Friedrich-Alexander-Universität Erlangen-Nürnberg
Institut für Informatik (IMMD)
Lehrstuhl für Künstliche Intelligenz

Ant-based methods for tasks of clustering and topographic mapping: improvements, evaluation and comparison with alternative methods

Diplomarbeit im Fach Informatik

vorgelegt von

Julia Handl

Betreuer:
Prof. Dr. Günther Görz, Prof. Dr. Marco Dorigo, Dr. Joshua Knowles

Beginn der Arbeit: 1. Juni 2003
Abgabe der Arbeit: 1. Dezember 2003

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäss übernommen wurden, sind als solche gekennzeichnet.

(Ort, Datum)

(Unterschrift)

Deutscher Titel

Ameisenbasierte Methoden für die Cluster-Analyse und die Generierung topographischer Karten: Verbesserung, Evaluierung und Vergleich zu Alternativ-Verfahren

Deutsche Zusammenfassung

Mit der stetigen Zunahme der Rechen- und Speicherkapazitäten von Großrechnern und Personalcomputern wird die Speicherung und Verarbeitung immer größerer Datenmengen möglich. Zur besseren Nutzung dieser Daten ist die Entwicklung von neuen und leistungsfähigeren Verfahren zur Analyse und Visualisierung notwendig. Die derzeit boomenden Forschungsfelder des "Document-Retrieval" und der Bioinformatik sind dabei treibende Kräfte.

Der Schwerpunkt dieser Arbeit sind zwei verschiedene Ansätze im Bereich des "Data-Mining": zum einen die Cluster-Analyse größerer Datenmengen, und zum anderen ihre Visualisierung mittels topographischer Karten. Beide haben im Rahmen der Datenanalyse beziehungsweise der Datenvisualisierung große Bedeutung und sind aus diesem Grund aktive Forschungsbereiche. Sowohl die Cluster-Analyse, als auch die Generierung topographischer Karten können als Optimierungsprobleme aufgefasst werden – die spezifischen Optimierungskriterien sind jedoch nicht eindeutig definiert, sondern generell äußerst problemabhängig. Diese Tatsache erschwert es, die Leistung eines individuellen Algorithmus losgelöst von einer bestimmten Anwendung zu beurteilen, und macht es ebenfalls schwierig, wenn nicht gar unmöglich, einen Algorithmus zu entwickeln, der mit gleichbleibend guter Qualität bei jeder denkbaren Datenverteilung anwendbar ist.

Daher ist eine gründlichen Evaluierung der Algorithmen und ihr Vergleich auf einer weiten Spanne von Benchmark-Daten unentbehrlich, da sie eben nur in unmittelbarem Bezug auf die Eigenschaften der verwendeten Daten und im direkten Vergleich mit anderen Algorithmen wirklich bewertet werden können.

In dieser Arbeit versuchen wir, diesem Anspruch im Zusammenhang mit einem bestimmten Algorithmus gerecht zu werden. Der behandelte Algorithmus ist ein heuristisches Verfahren, das vom Verhalten natürlicher Ameisen inspiriert ist, und dessen Verwendung sowohl für die Cluster-Analyse, als auch für die Generierung topographischer Karten vorgeschlagen worden ist. Eine gründliche Analyse der tatsächlichen Qualität der generierten Lösungen ist jedoch bisher nicht durchgeführt worden.

Unser Ziel einer solchen Analyse realisieren wir hier in mehreren Schritten. Wir führen zunächst eine Reihe von Veränderungen des Algorithmus ein, die seine Robustheit, die Qualität der erzeugten Lösungen und seine Laufzeit verbessern. Dies umfasst auch die Beschreibung einer Methode

zur Bestimmung angemessener Parameter für unterschiedliche Datenverteilungen, und einer Technik, die die objektive und unverfälschte Evaluierung der erzeugten Lösungen ermöglicht. Schließlich untersuchen wir die tatsächliche Leistung des Algorithmus und vergleichen ihn mit Standardtechniken für die Cluster-Analyse und die Generierung von topographischen Karten. Zu diesem Zweck verwenden wir eine Anzahl unterschiedlicher analytischer Evaluierungsfunktionen, einen umfangreichen Satz von künstlichen Benchmark-Daten, sowie zusätzliche Datensätze aus der Praxis.

Unsere Experimente unterstreichen die Eignung des Algorithmus für die Cluster-Analyse: die Qualität der erzeugten Lösungen ist überzeugend, und die Fähigkeit des Algorithmus, die Anzahl der vorhandenen Cluster automatisch zu bestimmen, sticht dabei besonders ins Auge. Für die Generierung topographischer Karten erweist sich der Algorithmus jedoch als weniger geeignet. Wir belegen, dass die generierten Darstellungen dem Anspruch der Bewahrung der Daten-Topologie kaum gerecht werden, und erläutern, warum bisherige Forschungsergebnisse zu einer falschen Einschätzung der tatsächlichen Leistung des Algorithmus führen konnten.

Abstract

All around us, impacting on all aspects of life, we are witnessing the storage and processing of increasingly large amounts of data as available computing power continues its inexorable rise. In order to fully benefit from all these data, the development of novel and improved techniques for both data analysis and data visualisation is imperative. Currently, two of the main driving forces in this respect are the research fields of document retrieval and bioinformatics.

In this thesis we focus on two tasks, *clustering* and *topographic mapping*, which are both of great importance within the context of data analysis and data visualisation and have been subject of active research in the past few years. Both clustering and topographic mapping can be considered as optimisation tasks – however, their respective optimisation criteria are not uniquely defined, but must usually be tailored according to the problem domain. This makes it (1) hard to evaluate the performance of an algorithm without respect to a particular application and (2) hard, if not impossible, to develop one best algorithm. These properties increase the necessity of a thorough evaluation and comparison of these techniques on broad ranges of benchmark data, as the quality of a particular method can only be judged with respect to particular data properties and its relative performance when compared to others.

In this thesis, we make an effort to satisfy these demands concerning one particular algorithm, *ant-based clustering and sorting*. Introduced in the field of nature-inspired heuristics, it has been claimed that this method can perform clustering and topographic mapping, but, so far, it has not been subject to thorough analytical analysis.

Towards the goal of a full evaluation of the algorithm, we proceed in several steps. First, we introduce several modifications that improve the robustness of ant-based clustering and sorting, and its performance in terms of quality and runtime. This includes a method for the derivation of appropriate parameter settings across differing data sets. Secondly, we describe a scheme that permits the unbiased interpretation of the obtained clustering results. Finally, we investigate the claims made on the algorithm's performance, and compare it to standard techniques for clustering and topographic mapping. This is done using a set of analytical evaluation functions and a range of synthetic and real data collections.

Our experiments confirm the ability of ant-based clustering and sorting to automatically identify the number of cluster inherent to a data collection and to produce high quality solutions. However, the results obtained for topographic mapping are very poor. We provide evidence that the solutions generated by the ant algorithm are barely topology-preserving, and we explain in details why results have – in spite of this – been misinterpreted in previous research.

Publications resulting from this work

Technical reports

Handl, J., Knowles, J. and Dorigo, M. (2003) Ant-based clustering: a comparative study of its performance with respect to K-means, average link and 1D-SOM. Technical Report TR-IRIDIA-2003-24. IRIDIA, Université Libre de Bruxelles. Brussels.

International conferences and workshops

Handl, J., Knowles, J. and Dorigo, M. (2003) On the performance of ant-based clustering. To appear in *Proceedings of the Third International Conference on Hybrid Intelligent Systems (HIS 2002)*. IOS Press.

Handl, J., Knowles, J. and Dorigo, M. (2003). Strategies for the increased robustness of ant-based clustering. To appear in *Postproceedings of the First International Workshop on Engineering Self-Organising Applications (ESOA 2003)*. Springer-Verlag.

