Abstract
Ant colonies have been observed to perform tasks similar to clustering. This observation is the inspiration for ant based clustering algorithms, which simulate this behavior on data. This paper investigates the performance of a particular ant clustering algorithm, coined ACLUSTER[16], against another, named ATTA[9], under the measures and datasets proposed by Handl et al.[9]. Based on performance results of both algorithms gathered from numerous runs, the results indicate weaknesses in the design of ACLUSTER, which are not present in ATTA. ACLUSTER is unable to generate a useful clustering, since it fails to reliably join fragments of homogeneous samples into single compact clouds, as well as to enforce spatial separation of those. On the other hand, ATTA’s performance is certainly striking.

1 Introduction
Ant colonies observed in the wild have been attributed with the remarkable habit of accumulating larvae and food in a distinctive order, reminiscent of clusterings of sets of data. This specific kind of behavior and countless of other phenomena are captured in the domain of Swarm Intelligence and have frequently been observed among social insects. A swarm made up by many agents, whose actions are defined by simple individual programs, can achieve intelligent behavior as a whole, often in the light of constraints which prevent centralized coordination of the masses, like the restriction on local information and a lack of direct communication. This is the study of an algorithm named ACLUSTER[16], proposed to generate a useful clustering of related samples, which are initially spread out randomly on real space, essentially by simulating a stochastic model of an ant colony. This real space, a discretized grid with toroidal boundary conditions, is traversed by a number of simple, equivalent agents with the ability to perform shifting operations on all samples situated in the cells of that space. The design of these agents is inspired by plausible principles derived from phenomenological observations of real ant colonies; the actions of each individual agent are triggered on the basis of local, spatially restricted information. There is not centralized control, so no planning whatsoever. There is no direct communication between the agents, with the exception of a pheromone map, another discretized grid, tailored in accordance the the size of the real space. This grid passively holds information for the agents in the form of markers, which indicate how often a certain cell was frequented in the recent history of the ant colony system. Just like pheromone trails attract ants to follow paths in the real world, these markers suggest directions which the agents choose to follow to traverse the grid. As the simulation evolves, the trails captured in the pheromone are subject to change. During traversal, the agents can pick up and drop samples at their current position. Since the samples dispersed over the real space can be compared with the help of a distance metric, the decision to initiate lifting, or placing action of a sample can be modelled in a way, which causes an ordering to emerge, based on the similarity between samples. This ordering, a spatial embedding of the original dataset on the grid can be viewed as a clustering generated by the simulation of an ant colony system on the dataset. A similar ant algorithm coined ATTA[9], produces a spatial embedding of samples in real space as well with one central difference: Its simulation of ant clustering behavior is based on a different ant colony model, underlying different principles and mechanisms. In the same publication, Handl et al.[9] propose datasets which are easy to reproduce and an expressive set of widely adopted measures to determine the quality of a clustering derived directly from the spatial embedding. Here ACLUSTER, proposed by Ramos et al.[16] is evaluated and diagnosed under exactly the same datasets and measures. This is a comparative study of two ant clustering algorithms, with particular emphasis on ACLUSTER, since it is the first time it was evaluated so rigorously.

This paper is arranged as follows: Section 2 clarifies the questions of interest to be addressed during the course of this study. The background of the topic is established in Section 3. Section 4 is dedicated to the outline of both algorithms. The datasets, experimental setup and measures employed in the evaluation of the
algorithm are covered in Section 5. Actual results await in Section 6 and the graphs located in the Appendix. The findings of this paper are concluded in Section 7.

2 Problem Statements

There is a number of reports on the application of ACLUSION to a clustering problem[17, 15, 16]. It is stated that this algorithm can generate useful, implicit clusterings of data. Unfortunately, no numeric result of an experimental application of ACLUDER has been published yet. There are several questions to be addressed in the course of this study:

- Can ACLUDER generate a useful clustering of a dataset?
- How well does it perform?
- How does its performance compare to ATTA's?

ACLUDER itself does not output an explicit clustering in the form of explicit memberships or centroids. Therefore, the quality of a clustering generated by ACLUDER could not have been evaluated analytically without the introduction of an extension. Handl et al.[9] employ Agglomerative Hierarchical Clustering to construct an explicit representation of clusters from a spatial embedding as it is generated by ACLUDER. To establish a common basis for diagnosis and comparison and ultimately, to be able to assess the quality of a clustering generated by ACLUDER on grounds of evidence in numerical form, the evulatory measures and synthetic datasets by Handl et al.[9] are used.

3 Background

This section is meant to clarify why the idea of deploying a simulation of an ant colony on clustering problems is particularly interesting.

