On the effects of direct communication between agents in ant-based clustering algorithms

Marco A. Montes de Oca, Leonardo Garrido, and José L. Aguirre

Centro de Sistemas Inteligentes, Tecnológico de Monterrey
Eugenio Garza Sada 2501, Col. Tecnológico
Monterrey, N.L. México C.P. 64849
{A00788072,leonardo.garrido,jlaguirre}@itesm.mx

Abstract. Ant-based clustering algorithms are distributed self-organized knowledge discovery tools inspired by the collective behavior of insect colonies. They are based on stigmergic communication among participating agents, that is, on the nonintentional indirect influence on the agents behavior through local modifications of the environment. Inspired by the oral trophallaxis phenomenon observed in some ant species, two different knowledge exchange strategies between agents in ant-based clustering algorithms are investigated. The impact on the final clustering quality is evaluated by comparing the clustering process development generated by each strategy. The investigated knowledge exchange strategies are: simple local memory sharing and the shared use of environment maps. It is shown that benefits on the final clustering are directly related to the usefulness of the exchanged knowledge and on the number of participating agents.

1 Introduction

Insect species are considered eusocial when overlapping generations of parents and offspring live together in an organizational unit or colony, division of labor occurs within the colony, and when they develop a physical structure or nest to live in. Ants, termites and some bee and wasp species are considered eusocial [4].

In eusocial insects, different activities are performed simultaneously by specialized individuals. This division of labor can be seen in activities such as nest cleaning and brood care. An example of such behavior can be seen in some ant species such as *Pogonomyrmex badius* and *Lasius niger* whose workers pile corpses of dead nestmates and defeated enemies on a distant spot away from their nest entrance [2, 14]. Brood care is a highly specialized activity that require, in some cases, the separation of larvae according to their development stage. In experiments, the ant *Leptothorax unifasciatus* gather its larvae together according to their size. Small larvae are aggregated near the center of a plate and larger larvae near the periphery [2].

All these collective behaviors and many others exhibited by social insects are the outcome of a process of self-organization [9]. Bonabeau et al. [2] define self-organization as "a set of dynamical mechanisms whereby structures appear at the

global level of a system from interactions among its lower-level components" (p. 9). They also identify four basic elements of self-organization, which are positive feedback, negative feedback, the amplification of fluctuations, and the existence of multiple interactions. In the case of ant corpse piling and brood sorting there is a key factor that mediates these processes. This factor is known as stigmergy. First proposed by Grassé (cited in [9]) to explain the construction of nests by the termites *Cubitermes* and *Macrotermes*. Grassé observed that when workers of *Macrotermes bellicosus* were placed in a container with some soil pellets, the insects carried about and put down pellets in an apparently random fashion after an exploration phase where they moved through the container without taking any action. At this stage, a pellet just put down by a termite worker is often picked up and placed somewhere else by another worker. When a pellet is placed on top of another, the resultant structure appears to be much more attractive and termites soon start piling more pellets nearby, making the dropping spot even more attractive [14]. Stigmergy then, is the nonintentional influence on the behavior of others through local environment modifications.

Inspired by corpse aggregation, brood sorting and nest building in colonies of social insects, computer scientists have created clustering algorithms for exploratory data analysis where insects are simulated by simple reactive agents that act in a two-dimensional grid. The basic clustering algorithm makes use of stigmergy as the only mean of communication between agents [3, 12].

Stigmergy, however, is not the only way social insects interact with each other. In most species, trophallaxis or liquid food exchange among members of the same colony, plays a key role in their social organization [14]. Consider the case of some termite species which require intestinal protozoa to derive benefits from cellulose. Their early instar nymphs are fed either by oral or anal trophallaxis. The latter infects them with symbiotic protozoa or bacteria contained in the proctodeal liquid. The subsocial association result of this codependence have evolved into a complex social and morphological structure [4].

In this paper we investigate the effects of two knowledge exchange strategies between agents in ant-based clustering algorithms inspired by the oral trophallaxis phenomenon. We compare the clustering process development generated by the Lumer and Faieta [12] algorithm that uses embedded short-term memories into agents with an extension of it where agents mutually explore their short-term memories. Moreover, an algorithm where agents build and share environment maps is also evaluated. We provide experimental evidence that shows the advantages of direct information exchange in the quality of the obtained clustering.

The remaining of this paper is organized as follows. In section 2, previous work on ant-based clustering using stigmergy as the only mean of communication between agents is presented. Section 3 describes the communication model and the knowledge representation strategies used. Section 4 details the experimental strategy used to measure the effects of introducing information exchange into the basic ant-based clustering algorithm. Section 5 presents the experiments results. In section 6, we analyze the experiments results and discuss the implications of

the introduction of information exchange bewteen agents in ant-based clustering algorithms. Finally, in section 7 we conclude and speculate on future extensions and ideas to further study the impact of information exchange between agents in ant-based clustering algorithms.

2 Previous Work

There are three main areas in which previous work have focused: the original spatial sorting algorithm proposed by Deneubourg et al. [3], a generalization of this model to exploratory data analysis introduced by Lumer and Faieta [12], and more recently, efforts to formally measure the advantages offered by ant-based clustering by Handl et al. [8]. The following subsections summarize these past works.

