Simplifying and improving ant-based clustering

Swee Chuan Tan, Kai Ming Ting, Shyh Wei Teng

SIM University, School of Business, Singapore
Monash University, Faculty of Information Technology, Australia

Abstract

Ant-based clustering (ABC) is a data clustering approach inspired from cemetery formation activities observed in real ant colonies. Building upon the premise of collective intelligence, such an approach uses multiple ant-like agents and a mixture of heuristics, in order to create systems that are capable of clustering real-world data. Many recently proposed ABC systems have shown competitive results, but these systems are geared towards adding new heuristics, resulting in increasingly complex systems that are harder to understand and improve. In contrast to this direction, we demonstrate that a state-of-the-art ABC system can be systematically evaluated and then simplified. The streamlined model, which we call SABC, differs fundamentally from traditional ABC systems as it does not use the ant-colony and several key components. Yet, our empirical study shows that SABC performs more effectively and efficiently than the state-of-the-art ABC system.

Keywords: Data Clustering, Ant-Based Clustering, Swarm-based Clustering

1. Introduction

Ant-based clustering (ABC) is inspired from the cemetery formation and brood-sorting activities found in real ant colonies. This approach is motivated by the collective problem solving ability found in simple social insects (ants, bees, termites, etc) [1]. Initially proposed by Deneubourg et. al [2], robotic ants was used to cluster physical objects. Deneubourg’s model was later extended by Lumer and Faieta [3] to cluster numerical data. Since then, a number of variants have been proposed; some interesting examples include: (i) combining ant-based clustering with K-means [4, 5], (ii) using pheromone in ant-based clustering [6], and (iii) specific extensions of Lumer and Faieta’s model: Ant-Q [7] and its successor ATTA [8].

Most of the recent studies in ant-based clustering have shown competitive results against traditional clustering methods; however, most of the proposed methods are geared towards incorporating more complex heuristics and mechanisms in order to improve clustering performance on real-world data. For example, AntClass [5] is a hybrid of Lumer and Faieta’s model and K-means [4]; Ant-Q [7] uses ant-like agents with memories and different movement speeds; and the ant-like agents in ATTA [8] can “look ahead” and “jump” to sites that contain the most similar...
items. Furthermore, these systems employ multiple ant-like agents and the complex interactions among the agents are difficult to elucidate.

While the current research direction inclines towards adding new mechanisms to improve clustering performance, we postulate that this direction is wrong. This is because when models become more complex, there would be less rooms for improvement. In this paper, we take the opposite approach—to explore ways to prune down the complexity of ant-based clustering and at the same time improve the clustering result. The investigation is guided by the principle of minimal (system) complexity—if two systems with different complexities can perform the same function, then the simpler system is preferred over the more complex one.

This study examines a well-cited system, ATTA [8], which is a state-of-the-art ABC system that has all the common characteristics of a typical ABC model. In Section 2, we begin by reviewing ant-based clustering and identifying the key components in ATTA. In Section 3, we introduce SABC, a simplified version of ATTA. Section 4 then provides experimental evidence showing that ATTA can be simplified, and the results help to justify the proposed configurations of SABC. It also shows that SABC performs more effectively and efficiently than ATTA. Finally, Section 5 provide some concluding remarks.

2. Ant-based clustering inspired by cemetery formation

The starting point of ant-based clustering method is interesting—real ants have been found to gather dead ants at some common areas within the colony [2], presumably to clean their nest. In the process of creating cemeteries, the ants communicate with each other indirectly via the distribution of dead ants within their nest. Each ant works autonomously and uses only local information (i.e., the proportion of dead ants that it has recently seen). There is neither supervisor nor master plan, yet the ants appear to work as a team to build the cemeteries.

Ant-based clustering was initially proposed by Deneubourg et al. [2], who used robotic ants to cluster physical objects. Initially, objects and ant-like agents are scattered over a two-dimensional (2D) space at random locations (c.f. Figure 1a). These autonomous ant-like agents explore their environment with random movements, and each of them operates with simple rules—an unloaded agent would probabilistically pick up an isolated object and transport it with random movement; a loaded agent would probabilistically drop its object if it comes across an area with many other objects. When the agents repeatedly perform these simple actions, groups of items are gradually created (c.f. Figure 1b).

Deneubourg’s work has sparked further investigations in several directions. One interesting direction is on its application to cluster analysis, proposed by Lumer and Faieta [3].

