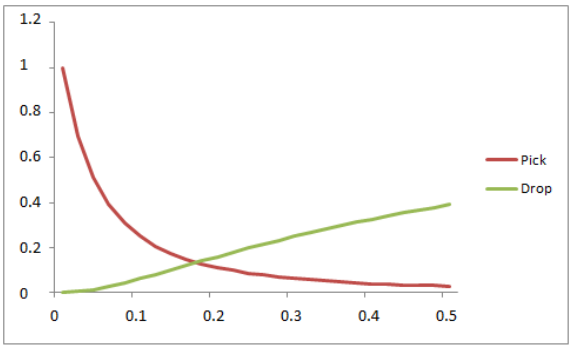

Figure 1. Graph of the probabilities of picking and dropping patterns

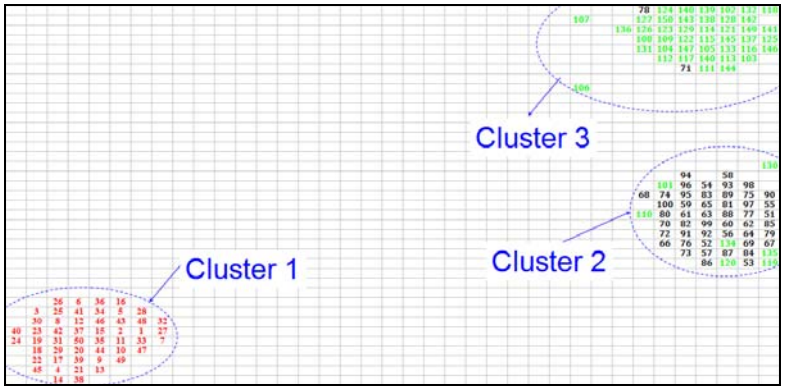

Figure 2. Result of the proposed algorithm for the IRIS database – best result.

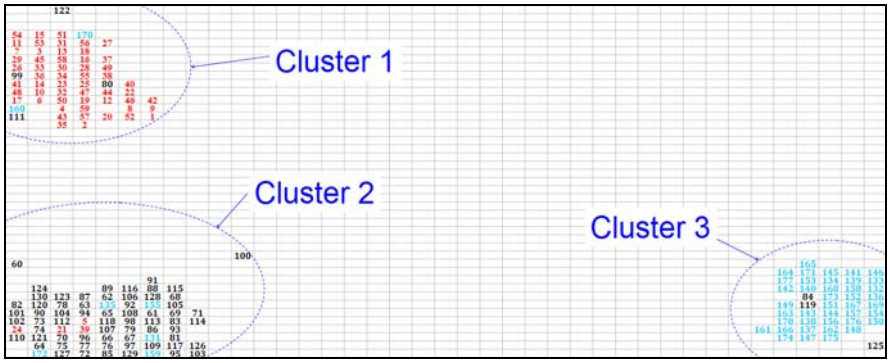

Figure 3. Result of the proposed algorithm for the WINE database – best result.