2.1 Spatial sorting

Ant-based clustering was introduced by Deneubourg et al. [3] using a model for spatial sorting. A group of agents exhibiting the same behavior move randomly over a toroidal square grid. In the environment there are objects that were initially scattered in a random fashion. The objects can be picked up, moved or dropped in any free location on the grid. An object is picked up with high probability if it is not surrounded by other objects of the same type and is dropped by a loaded agent if its neighborhood is populated by other objects of the same type and the location of the agent has no object on it. The probability p_p with which an object is picked up is given by

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

where k_1 is a constant and f is considered to be a measure of similarity of the neighborhood of the agent with respect to the object on the agent's location. If the object is surrounded by objects of the same type, then f should be large in comparison with k_1 in order to make p_p small. On the other hand, if the object is not surrounded by any object of the same type, then f should be small making p_p high. The probability p_d for an agent to deposit an object is given by

$$p_d = \left(\frac{f}{f + k_2}\right)^2 \tag{2}$$

where k_2 is another constant. If the object is surrounded by objects of the same type, then f should be large in comparison with k_2 in order to make p_d approach 1. On the other hand, if the object is not surrounded by any object of the same type, then f should be small making p_d small. As expected, p_d has an opposite behavior than p_p.

Having a robotic implementation in mind, Deneubourg et al. [3] proposed to compute f through a short-term memory each agent maintains. Keeping track

of the last T time units, an agent counts the number of objects it finds; f is the number of objects of a particular type encountered by the agent in the last T time steps divided by the maximum number of objects an agent can find in T time units.

2.2 Exploratory Data Analysis

Exploratory data analysis tries to identify structures and relationships within data. Cluster analysis is one possible approach to do so. In cluster analysis, the main problem is that of finding a partition of a data set so that similar objects end up in the same class or cluster [6].

Lumer and Faieta [12] generalized the spatial sorting algorithm to apply it to exploratory data analysis specifically, through the creation of clusters of related data. All classical clustering algorithms depend on a similarity or dissimilarity measure to determine whether two objects are similar or not. This algorithm is no exception. However, instead of taking as input a similarity or dissimilarity matrix, this algorithm starts with something very similar to what we described in the preceding section. A group of agents moving randomly on a toroidal square grid. On the grid, data elements scattered randomly that can be picked up, moved or dropped by the agents. This time however, the probabilities are computed differently. The probability of picking a data element i is defined as

$$p_p(i) = \left(\frac{k_p}{k_p + f(i)}\right)^2 \tag{3}$$

where k_p is a constant and $f(i)$ is a similarity density measure with respect to element i. Likewise, the probability of dropping a data element is given by

$$p_d(i) = \begin{cases} 2f(i) & \text{if } f(i) < k_d \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

where k_d is a constant. The similarity density $f(i)$ for an element i, at a particular grid location τ, is defined as

$$f(i) = \begin{cases} \frac{1}{s^2} \sum_{j \in Neigh(\tau)} \left(1 - \frac{d(i,j)}{\alpha}\right) & \text{if } f > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where s^2 is the size of the perception area centered at the location of the agent and α is a scaling factor of the dissimilarity measure $d(i,j)$ between elements i and j. If it is assumed that the elements can be represented as points in an n-dimensional space, then the Euclidean distance could be used as dissimilarity measure.

In experiments, Lumer and Faieta showed that the algorithm was capable of correctly classifying elements of a synthetic data set; however, the number of clusters found were, in general, more than those expected. To fix this problem, they proposed three modifications to the basic algorithm.

The first modification was the introduction of variability into the agent population. The elemement of change was the velocity of displacement. Velocities then range from 1 to V_{max} and represent the number of grid positions the agent can move in a single time unit along a given direction vector. This variability was coupled to the sensitiveness of each agent to dissimilarity in the following way

$$f(i) = \begin{cases} \frac{1}{s^2} \sum_{j \in Neigh(\tau)} \left(1 - \frac{d(i,j)}{\alpha\left(1+\frac{v-1}{V_{max}}\right)}\right) & \text{if } f > 0 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The result is that an agent moving slowly will be more selective in their similarity criterion than an agent moving rapidly. The effect of this variability is that the number of final clusters is closer to the expected than the model with homogeneous agents. Fast moving agents appear to create clusters in a coarser scale than slowly moving agents which place data elements more carefully.

The second modification was the introduction of a short-term memory into the agents so that they could keep track of the dropping location of the last m elements the agent had carried. The short-term memory tries to represent the environment state. This modification has the effect of lowering the probability that a just picked up element can create an independent cluster by itself, thus reducing the final number of clusters. This happens because every time an agent picks an element, it looks in its memory for the location of the closest element in attribute space. The agent, instead of moving randomly, moves directly towards that particular location. The knowledge representation of the environment state, i.e., the short-term memory, has problems when the environment is highly dynamic. Memorized dropping spots will not reflect the *current* environment state, particularily during early and intermediate clustering development stages. Our approach to overcome this problem using communication between agents and changing the knowledge representation is explained in section 3.

The third modification was that of behavioral switches. If an agent has not manipulated an item for a predetermined number of time units, it starts destroying the so far created clusters. This does not end up in a total cluster destruction because other agents that have not reached that state will reposition the disseminated objects.