Manuscripts in preparation

Handl, J., Knowles, J. and Dorigo, M. (2003) Ant-based clustering and topographic mapping. (To submit to the journal *Artificial Life*)

Acknowledgements

First of all, I would like to thank my supervisors Prof. Dr. Günther Görz, Prof. Dr. Marco Dorigo and Dr. Joshua Knowles for their support, help and feedback throughout the progress of this work. I am particularly indebted to Prof. Dr. Marco Dorigo for giving me the opportunity to spend six months at IRIDIA.

Many thanks to all of my colleagues at IRIDIA who have made my stay here so enjoyable: Vito, Shervin, Rodi, Max, Mauro, Halva, Elio, Christian, Carlotta and Bruno.

Finally, I am very much obliged to the German Academic Exchange Service for supporting me with another scholarship, and to Dr. Gabriella Kokai and Prof. Dr. Hans-Jürgen Schneider, who once again helped me to get the application on its way.

Contents

List of Figures

List of Tables

To my family

The purpose of computing is insight, not numbers.
(Hamming)

1 Introduction

The data volumes arising in the fields of document retrieval and bioinformatics are two prominent examples of a trend in a wide range of different research areas. Novel techniques (such as the Internet in document retrieval, multi-array experiments in bioinformatics, physical simulations in scientific computing and many more) give rise to enormous amounts of data, which can only be handled and processed by means of computers, which, in turn, their increased storage and computing power renders possible nowadays.

Still, this trend requires and triggers a continuous development in database technology and processing techniques. The automatic analysis of the data and the comprehensible presentation (of the data and/or the results of the analysis process) to humans is of particular importance in this context, as it is only when the data is interpreted, that it becomes meaningful and can provide new information and insight. The research field addressing these major challenges is generally referred to as *data-mining*, which is itself only one step in the *knowledge discovery* process.

The focus of this work is on two subproblems encountered in data-mining, namely *cluster analysis* and *topographic mapping*.

1.1 Cluster analysis

Cluster analysis is concerned with the division of data into homogeneous subgroups. Informally, the objectives of this division are twofold: data items within one cluster are required to be similar to each other, while those within different clusters should be dissimilar.

Formally, the *clustering problem* can be defined as an optimisation problem [7]:

Definition 1.1: The clustering problem

INSTANCE:	A finite set X, a distance measure $\delta(i,j) \in R_0^+$ for $i,j \in X$, two positive integers K and B, and a criterion function $J(C,\delta(\cdot,\cdot))$ on a K-partition $C = \{C_1,\ldots,C_K\}$ of X and measure $\delta(\cdot,\cdot)$.
QUESTION:	Is there a partition of X into disjoint sets $C_1,\ldots,C_K$ such that $J(C,\delta(\cdot,\cdot)) \leq B$?
OPTIMISATION:	Find the partition of X into disjoint sets $C_1,\ldots,C_K$ that minimises the expression $J(C,\delta(\cdot,\cdot))$.

The number $R(K,N)$ of possible solutions for the division of a data set of size N into K partitions is given by the Stirling number of the second kind:

$$R(K,N) = \frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}(i)^N \approx \frac{K^N}{K!}$$

Hence, even with a fixed number of partitions K, the search space for the clustering problem grows exponentially and cannot be scanned exhaustively already for medium sized problems. Indeed, the clustering problem is known to be NP-complete in many of its definitions [7].

Definitions of the clustering problem vary in the optimisation criterion J_e and the distance function $\delta(\cdot,\cdot)$ used. A multitude of possible optimisation criteria exists, examples are the minimisation of the intra-cluster variances or the maximisation of the inter-cluster distances. Possible choices for the distance function include the Euclidean distance, the Cosine similarity or the Correlation coefficient. While it is know that an NP-complete clustering problem remains intractable regardless of the employed distance function, there are optimisation criteria like the minimum-variance criterion for which the conservation of NP-completeness has not yet been shown.[1]

As we will see in Section 2, the choice of the distance function and the optimisation criterion crucially determines the type of cluster that can be identified using one particular algorithm. Most clustering methods therefore intrinsically make assumptions on the number of clusters, their shape, or the degree of spatial separation between clusters.

The selection of a clustering algorithm must therefore always account for the properties of the clustering problem being tackled. Similarly, the

[1] However, no polynomial-time algorithm is known for the minimisation of the intra-cluster variance and NP-completeness has been proven for optimisation criteria, which, intuitively, appear to be simpler (such as the definition of an upper bound on all intra-cluster distances).

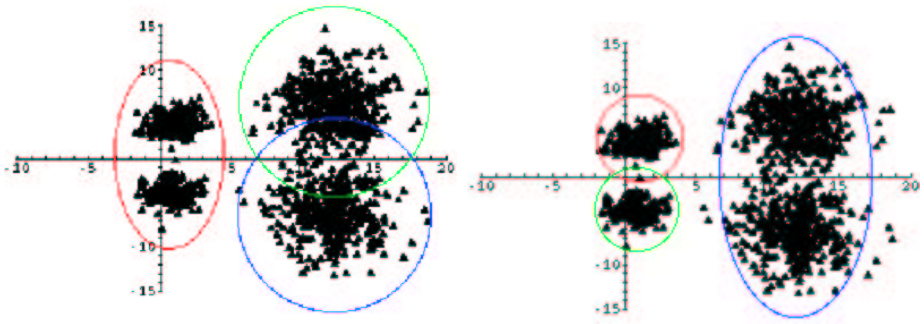


Figure 1: Two possible partitions of the same data set. Dependent on the optimisation criterion either of them could be considered of better quality. Different clustering algorithms will produce differing results.

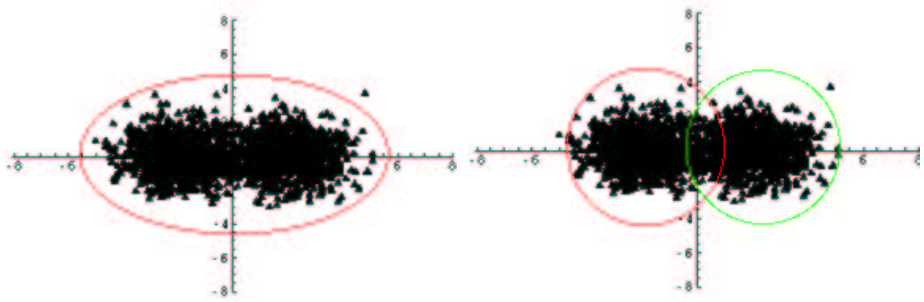


Figure 2: The exact number of clusters in the data is not always objectively clear. The displayed data set can be considered to consist of either one or two clusters.

performance of one particular clustering algorithm can only be judged with regard to a particular problem structure. This is also reflected by the difficulty to devise performance measures for the quality of a clustering solution.[2] Again, there is not one measure, which is valid for all application domains, but the available measures all imply some knowledge about the structure of the problem (e.g., the use of the minimum variance criterion makes sense only in the presence of circularly shaped clusters).[3]

[2]Note that this only applies to performance measures that do NOT compare to the correct partitioning. If the solution is a priori known (i.e., on benchmark data), performance measures are straightforward.

[3]This becomes intuitively clear if we think about clustering in terms of an optimisation problem. If an invariably valid evaluation function existed, we could use this function in an optimisation procedure, thereby obtaining an algorithm applicable regardless of the problem domain.

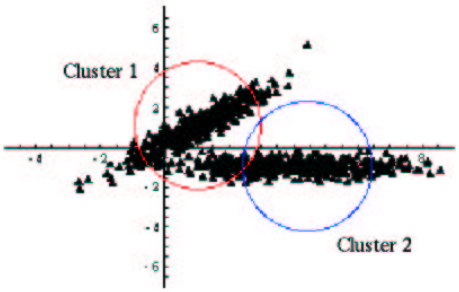


Figure 3: Two elliptical clusters. On this data set, the minimum variance criterion fails to separate the two clusters, as Cluster 2's left tail is closer to the cluster centre of Cluster 1.

1.2 Topographic mapping

We have seen above that the principal goal of cluster analysis is to identify related groups in the data. By this means, a notion of relation (in general similarity) is established between the data elements grouped within one cluster. Algorithms performing topographic mapping go one step further. Like clustering algorithms they aim to capture the main structure in the data. However, they are not limited to the detection of homogeneous groups within the data but they attempt to pinpoint additional information. This is (1) relationships between individual clusters (i.e., gradations in the degree of dissimilarity) and (2) relationships between data items belonging to one cluster (i.e., gradations in the degree of similarity).