3.1 Ant Colony Systems

Ant Colony Systems are popular examples of Swarm Intelligence. The dynamics of an ant colony have been observed and translated into realistic models. Ant Colony Systems have attracted great interest among the computer science community since they have been found a useful framework to generate imperfect but good solutions to problems that are traditionally difficult[12], e.g. the infamous Travelling Salesman Problem[7, 6]. The application of Ant Colony Systems on clustering problems has many intuitive links to aspects of the cognitive and social biology of real ant colonies for foraging or breeding. It has been observed that collectives of Messor sancta[3] and Pheidole pallidula[8] arrange corpses in clusters depending on the size of the individual objects. The ant coined Leptothorax unifasciatus arranges larvae in accumulations according to their respective size[19].

3.2 Clustering

The revelation of natural clusters in data is a standard requirement of many scientific endeavors. The task in a clustering problem is to partition a set of elements into homogeneous and well-separated subsets based on their respective properties, such that, depending on a distance measure, the constraints of minimal intracluster and maximal intercluster distance are satisfied - similar elements are to be placed close to each other, while dissimilar elements need to be clearly separated. Clustering belongs to the class of NP-hard problems[20, 2] in many of its aspects. Extensive search for the optimal clustering is computationally infeasible[21]. Therefore a variety of heuristic algorithms has been proposed and applied successfully to reduce the search space and generate suboptimal but reasonably good solutions in feasible time. Unfortunately, many of these traditional clustering techniques share a number of important drawbacks[11]:

- The number of desired clusters has to be specified a priori, which has an immediate effect on the quality of clusters generated.
- Adding new samples of data may void previous clusters and require recomputation.

3.3 Ant Based Clustering

It is claimed that ant based clustering can overcome these drawbacks, allowing for an explicit determination of the number of clusters inherent to the data[8], as well as giving the flexibility to incorporate new samples into an existing embedding[18], potentially interesting for applications of online clustering, which is not a subject of this paper. Both algorithms are directly linked to various papers which model aspects of ant colonies that are interesting for clustering tasks. The sorting behavior of ants is subject in Deneubourg et al.[5], Lumer and Fäi- eta [1], Bonabeau et al.[3] and certainly Wilson[22]. For readers interested in the full picture, a general overview of phenomena of Swarm Intelligence can be found in Bonabeau et al.[3].

4 Algorithms

The fundamental difference between the algorithms examined by Handl and Ramos is the absence of a pheromone mechanism in ATTA. Instead, Handl's ants use an internal memory of limited capacity to determine their course of movement between positions together with random walk, to reach locations to pick up and drop samples of data in the clustering process. Ramos' algorithm employs pheromone trials, a mechanism which is inspired by the sociobiological workings of real ant colonies[4] to a substantially higher degree.

1Compare: K-means and derivatives
2See: Hierarchical clustering techniques[21]
During the course of this study, ACLUSTER was implemented based on the original design and source code of Handl et al. [10], published under the GNU General Public License. This original framework was designed to investigate the performance of ATTA and was implemented in C++, an fast, compiled and object-oriented language. A perfect fit, since ACLUSTER is required to be run for typically $10^6$ iterations, each involving relatively many calculations, because of the stochastics that are involved.

4.1 ACLUSTER

The details of ACLUSTER are depicted in Algorithm 1. As a side note, a striking resemblance with an ant algorithm proposed by Lumer and Faieta [1] can be observed, with two notable differences:

- ACLUSTER applies a pheromone mechanism for the movement of ants proposed by Chialvo et al. [4], while Lumer and Faieta use random movement.
- The functions for picking / dropping decisions differ.

**Ant Movement**

The spatial state of an individual ant can be expressed in the tuple $s = (p, \theta)$ with position $p$ and orientation $\theta$. Ants are reactive agents, strictly acting on the perception of their immediate neighborhood. Consequently, a bio-inspired model needs to facilitate the mapping between a state $s$ and state $s'$ a time instant later exclusively on information contained in the cells adjacent to the cell where a particular ant exists.

**Pheromone Mechanism:** Ants mark their trails with pheromone[3] continuously. The concentration of pheromone in a specific cell has an immediate effect on the likelihood of an ant in the neighborhood to transfer to that cell. Two interpretations of how ants process pheromonal information have emerged. Osmotropotaxis denotes an instantaneous pheromone following behavior based on the difference of concentrations between two cells. Phenomenologically, it originates from the difference in the perception of pheromone between the antennae of the ant. This characteristic makes the underlying model a Markov process, since a state at a specific moment in time depends on the preceding state exclusively.