2.3 Clustering Validation

In an effort to formally evaluate the clustering quality of ant-based clustering algorithms, Handl et al. [8] applied four validity measures to the resulting clustering from a modified version of the Lumer and Faieta [12] algorithm. For the interested reader, the modifications introduced by Handl et al. can be found in [7].

The four validity measures used by Handl et al. [8] were:

The F_1-Measure. The harmonic mean of recall and precision, also known as the *F_1-Measure*. Commonly associated to the information retrieval field, recall and precision are measures that give us some idea of how well a clustering

algorithm is identifying the classes present in a database. In the context of classification, recall is defined as $r(i,j) = \frac{n_{ij}}{n_i}$ where n_{ij} is the number of elements of class i in cluster j and n_i is the number of elements of class i. Precision is defined as $p(i,j) = \frac{n_{ij}}{n_j}$ where n_j is the number of elements in cluster j. For a class i and a cluster j the F_1-*Measure* is defined by

$$F_1(i,j) = \frac{2\,p(i,j)\,r(i,j)}{p(i,j) + r(i,j)}$$

The overall F_1-*Measure* for the classification generated by the clustering algorithm is given by

$$F_1 = \sum_i \frac{n_i}{n} \max_j \{F_1(i,j)\} \tag{7}$$

where n is the size of the data set. F_1 is limited to the interval $[0,1]$ with a value of 1 with a perfect clustering.

The Rand Statistic. Determines a similarity measure between the known perfect classification C and the partition generated by the clustering algorithm P. It is defined as

$$R = \frac{a+d}{a+b+c+d} \tag{8}$$

where

- a is the number of pairs where both elements belong to the same class in C and to the same group of the partition P.
- b is the number of pairs where both elements belong to the same class in C and to different groups in P.
- c is the number of pairs where both elements belong to different classes in C and to the same group in P.
- d is the number of pairs where both elements belong to different classes in C and to different groups in P.

Note that $a+b+c+d = N(N-1)/2$ where N is the total size of the data set. The Rand statistic is limited to the interval $[0,1]$ with a value of 1 with a perfect clustering.

The intra-cluster variance. This measure tries to capture the idea that members that belong to the same cluster should be as close to each other as possible. It is given by

$$I = \sum_{i=1}^{n} \sum_{p \in C_i} \|p - \mu_i\|^2 \tag{9}$$

where n is the total number of groups in the partition generated by the clustering algorithm, and μ_i is the centroid of group i. This measure is to be minimized.

The Dunn index. This measure will give a high value whenever the partition gives compact and well separated clusters. It is given by

$$D = \min_{c,d \in C} \left\{ \frac{\|\mu_c - \mu_d\|}{\max_{e \in C}\{diam(e)\}} \right\} \tag{10}$$

where μ_i is the centroid of cluster i and $diam(i)$ is the diameter of cluster i which could be considered a dispersion measure. It is defined as

$$diam(c) = \max_{x,y \in c} \{\|x - y\|\}$$

In this investigation we used these validity measures to evaluate the effects of direct communication between agents during the clustering process.

3 Communication Model and Knowledge Representation Strategies

In all previous attempts to perform clustering tasks using a simulated insect colony there is no direct interaction among agents. This is perhaps due to the fact that early works on ant-based clustering and sorting focused on robotic implementations [3,9,11,13] where direct real-time communication is much more complicated than in software simulations.

The communication model used in our approach uses as analogy the oral trophallaxis phenomenon among nestmates in colonies of social insects. Whenever a group of agents, of which at least one of them is in search for a dropping location coincide on the grid, it exchanges knowledge with its peers about the environment state. In general, the main purpose of this exchange is to better approximate the current environment state in order to bias an agent search for better dropping locations.

The communication model raises the question of *what* knowledge is to be exchanged. To try to answer this question, we first use a natural extension of the short-term memory model proposed by Lumer and Faieta [12]. Whenever a loaded agent coincides on the grid with one or more agents, it explores their memory to look for a closer object to its load than the object that was guiding it on its search for an appropriate dropping location. If it finds it, then it redirects itself towards that object. This is the first knowledge representation strategy explored in this paper.

The second strategy explored in this work is the creation of environment maps. In an environment map, an agent wants to represent the spatial distribution of data elements. A succinct representation of a spatial distribution is accomplished through the use of self-organizing maps or SOM's [10]. However, because of the limitations of conventional SOM's; namely, the *a priori* fixed number of neurons and the problem of "dead" neurons or neurons that do not update their weight vectors due to a misplacing in the input space, each agent must create a dynamic self-organizing map. In order to accurately represent the environment, an agent should update its map as it explores the environment, regardless whether it is in search for a dropping location or just wandering through the environment.

In the environment map representation, both the agent behavior and the exchange of knowledge between agents is different. If an unloaded agent picks up a data element from the grid, it classifies it using its neural network and

moves directly to the location of the winner neuron. Whenever a loaded agent coincides on the grid with one or more agents, it classifies its load using its peers neural networks to determine the closest winner neuron. If it is closer in the attribute space than its own winner neuron, then the agent redirects itself towards the location of that neuron on the grid. The particular implementation of the dynamic self-organized map should not change the final effects as long as it accurately represents the spatial distribution of data elements.