Algorithms performing topographic mappings try to capture neighbourhood relations by providing a two-dimensional (possibly also three- or higher-dimensional) visualisation of the high-dimensional data-space (in the following the low-dimensional space used for visualisation will be referred to as *map-space*). In the literature the resulting *topographic maps* are also termed *neighbourhood-preserving*, *topological*, *topology-preserving*, *ordering* or *systematic* maps.

The main assumption underlying this scheme is that the data items, although located within a high-dimensional space, are restricted to a lower-dimensional manifold (embedded in the high-dimensional space), which can therefore be 'unrolled' in map-space. It is intuitively clear that the dimensionality of this manifold crucially determines the quality of the topographic map that can possibly be obtained. A perfect topographic mapping only exists if the dimensionality intrinsic to the manifold coincides with that of the map-space. A deviation in dimensionality will naturally involve a certain loss of neighbourhood information. In data visualisation, we will, in the majority of cases, have to deal with such a reduction of dimensionality.

Formally, the definition of topological equivalence is as follows [25]:

Definition 1.2: Topological equivalence

Let $\langle X, \delta(.,.)\rangle$, $\langle Y, d(.,.)\rangle$ be identical metric spaces with countable dense subsets. Let $\Theta : X \to Y$ be a bijection such that:

$$\forall i,j,k,l \in X : \delta(i,j) < \delta(k,l)$$
$$\Rightarrow d(\Theta(i),\Theta(j)) \leq d(\Theta(k),\Theta(l))$$

$$\forall i,j,k,l \in Y : d(i,j) < d(k,l)$$
$$\Rightarrow \delta(\Theta^{-1}(i),\Theta^{-1}(j)) \leq \delta(\Theta^{-1}(k),\Theta^{-1}(l))$$

Then Θ is a homeomorphism, and X and Y are topologically equivalent (see [25] for a proof).

This theorem shows that a 'perfectly neighbourhood-preserving' map is not uniquely defined. It depends on the definition of a *neighbourhood* in both data- and map-space (reflected by the distance functions $\delta(\cdot,\cdot)$ and $d(\cdot,\cdot)$). Also, the above notion of topological equivalence only requires the preservation of similarity-orderings (relative distances) within the data. Dependent on the application, additional constraints on the mapping might also be possible, for example it could be desirable to preserve similarities (absolute distances).

In order to judge the performance of different neighbourhood-preserving algorithms, analytical means to assess the quality of the resulting maps are crucial. Again, as with the clustering problem, the quality of a solution cannot be judged without regard to the application context.

At first sight, the derivation of criteria for a perfectly neighbourhood-preserving map appears to be rather straightforward, given the aim of distance-preservation and a metric distance measure within data- and map-space. Yet, both the definition of the neighbourhood and the aim of distance-preservation already introduce a large amount of problem-specific assumptions. In fact it is not clear at all under which circumstances an intuitively 'perfect' mapping (i.e., perfect in the sense of similarity-preservation) is superior to all other ones. In many applications different mappings might be more useful: those that sacrifice precise similarity-preservation in order to accentuate the principal structures.

It is even more difficult to devise evaluation functions for the degree of discrepancy to a perfect mapping. Mappings with topological defects will have to play a certain trade-off between the preservation of local neighbourhood-structures and global relationships. Which of these aspects is desired to dominate the mapping process very much depends on the application.

All of these issues will be further explored in Section 3.

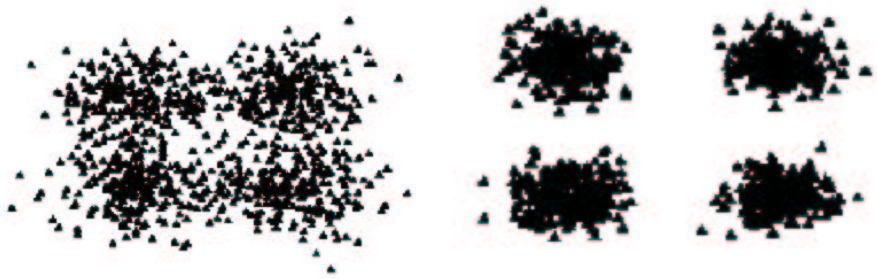


Figure 4: Two different topographic mappings of the same data set. The left map reproduces the exact data positions, while the right one emphasises the underlying phenomenon. It depends on the application, which representation is more desirable.

1.3 Implications for ant-based clustering and sorting

We have seen above, that both clustering and topographic mapping are highly application-dependent tasks. This makes clear that the notion of the performance of a particular algorithm for clustering or topographic mapping only becomes meaningful within the context of a specified problem domain. The problem type will also be decisive for the kind of evaluation function to be used.

Hence, it is only through detailed analysis and comparison of algorithms on different kinds of test problems, that we can gain increased insight into the performance of one particular method. This comprises information on (1) which types of problems the method is suited for and (2) how the results on these problems compare to those obtained with alternative (suitable) methods.

The aim of this work is to apply these principles to the evaluation of the performance of ant-based clustering and sorting, an algorithm emerging from the field of nature-inspired heuristics, which has recently been applied to tasks of clustering and topographic mapping. In particular, it has been claimed that this algorithm is, like self-organising maps, capable of performing both of these tasks simultaneously.

Yet, so far there hasn't been any thorough analysis of the algorithm's real performance. Experiments on artificial data have been limited to very few and simple toy problems. The evaluation of the results on this data has mostly been based on visual observation and only very little analytical evaluation has been carried out. In fact, those analytical results available even raise doubts on the algorithm's suitability for tasks of topographic mapping. The reported experiments on real data are similarly limited and have even less been subject to analytical evaluation. Finally, for both artificial and real test data there have been hardly any comparisons to alternative methods.

In an effort to fill this gap it is the main goal of this work to compare

the performance of ant-based clustering and sorting to a number of standard techniques of both clustering and topographic mapping, employing a range of artificial and real test data and a number of selected analytical evaluation functions.

The remainder of this thesis is structured as follows. In an attempt to provide an overview on the wide research field of cluster analysis, Section 2 introduces the main categories of clustering algorithms. A selection of algorithms for topographic mapping are then discussed in Section 3. Section 4 summarises the basic concepts of ant-based clustering and sorting and gives a survey of previous research in this field. The literature review closes in Section 5 with an overview of analytical measures for the evaluation of clustering and mapping results.

The second part of this thesis describes our work purely related to ant-based clustering and sorting. Section 6 starts with a description of the algorithm used, and then introduces a number of algorithmic changes that improve the algorithm's robustness, and its performance in terms of quality and runtime. Experimental results demonstrating the impact of individual modifications are presented. In preparation of a thorough analytical evaluation of the algorithm, we present a scheme to derive parameter settings across arbitrary data sets, and a method that permits the unbiased evaluation of the clustering results.

The third part of this thesis finally consists of the comparative study, which has been the main goal of this work. It starts with a description of the experimental setup in Section 7, where the selection of algorithms, evaluation functions and benchmarks is motivated, and a short description of each of these is given. Subsequently, the results obtained for clustering and topographic mapping are presented. Section 8 discusses future work and concludes.

1.4 Notation

In the above definitions of clustering and topographic mapping, we have introduced the distance functions $\delta(i,j)$ and $d(\Theta(i),\Theta(j))$. $\delta(i,j)$ gives the dissimilarity between the data items i and j (in data-space): in this work, we use numerical data only, and $\delta(x_i,x_j)$ is either the Euclidean distance or the Cosine distance between the data vectors of elements i and j (this depends on the type of data used and is detailed in Section 7). The mapping operation Θ assigns a position in map-space to each data item, and $d(\Theta(i),\Theta(j))$ gives the Euclidean distance between the map-positions of two data items.

We will, in the following, use the following simplified notation: for two data items i and j, $\delta(i,j)$ indicates the dissimilarity of the data items, and $d(i,j)$ gives the distance of their assigned positions in map-space.