With every iteration of the model, the concentrations in the pheromone layer are adjusted. Since close clouds of objects are quite relevant for a clustering problem, the number of neighboring samples $n$ to a cell $i$ is taken into account: The level of pheromone in every cell occupied by an ant is increased by $\eta + \frac{n}{2}$, giving more emphasis to cells located in saturated neighborhoods. The concentration of pheromone in cells that are not coincident with an ant’s position is decreased by some constant value $K$.

Needless to say, $\eta >> K$ is a necessary condition for stable pheromone trails to emerge. The importance of the pheromone layer for the cognitive processes in the ant colony are brought to an intuitive and fascinating interpretation by Chialvo et al.[4]:

Since every ant is restricted to local information and does not benefit from any internal memory, information of the transitions by the individual not held by the individual. The pheromone layer serves as the externalized collective memory of the whole. It captures the recent

\[
\text{Algorithm 1: ACLUSTER}
\]

\begin{verbatim}
input : Dataset $s$
output: Spatial embedding of $s$

1 for every sample $s_i$ do
  2 Place $s_i$ randomly on grid

3 for every agent $a_k$ do
  4 Place $a_k$ randomly on grid

5 for iteration $j = 1$ to $j_{\text{max}}$ do
  6 for all agents $a_k$ do
    7 sum = 0
    8 */ Let Pos(x) denote the position of $x$ on the grid
    9 if $a_k$ is empty AND Pos($a_k$) = Pos($s_i$) then
      10 for all nonempty neighbors $n$ of $s_i$ do
        11 Compute $p_{\text{pick}}$ according to Eq. 6
        12 Draw random number $r \in [0, 1]$
        13 if $r \leq p_{\text{pick}}$ then
          14 sum = sum + 1
        15 if sum $\geq \frac{n}{2}$ then
          16 Agent $a_k$: Pick up item $s_i$
        17 else if $a_k$ carrying $s_i$ AND Pos($a_k$) is empty then
          18 for all nonempty neighbors $n$ of $s_i$ do
            19 Compute $p_{\text{drop}}$ according to Eq. 7
            20 Draw random number $r \in [0, 1]$
            21 if $r \leq p_{\text{drop}}$ then
              22 sum = sum + 1
            23 if sum $\geq \frac{n}{2}$ then
              24 Agent $a_k$: Drop sample $s_i$
            25 Compute $W(\sigma)$, $P_k$ from Eq. 1, 2
            26 Move agent $a_k$ accordingly to an empty cell
            27 Increase pheromone $p_{h}$ at Pos$_k$ to $p_{h} = p_{h} + \eta + \frac{n}{\alpha}$
        28 Globally decrease pheromone by $K$
\end{verbatim}
history of how the environment was traversed by the swarm, and emphasizes areas that are more frequented. With the emergence of pheromone trails, there is potential information on how to navigate between these areas of interest. The distant history of traversals is forgotten after \( \tau = \frac{1}{K} \) iterations, due to the constant evaporation at rate \( K \) at every iteration.

The pheromone weighting function depicted in Equation 1 accepts three parameters; \( \sigma \) is associated with the pheromone density encountered at a specific cell. The parameter \( \beta \) corresponds to the osmotropotaxic sensitivity of the ant - the noise gain. High values for \( \beta \) will increase the the certainty with which an ant follows a pheromone gradient. Conversely, low values for \( \beta \) will lead to more random movement. The sensory capacity of an ant is encoded into the parameter \( \frac{1}{\gamma} \); an ant has a limited dynamic range for the perception of pheromone; high concentrations of pheromone degrade its ability to sense pheromone.

\[
W(\sigma) = \left(1 + \frac{\sigma}{1 + \gamma \sigma}\right)^{\beta} \tag{1}
\]

**Orientational Bias:** Ants are more likely to continue walking along a straight line; u-turns are less likely to occur in nature[4]. These observations have been captured and incorporated into the model by means of the weights displayed in Table 1 for a neighborhood of \( 3 \times 3 \) cells, assuming its orientation coincides with the direction of the arrow in the center of the table.

**Transition Probabilities:** It has been observed that the perception and processing of pheromone is inherently noisy in nature. Therefore the transition mechanism in the model involves the use of probabilities to facilitate movement with a certain degree of weighted randomness. This mechanism allows for the emergence of trail-following behavior and is modelled according to a proposal by Chialvo et al.[4], captured in Equation 2. Let \( N \) be the cells adjacent to \( i \), the immediate neighborhood. To move from cell \( i \) to a specific cell \( k \), a transition probability \( P_{ik} \) is computed. This is done by relating the weight for a particular cell \( k \) in the denominator of Equation 2 to the total weight of the neighborhood \( N \) in the nominator.

\[
P_{ik} = \frac{W(\sigma_k) \cdot w(\Delta_k)}{\sum_j W(\sigma_j) \cdot w(\Delta_j)} \tag{2}
\]

\( P_{ik} \) depends on the two parameters identified earlier in the text.

