Ant-based clustering and topographic mapping

J. Handl[*] J. Knowles[*] M. Dorigo[†]

Abstract

Ant-based clustering and sorting is a nature-inspired heuristic first introduced as a model for explaining two types of emergent behavior observed in real ant colonies. More recently, it has been applied in a data-mining context to perform both *clustering* and *topographic mapping*. Early work demonstrated some promising characteristics of the heuristic but did not extend to a rigorous investigation of its capabilities. In this paper, we describe an improved version, called ATTA, incorporating adaptive, heterogeneous ants, a time-dependent transporting activity, and a method (for clustering applications) that transforms the spatial embedding produced by the algorithm into an explicit partitioning. ATTA is then subjected to the most rigorous analytical evaluation of an ant-based clustering and sorting algorithm undertaken to date: we compare its performance to standard techniques for clustering and topographic mapping using a set of analytical evaluation functions and a range of synthetic and real data collections. Our results demonstrate the ability of ant-based clustering and sorting to automatically identify the number of clusters inherent to a data collection, and to produce high quality solutions; indeed, we show that it is particularly robust for clusters of differing sizes, and overlapping clusters. The results obtained for topographic mapping are, however, disappointing. We provide evidence that the solutions generated by the ant algorithm are barely topology-preserving, and we explain in detail why results have—in spite of this—been misinterpreted (much more positively) in previous research.

Keywords:- Ant-based heuristic, ant-based clustering and sorting, clustering, topographic mapping, swarm intelligence.

[*]School of Chemistry, The University of Manchester, PO Box 88, Sackville Street, M60 1QD Manchester, UK. j.handl@postgrad.manchester.ac.uk, j.knowles@manchester.ac.uk. Part of this work was done while J. Handl and J. Knowles were at IRIDIA.

[†]IRIDIA, Université Libre de Bruxelles, 50, Av. F. Roosevelt, CP 194/6, B-1050 Brussels, Belgium. mdorigo@ulb.ac.be

1 Introduction

The large volumes of data arising today in fields such as bioinformatics and electronic document retrieval exemplify a trend that is occurring throughout our 'knowledge economy'. Novel technologies (such as the Internet in document retrieval, micro-array experiments in bioinformatics, physical simulations in scientific computing and many more) give rise to enormous warehouses of data, which can only be handled and processed by means of computers. While the ever-increasing storage and speed of computers facilitates this trend, it nonetheless relies upon a continuous development in database technology and information processing techniques. The automatic analysis of data and their lucid presentation to humans is crucial in this context, as it is only when the data are interpreted, that they become meaningful and can provide new knowledge and insight. The research field addressing these major challenges is generally referred to as *data-mining*. In the following, we focus on two subproblems encountered in data-mining, namely *cluster analysis* and *topographic mapping*.

Cluster analysis is concerned with the division of data into homogeneous groups such that data items within one cluster are similar to each other, and those within different clusters are dissimilar. Clustering methods have been studied for many years, but they continue to be the subject of active research. Due to this, many clustering methods are available nowadays, differing not only in the principles of the underlying algorithm (which determine runtime behavior and scalability) but also in other characteristics, such as the types of attributes handled or the shapes of identifiable clusters. To date, the four main classes of clustering algorithms are *partitioning methods*, *hierarchical methods*, *density-based clustering* and *grid-based clustering*. For an extensive survey the reader is referred to [15].

Algorithms performing topographic mapping go one step further than those for mere clustering. They are not limited to the detection of homogeneous groups within the data but aim to capture neighborhood relations within a two-dimensional (or possibly three- or higher-dimensional) visualization of a high-dimensional data-space. This is realized by pinpointing additional information in the form of (1) relationships between individual clusters and (2) relationships between data items belonging to the same cluster. In this sense, a kind of 'sorting' of the data items in two (or more) dimensions is obtained.

In the literature the resulting *topographic maps* are also termed *neighborhood-preserving*, *topological*, *topology-preserving*, *ordering* or *systematic* maps. We can distinguish between two main types of algorithms used for producing topographic maps. On the one hand, there are theoretically well-founded and mathematically exact algorithms, which are usually based on the explicit computation of a mapping from data- to map-space. Unfortunately, these techniques do not necessarily generate the most visually-informative representation and may often be computationally expensive. On the other hand, there are approximation methods that gradually improve an optimization criterion (either implicitly or explicitly). Even though the latter can be prone to suboptimal solutions, they have been shown to perform well in many applications. For a discussion of different algorithms for topographic mapping see [3].

Ant-based clustering and sorting—the focus of our work—is a local, distributed heuristic that has been applied to both of the above tasks: clustering [26] and topographic mapping [23, 18, 21, 20, 13, 14, 27]. Indeed, some work claims that, like self-organizing maps, ant-based clustering and sorting is capable of performing both of these tasks simultaneously [22, 20]. The analyses of the algorithm's performance undertaken to date, however, do not fully support these claims. Experiments on artificial data have been restricted to very few and simple toy problems. The evaluation of the results on these data has mostly been based on visual observation and only very little analytical evaluation has been carried out. On the contrary, those analytical results that are available tend to raise doubts concerning the heuristic's suitability for topographic mapping [13]. The reported experiments on real data are similarly limited and have been subject to even less analytical evaluation. Finally, for both artificial and real test data, there have been no convincing comparisons to alternative methods.

In an effort to close this gap, it is *the main goal of this work* to evaluate thoroughly the performance of ant-based clustering and sorting, and to compare it to a number of standard techniques for both clustering and topographic mapping. We make this evaluation not on the basic algorithm, but on a much-improved version, called ATTA, which we present in detail here. This algorithm remains faithful to the underlying principles of ant-based clustering and sorting but introduces a number of modifications that are crucial to enable its analytical evaluation. In particular, the changes improve the spatial separation of the clusters, and enable the algorithm to be run robustly over a range of data sets.

To obtain a thorough understanding of all aspects of the capabilities of ant-based clustering and sorting, we employ a range of artificial and real test data. These enable us to, respectively, control different properties of the data sets and to verify the overall performance on realistic benchmarks. In each case, we employ a number of selected analytical evaluation functions to give a precise picture of the performance achieved by ATTA and by the algorithms against which we compare it, on these data sets. Some of the results on the clustering performance of our algorithm have been previously reported in [11, 12] in a less complete form.

The remainder of this paper is structured as follows. Section 2 summarizes the basic concepts of ant-based clustering and sorting and gives a survey of previous research in this field. Section 3 describes the improved algorithm used in this paper, giving an account of each modification made, and the advantage it confers. We present a version of the algorithm that returns an explicit partitioning of the data so that clustering results can be unambiguously obtained. In preparation for a thorough analytical evaluation of the algorithm, we also present a scheme to derive parameter settings across arbitrary data sets. We then proceed to describe our evaluation and comparative study. In order to study the algorithm's performance at clustering and topographic mapping, each *in isolation*, the evaluation has been divided into two parts. First, in Section 4, we present the algorithms, evaluation functions and test data used for clustering and discuss the results obtained. Subsequently, in Section 5, we describe the equivalent for topographic mapping. Finally, Section 6 discusses future work and concludes.

2 Ant-based clustering and sorting

Ant-based clustering and sorting was first introduced by Deneubourg et al. [4] to explain different types of naturally-occurring emergent phenomena. It is an instance of the broad category of *ant algorithms* [5], that is, algorithms that model 'some behavior' observed in real ants.[1] In the case of ant-based clustering and sorting, two related types of natural ant behavior are modeled [2]. When *clustering*, ants gather items to form heaps; an example of this being the cemetery formation (i.e., the clustering of dead corpses) observed in the species *Pheidole pallidula*. And when *sorting*, ants discriminate between different kinds of items and spatially arrange them according to their properties; a type of activity that can, for example, be observed in nests of *Leptothorax unifasciatus* where larvae are arranged as a function of their size.

In their paper [4], Deneubourg et al. proposed a continuous model to describe these behaviors. From this, a discrete Monte Carlo model was derived, which was experimentally validated. In the computer simulation ants were represented as simple agents, which randomly moved in their environment, a square grid with periodic boundary conditions. Items that were scattered within this environment could be picked up, transported and dropped by the agents. These operations were biased by the distribution of items within the agents' local neighborhoods, such that items that were either isolated or surrounded by dissimilar ones were more likely to be picked up, and then tended to be dropped again in the vicinity of similar ones. As a result, a clustering and sorting of the items on the grid was obtained.

Deneubourg et al.'s [4] work mainly aimed at deriving a model applicable to collective robotics, but it was soon applied to data analysis. Lumer and Faieta [22] introduced a number of modifications to the model that both enabled the handling of numerical data and improved solution quality and the algorithm's convergence time. It was in this paper, that the ants' sorting process was for the first time termed a "heuristic mapping of a possibly high-dimensional and sparse data set on a plane, in a way which preserves neighborhood relationships as much as possible", that is, as an approximate topographic mapping (as opposed to a pure clustering). However, it is our contention that the results presented in this paper, although understandably leading to initial excitement and activity, do not suffice in demonstrating a true topology-preserving capability. In particular, the analytical measures used (mean local fit and grid entropy) do not capture the preservation of inter-cluster relationships at all and reflect only a limited or 'loose' intra-cluster sorting.

[1] To date, the metaheuristic Ant Colony Optimization, ACO [6, 7, 8], is the best known and most successful algorithm of this type.

Nonetheless, Lumer and Faieta's [22] work was followed by a number of 'real-world' applications of ant-based clustering and sorting to topographic mapping—partly motivated by the favorable scaling properties that result from the algorithm's local and distributed operation. In several papers the algorithm's use for the two-dimensional visualization of document collections in the form of topic maps [14, 27, 13] and its applicability to graph partitioning [19, 21, 20] were investigated. In one of these papers [20], the Pearson correlation of the dissimilarities between pairs of data items in data-space and their respective spatial distances on the grid was measured in order to support the claimed capability of the 'ants' to generate an "isometric embedding". While the results obtained were far from optimal, the deficit from a 'perfect' correlation of 1.0 was due—it was argued—to the gathering of "vertices with a small dissimilarity" without any further discrimination (i.e., merely a result of defects in highly local topology preservation). In Section 5 we will demonstrate that the implied assumption of good global topology preservation is not generally valid and provide an explanation for earlier mis-interpretations of the evidence arising from the use of the Pearson correlation.

The evaluation of ant based clustering and sorting in a *clustering* context has been equally limited. Only in [26] have both suitable measures and a satisfyingly large test suite been used. However, as the algorithm studied by Monmarché is a hybridization of the k-means algorithm and ant-based clustering and sorting (and differs significantly from the latter), the results obtained with this method are not pertinent to an evaluation of ordinary ant-based clustering and sorting.