The notion of finding 'natural groups' tends to imply that the algorithm should passively conform like a wet teeshirt.
(Anderberg)

2 Cluster analysis

Clustering problems arise in a variety of different disciplines ranging from sociology and psychology to commerce, biology and computer science. Many of these are tasks of knowledge discovery, but clustering can also be applied in different contexts, for example to tackle the issue of efficient coding/compression and data transfer in computer science. From an information theoretic point of view, the common theme in all of these applications is that large volumes of data have to be modelled in a compact way while retaining a maximum of information.[4]

Clustering methods have been studied for many years, but they continue to be the subject of active research. In recent years particular attention has been paid to scalability issues, that is, the applicability of clustering methods to very large and/or high-dimensional sets of data. Due to the long tradition of research, there is a wide range of clustering methods available nowadays, which differ not only in the principles of the algorithm used (which of course determine runtime behaviour and scalability) but also in many of their most basic properties, such as:

- **The types of attributes handled**
  Many algorithms can only handle numerical data (as opposed to categorical data and proximity data) as they require, for example, the explicit computation of cluster centres.

[4]Information-theoretical principles are therefore applicable in cluster analysis. As an example, minimum description length (MDL, [67]) can be applied to determine the optimal number of clusters (given a specific cluster model) within a given data set [13].

- **The shapes of identifiable clusters**
  Clustering algorithms that use an explicit optimisation criterion might be restricted to the identification of a particular type of cluster, for example convex or circularly shaped clusters.

- **The kind of partitioning generated**
  While most algorithms use 'hard' assignments (i.e., each data element is assigned to exactly one cluster), 'fuzzy' approaches (where gradual and multiple memberships are possible) also exist. Furthermore, some algorithms pay particular attention to outliers within the data, whereas others need to assign each data element to a cluster, with the effect that (1) outliers are not identified and (2) the final solution might be significantly affected by them.

The four main classes of clustering algorithms available nowadays are *partitioning methods*, *hierarchical methods*, *density-based clustering* and *grid-based clustering*. In the following, we will shortly introduce the basic concepts of these categories and, for each one of them, we will survey the best-known representatives. For a more extensive survey the reader is referred to [6].

2.1 Partitioning methods

Partitioning methods are among the most popular approaches to clustering, which is mainly due to their ease of implementation and their favourable runtime behaviour. Clustering methods of this type start with an initial partitioning of the data, and iteratively improve it by means of a greedy heuristic: data items are repeatedly reassigned to clusters according to a specific optimisation criterion.

The *K-means* algorithm [53] is the best-known algorithm within this class. The criterion-function used by K-means is that of *minimum variance*, that is, the sum of squares of the differences between data items and their assigned cluster centres is minimised. Starting from a random partitioning, the algorithm repeatedly (1) computes the current cluster centres and (2) reassigns each data item to the cluster centre closest to it.[5] K-means terminates when no more reassignments take place, which is usually the case after only few iterations. This algorithmic scheme therefore results in a runtime complexity linear in the number of data elements.

However, like all clustering algorithms, K-means also has its limitations, some of which can only partially be overcome.

- The final result highly depends on the initial partitioning as K-means'

[5] This is the batch version of K-means. Note that a second 'online' version exists, in which cluster centres are recomputed after each individual point relocation.

greedy optimisation approach is prone to converge to suboptimal solutions.

- The number of clusters K has to be provided as an input parameter, which poses a problem in applications where the correct number of clusters is not known a priori.
- K-means frequently generates empty clusters. If empty clusters are continuously reinitialised in order to enforce the generation of K clusters, this can lead to convergence problems.
- Due to the use of the minimum variance criterion, K-means is best-suited for the identification of spherically shaped clusters. It can completely fail for clusters with other shapes.[6]
- As K-means requires the computation of explicit cluster centres, it can only be applied to numerical data.
- The algorithm does not scale well for high-dimensional data, as the distances between the cluster centres and all data items have to be recomputed in each iteration.[7]

Due to the popularity of the algorithm, much work has been done to overcome several of these limitations.

K-means' disposition to converge to suboptimal solutions is commonly reduced by repeatedly running the algorithm (starting from different initial partitionings) and keeping the solution with the minimum variance only. Also, different heuristic initialisation schemes have been introduced in the literature (see [62] for a comparison).

Similarly, the most suitable number of clusters K can usually be determined (automatically or semi-automatically) by the generation of multiple partitionings (using different numbers of clusters) and their evaluation under one or several performance measures. In plots of the performance versus the cluster number, the best K then generally shows in a significant 'knee', but it may also be determined using advanced optimisation schemes like simulated annealing or genetic algorithms [33].

Bisection K-means [73] employs ideas from divisive hierarchical clustering (cf. Section 2.2) to improve K-means' performance in terms of quality. Starting from only one cluster, an iteration of bisection steps is used, which are simply based on local K-means applications (with $K = 2$). Steinbach et al. show that this simple strategy provides results competitive to those

[6] In these cases it is only if the clusters are very well separated that K-means will perform well.

[7] Note, that no precomputed distances can be used, as the cluster centres do not correspond to actual data points in the set and can therefore change in each iteration.

obtained with hierarchical clustering methods, while maintaining K-means' favourable time-complexity.

In order to permit working with any attribute type, *K-modes* [39] and *K-medoids* [43] algorithms have been introduced, which use a cluster representation based on modes[8] and medoids [9] respectively. While the K-medoids algorithms have the additional advantage of a reduced sensitivity to outliers, their original representative PAM ("Partitioning around Medoids", [43]) suffered from scalability problems. This weakness has been addressed by the use of a simple sampling technique in CLARA ("Clustering Large Applications", [43]), where a partitioning is computed for several small subsets of the data. The resulting clusters are used as stereotypes to partition the complete data set, the objective values for all corresponding partitionings of the entire set are computed, and the best solution is selected. A further development is the algorithm CLARANS ("Clustering Large Applications based upon Randomised Search", [59]), which performs iterated local search in the space of all sets of medoids. Neighbouring sets in this search space are all those that differ by exactly one medoid.

Finally, while all methods above generate crisp clustering solutions, a fuzzy variant of K-means also exists. Many of the ideas presented in this section directly carry over to the application to *Fuzzy C-means* [31].

2.2 Hierarchical methods

Hierarchical algorithms take a quite different approach to cluster analysis. They do not start with an initial partitioning that already contains the correct (and final) number of clusters, but they successively generate partitionings of different granularities. Here, *divisive* and *agglomerative* methods can be distinguished. Divisive methods start with one single cluster (containing all data elements), and gradually refine it through the division of one cluster (possibly more) in each iteration. Agglomerative clustering algorithms [79] take the inverse approach. They start with the finest partitioning possible (i.e., *singletons*, each representing an individual cluster), and then merge the most similar clusters in each iteration. Both classes of methods terminate once an appropriate stopping criterion is met, for example, when a particular number of clusters has been reached. In spite of the higher potential of divisive strategies to capture the global relationships between clusters, most hierarchical algorithms are based on agglomerative strategies, as cluster division entails particularly high computational costs (because possible

[8] The mode of a cluster is defined as the vector that minimises the sum of its dissimilarity to all data items within the cluster. Here, the definition of the dissimilarity function can account for categorical attributes. The medoid is not necessarily an actual element of the data set.

[9] The medoid of a cluster is the data item, which best represents it (e.g., the data item that is closest to the precise cluster centre). Hence, it always corresponds to an actual data point, thus permitting the work with precomputed dissimilarity data.

subsets of one cluster have to be considered).

Classical hierarchical algorithms can be further categorised by means of their *linkage metric*, that is, the criterion used to decide which sets of points are to be subdivided or merged respectively in each iteration. The linkage metric has to be computed for all candidate sets of data points in the current partition: in agglomerative clustering this candidate set usually simply consists of all pairs of clusters, in divisive clustering subsets of clusters have to be analysed.