4 Experimental Setup

In our experiments we used two well-known test databases. Each one of them with different difficulty levels. In order to deal with them, we also modified the way an agent evaluates its neighborhood to compute a similarity density. In the following, a more detailed explanation of our experiment designs is presented.

4.1 Test Data

In order to evaluate the impact of the introduction of knowledge exchange between agents we used two real data collections from the UCI Machine Learning Repository [1]. These were:

- Iris Plant Database. This database is composed of 150 instances with 4 numeric attributes each. There are 3 classes composed of 50 instances each. Of these, one is linearly separable from the other two; the latter are not linearly separable from each other.
- Wine Recognition Database. This database is composed of 178 instances with 17 numeric attributes each. There are 3 classes in the database where class 1 has 59 instances, class 2 has 71 instances, and class 3 has 48 instances.

To eliminate the bias on similarity measures provoked by different scales within data attributes, we standardized all data sets. The similarity measure used in all our experiments was the cosine metric[1].

4.2 Agent Models

The agents picking and dropping probabilities were computed using expresions 3 and 4 respectively. However, because the cosine metric is a similarity measure, expresion 5 is not applicable. Therefore, for the local similarity density $f(i)$, we used

$$f(i) = \frac{1}{s^2} \sum_{j \in Neigh(\tau)} \left(\frac{1}{1 + e^{-S\frac{d(i,j)}{\alpha} + D}} \right) \tag{11}$$

where S is the steepness of the response curve and D serves as a displacement factor. In our experiments we fixed S to 5 because it provides a similarity value

[1] In preliminary experiments, it proved to give better results.

close to 0 when the cosine measure is minimum, that is, when the cosine mesure gives a value of -1, and D to 1 because this allows us to better distinguish vectors with separation angles between 0 and $\pi/2$.

This local similarity density function permits the integration of simple background knowledge. In particular, if it is known that data classes are difficult to separate, one can move the displacement factor near the maximum value of the similarity measure to better discriminate among very similar data elements. This expression also has the advantage of limiting the range of values α can take to $(0, 1]$, given that $-1 \leq d(i, j) \leq 1$, as is the case for the cosine metric.

Table 1 summarizes all other agent settings used in our experiments[2].

Table 1. Agent settings

Parameter	Value
k_p	0.1
k_d	0.15
α	0.7
Neighborhood size	5×5
Short-term memory size	8

All previously described settings were used for the Lumer and Faieta [12] basic algorithm and for the simple memory exchange algorithm. For the environment map building algorithm, we used growing neural gas networks to represent spatial data distribution. Introduced by Fritzke [5] as an approach to overcome some of the limitations of conventional self-organizing maps, a growing neural gas network consists of:

- a set A of units (or nodes). Each unit $c \in A$ has an associated *reference vector* $w_c \in \mathbf{R}^n$. The reference vectors can be regarded as positions in input space of the corresponding units.
- a set N of connections (or edges) among pairs of units. These connections are not weighted. Their sole purpose is the definition of topological structure.

The idea behind the training algorithm is to succesively add new units to an initially small network by evaluating local statistical measures gathered during previous adaptation steps. The network topology is generated incrementally by using a competitive Hebbian learning rule. The dimensionality depends on the input data and can vary locally.

The training algorithm basically begins with two randomly located units and an input signal that is to be learned. The learning rule used to adapt the

[2] The settings in tables 1 and 2 were derived experimentally, and proved to give good results for the given test databases.

reference vectors of the unit that is closest in the input space to the current input signal and its topological neighbors is

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \epsilon(\xi - \mathbf{w}^{(t)}) \tag{12}$$

where $\mathbf{w}$ is the reference vector of the adapting unit, ϵ is a constant called *learning rate*, ξ is the input signal being learned.

The squared error of the nearest unit in turn is accumulated, so that after λ input signals have been learned, a new unit is created half between the two neighboring units with the highest accumuated errors. The deletion of units happens whenever a unit is not topologically connected with any other unit. This occurs after that a given number of input signals have not fired a unit. This process continues until a stop criterion is met. The complete training algorithm can be found in [5].

The embedded growing neural gas networks have a parameter set on their own, we refer the reader to [5] for a detailed explanation of the meaning and effects of all these parameters. Table 2 summarizes the parameter set for the embedded growing neural gas networks used in our experiments.

Table 2. Embedded growing neural gas networks settings

Parameter	Value
Winner neuron learning rate ϵ_b	1
Neighboring neurons learning rate ϵ_n	0.005
Maximum edge age	50
Growing threshold λ	150
Local error decreasing rate α	0.5
Global error decreasing rate d	0.995

A learning rate of 1 is normally considered too high, however, due to the fact that the ant environment is highly dynamic, a learning rate of this characteristics is needed in order to allow the network learn the real data distribution on the grid. A crucial condition for the proper operation of this model.

4.3 Experiments Design

To observe the effects of information exchange between agents on the clustering process, we applied the four measures used by Handl et al. [8] (see subsection 2.3) every 10,000 simulation cycles to the partial clustering hitherto obtained. Each simulation cycle was composed of N individual actions, where N is the number of agents in the simulation. All three algorithms were tested 30 times with every database for 1,000,000 simulation cycles.