3 ATTA: An improved ant-based clustering and sorting algorithm

3.1 Questions of interest

Previous research on ant-based clustering and sorting has left several questions related to the algorithm's performance broadly unanswered, many of which fall into one of the following three categories.

- **Clustering performance**
  Seen purely as a clustering algorithm, how does the ant algorithm perform? How do its results compare to those obtained using classical clustering methods from the data-mining literature? In particular, is the algorithm competitive in terms of its runtime and of the quality of its solutions?

- **Sorting performance**
  To what degree is the spatial embedding generated by the ant algorithm topology-preserving? Are neighborhood relations preserved on a local or a global scale, or both? How do its results compare to those obtained using classical methods for topographic mapping from the data-mining literature? In particular, is the algorithm competitive in terms of its runtime and of the quality of its solutions?

- **Sensitivity to data properties**
  How robust is the ant algorithm's clustering and sorting performance with respect to different data properties? In particular, how strongly is it affected by the use of high dimensional and/or large data sets, by increasing overlap between clusters, or distinct deviations in the sizes of individual clusters?

The above questions reflect the main issues investigated in our work. In order to undertake a suitably objective and general analysis of them, a standard implementation of ant-based clustering and sorting cannot be employed, however. This is because the latter does not: (i) return an explicit partitioning of data, allowing clustering solutions to be evaluated without human intervention; or (ii) work robustly with the same parameter settings across different data sets. With regard to (i), the basic algorithm must be improved such that the embedding generated contains clearly separated, compact clusters, which can be unambiguously converted into an explicit clustering solution (or partition) by an *automatic process*, thereby removing the bias associated with human interpretation of the embedding. With respect to (ii), we seek an algorithm that is sufficiently robust that we can define, *a priori*, how we shall derive the parameters for a given data set, without any need of 'exploratory' runs.

In this section, starting off with a generic ant algorithm based on earlier work [4, 22, 13], we introduce a number of modifications that, together, overcome the two problems identified above. Experimental results demonstrating the impact of individual modifications are not presented in this paper but can be found in [10].

The algorithm that we finally arrive at is operationally different in a number of key areas from previous ant-based clustering and sorting approaches. To differentiate our approach from these, we call our algorithm, ATTA[2] for Adaptive Time-dependent Transporter Ants.

3.2 Basics

A basic, generic ant algorithm derived from the algorithms presented in [4, 22, 13] is described in Algorithm 1. It begins with an initialization phase, in which *#items* data items are randomly scattered on the toroidal grid (Line 2); each of *#agents* agents randomly picks up one data item (Lines 4–5); and each agent is placed at a random position on the grid (Lines 7–7). Subsequently, the clustering and sorting phase starts: this is a simple loop, in which one agent is randomly selected (Line 10); it performs a step of a given *stepsize* on the grid (Line 11); and it (probabilistically) decides whether to drop its data item or not (Lines 12–14). In the case of a 'drop'-decision, the agent drops the data item at its current grid position, or, if this grid cell is occupied by another data item, in the immediate neighborhood of it (the agent locates a nearby free grid cell by means of a random search) (Line 15-16). It then immediately searches for a new data item to pick up. This is done using an index that stores the positions of

[2] Also the name of a genus of leaf-cutter ant, which transports leaves.

all 'free' data items on the grid: the agent randomly selects one data item i out of the index, jumps to i's position on the grid and evaluates the neighborhood function $f(i)$. Based on the value of $f(i)$, which provides information on the similarity and density of the data items in i's local neighborhood, the agent then (probabilistically) decides whether to pick up i or not. It continues this search until a successful picking operation occurs (Lines 17–21). Only then is the loop repeated with another agent. After a pre-defined number of iterations the algorithm terminates and the output is a spatial *embedding* of the data on the toroidal grid.

The picking and dropping operations in the algorithm are probabilistic decisions. Their respective probabilities (given a grid position and a particular data item i), may be computed using Deneubourg et al.'s [4] density functions:

$$p_{pick}(i) = \left(\frac{k^+}{k^+ + f(i)}\right)^2, \tag{1}$$

and

$$p_{drop}(i) = \left(\frac{f^*(i)}{k^- + f(i)}\right)^2, \tag{2}$$

where commonly $k^+ = 0.1$ and $k^- = 0.3$, and $f(i)$ is Lumer and Faieta's [22] neighborhood function:

$$f(i) = \max\left(0.0, \frac{1}{\sigma^2}\sum_{j \in L}\left(1 - \frac{\delta(i,j)}{\alpha}\right)\right). \tag{3}$$

Here, $\delta(i,j) \in [0,1]$ is a dissimilarity function defined between points in data-space, $\alpha \in [0,1]$ is a data-dependent scaling parameter, and σ^2 is the size of the local neighborhood L (typically, $\sigma^2 \in \{9, 25\}$). The agent is located in the center of this neighborhood and its *radius of perception* in each direction is defined as $\frac{\sigma-1}{2}$.

3.3 Modified neighborhood function

As a first modification to this generic ant algorithm we introduce a small supplement to the neighborhood function. We redefine $f(i)$ as

$$f^*(i) = \begin{cases} \max\left(0.0, \frac{1}{\sigma^2}\sum_{j \in L}\left(1 - \frac{\delta(i,j)}{\alpha}\right)\right) & \text{if } \forall j \left(1 - \frac{\delta(i,j)}{\alpha}\right) > 0 \\ 0.0 & \text{otherwise,} \end{cases} \tag{4}$$

where the constraint $\forall j\ (1 - \frac{\delta(i,j)}{\alpha}) > 0$ serves the purpose of heavily penalizing high dissimilarities, which improves spatial separation between clusters.

3.4 Short-term memory with 'look-ahead'

The clustering process on the grid can be accelerated significantly by the use of a 'short-term memory' as introduced by Lumer and Faieta [22]. In Lumer and Faieta's approach, each agent remembers the last few carried data items and their respective dropping positions. When a new data item is picked up, the position of the 'best matching' memorized data item is used to bias the direction of the agent's random walk. Here, the 'best matching' item is the one of minimal dissimilarity $\delta(i,j)$ to the data item i currently being carried. We have extended this idea as follows.

In a multi-agent system the items stored in the memory might already have been removed from the remembered position. In order to determine more robustly the direction of the bias, we permit each agent to 'look ahead'. An agent carrying a data item i uses its memory to examine all remembered positions, one after the other. Each of them is evaluated using the neighborhood function $f^*(i)$, that is, the suitability of each of them as a dropping site for the currently carried data item i is examined.

Out of all positions evaluated, the one of 'best match' is the grid cell for which the neighborhood function yields the highest value. For the subsequent step of the agent on the grid, we replace the use of a biased random walk with an agent 'jump' directly to the position of 'best match'. However, this jump is only made with some probability, dependent on the quality of the match; the same probability threshold that we use for a dropping operation is used for this purpose (Equation 2 or 6, as appropriate). If the jump is not made, the agent's memory is de-activated, and in future iterations it reverts to trying random dropping positions until it drops the item successfully.

3.5 Increasing radius of perception

The size of the local neighborhood perceived by the agents limits the information used during the sorting process. It is therefore attractive to employ larger neighborhoods in order to improve the quality of the clustering and sorting on the grid. However, the use of a larger neighborhood is not only more expensive (as the number of cells to be considered for each action grows quadratically with the radius of perception), but it also inhibits the quick formation of clusters during the initial sorting phase.

For this reason we use a radius of perception that gradually increases over time. This saves computations in the first stage of the clustering process and prevents difficulties with the initial cluster formation. At the same time it accelerates the dissolution of preliminary small clusters.

In the current implementation, we start with an initial perceptive radius of 1 and linearly increase it to be 5 in the end. While doing so, we leave the scaling parameter $\frac{1}{\sigma^2}$ in Equation 4 unchanged, as its increase results in a loss of spatial separation.

3.6 Time-dependent modulation of the neighborhood function

As stated above, the spatial separation of clusters on the grid is crucial in order for individual clusters to be well-defined, enabling unambiguous (automatic) cluster retrieval. Spatial closeness, when it does occur, is, to a large degree, an artifact of early cluster formation. This is because, early on in a run, clusters will tend to form wherever there are locally dense regions of similar data items; and thereafter these clusters tend to drift only very slowly on the grid. After an initial clustering phase, we therefore use a short interlude (from time t_{start} to t_{end}) with a modified neighborhood function, which replaces the scaling parameter $\frac{1}{\sigma^2}$ by $\frac{1}{N_{occ}}$ in Equation 4, where N_{occ} is the number of occupied grid cells observed within the local neighborhood. Hence only similarity, not density, is taken into account, which has the effect of spreading data items out on the grid again, but in a sorted fashion; the data items belonging to different clusters will now occupy their own 'regions' on the grid. Subsequently, we turn back to using the traditional neighborhood function. Once again, clear clusters are formed, but they now have a high likelihood of being generated along the centers of these 'regions', due to the lower neighborhood quality at their boundaries.

In order to give a better idea of the functioning of this strategy, Figure 1 provides three snapshots of the spatial distribution on the grid, at different stages in the clustering process.

3.7 Modified threshold functions

We also introduce a set of threshold functions that are different from those proposed by Deneubourg et al. [4] or Lumer and Faieta [22]. The probability for a picking operation is now given by

$$p^*_{pick}(i) = \begin{cases} 1.0 & \text{if} f^*(i) \leq 1.0 \\ \frac{1}{f^*(i)^2} & \text{otherwise,} \end{cases} \tag{5}$$

and for a dropping operation

$$p^*_{drop}(i) = \begin{cases} 1.0 & \text{if} f^*(i) \geq 1.0 \\ f^*(i)^4 & \text{otherwise.} \end{cases} \tag{6}$$

Equations 5 and 6 have been experimentally derived, but, in order to further clarify their characteristics, the graphs in Figure 2 compare them to those used by Deneubourg et al. [4] (with $k^+ = 0.1$ and $k^- = 0.3$).

The new threshold functions make picking operations deterministic for the entire range $f^*(i) \in [0.0, 1.0]$. Only for neighborhood values $f^*(i) \geq 1.0$ do picking operations become less likely. Analogously, dropping of data items is now completely deterministic for neighborhood values $f^*(i) \geq 1.0$. However, the dropping probabilities in the 'low-similarity' region of the density function are lower, making this strategy more conservative than the original one (Equation 2).