2.1. Lumer and Faieta’s Model

To extend Deneubourg’s model [2] for numerical data analysis, Lumer and Faieta [3] proposed a neighbourhood function to compute the average similarity between an item $i$ and each item $(j)$ in its surroundings:

$$f(i) = \max \left( 0, \frac{1}{\sigma^2} \sum_{j} \left( 1 - \frac{\delta(i,j)}{\alpha} \right) \right). \quad (1)$$
where $\alpha$ is a threshold parameter for the distance measure $\delta(i, j)$ between a currently picked data item $i$ and all the data items $j$ in the neighbourhood of $i$. The size of the ant’s perception region (or neighbourhood) is $\sigma^2 = (2r + 1)^2$, where $r$ is the perceptive radius and $\sigma^2$ is typically 9 or 25. Intuitively, a high $f(i)$ means that $i$ is surrounded by many similar items, otherwise $f(i)$ is low.

The probabilities of picking up and dropping an object $i$ are shown in the following two equations respectively:

$$
p_{\text{pick}}(i) = \left(\frac{k^+}{k^+ + f(i)}\right)^2; \quad (2)
$$

and

$$
p_{\text{drop}}(i) = \begin{cases} 
2f(i), & \text{if } f(i) < k^- \, , \\
1, & \text{otherwise}.
\end{cases} \quad (3)
$$

Here, $k^+$ and $k^-$ are parameters between 0 and 1. These parameters decide how influential $f(i)$ is on $p_{\text{pick}}(i)$ and $p_{\text{drop}}(i)$. An unloaded ant is likely to pick up an item $i$ if $f(i)$ is small with respect to $k^+$; this means that the item is either isolated or is surrounded by many dissimilar items. Similarly, a loaded ant is likely to drop an item $i$ if $f(i)$ is large with respect to $k^-$; this means that the ant encounters many similar items. Consequently, this positive feedback mechanism promotes the formation of clusters on the 2D space.

The basic Lumer and Faieta’s algorithm, which is adapted from the one presented in Handl’s thesis [9], is presented in Algorithm 1. Initially, the items are randomly scattered on a 2D grid and each agent is loaded with an item. When the main loop starts, each ant moves randomly and begins to pick up and drop items. The random function $\text{Rand}[0, 1]$ generates a random value in $[0, 1]$.

---

**Algorithm 1 Basic Ant-based Clustering**

**INITIALIZATION**

Randomly scatter $n$ items on a 2D toroidal grid

Let $G$ be a population of agents

Each agent in $G$ is randomly assigned (or loaded with) an item

**MAIN LOOP**

for iteration = 1 to maxIteration do

\hspace{1em} $g := \text{an agent randomly selected from } G$

\hspace{1em} Let the item carried by $g$ be $i$

\hspace{1em} $g$ performs a random move on the grid

\hspace{1em} if $p_{\text{drop}}(i) > \text{Rand}[0, 1]$ then

\hspace{2em} $g$ drops item $i$ at its current location

\hspace{2em} PICK := false

\hspace{2em} while ¬PICK do

\hspace{3em} $g$ moves on the grid randomly until it encounters an item $q$

\hspace{3em} if $p_{\text{pick}}(q) > \text{Rand}[0, 1]$ then

\hspace{4em} PICK := true

\hspace{4em} $g$ is loaded with $q$

\hspace{3em} end if

\hspace{2em} end while

\hspace{1em} end if

end for

---

In Lumer and Faieta’s model, each ant has a short-term memory recording a list of items that the ant has recently visited. When an ant picks up an item, it searches through this memory to find the item that is the most similar to the picked item, and then gears its movement towards the position of the most similar item. We call this strategy Matching Search.

The basic Lumer and Faieta’s model, has formed the basis for many subsequent works that adopt the cemetery formation approach to data clustering. In the next section, we describe a significant recent extension.
2.2. ATTA

Handl et al. proposed seven new strategies [8] to improve Lumer and Faieta’s model:

a) The neighbourhood function is modified as follows:

\[
f^+(i) = \max \left( 0, \frac{1}{N_{\text{size}}} \sum_j \left( 1 - \frac{\delta(i,j)}{\alpha} \right) \right), \tag{4}
\]

where \( \alpha \) is a threshold parameter for the distance measure \( \delta(i,j) \) between a currently picked up data item \( i \) and all the data items \( j \) in the neighbourhood of \( i \). \( N_{\text{size}} \) is fixed at 8 regardless of the size of the neighbourhood. With this setting, ATTA tends to over-penalise \( f^+(\cdot) \) during the initial stage of the clustering process; that is, when the perceptive radius is one, the neighbourhood is small and \( f^+(\cdot) \) tends to be very small. As a result, ATTA can not perform any effective clustering at this stage. This begs the question of whether \( N_{\text{size}} \) can be adjusted to improve ATTA right at the outset of the clustering process.