- The the difference of the ants’ orientation between two successive instants in time captured by parameter \( \Delta \). The weights are chosen into accordance to Table 1.
- The pheromone density in a cell corresponding to parameter \( \sigma \).

Finally, the outcome of a Monte Carlo Simulation determines the target of the transition.

**Shifting Behavior**

The goal of the ant colony is to create compact accumulations of similar samples. It is logical that a sample located in a heterogeneous neighborhood should be picked up for relocation into homogeneous neighborhoods during the iterations of the simulation.

Let \( d \in [0, 1] \) denote the similarity between two samples, based on a distance metric. Both samples are identical if \( d = 0 \), and maximally different if \( d = 1 \), since all distances are normalized with regards to the maximal distance \( d_{\text{max}} \) over the entire dataset. The number of samples contained in the cells adjacent to \( i \), further referred to as neighbors, is denoted by \( n \). In this setup, \( n \in \{1, 2, 3, 4, 5, 6, 7, 8\} \); every cell can have up to eight neighbors.

Samples located in sparse neighborhoods are more likely to be picked up than others in saturated neighborhoods. The threshold function expressed in Equation 3 returns a weight depending on the number of neighbors of a cell, with \( \varphi \) being a threshold constant. For a broader understanding, the reader is referred to the formalization by Bonabeau et al.[3].

\[
\chi = \frac{n^2}{n^2 + \varphi^2} \tag{3}
\]

Whether it is appropriate to drop a sample of data at a specific cell is evaluated using a heuristic on its resemblance with its potential future neighbors. The function of Equation 4 returns a weight depending on the parameter \( d \), the similarity between two samples and is influenced by \( k_1 \), a threshold constant.

\[
\delta = \left( \frac{k_1}{k_1 + d} \right)^2 \tag{4}
\]

Likewise, Equation 5 is used to evaluate whether a sample located at a specific cell should be lifted from its position for relocation. Please note that the slope of the function is determined by threshold constant \( k_2 \).

\[
\varepsilon = \left( \frac{d}{k_2 + d} \right)^2 \tag{5}
\]

Table 1: Directional Weights

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>1/4</td>
<td>1</td>
<td>1/4</td>
</tr>
<tr>
<td>1/12</td>
<td>1/20</td>
<td>1/12</td>
</tr>
</tbody>
</table>

(v. October 22, 2007, p.4)
These threshold functions are the primitives for Equations 6 and 7, which were originally defined by Deneubourg et al.\[5\] and yield probabilities for picking up, or dropping a sample at site $i$ based on examining the cells adjacent to it.

$$p_{\text{pick}} = (1 - \chi) \cdot \varepsilon \quad (6)$$

$$p_{\text{drop}} = \chi \cdot \delta \quad (7)$$

As noted before, the behavior found in an ant colony in nature is inherently noisy. In ACLUSTER, the final decision to pick up or drop a sample is generated on the basis of these probabilities by running Monte Carlo Simulations for every of the $n$ neighbors followed by a democratic decision: If more or equal to $n/2$ runs vote for a positive decision, the sample is picked up.

The relationship between the number of items $n$ in the neighborhood, their dissimilarity $d$ and the resulting probabilities $p_{\text{pick}}$ and $p_{\text{drop}}$ is depicted in Figures 1 and 2 for convenience: The probability for picking up a sample located in sparse and heterogeneous neighborhoods is higher than in saturated and homogeneous neighborhoods. A sample will be dropped with higher probability in neighborhoods that are dense and homogeneous than in neighborhoods which are heterogeneous and contain few samples. Please note that a number of alternative threshold functions exist; the interested reader is referred to [17, 9].

### 4.2 ATTA

ATTA has been discussed and evaluated in detail in Handl et al.[9] and the thesis paper by Handl[8]. The evaluatory framework, as well as the datasets, have been borrowed from these publications. This paper has a focus on ACLUSTER, therefore the outline of ATTA is discussed in less depth. The inner mechanisms of ATTA are translated into intuitive terms and listed in Algorithm 2. The ants employed in ATTA use an internal memory of limited capacity to store positions of samples that were encountered earlier. During iteration, each ant can jump directly to these positions with a certain probability. The higher the similarity between a carried item and the neighborhood of a potential position for dropping, the higher the probability to jump to it. Finally, a dice is thrown to choose from the available targets; if the memory of an ant is empty, the ant is moved randomly.