Note that the above given Equations 5 and 6 are quite different from the ones suggested in previous work and are *not appropriate for the basic ant algorithm.* They have been experimentally derived for use with our enhanced version (which includes the short-term memory with lookahead, the increasing radius of perception, and the interlude with the modified neighborhood function discussed above), for which they significantly speed up the clustering process. In particular, they have to be seen in the context of the shift in the range of attainable values $f^*(i)$ resulting from our increase of the radius of perception (see Section 3.5). In the starting phase of the algorithm, $f^*(i)$ is limited to the interval $[0, 1]$; the upper bound, however, increases with each increment of the neighborhood radius, such that, in our implementation, $f^*(i)$ can yield values within the interval $[0, 15]$ after the last increment. Consequently, the picking operation is purely deterministic in the beginning, and, at this stage, it is solely the dropping operation that favors dense and similar neighborhoods. Gradually, with the rise of $f^*(i)$, an additional bias towards the picking of misplaced data items is introduced. The shift of the values of $f^*(i)$ combined with the use of the threshold functions $p^*_{pick}(i)$ and $p^*_{drop}(i)$ (Equations 5 and 6) have the effect of decreasing the impact of density for the dropping threshold while, simultaneously, increasing it for the picking threshold. This results in an improved spatial separation between clusters.

3.8 Parameter settings

The modifications described above improve the reliability with which clearly-separated, compact clusters are obtained on the grid, which is the prerequisite for automatic cluster retrieval (see Section 3.9). Our other goal of setting parameters without the need for exploratory runs is addressed in this section.

Several of the parameters of our algorithm can be set independently of the data. These include the number of agents, which we set to be 10, the size of the agents' short-term memory, which we equally set to 10, and the starting and end time of the interlude with the modified neighborhood function t_{start} and t_{end}, which we set to $0.45 \cdot \#iterations$ and $0.55 \cdot \#iterations$, respectively.

3.8.1 Parameters to be set as a function of the size of the data set

A number of other parameters should, however, be selected as a function of the size of the data set tackled, as they otherwise impair convergence speed. Given a set of size $\#items$, the grid (comprising a total of $\#cells$ grid cells) should offer a sufficient amount of 'free' space to permit the quick dropping of data items (note that each grid cell can only be occupied by one data item). This can be achieved by keeping the ratio $r_{occupied} = \frac{\#items}{\#cells}$ constant. A good value, found experimentally, is $r_{occupied} = \frac{1}{10}$. We obtain this by using a square grid with a resolution of $\lceil\sqrt{10 \cdot \#items}\rceil \times \lceil\sqrt{10 \cdot \#items}\rceil$ grid cells. The step size of the agents (which they use only if their memory is de-activated) is set to $stepsize = \lceil\sqrt{20 \cdot \#items}\rceil$. Note that the use of such a large step size speeds up the clustering process.

The total number of iterations has to grow with the size of the data set. Linear growth proves to be sufficient, as this keeps the average number of times each grid cell is visited constant. Here, $\#iterations = 2000 \cdot \#items$, with a minimal number of 1 million iterations imposed.

3.8.2 Activity-based α-adaptation

An issue addressed previously in [13] is the automatic determination of the parameter α (recall that α is the parameter scaling the dissimilarities in the neighborhood function $f^*(i)$), upon which the functioning of the algorithm crucially depends. During the sorting process, α determines the percentage of data items on the grid that are classified as similar, such that: a too small choice of α prevents the formation of clusters on the grid; on the other hand, a too large choice of α results in the fusion of clusters, which should be separated.

Setting this parameters is not straightforward, as a suitable choice for α is highly dependent on the structure of the data set. However, an unsuitable value is reflected by an excessive or extremely low sorting activity on the grid. Therefore, an automatic adaptation of α may be obtained through the tracking of the amount of activity, which is reflected by the frequency of the agents' successful picking and dropping operations.

To achieve this, we use a heterogenous population of agents in which each agent makes use of its own independent parameter α. All agents start with an α parameter randomly selected from the interval [0, 1]. An agent considers an adaptation of its own parameter after it has performed *#active* moves. During this time, it keeps track of the number of failed dropping operations *#fail*. The rate of failure is determined as $r_{\#fail} = \frac{\#fail}{\#active}$ where *#active* is fixed to 100. The agent's individual parameter α is then updated using the experimentally derived rule

$$\alpha \leftarrow \begin{cases} \alpha + 0.01 & \text{if } r_{fail} > 0.99 \\ \alpha - 0.01 & \text{if } r_{fail} \leq 0.99. \end{cases}$$

The value of the parameter α is kept adaptive during the entire sorting process. This makes the approach more robust than an adaptation method with a fixed stopping criterion. Also, it allows for the specific adaptation of α within different phases of the sorting process.

The algorithm defined by the preceding sections—starting with the basic algorithm given in 3.2, and including the modifications described in 3.3–3.7, as well as the method for deriving parameter settings above—constitutes our improved ant-based clustering and sorting algorithm, ATTA.

3.9 ATTA-C: cluster-retrieval

To enable ATTA's use for pure clustering tasks it is necessary that the two-dimensional embedding generated by the algorithm be converted into a *clustering*: that is, an explicit partitioning of the data set. To achieve this, a version of the algorithm, ATTA-C (ATTA for clustering), which includes an explicit cluster-retrieval step, was devised.

Methods for retrieving the 'visually obvious' clusters from the grid have been used in some earlier work. In [25] methods from pattern recognition were employed to identify the size of the clusters on the grid and to examine their density. In [21] the k-means algorithm was used towards this goal. This latter approach, however, has the disadvantage that the number of clusters must be specified, which either requires user interaction (if the visually apparent number of clusters is provided by a human) or distorts the final results (if it is assumed that the correct number of clusters is identified in all cases).

To avoid these difficulties, we elect to apply instead a clustering algorithm that does not require k, namely agglomerative hierarchical clustering. This algorithm starts by assigning each data item on the grid to an individual cluster, and proceeds by merging the two least distant clusters (in terms of their spatial distance on the grid) in each iteration, until a stopping criterion is met.

Given compact clusters and a distinct spatial separation between clusters, an agglomerative algorithm based on the single link criterion will clearly work very well and a stopping criterion for the clustering algorithm can easily be derived. However, as data items around the cluster borders are sometimes slightly isolated (so that they are prone to mislead the single link metric by establishing individual clusters or 'bridging' gaps between clusters), we have introduced an additional weighting term $weight(C_1, C_2)$, which encourages the merging of these elements with the 'core' clusters.

Given two clusters C_1 and C_2 on the grid and, without loss of generality, $|C_1| \leq |C_2|$, we define their spatial distance in grid-space as

$$weighted\_singlelink(C_1, C_2) = singlelink(C_1, C_2) \cdot weight(C_1, C_2).$$

Here, $singlelink(C_1, C_2)$ is the standard linkage metric of single link, that is, the minimal distance between all possible pairs of data elements i and j with $i \in C_1$ and $j \in C_2$. In our case, the distance between two data elements is given by the Euclidean distance between their *grid positions*. The term $weight(C_1, C_2)$ is a scaling factor taking the relative sizes of the clusters into account:

$$weight(C_1, C_2) = 1.0 + \log_{10}(1.0 + 9.0 \cdot \frac{|C_1|}{|C_2|}).$$

Clearly, $weight(C_1, C_2)$ is restricted to the range $(1, 2]$.

The radius of perception used within the last phase of the ant-based algorithm reflects the minimum distance clusters should have on the grid, and therefore provides a suitable stopping criterion for the agglomerative clustering algorithm. The merging of clusters stops once all the distances between the clusters have reached a value larger than this radius.

3.10 ATTA-TM: topographic mapping

To differentiate between the different algorithms for clustering and topographic mapping, we use the denomination ATTA-TM for the topographic mapping version of ATTA, which differs only in that it does not include the cluster-retrieval step described above.

4 Analyzing ATTA-C's clustering performance

Having explicitly defined an improved algorithm for ant-based clustering and sorting, ATTA, in the previous section, we are now in a position to proceed with our performance analysis. In this section, the *clustering* performance of ATTA-C is evaluated and compared to three alternative clustering methods and to one statistical method for the automatic determination of the number of clusters in the data. A range of synthetic and real data sets, and three distinct analytical evaluation functions, are used for this purpose. Both the experimental setup and the clustering results are presented in this section.

4.1 Algorithms for clustering

As methods for comparison with ATTA-C we select three of the most popular clustering algorithms from the literature, as well as a statistical method for determining the number of clusters inherent to a data set. These are, respectively: the partitioning method k-means [24], a hierarchical agglomerative approach based on the linkage criterion of average link [31], one-dimensional self-organizing maps (1D-SOM, [16]), and the Gap statistic [29].

While k-means has a favorable (linear) runtime behavior, average link agglomerative clustering is known for its high quality solutions, such that a comparison against both of them provides a good basis for judgments on the performance of the ant algorithm. The choice of 1D-SOM was predominantly motivated by the fact that SOMs simultaneously perform the tasks of clustering and topographic mapping—exactly the claim made for ant-based clustering and sorting in [22, 20].

Both k-means and 1D-SOM require the a priori determination of the desired number of clusters k; for average link agglomerative clustering an appropriate stopping criterion must be derived. In order to avoid the introduction of an additional source of variation, we have decided to directly provide the correct number of clusters k to each of the three algorithms, thus giving the same advantage to each of them. Hence, in our study, ATTA-C is the only clustering algorithm that automatically derives k from the data. However, in order to evaluate how well this task is performed by ATTA-C, we also compare it to the Gap statistic, a method not for clustering (as a whole) but just for determining the number of inherent clusters in a data set. The implementation of these four 'competitor' methods is briefly described below.

4.1.1 k-means

Starting from a random partitioning, the k-means algorithm repeatedly (i) computes the current cluster centers (i.e., the average vector of each cluster in data-space) and (ii) reassigns each data item to the cluster whose center is closest to it. It terminates when no more reassignments take place. By this means, the intra-cluster variance, that is, the sum of squares of the differences between data items and their associated cluster centers, is locally minimized.

Our implementation of the k-means algorithm is based on the batch version of k-means, that is, cluster centers are only recomputed after the reassignment of all data items. As k-means can sometimes generate empty clusters, these are identified in each iteration and are randomly reinitialized. This enforcement of the correct number of clusters can prevent convergence, and we therefore set the maximum number of iterations to 1000. To reduce suboptimal solutions k-means is run repeatedly (20 times) using random initialization, and only the best result in terms of intra-cluster variance is returned.

4.1.2 Average link agglomerative clustering

As a second method, an agglomerative hierarchical clustering algorithm based on the *average link* linkage metric is used. The algorithm starts with the finest partitioning possible (i.e., singletons) and, in each iteration, merges the two least distant clusters. The distance between two clusters C_i and C_j is computed as the average dissimilarity between all possible pairs of data elements i and j with $i \in C_i$ and $j \in C_j$.