b) To penalise items with a high dissimilarity in the same grid-neighbourhood, an additional constraint is used to set the neighbourhood function to zero when a neighbour is of a distance more than \( \alpha \) away from \( i \):

\[
f^*(i) = \begin{cases} 
  f^+(i), & \text{if } \forall j, \delta(i,j) < \alpha \\
  0, & \text{else.} 
\end{cases} \tag{5}
\]

Previous research has claimed that this heuristic improves clustering quality[8], but its effect on runtime is still not known.

c) The perceptive radius \( (r) \) of the ants is increased from one to five, over equal time intervals of the clustering process. This mechanism aims to increase the spatial distances between any two clusters on the grid. The problem here is that the time to evaluate the neighbourhood function increases quadratically with increasing \( r \). In this paper, we will explore if \( r \) can be reduced without affecting clustering quality.

d) A new set of pick up and drop rules (also called the response threshold functions) are used to speed up the clustering process and improve clustering accuracy:

\[
p^*_{\text{pick}}(i) = \begin{cases} 
  1, & \text{if } f^*(i) \leq 1 \\
  \frac{1}{f^*(i)^2}, & \text{else;} 
\end{cases} \tag{6}
\]

and

\[
p^*_{\text{drop}}(i) = \begin{cases} 
  1, & \text{if } f^*(i) \geq 1 \\
  f^*(i)^4, & \text{else.} 
\end{cases} \tag{7}
\]

e) To increase the spatial separation between any two clusters on the 2D space, a process to disperse data items is introduced in the midst of the clustering process. This mechanism is called the “Time-dependent modulation of neighbourhood function” (TMNF). This is a kind of behavioral switch because normal ant behaviour, is to group similar items together, rather than to disperse them. Note that TMNF shares the same purpose as increasing \( r \), so it is necessary to determine if TMNF is really required.

f) Similar to Lumer and Faieta’s model, each ant has a short-term memory. To find the best possible drop site for each picked item \( i \), each ant finds the memorised item with grid neighbours that are the most similar to \( i \). We call this heuristic the Lookahead Search strategy. Note that a short-term memory with a size \( m \) can only remember at most \( m \) cluster locations. This can become a problem when the number of clusters is more than \( m \). In ATTA, \( m = 10 \).

g) To automatically estimate the parameter \( \alpha \) used in Equation 4, ATTA uses a population of ants to estimate multiple \( \alpha \)-values during the clustering process. This mechanism is important in ensuring that ATTA is able to detect the number of clusters automatically.
3. Streamlining ant-based clustering

In this section, we explore an important question: How far can we prune the complexity of an ABC algorithm and still preserve its functional capability? We present a Simplified version of Ant-based Clustering (SABC) that only employs effective heuristics in ABC. The proposed method differs from existing ABC methods in the following ways:

a) The first fundamental difference is that SABC does not employ an ant-colony. It is a monolithic stochastic clustering algorithm (i.e., equivalent to one ant) rather than a multi-agent swarm-based clustering algorithm. This difference is significant because it challenges the premise of swarm-based clustering, which asserts that multiple agents are essential in the formalism.

b) We avoid the trappings of mimicking nature – SABC does not employ two common components found in existing ant-based clustering models:

- SABC does not employ any behavioural switches such as switching ant behaviour to destroy clusters (as in Lumer and Faieta’s model [3]) or dispersing data items (i.e., TMNF [9]) during the clustering process.
- SABC avoids the use of a short-term memory by using an improved search strategy to find drop sites for each picked item. It uses the Matching Search strategy (in the Lumer and Faieta’s model) during the first 70% of the clustering process, this allows clusters to be built up quickly; the rest of the clustering process uses ATTA's Lookahead Search strategy to produce more homogenous clusters. Each time, when the algorithm needs to find a drop site, ten data items are randomly drawn from the dataset, and the grid-positions of these items are used as potential drop sites by the search heuristic. Unlike ATTA, these randomly-generated locations differ over time; thus this mechanism overcomes the limit on the number of clusters that can be detected.

c) SABC uses a set of improved parameter settings:

- Using a neighbourhood size of four (i.e., \( N_{size} = 4 \)) instead of eight in Equation 4. This setting allows more effective clustering because the neighbourhood function acts like an average similarity function right at the start of the clustering process.
- Using a maximum perceptive radius \( R_{max} \) of three instead of five. This setting creates a clearance of at least two grid-cells between any two distinct clusters on the 2D grid-space. Such a clearance is sufficient for avoiding cluttering different clusters on the grid.
- Adopting the automatic \( \alpha \)-adaptation scheme, but simplifying it by estimating only one \( \alpha \)-parameter (due to the use of one ant) rather than ATTA's approach of estimating multiple \( \alpha \)-parameters.