In our experiments, information exchange only occurs whenever two or more agents meet at a point on the grid. We can expect therefore, that the probability of an encounter raises as the number of agents is increased, or more precisely, as the agent density of the environment increases. By having more agent encounters over time, the effects of information exchange should have more impact. We tried with populations of 10, 20 and 30 agents within an environment of 100×100 locations in all of our experiments.

In order to collect information from the spatial partition formed by the agents on the grid, we applied an agglomerative hierarchical algorithm on the data using Euclidean distance and the single linkage strategy. The maximum distance to merge clusters was set equal to the size of the agents neighborhood. The distance used to compute the variance and the clusters diameters in the attribute space was also the Euclidean distance.

5 Results

Results of all algorithms applied to the test databases are summarized and presented grouped by validity measure. Subsection 5.5 presents the results on the number of clusters found. Results with 20 agents are skipped because, in all cases, they showed a behavior in between the results with 10 and 30 agents.

5.1 F_1-Measure

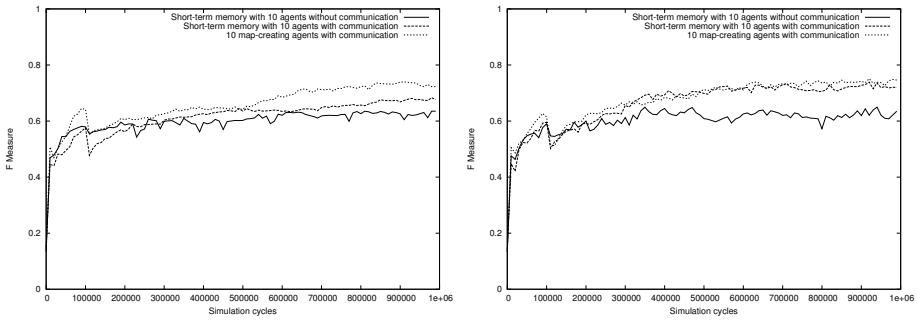
Figure 1 shows the F_1-Measure scores over time for all algorithms on the Iris Plant and Wine databases using 10 agents. After a certain number of simulation cycles, it can be seen that the algorithms where agents can exchange knowledge have higher scores than the algorithm where they cannot. With respect to the information exchange algorithms, the algorithm with map-creating agents performs slightly better than the algorithm with simple memory exchange agents.

Figure 2 shows the F_1-Measure scores over time for all algorithms on the Iris Plant and Wine databases. This time, however, using 30 agents. On the wine database, the algorithm with map-creating agents performs better than the other two algorithms at initial stages. However, all three versions converge to approximately the same score near the end of the simulation.

5.2 Rand Statistic

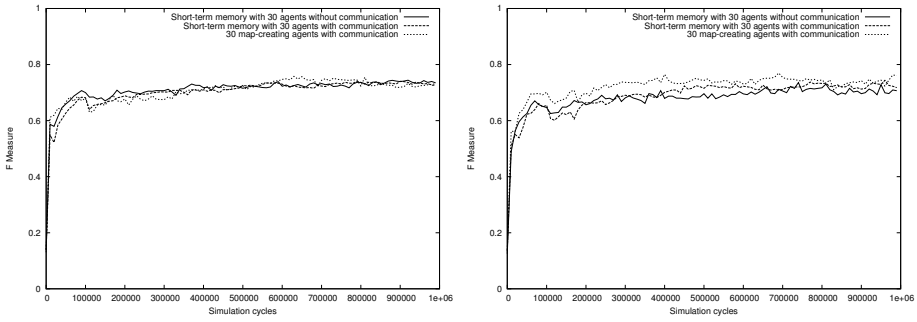
The Rand statistic scores over time for both algorithms on the Iris Plant and Wine databases using 10 agents are shown in figure 3. In both cases, it can be seen that the two algorithms with information exchange have higher scores than the algorithm without information exchange.

In figure 4 almost the same situation than that of the F_1-Measure can be observed, that is, with 30 agents both kind of algorithms behave almost the same way. Again, the algorithm with map-creating agents score higher during the first stages; after that, it converges to the same score than the two other algorithms.



(a) Iris database (b) Wine database

Fig. 1. F_1-Measure scores over time for all three algorithm versions on the Iris Plant and the Wine databases using 10 agents. A score value of 1 means perfect classification.

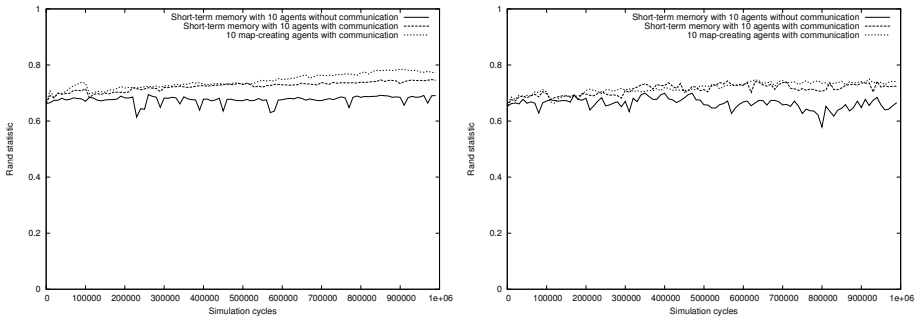


(a) Iris database (b) Wine database

Fig. 2. F_1-Measure scores over time for all three algorithm versions on the Iris Plant and the Wine databases using 30 agents. A score value of 1 means perfect classification.