Our implementation of average link agglomerative clustering uses a distance matrix of the pairwise distances between clusters that is incrementally updated. Initially, this is simply the data set's complete dissimilarity matrix. After each merging step this matrix is updated using Ward's formula: the fusion of the two clusters C_i and C_j to form cluster C_k requires the computation of the distances between C_k and each remaining cluster C_l; these distances are given as

$$\delta(C_k, C_l) = \frac{|C_i|}{|C_i| + |C_j|} \cdot \delta(C_i, C_l) + \frac{|C_j|}{|C_i| + |C_j|} \cdot \delta(C_j, C_l).$$

The algorithm terminates when the correct number of clusters has been obtained.

4.1.3 One-dimensional self-organizing maps

SOMs are two-layered unsupervised neural networks that adapt a set of weight vectors to approximately model the input data. Each of these weight vectors is associated with one of the neurons in the neural layer that are arranged in the form of a grid (which is one-dimensional, if applied to clustering), and are randomly initialized.

During training, individual input vectors $\vec{x}$ are successively presented to the network, and, for each of these stimuli, the best matching unit (BMU) $\vec{y}$ (i.e., the most similar weight vector) and its local neighbors are determined and updated according to the learning rule

$$\vec{w} \leftarrow \vec{w} + \epsilon \cdot h(\vec{y}, \sigma)(\vec{x} - \vec{w}).$$

Here, ϵ is the learning rate and $h(\vec{y}, \sigma)$ is a *neighborhood function* (centered around the BMU $\vec{y}$) whose spread is determined by the parameter σ.

Our implementation of 1D-SOM is based on the guidelines given in the description of the SOM Toolbox [30]. The correct number of clusters k is used to set the grid resolution to $1 \times k$ rectangular grid cells; the weight vectors are uniformly randomly initialized. The SOM is trained in two training phases, a first 'coarse' approximation phase and a second fine-tuning phase. The first phase starts with a neighborhood size of $\sigma_1^{start} = \max(1.0, \frac{1}{4}k)$, which is exponentially decreased to $\sigma_1^{end} = \max(1.0, \frac{1}{4}\sigma_1^{start})$. The learning rate during this phase is $\epsilon_1 = 0.5$. The second phase starts with the final neighborhood size of phase 1, that is, $\sigma_2^{start} = \sigma_1^{end}$, and continues to decrease it to $\sigma_2^{end} = 1.0$. The learning rate in phase 2 is $\epsilon_2 = 0.05$. The number of iterations for each phase are $it_1 = 10$ and $it_2 = 40$ respectively, and, in each iteration, all data items are presented to the SOM in random order. Finally, in the classification step all data items are assigned to the best matching output neuron. Each output neuron is interpreted as one cluster.

4.1.4 The Gap statistic

For the evaluation of ATTA-C's performance at identifying the correct number of clusters in the data, we additionally compare against the results returned by the Gap statistic, a recently proposed automated method for the determination of the number of clusters in a data set [29].

The Gap statistic is based on the expectation that the most suitable number of clusters can be identified by a significant 'knee' in a plot of the observed clustering quality obtained by an algorithm[3] as a function of the number of specified clusters, k. Thus, to compute the statistic, the clustering problem is first solved for a range of different values of k and, for each k, the resulting partitioning $C = \{C_1, \ldots, C_k\}$ is evaluated by means of some chosen measure of clustering solution quality. Most clustering quality measures are, unfortunately, not independent of the number of clusters; a better evaluation may result only because k is larger/smaller. To overcome this, the Gap statistic uses a number of reference curves to normalize the quality curve. The purpose of the reference curves is to estimate, for a range of values of k, the contribution to observed quality that arises only from the k-dependent bias in the measure, and not from a particular choice of k reflecting the true number of clusters in the data.

To implement the method, B reference curves $R_b(k)$ (with $b \in \{1, ...B\}$) are computed, which are the performance curves obtained when the same clustering algorithm is applied, again using a range of values of k, to B different, uniformly random data distributions (i.e., having *no* clear cluster structure) of the same size and dimension as the real data set. Using these, the normalized performance curve ('Gap curve') for a quality curve $Q(k)$ is given as

$$Gap(k) = \frac{1}{B}\left(\sum_{b=1}^{B} \log(R_b(k))\right) - \log(Q(k)).$$

The most suitable number of clusters is then determined as the smallest k, such that

[3]One in which k is given as a parameter.

$$Gap(k) \geq Gap(k+1) - s_{k+1}, \qquad \text{where} \qquad s_k = sd_k\sqrt{1+\frac{1}{B}},$$

and sd_k is the standard deviation within the set of the B log-scaled reference points $R_b(k)$.

For our implementation of the Gap statistic we use the above described k-means algorithm and use the intra-cluster variance (see Section 4.2.3) as the quality measure. We compute the quality curves for $k \in \{1, ..., 20\}$, and, for each k, we generate $B = 20$ reference curves.

4.2 Evaluation functions

The clustering results of the different algorithms are compared using three analytical evaluation measures. The first of these makes use of the correct classification, which is known for all of our test sets. The other two estimate intrinsic properties of a clustering solution.

4.2.1 The F-Measure

The *F-Measure* [28] uses the ideas of precision and recall from information retrieval. Each *class* i (as given by the class labels of the particular benchmark data set) is regarded as the set of n_i items desired for a query; each *cluster* j (generated by the algorithm) is regarded as the set of n_j items retrieved for a query; n_{ij} gives the number of elements of class i within cluster j. For each class i and cluster j precision and recall are then defined as $p(i,j) = \frac{n_{ij}}{n_j}$ and $r(i,j) = \frac{n_{ij}}{n_i}$, and the corresponding value under the F-Measure is

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

where we chose $b = 1$, to obtain equal weighting for $p(i,j)$ and $r(i,j)$. The overall F-value for the partitioning is computed as

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

where n is the total size of the data set. F is limited to the interval $[0,1]$ and should be maximized.

4.2.2 The Dunn Index

The *Dunn Index* [9] determines the minimal ratio between inter-cluster distance and cluster diameter for a given partitioning. Thus, it captures the notion that, in a good clustering solution, data elements within one cluster should be much closer than those within different clusters. It is defined as

$$D = \min_{c,d \in C} \left[\frac{\delta(\mu_c, \mu_d)}{\max_{e \in C}\left[diam(e)\right]} \right],$$

where the diameter $diam(c)$ of a cluster c is computed as the maximum intra-cluster distance, and $\delta(\mu_c, \mu_d)$ is the distance between the centroids of clusters c and d. It is to be maximized.

4.2.3 The Intra-cluster variance

The *Intra-cluster variance* computes the sum of squared deviations between all data items and their associated cluster center, which reflects the common agreement that data elements within individual clusters must be similar. It is given by

$$I = \sum_{c \in C} \sum_{i \in c} \delta(i, \mu_c)^2,$$

where C is the set of all clusters, μ_c is the centroid of cluster c and $\delta(i, \mu_c)$ is the distance function employed to compute the deviation between each data item i and its associated cluster center. It is to be minimized. However, as all other measures used are to be maximized, we will, for ease of interpretation, use the transformed value $I^* = 10.0 - I$ in our boxplots.

4.3 Experimental data

Two different types of benchmark data sets are used. First, seven real data sets from the Machine Learning Repository [1], which we summarize in Table 1. Second, a range of two-dimensional synthetic data sets that permit the modulation of specific data properties.

In the latter, each cluster is described by a two-dimensional normal distribution $N(\vec{\mu}, \vec{\sigma})$. The number of clusters, the sizes of the individual clusters, and the mean vector $\vec{\mu}$ and vector of the standard deviation $\vec{\sigma}$ for each normal distribution are manually fixed. In each run of the experiments, a new set of data is sampled from these distributions. These synthetic benchmarks are variations of the *Square* data set, a data set that has been frequently employed in the literature on ant-based clustering and sorting. It is two-dimensional and consists of four clusters of equal size (250 data items each), which are generated by normal distributions with a standard deviation of 2 in both dimensions and are arranged in a square. The data sets *Square1* to *Square7* only differ by the distance between the individual clusters (i.e., the length of the edges of the square), which are 10, 9, 8, 7, 6, 5 and 4 respectively. They were generated in order to study the relative sensitivity of the algorithms to increasing overlap between clusters. In the *Sizes1* to *Sizes5* data sets, edge length and standard deviation are kept constant, and, instead, they differ in the sizes of the individual clusters. In particular, the ratio between the smallest and the largest cluster increases from the *Sizes1* (where it is 2) to the *Sizes5* data (where it is approximately 10). This is used to investigate the algorithms' sensitivity to unequally-sized clusters.

4.3.1 Data pre-processing

All types of data are subject to a number of preprocessing steps: missing values are replaced by zeros; the data vectors are normalized in each dimension; dissimilarities between data vectors are computed using Euclidean distance (for the synthetic data) and Cosine similarity respectively (for the real data), and are normalized to lie within the interval $[0, 1]$.

4.4 Results

We begin by presenting results for the synthetic data sets, focusing on the sensitivity of the algorithms to two important data properties, degree of overlap of clusters, and differences in cluster sizes. Results on the real data sets follow on.

4.4.1 Sensitivity to overlapping clusters

We study the sensitivity to overlapping clusters using the *Square1* to *Square7* data sets. It is clear that the performance of all four algorithms necessarily has to decrease with a shrinking distance between the clusters, as points within the region of overlap cannot be correctly classified. It is however interesting to see whether the performance of the individual algorithms degrades gracefully or more catastrophically, as a graceful degradation would indicate that the main cluster structures are still correctly identified.

Figure 3a shows a plot of the algorithms' performance (as reflected by the F-measure) versus the distance between neighboring clusters. A number of trends can be observed in this graph. There is the very strong performance of k-means, which performs best on the first four data sets. The 1D-SOM starts on a lower quality level, but its relative drop in performance is less than that of k-means: it clearly profits from its topology preserving behavior, which makes it less susceptible to noise. Average link agglomerative clustering, in contrast, has trouble in identifying the principal clusters and performs quite badly, especially on the data sets with a smaller inter-cluster distance.

The results of ATTA-C are very close to those for k-means on the simplest data set, *Square1*, but its performance drops slightly more quickly. Still, it performs significantly[4] better than average link on the first five test sets. Also, in spite of the fact that the clusters 'touch', the ant-algorithm reliably identifies the correct number of clusters on the first five test sets (Figure 3b), and it can thus be concluded that the algorithm does not rely on the spatial separation between clusters, but that distinct changes in the density distribution are sufficient for it to detect the clusters. Notice also from Figure 3b that ATTA-C is significantly better than the state-of-the-art, Gap statistic on these data.

For the *Square6* and *Square7* test data, the performance of ATTA-C drops significantly, as it fails to reliably detect the four clusters. For the *Square6* test set the number of identified clusters varies between 1 and 4, for *Square7* only

[4]Based on tabulated mean and standard deviation values given in [10].