Aside from these differences, SABC is similar to existing ant-based clustering methods in the following ways:

- Like most ABC models, SABC uses a 2D grid as a workspace to perform clustering of data items.
- SABC uses the neighbourhood function (i.e., Equation 5) and response threshold functions (i.e., Equations 6 and 7) in ATTA.

4. Experimental setup and results

We conduct the experiments in two phases. The first phase examines the effects of different components in ATTA. The main aim of this phase is to identify effective as well as redundant components. The results obtained from this phase help to justify the SABC configurations proposed in Section 3. The second phase then gives a head-to-head comparison between ATTA and SABC.

In total, we use 25 datasets, 22 of which were previously used for testing ATTA [9]. There are 10 synthetic datasets, including the Squares series, Sizes series, Skew, VaryDensity and Triangle [11]. Characteristics of these datasets are shown in Figure 2. In addition, we have the xDyC series, \( x = \{ 2, 10, 100 \} \) is the number of dimensions, and \( y = \{ 4, 10 \} \) is the number of clusters. As a result, there are six variations of xDyC datasets in terms of dimensionality.
Table 1: Real-world datasets adopted from the repository of the UCI Machine Learning databases [10]. \( N \) is the number of instances in the dataset; \( N_i \) is the number of instances for cluster \( i \); \( D \) is the dimensionality and \( C \) is the number of clusters.

| Dataset     | \( N \) | \( N_i \) | \( D \) | \( C \) |
|-------------|--------|----------|--------|--------|
| Wisconsin   | 699    | 458, 241 | 9      | 2      |
| Dermatology | 366    | 112, 61, 72, 49, 52, 20 | 34     | 6      |
| Zoo         | 101    | 41, 20, 5, 13, 4, 8, 10 | 16     | 7      |
| Wine        | 178    | 59, 71, 48 | 13     | 3      |
| Iris        | 150    | 50, 50, 50 | 4      | 3      |
| Digits      | 3498   | 363, 364, 364, 336, 364, 335, 336, 364, 336, 336 | 16     | 10     |
| Yeast       | 1484   | 463, 429, 244, 163, 51, 44, 37, 30, 20, 5      | 8      | 10     |

Figure 2: Example of ten synthetic datasets. Each dataset contains 1000 instances and four bivariate Gaussian clusters. Squares series contain increasingly overlapping clusters. Sizes series contain clusters with increasingly uneven sizes. Triangle contains clusters with different shapes; VaryDensity contains clusters with different densities; and Skewed contains clusters with different distances between their centroids.

and number of clusters. The real-world datasets include Wisconsin, Iris, Wine, Dermatology, Zoo, Yeast and Digits, all taken from UCI Machine Learning Repository [10], with details given in Table 1. Further details of these datasets can be found in Handl’s thesis [9].

For each dataset, we perform 50 independent runs and measure the performance using three evaluation measures. These measures are the number of clusters detected, the runtime (in seconds) taken and the F-measure [12].

The F-Measure is one of the key measures used for evaluating ATTA. It quantifies how well a set of generated clusters match the actual classes in the dataset. Let \( n_i \) (respectively \( n_j \)) be the number of items in class \( i \) (respectively cluster \( j \)), then \( n_{ij} \) is the number of items in class \( i \) within cluster \( j \). The F-Measure is then defined as:

\[
F(i, j) = \frac{2 \cdot prec(i, j) \cdot rec(i, j)}{prec(i, j) + rec(i, j)}
\]

for each class \( i \) and cluster \( j \), \( prec(i, j) = n_{ij}/n_j \) and \( rec(i, j) = n_{ij}/n_i \) are defined as the precision and the recall respectively. Note that this objective measure treats cluster retrieval like an information retrieval problem [12]. The overall F-Measure can be computed as:

\[
F = \sum_i \frac{n_i}{n} \max_j \{F(i, j)\}, \quad (8)
\]

where \( n \) is the total number of items in a dataset. F-Measure is in the interval \([0, 1]\) and equals 1 as perfect clustering.