The three most popular linkage metrics are those of *single link*, *complete link* and *average link*, which are also named *graph metrics*.[10] They operate on two sets of points C_1 and C_2, which correspond to individual clusters in the agglomerative case and to one of many possible bisections of a cluster in the hierarchical case. Each of these three linkage metrics then requires the computation of the distances between all $N = |C_1||C_2|$ possible pairs of data items i and j, where $i \in C_1$ and $j \in C_2$. Their values are defined as

$$\delta_{slink}(i,j) = \min_{i \in C_1, j \in C_2} \delta(i,j)$$

$$\delta_{clink}(i,j) = \max_{i \in C_1, j \in C_2} \delta(i,j)$$

$$\delta_{alink}(i,j) = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} \delta(i,j)$$

Contrarily to this cluster representation purely based on point sets, *geometric metrics* explicitly describe clusters by means of their cluster centre. Examples of this type of metric are those based on the distance between cluster centroids, the distance between cluster medians, or the impact of a merging operation on the intra-cluster variance. For more details on linkage metrics and their efficient computation see [60].

While the advantages of hierarchical linkage-based clustering methods are their simplicity, the implicit generation of a hierarchical structure (which permits the easy retrieval of clustering information on different levels of granularity) and the capacity of most linkage metrics to work with any kind of data, they are also limited in a number of respects.

- The hierarchical representation does not necessarily reflect any significant information about actual meaningful relationships between clusters.
- If the correct number of clusters is not known, it can be difficult to derive an appropriate stopping criterion.

[10] This name refers to the operation of these metrics on the completely connected graph whose vertices represent the individual data items and whose edges give the distances between them.

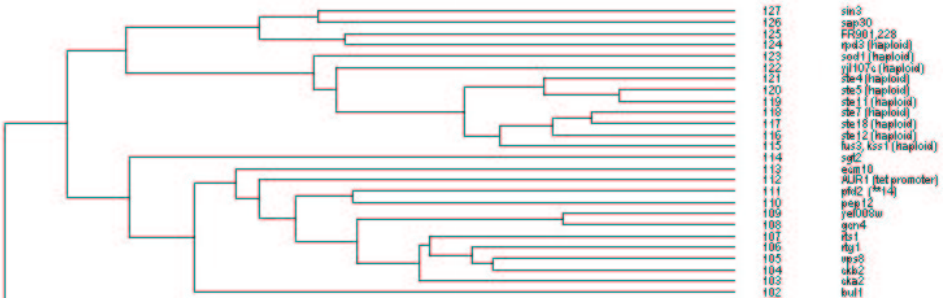


Figure 5: Part of the dendrogram generated by a hierarchical clustering algorithm.

- The classical hierarchical clustering methods never reconsider the choices taken in the past, which can result in highly suboptimal solutions.
- The applied linkage metric significantly affects the final clustering solution, as it incorporates assumptions about the structure of the data.[11]
- Hierarchical clustering methods based on linkage metrics do not scale well for large data sets, as they have an overall time complexity quadratic in the number of data elements.

Attempts to improve on hierarchical methods have mainly been concerned with the development of new cluster representations and splitting and merging criteria, as these crucially affect the algorithms' time performance and their ability to handle outliers or to detect clusters of various shapes.

The algorithm BIRCH ("Balanced Iterative Reducing and Clustering using Hierarchies", [82]) uses a two-level approach, which first compresses data in a hierarchical tree structure and, subsequently, applies a clustering algorithm to individual leaf nodes. While these features ensure scalability and make the algorithm particularly interesting for the use with large data sets, BIRCH performs badly for non-spherical and unbalanced clusters.

CURE ("Clustering using Representatives", [29]) combines ideas of graph and geometric linkage metrics. It describes clusters by a representative set of points (rather than by all points or just the cluster centre or medoid), which are chosen to be 'well distributed' and are 'shrunk' towards the cluster centre. An agglomerative clustering algorithm based on the single link metric is applied to these representative point sets. CURE has been extended for the use with categorical data, resulting in the algorithm ROCK ("Robust Clustering Algorithm for Categorical Data", [30]), which works on a derived sparse proximity graph (vertices are removed according to a user-specified

[11] For example, the single link metric only works well with clusters that are spatially well separated. On the other hand it is the only one of out the three linkage criteria introduced above that has the capacity to discover clusters of non-convex shapes.

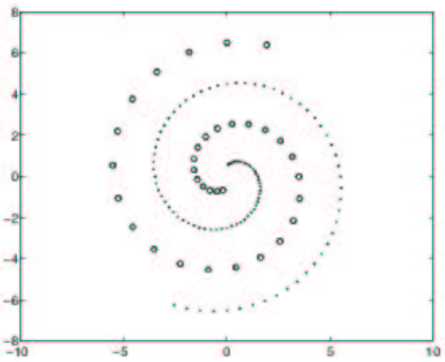


Figure 6: The *spiral* data set consisting of two intertwined spirals. This is one of the typical benchmark data sets used to illustrate an algorithm's capability to identify clusters of arbitrary shape.

threshold) and applies a distance measure based on shared neighbours (links) of data points. Both algorithms can detect non spherically-shaped clusters and are insensitive to outliers; however, their performance depends very much on the (user-defined) shrinking factor and the method of point selection.

Finally, CHAMELEON [42] also operates on a sparse proximity graph. Different to ROCK, vertex removal is based on a k-nearest neighbours strategy, and the merging criterion takes the differences between cluster intra and inter-connectivity into account. It uses a two-stage process: based on a graph-partitioning method a number of small clusters are initially constructed, which are then combined using an agglomerative approach. While CHAMELEON has been shown to be very effective in clustering, it cannot handle noise and requires several crucial user-specified parameters (including a scaling parameter for the linkage criterion and the number of desired nearest neighbours).

2.3 Density-based methods

CHAMELEON's merging criterion incorporates the notion of clusters as *connected components*, which is also one of the underlying principles of *density-based* clustering methods. These algorithms directly apply the concepts of *density*, *connectivity* and *boundary* to identify clusters in the input data. In this context, two main approaches exist: (1) Algorithms of *density-based connectivity*, which analyse the density and connectivity for individual data points. (2) Algorithms using *density functions*, which attempt to model the underlying data distribution. A particular strength of density-based algorithms is their ability to identify clusters of arbitrary shape.

The main representative of the first category is the algorithm DBSCAN ("Density Based Spatial Clustering of Applications with Noise", [21]). It uses the ϵ-*neighbourhoods* $NH_\epsilon(i)$ of a data item i and the set of *core objects* CO

$$NH_\epsilon(i) = \{j \in X |\ \delta(i,j) \leq \epsilon\}$$

$$CO = \{i \in X |\ |NH_\epsilon(i)| \geq MinPts\}$$

to define the notions of *density-reachability* $dr(i, co)$ of item i with respect to a core-object $co \in CO$, and the *density-connectivity* $dc(i, j)$ of two data items i and j

$$dr(i, co) \leftrightarrow \exists e_0..e_n \in X : e_i \in N_\epsilon(e_{i-1}) \wedge e_0 = co \wedge e_n = i$$

$$dc(i, j) \leftrightarrow \exists co \in CO : (dr(i, co) \wedge dr(j, co))$$

Here, $e_0..e_n$ is a 'chain' of data elements that connect data item i and core object co, and ϵ and $MinPts$ are user-defined parameters. All core objects are considered *internal points* of the cluster, while all those that are merely density-reachable make up the *cluster borders*. Data points that are not density-reachable from any core object are considered as *outliers*. The notion of neighbourhood can be easily extended as long as the *reflexivity* and *symmetry* of the neighbourhood relation is ensured.[12] Limitations of DBSCAN are:

- The effective computation of the ϵ-neighbourhood poses a problem for high-dimensional data (for low-dimensional data indexing schemes exist that allow an overall worst-case complexity of O(Nlog(N)).
- The parameters ϵ and $MinPts$ crucially determine the outcome of the clustering process. Also, if cluster densities vary, different values may be required across the same data set. This challenge is tackled in the algorithm OPTICS ("Ordering Points to Identify the Clustering Structure", [3]).