1 cluster is identified. However, a plot of the theoretical density distribution along the 'edge' between two neighboring clusters in this data set puts this failure into perspective: Figure 4 makes clear, that, due to the closeness of the clusters, the density gradient is very weak for the *Square6* data, and the distribution of data items is nearly uniform for the *Square7* data. Notice that the other three algorithms are not affected by this lack of gradient because they have been explicitly provided with the correct value of k.

4.4.2 Sensitivity to differing cluster sizes

The sensitivity to unequally-sized clusters is studied using the *Sizes1* to *Sizes5* data sets. Again, we show the algorithms' performance on these data sets as reflected by the F-measure (Figure 5a).

ATTA-C performs very well on all five test sets, in fact it is hardly affected at all by the increasing deviations between cluster sizes. Out of its three rivals, only average link agglomerative clustering performs similarly robustly. 1D-SOM is very strongly affected by the increase of the ratio between cluster sizes, and the performance of k-means also suffers. Notice again that the performance of the Gap statistic is very weak when compared to ATTA-C (see Figure 5b).

4.4.3 Summary of the performance on real data

To gain an impression of the average performance of the algorithms on more general data, we analyze their performance under the different measures on the real data sets. Boxplots [32] for the results obtained in 50 independent runs are shown in Figure 6. Overall, the picture obtained on these data sets is far less uniform. No algorithm consistently dominates the others; in particular, a clear discrepancy can be observed between the ratings returned by the individual measures.

- In contrast to its behavior on the synthetic benchmarks, ATTA-C frequently fails to identify the correct number of clusters on these data sets. In particular, the number of clusters determined on the *Dermatology*, the *Digits*, the *Yeast* and the *Zoo* data sets is distinctively too low, which seriously affects the values obtained under the F-Measure and the Intra-Cluster Variance. In spite of this, the values obtained are the worst only on one of these test sets (the *Digits* data) and ATTA-C even shows a very good performance under the Dunn Index. These results indicate that the clusters identified by ATTA-C must correspond to actual structure within the data. Seemingly, the data sets contain cluster structures on various levels. While several classes in the data are not well-separated (neither spatially nor by a clear density gradient), more distinct cluster structures can be observed on a coarser level (i.e., there seems to be a clear spatial separation between certain groups of clusters). ATTA-C only manages to identify these upper-level structures and fails to further distinguish between groups of data within them. Our impression of a lack of separation between some classes of this data is confirmed by both the poor performance of all other clustering methods and the number of clusters predicted by the Gap statistic, which is frequently too low (results not shown here, see [10] for the full result tables).

- The results obtained on the *Zoo* data set reveal a hitherto unnoticed weakness of ant-based clustering and sorting. Due to the current definition of the neighborhood function, the algorithm requires clusters to have a minimum size in order to construct stable clusters on the grid. The *Zoo* data contains several extremely small clusters (the smallest is of size 5) that ATTA-C fails to identify and consequently assigns to larger clusters. While this is a clear limitation of the algorithm, it could possibly be overcome through the use of a modified neighborhood function.

- The runtimes of the four algorithms, provided in [10], show that, while ATTA-C is relatively slow on small and low-dimensional data sets, it has a favorable scaling behavior. Thus, with an increase in the dimensionality and the size of the data set, it quickly starts to outperform its rivals. For an increase in the number of data elements, a quadratic rise in runtime can be observed for average link agglomerative clustering, whereas k-means, 1D-SOM and ATTA-C all scale linearly. On the other hand, only ATTA-C and average link agglomerative clustering (which can work with a precomputed dissimilarity matrix) are not affected by a rise in dimensionality.

4.4.4 Discussion

In conclusion, we can say that there are a number of properties that make ant-based clustering and sorting an interesting candidate as an algorithm for cluster analysis.

It is one of the few clustering algorithm that has the intrinsic capability to identify the number of clusters k in the data. Most other clustering methods (like the considered k-means, average link agglomerative clustering and 1D-SOM) rely on the specification of k as an input parameter. This requires a priori knowledge or the interaction with another algorithm, which can be problematic, as current methods for the automatic determination of the number of clusters in a data set are rather limited (indeed we have seen here that the Gap statistic is not reliable). On the other hand, the impossibility to specify k can also be considered a disadvantage of ant-based clustering and sorting. In specific applications the user might have precise ideas about the number of clusters to be identified, and in ant-based clustering and sorting, there is currently no possibility to adjust the number of clusters that are to be generated.

As far as the quality of the generated partitionings is concerned, ATTA-C's robustness with respect to different data properties is quite impressive. In our comparison, it is the only algorithm to perform consistently well on all synthetic data sets, whereas k-means, 1D-SOM and average link agglomerative clustering all have their individual problems. Both k-means and 1D-SOM are significantly affected by differences in the sizes of the clusters and the performance of average link agglomerative clustering strongly decreases for an increasing overlap between clusters.

On the other hand, the results obtained on real data also show the limitations of ant-based clustering and sorting: if cluster structures on several levels exist, it only identifies the upper level ones. A recursive application of the algorithm on the resulting clusters might be a way, to be tested in future work, to overcome this problem.

5 Analyzing ATTA-TM's performance for topographic mapping

Having established an objective evaluation of ATTA-C's *clustering* performance in the last section, we now proceed to study the suitability of ATTA-TM to tasks of *topographic mapping*. For this purpose, we select two alternative approximation algorithms to compare against, and we use an additional randomized approach to obtain a lower bound on mapping quality. Again, three distinct analytical evaluation functions are used and we employ the same benchmark data sets as used for clustering.

5.1 Algorithms for topographic mapping

From the pool of different approaches to topographic mapping, we select two well-known approaches: non-metric multidimensional scaling and two-dimensional self-organizing maps.

5.1.1 Non-metric multidimensional scaling

Non-metric multidimensional scaling [17] is implemented using a gradient descent method to (locally) minimize non-metric stress. After an initial random placement of all data items close to the origin of the two-dimensional map, the following iteration scheme starts. At the beginning of each iteration all data items are randomly permuted. One after the other, each data item is then 'pinned' and the positions of all other data items are shifted (in horizontal and vertical direction) with respect to the pinned item. With data element p pinned, the update of item i's position in direction x is

$$dx = -lr * \frac{d(p,i) - \delta(p,i)}{d(p,i)} \cdot d_x(p,i),$$

where lr is the learning rate which we set to 0.05, $d(p,i)$ is the Euclidean distance between the data items in map-space, $d_x(p,i)$ is the distance in direction x between the data items in map-space, and $\delta(p,i)$ is the dissimilarity between the data items in data-space. The update dy (in direction y) is computed analogously. The algorithm is stopped after 10 iterations have been performed.

(It was empirically observed that improvements to the mapping rarely occur after the fifth iteration).

5.1.2 Two-dimensional self-organizing maps

Two-dimensional self-organizing maps [16] are a straightforward extension of one-dimensional self-organizing maps. All our implementation details are equivalent to the 1D case (see section 4.1.3), except for the use of a square grid with a resolution of $\lceil\sqrt{\#items}\rceil \times \lceil\sqrt{\#items}\rceil$ (i.e, the number of grid cells is equal to the size of the data set tackled).

5.1.3 Lower bound

A lower bound on map quality is obtained using the known cluster structure of the data. For each cluster (as described by the class labels), we randomly determine a position on the grid. The data elements of the individual clusters are then randomly scattered on the grid cells in the close proximity of the respective grid position (one data item per grid cell only).

During the generation of the cluster positions care is taken that clusters have a sufficient spatial separation: the Euclidean distance between each pair of cluster positions must be at least $\sqrt{\#cells}/k$, otherwise one of the 'colliding' positions is regenerated. Here, $\#cells$ gives the overall number of grid cells and k is the number of clusters in the data set.

It is intuitively clear that the resulting 'mapping' is not topology-preserving to any significant degree, as it does not provide more information than a pure clustering.

5.2 Evaluation measures

Three distinct analytical measures are used to evaluate the quality of the resulting topographic mappings. Each of these reflects different aspects of the mapping quality, so that, combined, they provide an overall picture of the algorithms' performance.

5.2.1 Overall Pearson correlation

The *Pearson correlation coefficient* provides information on the degree of linear relationship between the distributions of two variables. In the context of topographic mappings it can therefore be employed to determine the degree to which a mapping preserves a linear relationship between the distances in data-space (described by distribution X) and those in map-space (described by distribution Y):

$$P = \frac{covariance(X,Y)}{variance(X) \cdot variance(Y)}.$$

P takes values in the interval $[-1,1]$, with 1 signifying perfect positive correlation, -1 signifying perfect negative correlation, and 0 signifying a complete lack of linear correlation. N.B. The Spearman rank correlation, which is able to capture non-linear correlations, has also been used. However, as the results obtained using this measure do not differ significantly from those under the Pearson correlation, we do not report them here.

5.2.2 Inter-cluster Pearson correlation

When the real cluster distribution is known, the correlation of the respective distances between cluster centers in data-space and in map-space can additionally be computed. This gives insight into the preservation of global relationships, that is, the relative distances between clusters. We refer to this measure as the *inter-cluster Pearson correlation*.

5.2.3 Intra-cluster Pearson correlation

Similarly, the Pearson correlation can be applied to obtain more detailed information on the sorting within individual clusters. The correlation coefficient computed for the dissimilarities of the data items belonging to individual clusters provides information about the preservation of more local relationships. We refer to this measure as the *intra-cluster Pearson correlation*.

5.3 Results

Extensive experiments carried out on both synthetic and real data permit us to gain an impression of the algorithms' strengths and weaknesses. The data sets used were the same as those employed for our study on clustering, that is, we obtained results on the *Square*, the *Sizes* and the real data sets from the Machine Learning Repository (see Section 4.4). To further evaluate topology preservation, we also ran experiments on synthetic, randomly generated data sets of up to 100 dimensions (see [10]). In each case, 50 independent runs of each algorithm were performed. Tables giving the mean value and standard deviation for each of our three measures for all algorithms and data sets can be found separately in [10]. Here we provide representative plots and a summary table only. In the following, we discuss our main findings:

5.3.1 Summary of the performance on synthetic and real data

- In all of our experiments, multidimensional scaling shows a very strong performance and it is the clear winner under all of the measures that reflect the preservation of global relationships in the data, that is, the overall Pearson correlation and the inter-cluster Pearson correlation. For the two-dimensional data (e.g., the *Square* data sets, see Figure 7a), where no topological defect needs to occur, all correlation values are close to the maximum value of 1.0, and a scatterplot of the distances in data-space versus those in map-space reveals that distance ratios are nearly perfectly preserved (see Figure 8a). With an increase in dimensionality both global and local topology-preservation do suffer, however. (See Table 2 and [10] for the full result tables.)