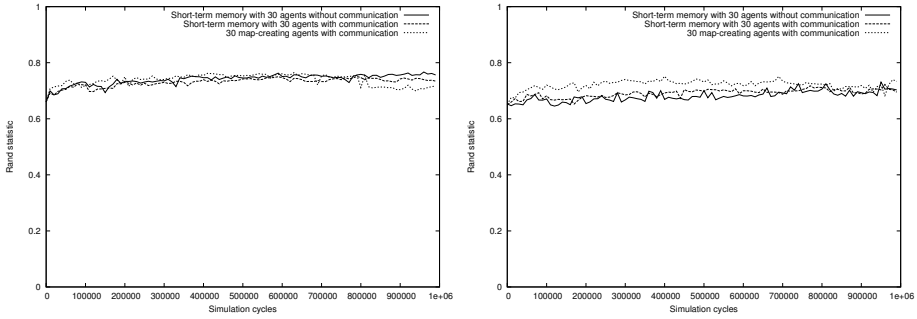
5.3 Intra-cluster Variance

The plots seen in figure 5 correspond to the intra-cluster variance scores for all three algorithms with 10 agents on the two test databases. As was the case with the F_1-Measure and the Rand statistic, the algorithms where agents can exchange information perform better than the short-term memory algorithm without direct communication among agents. It can also be seen that the algorithm with map-creating agents starts with a relatively poor performance and gets better as the simulation develops. Remember that this measure is to be minimized.



(a) Iris database (b) Wine database

Fig. 3. Rand statistic scores over time for all three algorithm versions on the Iris Plant and the Wine databases using 10 agents. A score value of 1 means perfect classification.



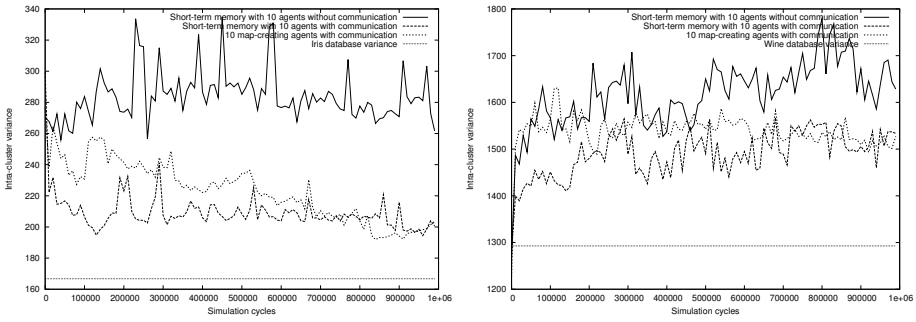
(a) Iris database (b) Wine database

Fig. 4. Rand statistic scores over time for all three algorithm versions on the Iris Plant and the Wine databases using 30 agents. A score value of 1 means perfect classification.

Figure 6 shows how the total intra-cluster variance gap between the two versions gets smaller as the number of agents increases. In both databases, the curves even swap making the information exchange versions the worst.

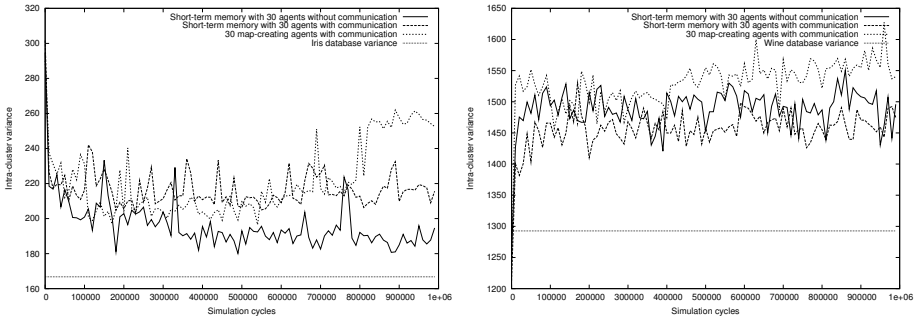
5.4 Dunn index

Figure 7 shows the Dunn index scores for all three algorithm versions with 10 agents on the two test databases. In this case, the performance of the short-term memory algorithm without information exchange is better in the Iris database test and worse in the Wine database test compared with the information exchange algorithms.



(a) Iris database (b) Wine database

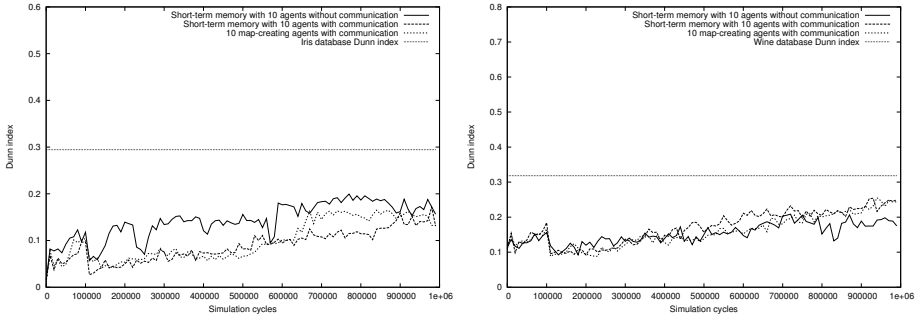
Fig. 5. Total intra-cluster variance over time for all three algorithm versions on the Iris Plant and the Wine databases using 10 agents. As a reference, the total intra-cluster variance of the correct clustering is shown.