All the results reported in this section are based on the original C++ implementation of ATTA [9]. All experiments have been conducted on a Linux machine with AMD Opteron Processor 265, 1.8GHz, 1M cache and 4GB RAM.
Table 2: Overall results of ATTA’s components using 10 Synthetic datasets. Row (1) contains the baseline results. Results in Rows (2) to (6) are to be compared against Row (1). #Cluster is the number of clusters detected and the runtime is in seconds. Each entry contains mean ± standard deviation.

| Model Description                                                                 | F-Measure   | #Clusters   | Runtime     |
|----------------------------------------------------------------------------------|-------------|-------------|-------------|
| (1) Baseline results of the original ATTA implementations                        | 0.87 ± 0.03 | 3.53 ± 0.25 | 14.35 ± 3.70 |
| (2) Results of ATTA without ant-colony, i.e., using only “1 ant”                 | 0.87 ± 0.03 | 3.53 ± 0.25 | 14.58 ± 3.76 |
| (3) Results of ATTA using a maximum perceptive radius ($R_{max}$) of 3; instead of 5 | 0.87 ± 0.02 | 3.53 ± 0.27 | 8.34 ± 0.58  |
| (4) Results of ATTA without the constraint (in Equation 5) that penalises the neighbourhood function when highly dissimilar items are found in the same neighbourhood | 0.82 ± 0.06 | 3.11 ± 0.37 | 34.07 ± 2.33 |
| (5) Results of ATTA without using short-term memory                               | 0.88 ± 0.02 | 3.54 ± 0.17 | 29.66 ± 4.89 |
| (6) Results of ATTA without Time-dependent Modulation of Neighbourhood Function   | 0.89 ± 0.02 | 3.71 ± 0.22 | 14.86 ± 1.67 |

4.1. Results of Evaluation of ATTA’s Components

Here, we present the results of evaluating the components in ATTA. Table 2 shows the results of the comparative experiments. Note that Row (1) of the table contains the clustering results of the original ATTA implementation, and this will be the baseline. The results are obtained by taking average of the results obtained from the ten synthetic datasets as shown in Figure 2. This gives an indication of the algorithms’ overall performance. In the following, we summarise the findings.

Effects of removing the ant-colony: The motivation for using multiple agents is based on the assumption that a single agent alone cannot solve the clustering problem effectively. Row (2) of Table 2 shows that ATTA using only a single ant, performs comparably with ATTA using multiple ants (as shown in Row (1)), both in terms of clustering quality and runtime. Thus, there is also no need to use a population of ants. This implies that existing Swarm-based clustering algorithms can be simplified into simple monolithic (i.e., single-agent) stochastic clustering algorithms.

A recent theoretical proof [13] has shown that a single ant can also be used to perform a simplistic task of clustering objects with known types. However, this proof is based on a toy example and cannot be generalised to the task of clustering complex real-world data. As will be shown in Section 4.2, we will provide the first thorough empirical evaluation showing that ant-colony can be removed from a state-of-the-art ant-based clustering system, and the evaluation will be based on non-trivial, real-world data clustering tasks.

Effects of reducing the maximum perceptive radius: Row (3) of Table 2 shows that using $R_{max} = 3$ (instead of $R_{max} = 5$) reduces the average runtime from 14.35 seconds to about 8.34 seconds. The runtime is effectively reduced because the maximum number of items in each neighbourhood is reduced from 121 to 49. Note also that this is achieved without sacrificing the clustering accuracy, as indicated by the average F-Measure and number of clusters detected, which are similar to that of the original ATTA shown in Row (1).

Effects of removing the penalising constraint in the neighbourhood function: Row (4) of Table 2 shows that removing the additional constraint (Equation 5) in the neighbourhood function not only degrades the clustering accuracy; but also doubles the runtime! This effect on the runtime was not known previously. Furthermore, the runtime result is counterintuitive because there is no apparent reduction in the same number of distance computations, judging from the equation alone.

We find that the improvement in runtime is due to the efficient design of neighbourhood function algorithm in ATTA. As soon as a dissimilar item is found in a neighbourhood (i.e., $\delta(i, j) < \alpha$), ATTA’s neighbourhood function returns a zero immediately and stops any further distance computations. Thus, ATTA’s neighbourhood function runs
a lot faster than a neighbourhood function that evaluates all items in the neighbourhood. One example of a slow implementation is the one in Lumer and Faieta’s model (see Equation 1).