- While MDS clearly preserves local topology to a significant degree, it is, in this respect, repeatedly outperformed by the 2D-SOM, for high-dimensional data. When looking at the correlation values obtained by 2D-SOM (see Figure 7b), it is notable that the overall correlation and the intra-cluster correlation almost perfectly coincide: the overall Pearson correlation obtained is a result of local topology-preservation only. The value of the inter-cluster correlation confirms this. Even for two-dimensional data (where no topological defect would need to occur), its average value is close to 0 with an extremely high standard deviation, showing that cluster positions are determined virtually at random.

- These results accord with the particular focus of the two methods, MDS and 2D-SOM. While MDS trades off both global and local topology-preservation, 2D-SOM is primarily concerned with the optimization of local topology. Whilst it could be argued that local topology-preservation might induce topology-preservation on a more global scale, our analytical evaluation shows that this is not the case. Consequently, 2D-SOM is quite far from achieving a perfect preservation of distance ratios even for two-dimensional data (also see the scatterplot in Figure 8b). Its advantage is, however, that, due to its focus on local topology, it seems to cope much better with a rise in dimensionality (see Table 2).

- ATTA-TM never achieves an overall Pearson correlation to match MDS. The values it obtains are, however, reasonably high (up to a mean value of 0.798703 achieved on the Iris data set). This is in agreement with the results reported by other researchers [20] leading them to claim that the algorithm exhibited a topology-preserving behavior. However, these results are strongly at odds with those obtained under the measures of intra- and inter-cluster correlation. The local topology preservation of the ant-based algorithm amounts only to a weak 'sorting' (see Table 2, where intra-cluster correlations are summarized). Similarly, the global topology preservation is weak: the average inter-cluster correlation is often very low with large standard deviations, indicating a random placement of the clusters (analogously to 2D-SOM). Scatterplots of the distances in map-space versus those in data-space give an additional idea of how poor the topology-preservation really is (see Figure 8c).

- Runtimes (provided in [10]) show that the ant-based algorithm has a favorable (linear) scaling behavior in comparison to its rivals, which are both quadratic in the number of data items.

5.3.2 The pitfalls of the Pearson correlation

The discrepancy observed between the overall Pearson correlation, the inter-cluster correlation and the intra-cluster correlation obtained for ATTA-TM seems particularly strange, when one might assume that inter- and intra-cluster correlation comprise the overall correlation value. Our results show that this is not necessarily true.

The high overall correlation values obtained for ATTA-TM primarily emerge as an artifact of the preservation of cluster memberships and the clear spatial separation between clusters on the grid: this ensures that small distances in data-space are matched by small distances in map-space and, similarly, large ones are matched by large ones. Such a matching of extremes without any more precise discrimination is enough to yield reasonably high overall correlation values, particularly on data sets with large inter-cluster distances.

5.3.3 Is ant-based clustering and sorting better than random cluster mapping?

In order to demonstrate the above more clearly, that is, to show that a high Pearson correlation value can emerge purely as a result of clustering and no sorting, we evaluate the mappings generated by our lower bound method. Recall that this method generates an embedding that preserves cluster memberships, but randomly positions the clusters and performs no sorting within individual clusters.

When we look at the scatterplots obtained for both ATTA-TM and the lower bound method, clear similarities can be observed (compare Figure 8c and Figure 8d). Analytical evaluation reveals that the Pearson correlation obtained for the lower bound is almost as high as that for ATTA-TM. This is the case for most data sets, except for a number of the real data benchmarks where the correlation values obtained by random cluster mapping are close to zero. The latter seems to be an indication that at least some of the classes in these data are spatially not (well) separated, such that their mapping to spatially well separated clusters has a negative impact on the overall correlation. By comparison, the high correlation values obtained for ATTA-TM show that those clusters identified by the ant algorithm correspond to distinctive structures within the data (recall from the previous section that ATTA-C frequently discovers a too low number of clusters on the real data).

5.3.4 Data-dependency

One may observe that, differently from MDS and 2D-SOM, the performance of ATTA-TM varies significantly as a function of the data set tackled. In particular, the results obtained under the intra-cluster and the inter-cluster correlation are far from uniform across the range of data sets. Particularly interesting trends can be seen on the *Square* data sets.

First, the *inter*-cluster correlation differs for different degrees of overlap between the clusters (see Figure 7c). Recalling the clustering results presented in Section 4.4 we can understand this phenomenon more clearly. We have seen that, for the *Square6* and the *Square7* data sets, the density gradient is not sufficient for ATTA-TM to identify individual clusters, and the algorithm therefore gathers all four clusters in an individual cluster on the grid. Nonetheless, visual representation of this resulting cluster shows a rough sorting of the data elements into four distinct regions, which are sorted in the correct order. This is the reason for the high inter-cluster correlation obtained on this data. In contrast, on data in which the clusters are actually identified and separated on the grid, the locations of the centers is random, hence the inter-cluster correlation measure is lower and has a higher standard deviation.

Secondly, distinct variation in the *intra*-cluster correlation can also be observed (see Figure 7c). In particular it increases with an increasing overlap between clusters. This effect is caused by the change in distance ratios: the maximum distance found between elements decreases relatively to the distance between elements belonging to the same cluster. This permits the ant algorithm to obtain a better intra-cluster sorting, as it succeeds in distinguishing more precisely between gradations in distance.

5.3.5 Discussion

In conclusion, we must state that we do not find ant-based clustering and sorting to be a satisfactory method for topographic mapping. Both its capacity for intra- and inter-cluster sorting is very limited and unreliable and it therefore does not prove to be competitive to the established methods of MDS and 2D-SOM. (That these results apply equally to more basic versions of ant-based clustering and sorting can be deduced from results given in [10], where it is shown that both intra- and inter-cluster Pearson correlation have been improved by the modifications made to the basic algorithm.) In particular, our results indicate that the overall Pearson correlation is unreliable as a means of characterizing topology-preservation, unless great care with it is taken. This observation explains why earlier work on ant-based clustering and sorting reported that it exhibited topology-preserving properties, whereas we find little evidence for such behavior. Using the measures of intra- and inter-cluster correlation, and a comparison with random mappings, we have been able to distinguish true topology preservation from 'mere' clustering. We find the latter a more realistic explanation for the high values of Pearson correlation reported in the literature.

6 Conclusion

In the preceding sections we have presented the key issues and insights resulting from our work with ant-based clustering and sorting, which we hope will serve to answer some of the open questions about the approach. The main goal of this work has been to determine the true potential of ant-based clustering and sorting in data-mining applications. To achieve this, and to advance the state-of-the-art in ant-based clustering and sorting, we have presented an improved version of the heuristic, called ATTA, which incorporates improvements leading to gains in efficiency, robustness and overall solution quality.

The extensive evaluation and comparison of ATTA we have carried out seems to indicate that the use of ant-based clustering and sorting for clustering tasks is indeed justified. In cases where the number of clusters in the data is not known, the application of the heuristic is particularly promising. However, as far as topographic mapping is concerned, we would advise against its use, as much better methods exist for this purpose. (Its sole advantage seems to be a linear runtime.)

There are a number of directions in which research on ant-based clustering and sorting can be continued. Indeed, we are convinced that there is still room for improvement of the algorithm itself, though it will become increasingly difficult to obtain more than marginal performance gains. In our opinion, the hybridization of the algorithm with alternative clustering methods might therefore be a more rewarding and promising line of research.

Finally, we find it particularly interesting to reflect on, and in future to analyze, the working principles that ant-based clustering and sorting shares with other clustering methods. For example, one can regard ant-based clustering and sorting as a randomized partitioning method with a cluster representation based on representative point sets. Unlike in other clustering algorithms, this point set is not fixed but constantly being re-sampled. Ant-based clustering and sorting also has certain similarities with density-based methods, in particular the scaling parameter α plays a role similar to the density thresholds in density-based clustering. Studying the algorithm's working principles in more detail remains a fascinating prospect and might yield a slimmer version of ant-based clustering and sorting that incorporates the algorithm's essential mechanisms only.

Acknowledgements

Julia Handl thanks Prof. Günther Görz, University of Erlangen-Nuremberg, Germany, for supervision, and gratefully acknowledges support from scholarships of the German Academic Exchange service (DAAD) and the Gottlieb-Daimler and Karl-Benz Foundation, Germany. Joshua Knowles was supported by a CEC Marie Curie Research Fellowship while working at IRIDIA, and is now supported by a David Phillips Fellowship from the British Sciences Research Council (BBSRC), UK. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Research Director, and from the ''ANTS" project, an ''Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium.

References

1. Blake, C. & Merz, C. (1998). UCI repository of machine learning databases. Technical report, Department of Information and Computer Sciences, University of California, Irvine.
http://www.ics.uci.edu/~mlearn/MLRepository.html

2. Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence — From Natural to Artificial Systems.* New York, NY: Oxford University Press.

3. Carreira-Perpiñán, M. Á. (2001). *Continuous latent variable models for dimensionality reduction and sequential data reconstruction.* PhD thesis, Department of Computer Science, University of Sheffield, UK.

4. Deneubourg, J.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chrétien, L. (1991). The dynamics of collective sorting: Robot-like ants and ant-like robots. In *Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 1* (pp. 356–365). Cambridge, MA: MIT Press.

5. Dorigo, M., Bonabeau, E., & Theraulaz, G. (2000). Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8), 851–871.

6. Dorigo, M. & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization* (pp. 11–32). London, UK: McGraw Hill.

7. Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, 5(2), 137–172.

8. Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization.* Cambridge, MA: MIT Press.

9. Halkidi, M., Vazirgiannis, M., & Batistakis, I. (2000). Quality scheme assessment in the clustering process. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, Vol. 1910 of Lecture Notes in Computer Science (pp. 265–267). Heidelberg, Germany: Springer-Verlag.

10. Handl, J. (2003). Ant-based methods for tasks of clustering and topographic mapping: extensions, analysis and comparison with alternative methods. Masters thesis. University of Erlangen-Nuremberg, Germany. http://www.handl.julia.de

11. Handl, J., Knowles, J., & Dorigo, M. (2003). On the performance of ant-based clustering. In *Design and Application of Hybrid Intelligent Systems*, Vol. 104 of Frontiers in Artificial Intelligence and Applications (pp. 204–213). Amsterdam, The Netherlands: IOS Press.

12. Handl, J., Knowles, J., & Dorigo, M. (2004). Strategies for the increased robustness of ant-based clustering. In *Engineering Self-Organising Systems*, Vol. 2977 of Lecture Notes in Computer Science (pp. 90–104). Heidelberg, Germany: Springer-Verlag.

13. Handl, J. & Meyer, B. (2002). Improved ant-based clustering and sorting in a document retrieval interface. In *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature*, Vol. 2439

of Lecture Notes in Computer Science (pp. 913–923). Berlin, Germany: Springer-Verlag.