(a) Iris database (b) Wine database

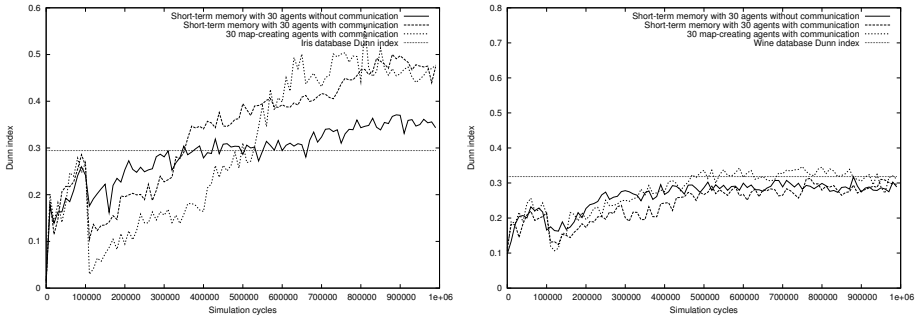
Fig. 6. Total intra-cluster variance over time for all three algorithm versions on the Iris Plant and the Wine databases using 30 agents. As a reference, the total intra-cluster variance of the correct clustering is shown.

Figure 8 shows the Dunn index scores for all three algorithms with 30 agents on the two test databases. In this case, the performance of the information exchange algorithms is better in the Iris database and almost the same in the Wine database test compared with the short-term memory algorithm. This behavior can be explained if we recall that in the Iris database two clusters are not linearly separable, thus making the algorithm consider them as one. This clearly increases the measure that tries to identify compact and well separated clusters.



(a) Iris database (b) Wine database

Fig. 7. Dunn index score over time for all three algorithm versions on the Iris Plant and the Wine databases using 10 agents. As a reference, the Dunn index for the correct clustering is shown.



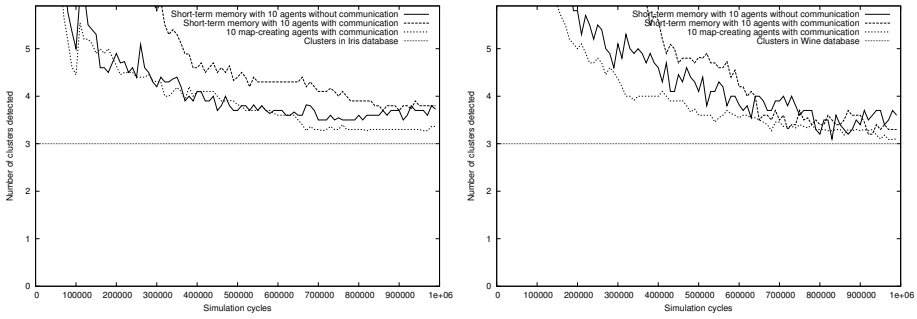
(a) Iris database (b) Wine database

Fig. 8. Dunn index score over time for all three algorithm versions on the Iris Plant and the Wine databases using 30 agents. As a reference, the Dunn index for the correct clustering is shown.

5.5 Number of clusters

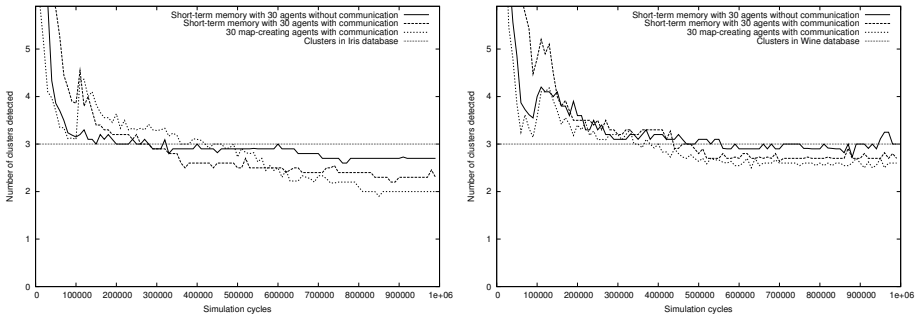
One of the main features that make ant-based clustering an appealing approach to clustering is that it is capable, on average, of finding the correct number of clusters within the data. This is confirmed in figures 9 and 10.

Figure 10 shows how the number of clusters decreases as the number of agents increases. In both figures, the algorithm with map-creating agents converges faster than both, the memory exchange algorithm and the short-term memory. This means that in this algorithm, the participating agents waste less time searching for appropiate dropping locations. Instead, they concentrate on



(a) Iris database (b) Wine database

Fig. 9. Number of clusters found by all three algorithm versions on the Iris Plant and the Wine databases using 10 agents. As a reference, the correct number of clusters is shown.