**Effects of not using the short-term memory:** To isolate the effects of the short-term memory with size \( m \), a set of \( m \) random locations are generated and used by an ant every time it seeks to find a drop site. Row (5) of Table 2 shows that ATTA without a memory has a runtime at least double of ATTA with memory, although no significant improvement in accuracy is observed. This is expected because the memory biases the search towards promising drop sites rather than randomly searching the entire space.

**Effects of not using Time-dependent modulation of neighbourhood function (TMNF):** Recall that the purpose of TMNF is to separate the clusters created on the grid [9]. However, another mechanism that could also increase grid-cluster separation is the progressive increment of the perceptive radius. Since TMNF and increasing perceptive radius both attempt to increase the spatial separation between clusters on the grid, it is unclear whether TMNF really does contribute to this objective.

Row (6) of Table 2 shows that ATTA without using TMNF produce an average F-Measure of 0.89 and an average runtime of about 14.9 seconds on all datasets, which are not significantly different from the baseline (i.e., Row (1)), indicating that TMNF is a redundant component in ATTA.

The above results collectively justify the proposed configurations of SABC presented in Section 3. In the next section, we will provide a head-to-head comparison of SABC and ATTA.

### 4.2. Clustering Results

Despite its simplicity, Table 3 shows that the quality of clustering solutions produced by SABC is comparable to ATTA. It shows that SABC is able to detect elongated cluster (found in the Triangle dataset) as well as relatively overlapping clusters in the Square series. For highly overlapping clusters that are difficult to identify visually (see Square5 and Square7 in Figure 2), SABC reports less number of clusters. ATTA, on the other hand, tends to subdivide clusters, regardless of whether the clusters are overlapped or elongated.

Notice that ATTA detects an average of 10.02 clusters for the 2D10C dataset and this seems to contradict our earlier criticism that it can only detect at most 10 clusters. But recall that ATTA initially scatters all the items randomly on a 2D surface; and these items can be seen as singleton clusters. If ATTA is unable to successfully group these items into 10 clusters at the end of the clustering process, it will report more than 10 clusters. This is exactly what happened in this case.

It is interesting to note that SABC runs well without using TMNF, which is originally used in ATTA to ensure sufficient spatial separation between clusters on a two-dimensional grid. Our results show that a simple increase in perceptive radius to three is all that is required to achieve the same effect. In the same vein, SABC runs well without using the short-term memory.

In terms of runtime, Table 3 shows that SABC is significantly faster than ATTA in most cases. This is expected because SABC uses a maximum perceptive radius of three, instead of five as is the case in ATTA. Not only that the runtime is improved due to the use of smaller perceptive radius, the clustering quality is not compromised.

Notice that the runtime of ATTA increases as the clusters become more and more overlapped from Square1 to Square7. This problem is due to a sentinel loop in which the ants have indefinite search time when seeking for items to pick up; and the indefinite search time varies with respect to changes in the characteristics of data [14]. In contrast, SABC has a more consistent runtime due to its use of smaller perceptive radius.

Table 4 shows that SABC produces comparable and sometimes better results than ATTA, in terms of their clustering quality. Note that we are using the original implementation of ATTA here, so the results of ATTA are similar to that previously published [9]. The only exception is the Zoo dataset, in which the clustering quality is lower than that published by Handl. One conjecture is that the use of \( N_{\text{size}} = 8 \) in ATTA reduces the chance for some of the small clusters in the Zoo data to build up on the grid. This is plausible because when we use the SABC model, which uses \( N_{\text{size}} = 4 \), the results of SABC on the Zoo data become very similar to the original published results. To verify this, we test ATTA using \( N_{\text{size}} = 4 \) and find that its F-Measure on the Zoo dataset improves from 0.71 to 0.82. Similarly, we find that the average F-Measure of ATTA on the Dermatology dataset, improves from 0.82 to 0.89, when the \( N_{\text{size}} \) is changed from 8 to 4.