14. Hoe, K., Lai, W., & Tai, T. (2002). Homogeneous ants for web document similarity modeling and categorization. In *Proceedings of the Third International Workshop on Ant Algorithms*, Vol. 2463 of Lecture Notes in Computer Science (pp. 256–261). Heidelberg, Germany: Springer-Verlag.

15. Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31, 264–323.

16. Kohonen, T. (1995). *Self-Organizing Maps.* Berlin, Germany: Springer-Verlag.

17. Kruskal, J. B. (1964). Nonmetric multidimensional scaling: a numerical method. *Psychometrica*, 29, 115–129.

18. Kuntz, P., Layzell, P., & Snyers, D. (1997). A colony of ant-like agents for partitioning in VLSI technology. In *Proceedings of the Fourth European Conference on Artificial Life* (pp. 417–424). Cambridge, MA: MIT Press.

19. Kuntz, P. & Snyers, D. (1994). Emergent colonization and graph partitioning. In *Proceedings of the Third International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3* (pp. 494–500). Cambridge, MA: MIT Press.

20. Kuntz, P. & Snyers, D. (1999). New results on an ant-based heuristic for highlighting the organization of large graphs. In *Proceedings of the 1999 Congress on Evolutionary Computation* (pp. 1451–1458). Piscataway, NJ: IEEE Press.

21. Kuntz, P., Snyers, D., & Layzell, P. (1998). A stochastic heuristic for visualising graph clusters in a bi-dimensional space prior to partitioning. *Journal of Heuristics*, 5(3), 327–351.

22. Lumer, E. & Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. In *Proceedings of the Third International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3* (pp. 501–508). Cambridge, MA: MIT Press.

23. Lumer, E. & Faieta, B. (1995). Exploratory database analysis via self-organization. Unpublished manuscript. Results summarized in [2].

24. MacQueen, L. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1 (pp. 281–297). Berkeley, CA: University of California Press.

25. Martin, M., Chopard, B., & Albuquerque, P. (2002). Formation of an ant cemetery: Swarm intelligence or statistical accident? *Future Generation Computer Systems*, 18(7), 951–959.

26. Monmarché, N. (2000). *Algorithmes de fourmis artificielles: applications à la classification et à l'optimisation.* PhD thesis, Laboratoire d'Informatique, Université de Tours, France.

27. Ramos, V. & Merelo, J. (2002). Self-organized stigmergic document maps: Environment as a mechanism for context learning. In *Proceedings of the First Spanish Conference on Evolutionary and Bio-Inspired Algorithms* (pp. 284–293). Mérida, Spain: Centro Univ. Mérida.

28. Rijsbergen, C. V. (1979). *Information Retrieval, 2nd edition.* London, UK: Butterworths.

29. Tibshirani, R., Walther, G., & Hastie, T. (2000). Estimating the number of clusters in a dataset via the Gap statistic. Technical Report 208, Department of Statistics, Stanford University, CA.

30. Vesanto, J., Himberg, J., Alhoniemi, E., & Parkankangas, J. (2000). SOM Toolbox for Matlab 5. Technical Report A57, Neural Networks Research Centre, Helsinki University of Technology, Espoo, Finland.

31. Vorhees, E. (1985). *The effectiveness and efficiency of agglomerative hierarchical clustering in document retrieval.* PhD thesis, Department of Computer Science, Cornell University, Ithaca, NY.

32. Weisstein, E. W. (1999). Box-and-whisker plot. From MathWorld — A Wolfram Web Resource.
http://mathworld.wolfram.com/Box-and-WhiskerPlot.html

Algorithm 1 Basic ant algorithm

```
procedure BASIC_ALGORITHM
    /* INITIALIZATION PHASE */
    Randomly scatter data items on the toroidal grid
    for each $j$ in 1 to $\#agents$ do
        $i$ := random_select($free\_data\_items$)
        pick_up($agent(j), i$)
        $g$ := random_select($empty\_grid\_locations$)
        place_agent($agent(j), g$)
    end for
    /* MAIN LOOP */
    for each $it\_ctr$ in 1 to $\#iterations$ do
        $j$ := random_select($all\_agents$)
        step($agent(j), stepsize$)
        $i$ := carried_item($agent(j)$)
        $drop$ := drop_item?($f(i)$) // see equations 2 and 3
        if $drop$ = TRUE then
            drop($agent(j), i$)
            $pick := FALSE$
            while $pick$ = FALSE do
                $i$ := random_select($free\_data\_items$)
                $pick$ := pick_item?($f(i)$) // see equations 1 and 3
            end while
            pick_up($agent(j), i$)
        end if
    end for
end procedure
```

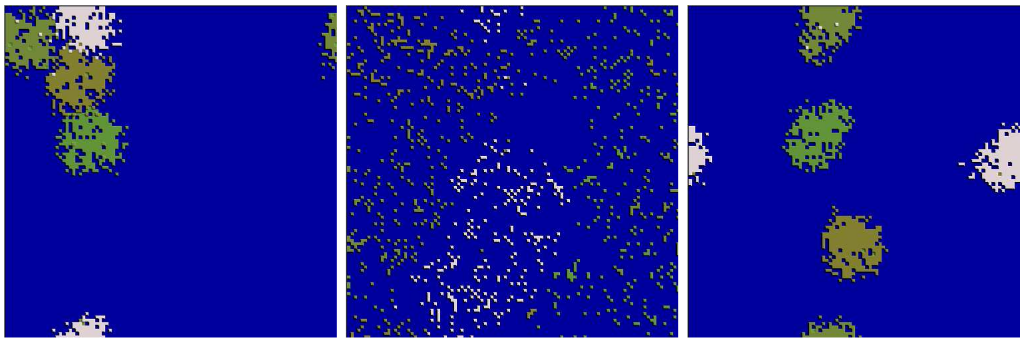

Figure 1: Spatial distribution on the grid at different stages of the run of our algorithm (with all modifications). (a) After the initial clustering phase: the clusters touch. (b) After the interlude with the modified neighborhood function: the clusters have 'dissolved' with data items spread out but remaining sorted. (c) Final result: the clusters are clearly separated.

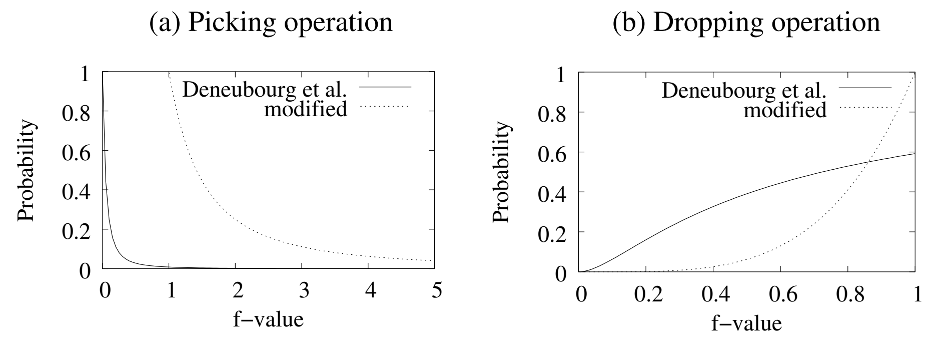

Figure 2: Old and new threshold functions. (a) Picking operation. (b) Dropping operation.

Table 1: Summary of the used real data sets from the Machine Learning Repository. N is the total number of data items in the data set, N_i is the number of items for cluster i, and D and C give the dimensionality and the number of clusters respectively.

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

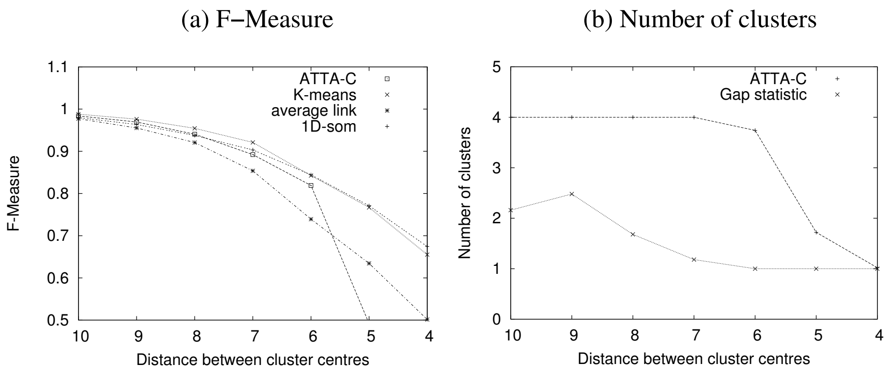

Figure 3: Performance as a function of the distance between the cluster centers on the *Square* data sets. Mean values over 50 runs. (a) F-measure (b) Number of identified clusters.

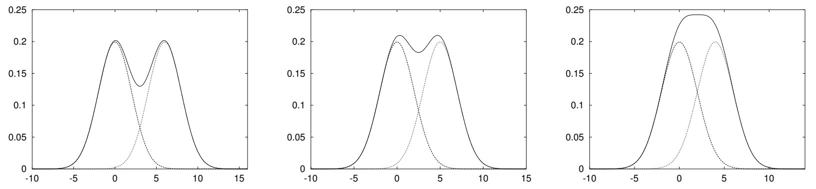

Figure 4: Theoretical density distribution along the connecting line between two cluster centers in the *Square5*, the *Square6* and the *Square7* test set (from left to right). Constructed as the superimposition of two one-dimensional normal distributions with standard deviation 2 and a distance of 6, 5 and 4 respectively.

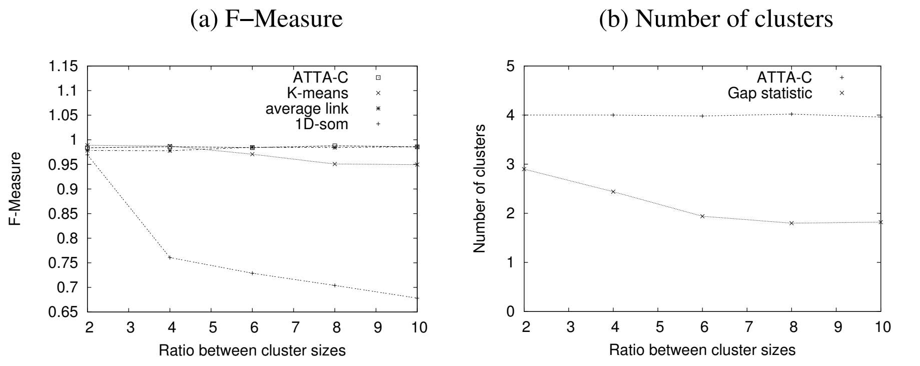

Figure 5: Performance as a function of the ratio between cluster sizes on the *Sizes* data sets. Mean values over 50 runs. (a) F-measure (b) Number of identified clusters.