The second type of density-based clustering algorithm takes a quite different approach. The precursor in this category, the algorithm DENCLUE uses a *distance function*, which is the superposition of several *influence functions*, to model the density distribution:

$$f^L(i) = \sum_{j \in L} f(i, j)$$

[12] The reflexivity and symmetry of the neighbourhood relation are necessary in order to guarantee that $dc(x_1, x_2) \leftrightarrow dc(x_2, x_1)$. This ensures that the algorithm is independent of data ordering!

Here, L is usually restricted to a local neighbourhood around j (in order to make the approach computationally feasible) and $f(i,j)$ is a decreasing function of the distance $\delta(i,j)$. The algorithm uses hill-climbing techniques to identify local maxima of the density function, which are then interpreted as clusters. Its advantages include the handling of outliers and its time complexity, which scales sublinearly due to the additional integration of grid-based techniques.

2.4 Grid-based methods

Unlike the first three categories of clustering algorithms, the class of *grid-based methods* does not describe a set of algorithms with similar cluster concepts and a similar construction process, but rather a very versatile set of methods that can themselves be partitional, hierarchical or density-based and which, at some stage during the clustering process, employ a segmentation of the data-space.

A partitioning of the input space in hyperrectangles is particularly advantageous for the application to large data sets; additionally it potentially reduces noise and data-order dependency. Many partitioning methods implicitly use grid-based techniques as (possibly continuous) ranges of numerical values are discretised. Examples of hierarchical grid-based methods are the algorithms GRIDCLUST [70] and STING ("Statistical Information Grid-based Method", [80]). The two best-known grid-based algorithms are density-based: the algorithm CLIQUE ("Clustering in Quest", [1]) and its successor MAFIA ("Merging of Adaptive Finite Intervals", [58]).

A very interesting approach is taken in the algorithm *WaveCluster* [72], which employs methods from signal processing. In order to detect clusters, a Wavelet Transformation of the discretised data is followed by the application of a low-pass filter.[13]

[13] In the frequency domain cluster borders show as high frequencies; interior cluster points give rise to low frequencies with high amplitude values.

All things are difficult before they are easy.
(Fuller)

3 Topographic mapping

Like cluster analysis, methods of *topographic mapping* have their origin in the social sciences, where they have, for many decades, been applied to the visualisation and analysis of proximity data arising for example in sociology, psychology or archaeology.[14] By comparison, the use of map-like representations in computer science (and its fields of application) is a very recent development. Yet, a fundamental difference between the classical use and the domains tackled nowadays, is the enormous size of the data collections encountered in today's data-mining applications.

Visualisations in two and three dimensions are most familiar to humans and, for exactly this reason, are probably also the most intuitive. Most mapping-based approaches to visualisation have therefore focused on two- or three-dimensional representations, even though most of the methods applied for the mapping process itself can be generalised to arbitrary dimensions.[15]

We can distinguish between two main types of algorithms that can be applied to reduce the dimensionality of a data set. On the one hand, theoretically well-founded and mathematically exact algorithms that are usually based on the explicit computation of the mapping from data- to map-space. Unfortunately, these techniques do not necessarily generate the visually most effective representation and may often be computationally expensive. On the other hand, there are approximation methods that gradually improve an optimisation criterion (which does not always need to be explicitly known).

[14]Note that, in statistics, methods of topographic mapping are more commonly referred to by the term of *multidimensional scaling*. This term is however ambiguous, as it also refers to the actual class of multidimensional scaling algorithms (cf. Section 3.2) and, for the sake of clarity, we will therefore retain the term *topographic mapping* in the following.

[15]For some methods the dimensionality of the map-space might be restricted to be less or equal to that of data-space, as, for example, in the case of principal component analysis.

Even though these are usually prone to suboptimal solutions, they have been shown to perform well in many applications.

Apart from these distinctions on an algorithmic level, there are again basic differences as far as the algorithms' applications and results are concerned.

- **The types of attributes handled**
  Again, not all algorithms can handle data sets that are purely described by means of dissimilarity data, but many need an explicit data description by means of numerical vectors.

- **The level of topology-preservation**
  While some methods are predominantly concerned with the preservation of global relationships, others solely focus on the preservation of local neighbourhood-relations. For many applications combined approaches would be desirable.

In the following sections we attempt to give an overview of the best-known classes of methods for topographic mapping. First, *principal component analysis*, as a representative of linear projection methods, is introduced. Then, *multidimensional scaling* algorithms are surveyed and, finally *topology-preserving neural networks* are discussed.

3.1 Principal component analysis

The application of *principal component analysis* (PCA) to the task of dimensionality reduction is straightforward. Given a set of N data items, each described by a data vector of dimensionality D, these vectors are to be projected to an m-dimensional space with $m < \min(D, N)$, while maintaining a maximum of information. Towards this end, all data vectors are gathered in a data matrix I, with the rows corresponding to individual data vectors and the columns representing the values assumed by individual variables across the entire set of data. Next, the covariance matrix $Cov(I, I) = \overline{(I - \overline{I})(I - \overline{I})}$ is constructed and its eigenvectors and eigenvalues are computed. The first m *principal directions* (corresponding to orthogonal directions of maximal variance) are the eigenvectors associated with the m largest eigenvalues. Each data vector is projected to the space spanned by these m principal directions.

Thus, for a two-dimensional representation of a given data set, only the first two principal directions are selected. The two-dimensional vectors resulting from the projection of each data item onto these two directions directly reflect the new coordinates in the map-space. Unfortunately, the sole[16] use of PCA for the generation of topographic mappings does have several disadvantages.

[16] An alternative technique is to use PCA with only little dimensionality reduction (i.e.,

- The use of PCA is restricted to numerical data collections.
- In spite of the development of relatively efficient techniques for PCA (such as singular value decomposition (SVD) or Cholesky decomposition [63]), the time complexity remains $O(N^3)$.
- For large data sets the covariance matrix (which is of size $N \times N$) may be too large to be explicitly computed. Although neural network techniques for PCA exist, which may be applied in these cases,[17] these often have prohibitive runtimes.
- PCA only captures the structure of *linear manifolds* within the data. This limitation is overcome by non-linear projection methods, such as *Non-linear autoassociators* [69], *Kernel PCA* [71], *Projection pursuit* [23], or *Principal curves* [36].
- As PCA is not concerned with local structures in the data, it may entirely fail to reveal actual cluster structures. Again, non-linear projection methods can help in this context. The combination of several local linear PCAs also improves the modelling of the data and it has been studied by several authors (see e.g. [11, 17, 37]).

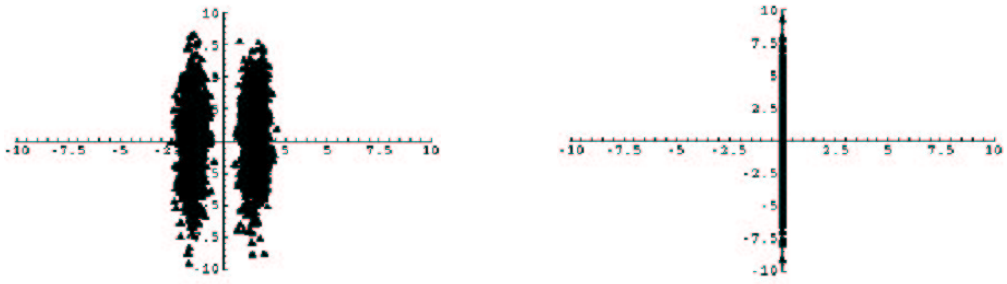
While the non-linear and local projection techniques mentioned above are better suited for the identification of cluster structures and of non-linear manifolds within the data, none of them is currently computationally as efficient as PCA. PCA therefore still remains one of the most commonly used techniques for dimensionality reduction. Additional advantages of PCA are its straightforward concepts, and the easy interpretability of the resulting projection space. For a more comprehensive survey of projection methods see [12].