In terms of the runtime performance, SABC runs a lot faster than ATTA on all real-world datasets.
Table 3: Results of ATTA and SABC on synthetic datasets. Each entry is the mean ± standard deviation. #Cluster is the number of clusters detected and the runtime is in seconds.

|        | ATTA          | SABC          |        | ATTA          | SABC          |
|--------|---------------|---------------|--------|---------------|---------------|
|        | Runtime       | #Cluster      | F-Measure | Runtime       | #Cluster      | F-Measure      |
| Square1| 12.78 ± 0.88  | 4.00 ± 0.00   | 0.98 ± 0.00 | 15.05 ± 5.86  | 4.00 ± 0.40   | 0.98 ± 0.04   |
|        | 9.37 ± 2.06   | 4.00 ± 0.00   | 0.98 ± 0.00 | 10.78 ± 4.73   | 4.00 ± 0.40   | 0.99 ± 0.03   |
| Square3| 13.21 ± 0.90  | 3.98 ± 0.14   | 0.94 ± 0.02 | 14.46 ± 4.15  | 4.00 ± 0.00   | 0.99 ± 0.04   |
|        | 9.40 ± 1.98   | 4.04 ± 0.28   | 0.94 ± 0.02 | 9.96 ± 4.09    | 4.02 ± 0.14   |
| Square5| 14.95 ± 1.07  | 3.96 ± 0.20   | 0.83 ± 0.03 | 12.19 ± 3.06  | 3.02 ± 0.89   | 1.00 ± 0.00   |
|        | 9.88 ± 2.02   | 3.02 ± 0.91   | 0.70 ± 0.12 | 7.26 ± 3.02    | 4.00 ± 0.00   | 1.00 ± 0.00   |
| Square7| 17.29 ± 3.88  | 1.14 ± 0.53   | 0.41 ± 0.04 | 22.81 ± 4.46  | 1.10 ± 0.30   | 1.00 ± 0.00   |
|        | 12.73 ± 4.13  | 1.10 ± 0.30   | 0.41 ± 0.04 | 26.60 ± 7.83   | 9.82 ± 0.63   |
| Sizes1 | 13.62 ± 1.05  | 4.00 ± 0.00   | 0.98 ± 0.00 | 23.18 ± 4.53  | 3.98 ± 0.14   | 1.00 ± 0.00   |
|        | 8.88 ± 1.99   | 3.98 ± 0.14   | 0.98 ± 0.02 | 25.88 ± 7.50   | 10.00 ± 0.00  |
| Sizes3 | 12.75 ± 0.64  | 4.00 ± 0.00   | 0.99 ± 0.00 | 21.89 ± 2.96  | 4.00 ± 0.00   | 1.00 ± 0.00   |
|        | 8.58 ± 1.54   | 4.00 ± 0.00   | 0.99 ± 0.00 | 15.08 ± 3.23   | 10.00 ± 0.00  |
| Sizes5 | 14.64 ± 2.20  | 3.96 ± 0.20   | 0.99 ± 0.02 | 19.77 ± 5.59  | 4.00 ± 0.00   | 1.00 ± 0.00   |
|        | 9.26 ± 2.04   | 4.00 ± 0.00   | 0.99 ± 0.19 | 9.84 ± 2.13    | 3.16 ± 0.37   |
| Triangle| 14.87 ± 1.35  | 4.98 ± 0.86   | 0.93 ± 0.05 | 18.77 ± 0.47  | 4.14 ± 0.40   | 1.00 ± 0.00   |
|        | 11.04 ± 2.61  | 4.14 ± 0.40   | 0.98 ± 0.02 | 14.04 ± 1.19   | 4.00 ± 0.00   |
|        | 11.28 ± 2.11  | 4.00 ± 0.00   | 1.00 ± 0.00 | 1.00 ± 0.00    |

5. Conclusion

Our postulation that current ABC systems are excessively complicated and their essential components are few, has been verified using our simplified system, SABC. This streamlined version is essentially a monolithic stochastic clustering algorithm that differs fundamentally from the traditional multi-agent ABC systems. The working principles behind each component in the streamlined system are easier to understand compared with the ones in traditional ABC systems.

By using all the data in the experiments conducted for ATTA, this study shows that SABC produces comparable and sometimes better results than ATTA on the synthetic problems, both in terms of clustering accuracy and the number of clusters detected. However, a distinction can be drawn when we examine the results on the real-world problems, where we see that SABC produces better results more frequently than ATTA.

Our empirical results also show that SABC runs faster than ATTA. SABC performs better because the architecture of SABC is a lot simpler than its full-fledged counterpart, ATTA. Since ATTA has already been shown to perform competitively against traditional clustering methods, we can expect a similar or better performance from SABC. This finding suggests that many existing swarm-based clustering systems can be simplified and improved by complexity reduction.
Table 4: Results of ATTA and SABC on real-world datasets. Each entry is the mean ± standard deviation. #Cluster is the number of clusters detected and the runtime is in seconds. K is the actual number of classes in each dataset.