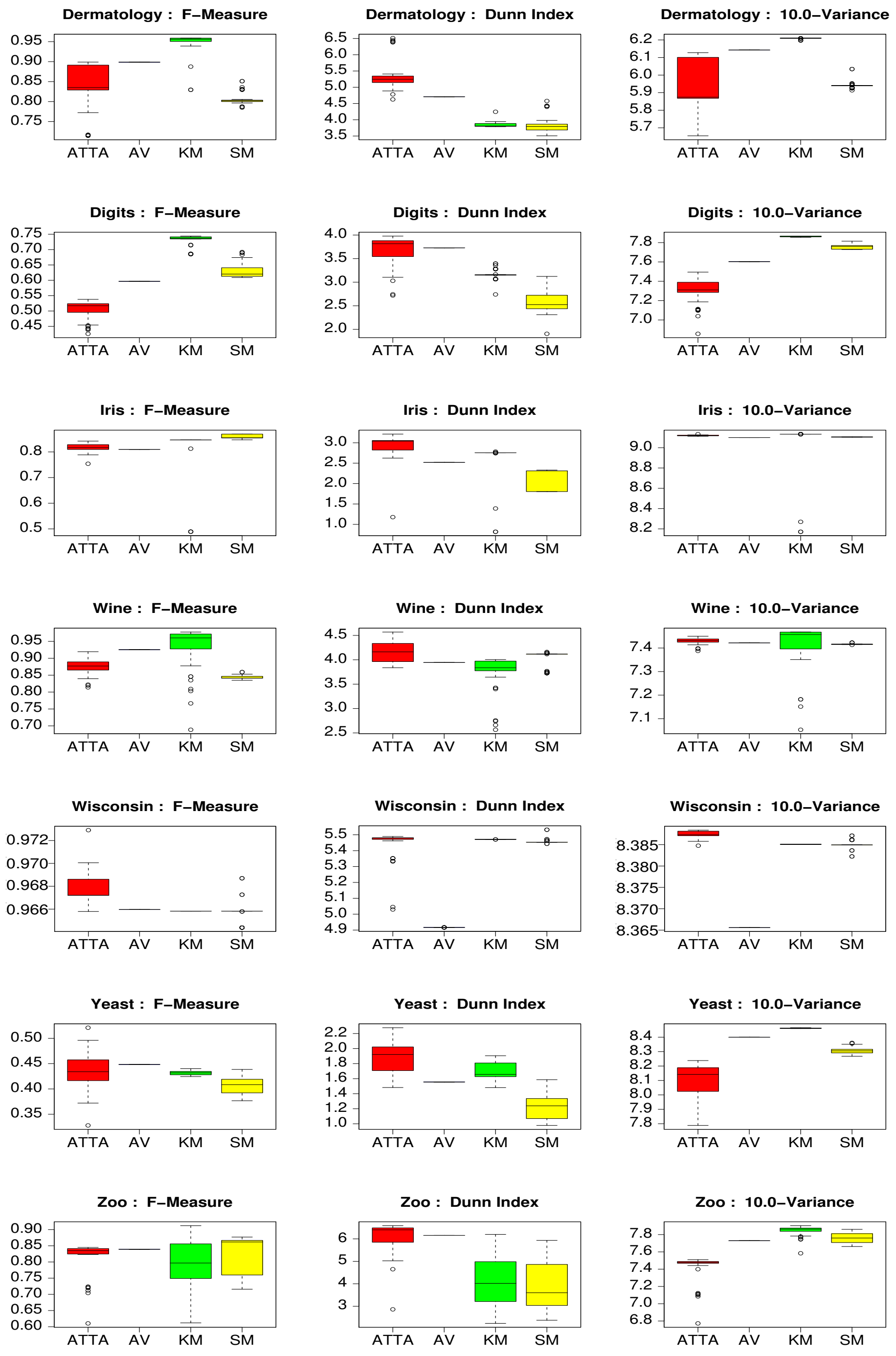

Figure 6: Boxplots [32] giving the distribution of F-measure, Dunn Index and Variance values achieved for 50 runs of each algorithm on the seven real data sets. Key: ATTA=ATTA-C, AV=average link agglomerative clustering, KM=k-means, SM=1D-SOM.

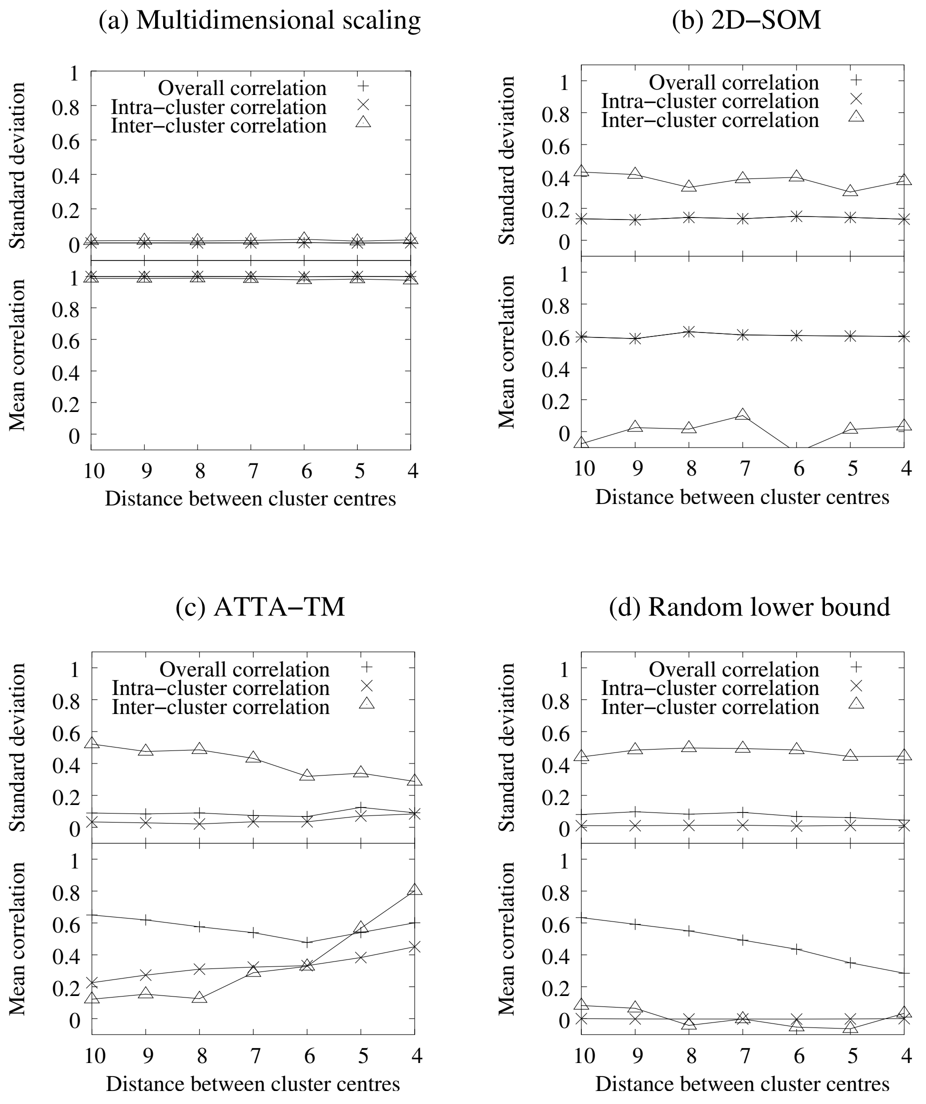

Figure 7: Results (mean and standard deviation over 50 runs) under the correlation measures for the *Square* data sets. (a) multidimensional scaling (b) 2D-SOM (c) ATTA-TM (d) the random lower bound.

Table 2: Mean intra-cluster correlation values for the algorithms over groups of the data sets: all the real data, all the *Sizes* data, all the *Square* data, the 10 dimensional and the 100-dimensional synthetic data.

Group	MDS	SOM	ATTA-TM	Lower bound
Real	0.8322889	0.5127919	0.4099512	-0.0211883
Sizes	0.9936002	0.5128746	0.2277882	-0.0005318
Square	0.9980671	0.6015531	0.3285400	-0.0014220
10D	0.6438175	0.5598625	0.0159658	0.0000151
100D	0.3703910	0.5629750	0.0108312	0.0006323

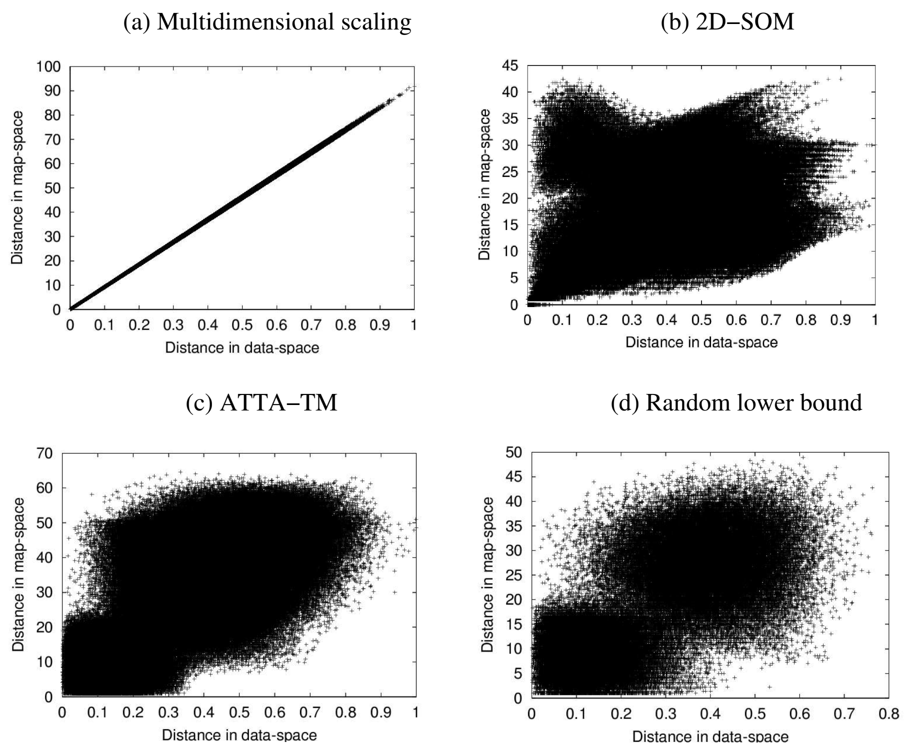

Figure 8: Scatterplots of the distances in data-space versus those in map-space for a solution generated on the *Square1* data set by (a) multidimensional scaling (b) 2D-SOM (c) ATTA-TM (d) the random lower bound..