3.2 Multidimensional scaling

The term multidimensional scaling (MDS, [81]) encompasses the traditional statistical techniques used to discover structure within a data set that is presented in the form of *proximity* data. That is, rather than dealing with data elements defined by points (vectors) in high-dimensional space, the

keeping most of the principal components) as a preprocessing stage to another visualisation technique. This has the major advantage that, due to the orthogonality of the coordinate system, variables within the space of principal components are uncorrelated. This property is, for example, exploited in Latent Semantic Indexing (LSI, [14]) in information retrieval.

[17] Neural networks (NNs) for PCA do not explicitly compute the covariance matrix. The memory consumption may therefore be limited to the storage of the input data and the principal components. Examples of NNs for PCA are *Autoassociators* [10], which learn the space of principal components by 'squeezing' data through a hidden layer of m units.



(a) (b)

Figure 7: A Principal Component Analysis reducing the data-space from 2D to 1D. The linear projection on the direction of maximum variance fails to reveal the cluster structure of the data.

individual items are merely described by their respective dissimilarities.[18] These so-called *disparities* do not necessarily have to be distances (in a metric space) in the mathematical sense, that is, they need not be positive, nor symmetric, nor fulfil the triangle inequality.

Starting from this proximity data, MDS constructs a low-dimensional representation of the data, while attempting to preserve relations between data items. The adequacy of the generated mapping is determined using an optimisation criterion, traditionally referred to as *stress* function. Various variations of the stress function exist and we review the two most popular ones in Section 5. In its general form its is given as

$$stress = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(f(\delta(i,j)) - d(i,j))^2}{scalefactor}}$$

where f is a monotonic function, N is the number of data items and, for each pair of data elements i and j, $\delta(i,j)$ is the associated disparity, and $d(i,j)$ is their distance in map-space.

There is the distinction between *metric* and *non-metric* multidimensional scaling, which refers to the type of proximity data being tackled (which, of course, determines what kind of optimisation criterion can be used). While metric multidimensional scaling can be applied to quantitative data (i.e., when absolute distances between data items are defined), non-metric multidimensional scaling tackles the problem of visualising qualitative data (i.e., when only similarity-orderings of the distances between data items are given).

[18] Of course every data set given in terms of high-dimensional data vectors can easily be transformed to proximity data by means of a distance function.

Dependent on the stress function, the low-dimensional representation can either be computed analytically (i.e., the mapping function is explicitly determined), or it has to be approximated through an iterative approach. The properties of the obtained embedding very much depend on the definition of stress that has been applied.

MDS has a number of properties that affect its suitability for topographic mapping.

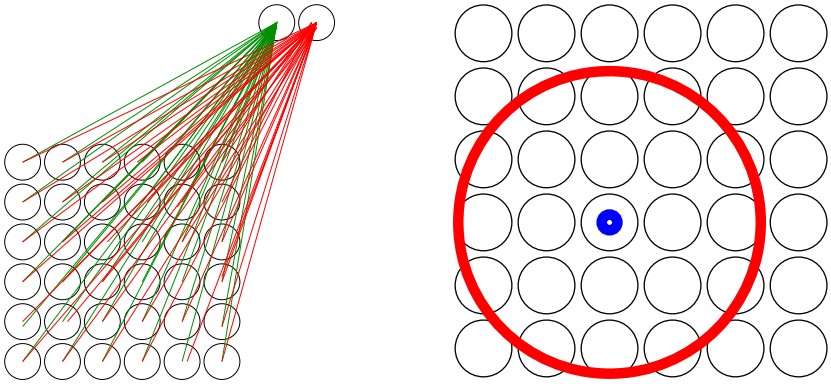
- Many stress measures tend to enforce the preservation of large distances while neglecting violations of neighbourhood relationships on a more local basis. This can prevent the identification of cluster structures.
- Results of the iterative optimisation approaches depend on the starting configuration and are likely to be suboptimal.
- Its time complexity is quadratic in the number of data items. Hence, while MDS is very fast for small data sets, its performance deteriorates quickly as the number of data elements increases.

Research efforts aimed at overcoming these weaknesses have mainly focused on amended stress measures, which use an additional scaling and weighting of the distances in order to improve the amount of structure perceivable within the generated mappings. Also, different initialisation and iteration schemes have been investigated.

3.3 Topology-preserving neural networks

The original and best-known example of topology-preserving neural networks are Kohonen's *Self-Organising Maps* (SOMs, [45]). 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 themselves arranged in the form of a rectangular (or possibly hexagonal) grid. If the grid (which can be one-, two- or even higher-dimensional) has the dimensionality of the desired map-space, the association of weight vectors with particular neurons (i.e., their respective coordinates on the grid) can be interpreted as a mapping of the high-dimensional data-space to this low-dimensional map-space.

When presented with new input data, a SOM traverses a three-stage process. First, all weight vectors are initialised. This can either be done randomly or with regard to the input data (e.g., by ensuring a uniform distribution of the weight vectors in the sector of data-space that is occupied by the input data). Next, during the *training phase*, individual input vectors $\vec{x}$ are successively presented to the network. For each of these vectors, the



(a)

Figure 8: (a) A two-dimensional self-organising map. In the example, the input data is also two-dimensional. Therefore, each of the map neurons is associated with a two-dimensional weight vector. The two neurons of the input layer are connected to all neurons of the map layer. (b) For each input data item the winning neuron is determined (here, marked by the blue dot). The weight vectors of all nearby neurons are then updated. For each neuron the update is a function of the difference between the input element and its own weight vector and its grid distance to the BMU.

best-matching unit (BMU, the neuron whose weight vector is most similar to the stimulus vector) is determined. The BMU and its local neighbours then[19] update each weight vector $\vec{w}$ according to the *learning rule*

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

Here, $h(\vec{x}, \sigma)$ is a *neighbourhood function* that is centred around the BMU $\vec{x}$ and whose spread is determined by the parameter σ (typically h will be of Gaussian or 'Bubble' shape). In order to ensure convergence, both the learning rate ϵ and the spread σ usually need to be decreased during the training process. Finally, once a SOM has been trained, it can be used for *classification*. When presented with new input data, each item will be mapped to its BMU on the grid.

One fundamental application of SOMs is their use for vector quantisation, as the main facets of the data are captured by the weight vectors. Besides, SOMs have certain (local) topology-preserving properties, a property that emerges from the update of neighbourhoods (rather than single neurons) during training. However, the topology-preservation obtained by SOMs is limited in some respects.

[19] An alternative for the learning process is to use *batch-training*, where the update of weight vectors is performed only after a sequence of input stimuli has been presented.

- While SOMs preserve local neighbourhood relations, they do not penalise false ones.
- Global relationships within the data are not necessarily captured.
- A further disadvantage of SOMs is the long training phase, which may be a drawback for some applications.

There has been extensive research on improvements related to the topology-preserving capacities of SOMs, and many extended versions have been developed. In particular, *dynamic models* have been introduced, which adapt their grid-topology during the learning process to better approximate the input data. Dynamic models comprise two sub-categories, *dimensionality reducing networks* on one hand and *topology constructing networks* on the other hand.

Examples for dimensionality reducing networks are the algorithms *GCS* ("Growing Cell Structures", [24]), *GSOM* ("Growing SOM", [5]) and *TS-SOM* ("Tree-Structure SOM", [46]). While all three algorithms adapt their grids by inserting additional neurons in crowded grid regions, only the GCS can additionally increase grid dimensionality. The TS-SOM is an example of a *hierarchical model*: it constructs a hierarchy of grid layers of different resolutions, resulting in a 'hyperpyramide', which renders searches for the BMU more effective.

Topology constructing networks modify the actual *topological connections* in map-space. An example is the algorithm *NeuralGas* [55] that uses a fixed number of neurons and constructs the connectivity matrix of these neurons during training only. *Growing Neural Gas* is an extension of this method, which can additionally introduce new neurons. The combination of these ideas with those of hierarchical models results in the algorithm *Split Net [64]*. For a detailed discussion of the different approaches we refer to [64].