| Dataset      | ATTA       | SABC       | Dataset      | ATTA       | SABC       |
|--------------|------------|------------|--------------|------------|------------|
| Wisconsin Iris | 12.03 ± 0.38 | 5.05 ± 0.12 | K            | 6.60 ± 0.17 | 2.29 ± 0.08 |
| Runtime      | 2.00 ± 0.00 | 2.00 ± 0.00 | #Cluster     | 2.98 ± 0.14 | 3.06 ± 0.24 |
| F-Measure    | 0.97 ± 0.00 | 0.97 ± 0.00 | F-Measure    | 0.82 ± 0.01 | 0.81 ± 0.01 |
| Dermatology Digits | 6.75 ± 0.20 | 2.02 ± 0.06 | K            | 46.85 ± 2.33 | 26.31 ± 1.04 |
| Runtime      | 4.08 ± 0.49 | 5.02 ± 0.25 | #Cluster     | 5.18 ± 0.80 | 8.82 ± 1.38 |
| F-Measure    | 0.82 ± 0.04 | 0.89 ± 0.01 | F-Measure    | 0.50 ± 0.05 | 0.57 ± 0.01 |
| Zoo Yeast    | 5.32 ± 0.23 | 1.77 ± 0.04 | K            | 15.55 ± 0.38 | 8.37 ± 0.14 |
| Runtime      | 2.94 ± 0.37 | 3.98 ± 0.38 | #Cluster     | 4.32 ± 1.32 | 6.58 ± 0.61 |
| F-Measure    | 0.71 ± 0.04 | 0.82 ± 0.03 | F-Measure    | 0.43 ± 0.04 | 0.43 ± 0.02 |
| Wine Yeast   | 6.04 ± 0.21 | 2.16 ± 0.06 |              |            |            |
| Runtime      | 3.00 ± 0.00 | 3.00 ± 0.00 |              |            |            |
| F-Measure    | 0.87 ± 0.02 | 0.87 ± 0.01 |              |            |            |

Finally, interested readers may contact James Tan (jamestanse@unisim.edu.sg) to request for the source codes.

Acknowledgments

The first author was partially supported by Monash Postgraduate Publication Award. The authors acknowledge Julia Handl for providing the source codes of her work.

References

[1] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, New York, 1999.
[2] J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, L. Chrétien, 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 I, MIT Press, Cambridge, 1991, pp. 356–363.
[3] E. Lumer, B. Faia, Diversity and adaptation in populations of clustering ants, in: Proceedings of the third International Conference on Simulation of Adaptive Behavior: From Animals to Animals 3, MIT Press, Cambridge, 1994, pp. 501–508.
[4] L. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. 1, University of California Press, Berkeley, 1967, pp. 281–297.
[5] N. Monmarché, M. Slimane, G. Venturini, On improving clustering in numerical databases with artificial ants, in: Proceedings of The Fifth European Conference on Artificial Life, Lecture Notes in Artificial Intelligence 1674, Springer Verlag, 1999, pp. 626–635.
[6] V. Ramos, A. Abraham, Evolving a stigmengic self-organized data-mining, in: Proceedings of the Fourth International Conference on Intelligent Systems, Design and Applications, Budapest, Hungary, 2004, pp. 725–730.
[7] J. Handl, B. Meyer, Improved ant-based clustering and sorting in a document retrieval interface, in: Proceedings of Seventh International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science 2439, Springer Verlag, Berlin, 2002, pp. 913–923.
[8] J. Handl, K. J., M. Dorigo, Ant-based clustering and topographic mapping, Artificial Life 12 (1) (2006) 35–61.
[9] J. Handl, Ant-based methods for tasks of clustering and topographic mapping: extensions, analysis and comparison with alternative methods, Master’s thesis, University of Erlangen-Nuremberg, Germany (2003).
[10] A. Asuncion, D. Newman, Uci machine learning repository, Tech. rep. (2007).
[11] J. Handl, K. J., An evolutionary approach to multiobjective clustering, IEEE Transactions on Evolutionary Computation 11 (1) (2007) 56–76.
[12] C. van Rijsbergen, Information Retrieval, 2nd edition, Butterworths, London, UK, 1979.
[13] L. Gauthier, P. Redoua, P. Harroueta, J. Tisseaua, A first mathematical model of brood sorting by ants: Functional self-organization without swarm-intelligence., Ecological Complexity 4 (4) (2007) 234–241.
[14] S. Tan, K. Ting, S. Teng, Reproducing the results of ant-based clustering without using ants, in: Proceedings of Congress on Evolutionary Computation 2006, IEEE Press, 2006, pp. 1760–1767.