$$p_{\text{pick}} = \begin{cases} 
1 & \text{if } f^*(i) \leq 1 \\
\frac{1}{f^*(i)} & \text{otherwise}
\end{cases} \quad (8)$$

$$p_{\text{drop}} = \begin{cases} 
1 & \text{if } f^*(i) \geq 1 \\
\frac{1}{f^*(i)} & \text{otherwise}
\end{cases} \quad (9)$$

$$f^*(i) = \begin{cases} 
\frac{1}{\sigma^2} & \text{if } \forall j \in N \left(1 - \frac{d(i,j)}{\alpha} \right) > 0 \\
\frac{1}{\sigma^2} \sum_{j} (1 - \frac{d(i,j)}{\alpha}), & \text{otherwise}
\end{cases} \quad (10)$$

The decisions to pick up or drop a sample at cell $i$ are probabilistic. These probabilities are calculated with the help of Equation 8 for picking and Equation 9 for dropping. Suppose an item is a candidate to be picked up or dropped at cell $i$, and let $N$ be cells around $i$ which hold samples. Both equations depend on the neighborhood function $f^*(i)$ displayed in Equation 10. Based on the similarity between $i$ and $j \in N$ expressed in $d(i,j)$, the computed value for $f^*(i)$ gives an indication of how well that sample at cell $i$ fits into its respective neighborhood $N$. The sensitivity of the function can be adjusted by $\alpha$, a data dependent scaling parameter and $\sigma$, a parameter used to incorporate the size of the neighborhood into the function. Please note that these parameters are subject to modulation during the operation of ATTA, to
Algorithm 2: \textit{ATTA}

\begin{algorithm}
\begin{algorithmic}
\State \textbf{input} : Dataset \(s\)
\State \textbf{output}: Spatial embedding of \(s\)
\For{every agent \(a_i\)}
\State Position \(a_i\) randomly, give it a sample to carry
\EndFor
\For{every remaining sample \(s_k\)}
\State Place \(s_k\) randomly on grid
\EndFor
\For{iteration \(j = 1\) to \(j_{\text{max}}\)}
\State Select an agent \(a_r\) randomly
\State Move \(a_r\) to a location stored in memory, otherwise move randomly
\If{\(a_r\) carries sample \(s\) on a free grid position}
\State Drop in accordance to probabilities from Equation 9
\EndIf
\While{\(a_r\) empty}
\If{Sample \(s\) on \(a_r\)’s position}
\State Pick up \(s\) in accordance to probabilities from Equation 8
\State Iterate over \(s\)
\EndIf
\State Store position in \(a_r\)’s memory
\EndWhile
\EndFor
\end{algorithmic}
\end{algorithm}

improve spatial separation between the samples. For a detailed description of this important feature, the reader is referred to the original sources[9, 8].

4.3 Deriving Clusters from a Spatial Embedding

The application of the analytical tools described in this Section requires an explicit clustering for evaluation. Both \textit{ACLUSTER} and \textit{ATTA} generate spatial orderings of samples in real space, which implicitly hold a clustering. Figure 3 shows such a spatial embedding generated by \textit{ACLUSTER}. Postprocessing with \textit{average agglomerative hierarchical clustering} derives explicit clusters. At initialization, \textit{average agglomerative hierarchical clustering} regards all samples as singleton clusters, and incrementally merges two pairs of clusters that are closest in terms of their average proximity during operation until a stopping criterion is met. This produces explicit clusters as shown in Figure 4, with each sample being a member of a specific cluster.

5 Methodology

5.1 Datasets

The datasets employed in the analysis of \textit{ACLUSTER} are identical to the synthetic datasets applied in Handl et al.[9, 8], to ensure comparability. All datasets are based on four Gaussian distributions \(N(\bar{\mu}, \bar{\sigma})\), with parameters mean \(\bar{\mu}\) and standard deviation \(\bar{\sigma}\). The structure of each cluster is provided by a \textit{cluster prototype}, simply a definition of the mean \(\bar{\mu}\) of the members held in that respective cluster. Each dataset holds a total of 1000 samples that are located in two dimensions. All datasets are generated using \(\bar{\sigma} = [2, 2]\) in both dimensions, with the exception of \(\textit{square0}\), where \(\bar{\sigma} = [1, 1]\). The distance between the centers of the four clusters, \(\Delta C_i\), and the number of samples allocated in a cluster, \(N_C\), is subject to modulation. This gives rise to two families of synthetic datasets, \(\textit{square0}[0\ldots7]\) and \(\textit{sizes}[1\ldots5]\), described in the following sections. The modulation of these features allows to examine the robustness of the algorithm on datasets with overlap between the samples and unequally sized clusters. An example of a dataset of each family in attribute space can be seen in Appendix 9, with obvious cluster centers \(C_1 \ldots C_4\).
Datasets: square[0 . . . 7]
The square datasets are varied with regards to the means \( \bar{\mu} \) of the four clusters such that the distance between their centers, \( \Delta_C \) is gradually reduced. This gives rise to an increase in overlap between the distinct clusters. With each cluster being constituted by 250 samples, all are equally weighted. The parameters of the individual sets contained in this family are summarized neatly in the top rows of Table 2.