(a) Iris database (b) Wine database

Fig. 10. Number of clusters found by all three algorithm versions on the Iris Plant and the Wine databases using 30 agents. As a reference, the correct number of clusters is shown.

exchanging elements among clusters in order to increment the resulting clustering quality.

6 Discussion

The probability of an encounter between two agents moving randomly raises as the number of agents is increased. This is confirmed in robotic experiments [9] where it has been reported that the number of collisions between robots raises as the number of robots is increased. In our experiments, we tried to take advantage

of this fact and use it to study the effect of increasing the knowledge exchange frequency between agents. However, the actual effect of knowledge exchange depends not only on the frequency of encounters but also on their effectiveness, i.e., knowledge exchange is beneficial as long as the exchanged knowledge is *useful*. In a clustering context, knowledge usefulness depends very much on how well an agent's knowledge represents the *current* environment state. Early in the clustering process this is not accomplished because of the high environment dynamism. When this phase finishes, once that some initial clusters exist, information exchange is more effective as the environment is less dynamic. This in part explains the relatively poor performance of the information exchange model over the first simulation cycles.

Our results suggest that even by just sharing memorized dropping spots, information exchange offers some important advantages over the model that relies on stigmergy as the only mean of interaction among agents. In particular, the quality of the clustering generated is improved and this is reflected by the F_1-Measure, Rand index and intra-cluster variance scores. This holds true, however, only when the number of agents is small. This fact suggests that there is a critical number of agents that can exchange information, that when surpassed, the effects could even be detrimental. Despite this, information exchange could be useful when dealing with large databases, where both, the number of agents and the environment dimensions must be increased, e.g., when one cannot afford the computational costs associated with increasing these variables in a clustering task.

7 Conclusions and Future Work

Two information exchange schemes are explored in this paper. A simple short-term memory exchange scheme where agents use their peers past experience to find favorable dropping locations on the grid and the exploitation of data distribution maps that agents build as they explore the environment. The result is that when an agent is in search for a dropping location, it can change its trajectory based on the knowledge of other agents.

There are many possibilities to explore direct communication among agents in ant-based clustering. In all of them one has to determine *what* information to exchange, *how* to exchange this information and *when* is most appropriate to do so. Some of this aspects have been investigated in the robotics field, and although their results are important, they are not directly aplicable to software implementations where physical restrictions need not be considered.

Future efforts will be focused on exploring different information exchange strategies that, while keeping the agents simple, could provide improvements in the quality of the resulting clustering of ant-based clustering algorithms. Among these, the exploration of indirect information exchange through deposition of information packages on the environment.

8 Acknowledgment

The authors would like to thank Julia Handl for clarifying some aspects of her work.

This research has been supported by the Technologies for Distributed Knowledge and Intelligent Agents Research Program CAT-011. Tecnológico de Monterrey, Campus Monterrey.

References

1. Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html] University of California, Irvine, Dept. of Information and Computer Sciences. 1998.
2. Bonabeau, E., Dorigo, M., and Theralauz, G.: Swarm Intelligence. From Natural to Artificial Systems. Oxford University Press. 1999.
3. Deneubourg, J.-L., Goss, S., Franks, N., Sendova-Franks A., Detrain C. and Chretien, L.: The Dynamics of Collective Sorting: Robot-like Ants and Ant-like Robots. Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats. MIT Press. 1991. 356–365
4. Elzinga, Richard J.: Fundamentals of entomology. Prentice Hall. 2000.
5. Fritzke, B.: A growing neural gas network learns topologies. Advances in Neural Information Processing Systems 7. Tesauro, G., Touretzky, D.S., and Leen, T.K. Editors. MIT Press. 1995.
6. Gordon. A. D. : Classification. Second Edition. Chapman & Hall/CRC. Monographs on statistics and applied probability 82. 1999.
7. Handl. J., Knowles. J., and Dorigo M.: Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1D-som. Technical Report TR/IRIDIA/2003-24. IRIDIA, Universite Libre de Bruxelles. Belgium. 2003.
8. Handl. J., Knowles. J., and Dorigo. M.: On the performance of ant-based clustering. Design and application of hybrid intelligent systems. Frontiers in Artificial Intelligence and Applications 104. IOS Press, Amsterdam, The Netherlands. 2003.
9. Holland, O., Melhuish, C.: Stigmergy, Self-Organization, and Sorting in Collective Robotics. Artificial Life. **5** (1999) 173–202
10. Kohonen, T., Huang, T.S., and Schroeder, M.R.: Self-organizing maps. Third Edition. Springer Verlag. 2000.
11. Kube, C.R., Zhang, H.: Collective Robotic Intelligence. Second International Conference on Simulation of Adaptive Behavior. 1992. 460–468
12. Lumer, Erick D., Faieta B.: Diversity and Adaptation in Populations of Clustering Ants. Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3. MIT Press. 1994. 501–508
13. Martinoli A., Mondada, F.: Collective and Cooperative Group Behaviours: Biologically Inspired Experiments in Robotics. Proceedings of the Fourth International Symposium on Experimental Robotics ISER-95. Khatib. O., Salisbury., J.K. Editors. Springer Verlag. 1995. 3–10
14. Wilson. Edward O.: The Insect Societies. The Belkap Press of Harvard University Press. 1971.