Datasets: sizes[1 . . . 5]
The sizes datasets impose unequal weights to the clusters, while retaining the same intercluster distance of \( \Delta_C = 10 \) over all sets. Instead of using a constant of 250 samples for each cluster, the number of samples constituting one of the clusters are gradually increased. For the parameters, you are referred to bottom rows of Table 2.

5.2 Parameters
Ramos omits the derivation of how to set the size of the environment, as well as the number of agents constituting the ant population, two parameters of crucial importance. Luckily, some reasonable relationship can be derived from [13], which is sadly a source of unconfirmed reliability. Accordingly, the following rules should be obeyed:

- A dataset of \( n_d \) samples is placed on a square grid with approximately \( 4n_d \) cells.
- The number of agents \( n_a \) is roughly \( \frac{n_d}{10} \).

All of the other parameters found in Ramos et al.[16] are listed in Table 3 along with their occurrence in chronological order. The parameters have been proposed in correspondence with empirical relationships, qualified by their ability to produce actual phenomena.

<table>
<thead>
<tr>
<th>Name</th>
<th>( C )</th>
<th>( \Delta_C )</th>
<th>( \bar{d} )</th>
<th>( C_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>square0</td>
<td>4</td>
<td>10</td>
<td>[1, 1]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>square1</td>
<td>4</td>
<td>10</td>
<td>[2, 2]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>square2</td>
<td>4</td>
<td>9</td>
<td>[2, 2]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>square3</td>
<td>4</td>
<td>8</td>
<td>[2, 2]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>square4</td>
<td>4</td>
<td>7</td>
<td>[2, 2]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>square5</td>
<td>4</td>
<td>6</td>
<td>[2, 2]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>square6</td>
<td>4</td>
<td>5</td>
<td>[2, 2]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>square7</td>
<td>4</td>
<td>4</td>
<td>[2, 2]</td>
<td>{200, 200, 200, 200}</td>
</tr>
<tr>
<td>sizes1</td>
<td>4</td>
<td>10</td>
<td>[2, 2]</td>
<td>{400, 200, 200, 200}</td>
</tr>
<tr>
<td>sizes2</td>
<td>4</td>
<td>10</td>
<td>[2, 2]</td>
<td>{571, 143, 143, 143}</td>
</tr>
<tr>
<td>sizes3</td>
<td>4</td>
<td>10</td>
<td>[2, 2]</td>
<td>{667, 111, 111, 111}</td>
</tr>
<tr>
<td>sizes4</td>
<td>4</td>
<td>10</td>
<td>[2, 2]</td>
<td>{727, 91, 91, 91}</td>
</tr>
<tr>
<td>sizes5</td>
<td>4</td>
<td>10</td>
<td>[2, 2]</td>
<td>{769, 77, 77, 77}</td>
</tr>
</tbody>
</table>

Table 2: Parameters of the Datasets

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \eta )</td>
<td>0.07</td>
<td>Sect. 4.1</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>400</td>
<td>Sect. 4.1</td>
</tr>
<tr>
<td>( K )</td>
<td>0.015</td>
<td>Sect. 4.1</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>0.2</td>
<td>Sect. 4.1, Eq. 1</td>
</tr>
<tr>
<td>( \beta )</td>
<td>3.5</td>
<td>Sect. 4.1, Eq. 1</td>
</tr>
<tr>
<td>( \varphi )</td>
<td>5</td>
<td>Sect. 4.1, Eq. 3</td>
</tr>
<tr>
<td>( k_1 )</td>
<td>0.1</td>
<td>Sect. 4.1, Eq. 4</td>
</tr>
<tr>
<td>( k_2 )</td>
<td>0.3</td>
<td>Sect. 4.1, Eq. 5</td>
</tr>
<tr>
<td>( j_{max} )</td>
<td>10^5</td>
<td>Sect. 4.1</td>
</tr>
</tbody>
</table>

Table 3: Static parameters of ACLAUSTER

5.3 Measures
Simply comparing the number of clusters found in the embedding with the number of clusters contained in the original data can give an idea of the performance of the algorithm, although on a very superficial level. The correct number of clusters is known beforehand, being a parameter for the generation of each dataset prior to clustering. In cases where the exact number is unknown, the Gap statistic[14] comes in handy. Naturally, assessing the quality of a clustering needs more expressive methods, which aid the diagnosis of potential inadequacies. These methods are the subject of this section.

F-Measure
Let \( m_i \) be the members of cluster \( i \) and \( m_j \) the members of class \( j \). Precision in Equation 11 measures the degree to which a cluster \( i \) is made up by members of class \( j \), while recall in in Equation 12 gives the fraction to which all members of class \( j \) are contained in a specific cluster \( i \). The F-Measure depicted in Equation 13 determines “the extend to which a cluster contains only objects of a particular class and all objects of that class” (Tan et al. [21, p. 549]), based on the measures for precision and recall. The weighting of precision and recall can be adjusted with the help of parameter \( \alpha \); typically \( \alpha = 1 \) is used, giving equal weight to both.

\[
\text{precision}(i, j) = \frac{|m_i \cap m_j|}{|m_i|} \tag{11}
\]

\[
\text{recall}(i, j) = \frac{|m_i \cap m_j|}{|m_j|} \tag{12}
\]

\[
F_\alpha(i, j) = \frac{(1 + \alpha) \cdot \text{precision}(i, j) \cdot \text{recall}(i, j)}{\alpha \cdot \text{precision}(i, j) + \text{recall}(i, j)} \tag{13}
\]

Dunn Index
The Dunn Index, shown in Equation 14, determines the quality of a clustering by relating the minimal intercluster distance \( \Delta \) between two centroids \( \mu_k \) and \( \mu_l \) to
the maximum intracluster distance, the diameter of a cluster \( m \).

\[
D = \frac{\min_{k \in C} \sum_{m \in C} [\Delta(\mu_k, \mu_l)]}{\max_{m \in C} \text{diam}(m)} \tag{14}
\]

**Intracluster Variance**

Members of the same cluster should be similar. This measure, displayed in Equation 15, sums the squared distance \( \Delta \) between each member \( i \) of cluster \( k \) and its respective centroid \( \mu_k \) over all clusters \( C \).

\[
I = \sum_{k} \sum_{i} \Delta(i, \mu_k)^2 \tag{15}
\]

6 Discussion

The reader will find the results of **ACLUSTER** arranged in figures in the Appendix, starting from page \( v \). Results of **ATTA** follow from page \( v \). These plots depict the means for each of the evaluatory measures as well as the standard deviation over ten independent runs of the algorithm on each dataset.

**Observations**

Figure 3 shows an exemplary spatial embedding of the *square0* dataset produced by **ACLUSTER** after \( 10^6 \) iterations of the ant colony simulation. As can be seen, the members of each of the four clusters \( C_1, C_2, C_3, C_4 \) are not arranged in single compact clouds. For example, the samples that share membership of of cluster \( C_3 \) are distributed in three disconnected areas in the exemplary embedding. This has a direct consequence on intracluster distances of the respective cluster. A clustering of high quality requires samples that share membership in a cluster in attribute space to be situated in close proximity in the embedding. This is clearly not the case in the example depicted in Figure 3. The results for the F-Measure depicted in Appendix 2(a) and 4(a) provide additional evidence. Since similar samples are fragmented in small accumulations at various locations it is unlikely for a cluster to hold all samples of one class. Thus, the algorithm scores a low recall value, and consequently a low value for the F-Measure. **ACLUSTER fails to generate compact clusters of homogeneous content.**

As stated in Section 3.2, a useful clustering requires the distance between allocations of samples that differ in attribute space to be maximized. As can be seen in the bottom center of Figure 3, accumulations of samples that are members of \( C_1, C_2, C_3 \) touch at their edges, which has a direct impact on intercluster distances in the derived clustering. Consequently, the intracluster variance shown in Appendix 2(b) and 4(b) is high over all datasets, an indication that the clusters found are heterogeneous accumulations of samples which lack purity. Since the F-Measure incorporates the precision component, it is able to support this observation as well. Additional evidence can be found in the high scores for Intracluster Variance arranged in Appendix 2(b) and 4(b), since the clusters generated from the spatial embedding span different classes. Finally, the particular low values for the Dunn Index arranged in Appendix 1(b) and 3(b) over all datasets once again reinforce the lack of spatial separation that was observed. The values of this last measure are also subject to high deviation. This suggests that **ACLUSTER** severely lacks a robust ability to generate clusters that are dense and clearly separated.

Needless to say, the correct number of clusters implicit to the datasets could not be revealed at all, expressed in Appendix 1(a) and 3(a). **ACLUSTER fails to separate clusters.** Investigating the robustness of the algorithm on the individual datasets is pointless in the light of this result.

In comparison, the results for **ATTA** are strikingly positive. **ATTA** robustly outperforms **ACLUSTER** on all *sizes* datasets by unravelling the correct number of clusters displayed in Appendix 7(a) and receives high scores on the F-Measure depicted in Appendix 8(a), which indicate the capability of this algorithm to generate clusters that are complete and well separated. The scores for the Dunn Index in Appendix 7(b) are additional evidence for this claim. This performance on the *sizes* dataset indicate its robustness towards unequally weighted clusters in the data. **ATTA**’s performance on the *squares* datasets degrades gracefully with the decreasing separation between the cluster centers modulated into the datasets. Still, the scores received on the F-Measure on these datasets shown in Appendix 6(a) and the results for the Dunn Index in Appendix 5(b) indicate a tendency of **ATTA** to preserve the internal structure of the data, even when delineation becomes more difficult.

The results of **ACLUSTER**, inferior in comparison with **ATTA**, are the repercussions of the Response Thresholds discussed in Section 4.1. These functions fail to eventually merge fragmented accumulations of similar samples into homogeneous clusters. Neither do they seem to penalize dissimilarities heavily enough, such that a clear spatial separation emerges. The resulting spatial embedding as shown in Figure 3 is full of flaws. The consequence is an explicit clustering as shown in Figure 4, that does not represent the internal structure of the data. Although **ATTA** makes use of very different mechanisms, its spatial embedding suffers from similar deficiencies during some phases of operation. The parameters of the function responsible for picking / dropping

---

\(^3\)For values of the Dunn Index < 1, the maximal diameter of a cluster is larger than the smallest distance between two clusters, potentially indicating overlap in attribute space.
ping decisions in ATTA-C are modulated to enforce an embedding which is more strict towards dissimilarities and effectively supports the separation of distinct clusters. The downside is that early accumulations of homogeneous samples are broken up and dispersed over the grid during its application. Therefore, the modulation of this function is applied in phases. Whether the introduction of this modification into ACLUSTER could improve the performance remains an interesting question. Despite this specific potential improvement in the future, a statement about the computational performance has to be made: The rules underlying the movement of the ant colony system in ACLUSTER have been inspired by phenomenological observations. As a consequence, the pheromone mechanism outlined in Section 1 requires local information for each ant in every iteration. Since every ant is bound to move no more than one cell at a time, many iterations are required to ensure sufficient traversal of the environment. Given \(10^6\) iterations, the typical running time of this algorithm is 30 times higher than of ATTA-C[9], where considerably fewer ants are used, all equipped with internal memory to store promising dropping locations, and the ability to jump multiple positions at once. It is yet to be determined whether there is some characteristic utility attached to a pheromone mechanism that can possibly outweigh the computational overhead necessary.

7 Conclusion

In the preceding sections the motivation for ant based clustering algorithms has been outlined together with two existing ant algorithms developed for this task. With the help of expressive measures and synthetic datasets by Handl et al.[9], it was possible to assess the performance of the algorithms. ACLUSTER has been meant to be an ant clustering system in Ramos[17]. The results indicate several weaknesses. The performance of the algorithm suffers severely from its inability to arrange similar samples in compact clusters and clearly delineate allocations of dissimilar samples. Visual inspection of spatial embeddings and the resulting clustering support these observations as well as numerical results. The reason for the inferior performance is arguably the application of the Response Thresholds introduced in Section 4.1. The functions deciding the spatial relocation of samples are unsuited to enforce complete and well separated clusters. ATTA is an example of an ant algorithm that is well capable of clustering tasks. It is able to detect the correct number of clusters in the data, and preserves the structure contained in the dataset. ATTA performs robustly on datasets of unequally weighted clusters.

References


(v. October 22, 2007, p.9)


Appendix 1: Evaluatory results of ACLUSTER on square[0..7]

(v. October 22, 2007, p.i)
Appendix 2: Evaluatory results of ACLUSTER on square[0..7] cont.

(v. October 22, 2007, p.ii)
Appendix 3: Evaluatory results of *ACLUSTER* on *sizes*[1..5]

(v. October 22, 2007, p.iii)
Appendix 4: Evaluatory results of ACluster on sizes[1..5] cont.

(v. October 22, 2007, p.iv)
Appendix 5: Evaluatory results of ATTA on square[0..7]

(v. October 22, 2007, p.v)
Appendix 6: Evaluatory results of *ATTA* on *square*[0..7] cont.

(v. October 22, 2007, p.vi)
Appendix 7: Evaluatory results of \textit{ATTA} on \textit{sizes[1..5]}

(v. October 22, 2007, p.vii)
Appendix 8: Evaluatory results of ATTA on sizes$[1..5]$ cont.

(v. October 22, 2007, p.viii)
Appendix 9: Two datasets

(a) *Square*1, with cluster centers $C_1 \ldots C_4$ at [1, 1], [-1, -1], [-1, 1] and [1, -1].

(b) *Sizes*5, with the majority of samples in $C_1$.