Go to the ant, thou sluggard;
consider her ways, and be wise.
(The Holy Bible)

4 Ant-based clustering and sorting

4.1 Nature-inspired algorithms

For many years now, researchers in the field of computer science have used nature as an inspiration to devise new efficient and effective algorithms. In particular processes observed in biology and physics have served as paradigms for state-of-the art methods in a large range of different application areas. Examples are artificial neural networks for feature extraction (e.g. in image processing) and robotics control, artificial immune-systems with their application in network security, and important heuristic optimisation methods such as evolutionary algorithms and simulated annealing. A variety of algorithms inspired by natural swarm behaviour have also been introduced [9]. Their use includes applications like the simulation of flocking behaviour in computer graphics, however, the main focus is on their use for optimisation. In this respect the two new fields of particle swarm optimisation and *ant colony optimisation* have only recently emerged.

4.2 Ant colony optimisation

Ant-colony optimisation (ACO, [18]) is a relatively new metaheuristic,[20] which has been successfully applied to a range of NP-hard optimisation

[20] A metaheuristic is a set of concepts useful for defining heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general framework which can be applied to different optimisation problems with relatively few modifications to make them adapted to a specific problem. Examples of metaheuristics include simulated annealing, tabu search, iterated local search, evolutionary algorithms and ant colony optimisation. (Definition from the European Metaheuristics Network)

problems, including quadratic assignment, timetabling, scheduling and frequency assignment.

It is inspired by the foraging behaviour of ant colonies and their astonishing capability to solve shortest path problems. Like its natural paradigm, ACO is a distributed process. The basic algorithm employs multiple elementary agents and positive feedback mechanisms in order to solve complex optimisation tasks. It imitates the trail-laying-trail-following behaviour observed in real ants, where ants deposit chemical substances, called *pheromones*, in order to transmit information about their selected path. This indirect communication via the environment has been first reported by Grassé [28] and is commonly referred to as *stigmergy*. In ACO algorithms, artificial pheromones are introduced to serve as *stigmergic variables*. An extensive introduction to the field is given in [19].

4.3 Ant-based clustering and sorting

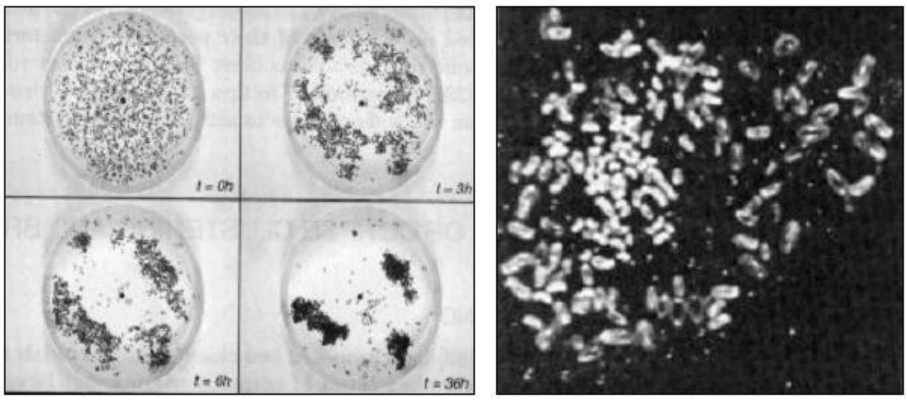
While ACO comprises a number of largely different implementations, it can itself be classed in the more general and broad category of *ant algorithms*, that is, algorithms that model "some behaviour" observed in real ants.[21] Ant-based clustering and sorting, the algorithm this project focuses on, also forms part of this category and indeed also works by positive feedback and local information processing. However, it differs from ACO in some fundamental respects.

- Ant-based clustering and sorting draws its inspiration from the clustering and sorting (not the foraging) behaviour observed in real ants.
- It is not a metaheuristic. In contrast, it tackles only the specific task of clustering and sorting.
- Unlike ACO it does not make use of artificial pheromones.
- In the basic algorithm no synergetic effect can be observed, that is, the algorithm's performance is mostly[22] independent of population size.

In the following, the basic principles of this particular ant algorithm will be described in detail, and previous research efforts related to the improvement, evaluation, comparison and application of the algorithm will be surveyed.

[21] Without doubt, however, ACO is currently the most successful and best-known type of ant algorithm.

[22] In order to ensure the function of the algorithm it is important that the number of ants is clearly inferior to the number of data items being sorted.



(a) (b)

Figure 9: Tasks of clustering and sorting in real ant colonies. (a) Clustering of corpses in *Pheidole pallidula*. (b) Sorting in *Leptothorax unifasciatus*.

4.4 Deneubourg's basic clustering model

Ant-based clustering and sorting was first introduced by Deneubourg et al. [16] in 1990. As its name implies, two types of natural ant behaviour are modelled by this algorithm. First, *clustering*, where ants gather items to form heaps. An example for this is the cemetery formation (i.e., the clustering of dead corpses) observed in the species of *Pheidole pallidula*. Second, *sorting*, where ants discriminate between different kinds of items and spatially arrange them according to their properties. This type of activity can, for example, be observed in nests of *Leptothorax unifasciatus* where larvae are arranged dependent on their size. In their paper, Deneubourg et al. proposed a continuous model to describe these behaviours and derived a Monte Carlo model, which was experimentally validated. The experiments reported were limited to the clustering of one or two types of data items and the main focus was on obtaining a model applicable to groups of robots.

In the Monte Carlo simulation ants are modelled by simple agents, which randomly move in their environment, a square grid with periodic boundary conditions. Data items that are scattered within this environment can be picked up, transported and dropped by the agents. The picking and dropping operations of each individual agent are biased by the probabilities

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

and

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

Here, $f(i)$ is an estimation of the fraction of data items in an ant's immediate environment that are similar to the data item the ant currently considers to pick up or drop respectively. In Deneubourg's algorithm this estimate is obtained using a short-term memory of each agent, where the contents of the last encountered grid cells are stored. This choice of the *neighbourhood function f(i)* was mainly motivated by the ease of its realisation for simple robots. The parameters k^+ and k^- determine the influence of the neighbourhood function $f(i)$, and were fixed to 0.1 and 0.3 respectively.

In the described model ants are thus likely to pick up data items that are either isolated or surrounded by dissimilar ones. On the other hand they tend to drop them in the vicinity of similar ones. In this way a clustering and sorting of the elements on the grid is obtained.

We can now see that the mechanism of positive feedback at work in this algorithm is quite different from the one used in ant colony optimisation. The ants do not employ artificial pheromones for indirect communication, but it is the distribution of data items itself that serves as stigmergic variable.

4.5 Behavioural studies

Several aspects of Deneubourg's model have been analysed by other researchers. Gutowitz introduced *complexity-seeking* ants that tend to concentrate their actions within regions of high interest, thus speeding up the clustering process [32]. Regions of high interest are determined using a local measure of complexity: cells in the grid can assume two possible states (occupied or empty) and the complexity measure simply counts the number of pairs of unequal neighbouring cells. Hence, complexity is 0 for regions that are either entirely occupied or entirely empty, while it is highest for a checkerboard pattern.

In [54], a *minimal model* for ant-based clustering was introduced. Here, deterministic pick up and deposition rules were used. The displacement rule was also modified in order to enable straighter movements of the agents, which improves time performance. Martin et al. also demonstrated that the algorithm's performance seems to be independent of the number of agents used.

4.6 Lumer and Faieta's extension to sorting

Shifting away from the use in robotics and towards the algorithm's application to data analysis, Lumer and Faeita introduced a number of modifications that enabled it to work with numerical data, and improved solution quality and the algorithm's convergence time.

- In Lumer and Faieta's work, data items are described by numerical vectors. The distance between these vectors is computed using the Euclidean distance. In general, the algorithm can be applied to all

those types of data, for which a measure of dissimilarity (or, equivalently, similarity[23]) between individual data items can be defined.

Using this kind of discrimination between data elements, the neighbourhood function is redefined[24] as

$$f(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j (1 - \frac{\delta(i,j)}{\alpha}) & \text{if } f(i) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

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 neighbourhood (typically, $\sigma^2 \in \{9, 25\}$). The agent is located in the centre of this neighbourhood; its *radius of perception* in each direction is therefore $\frac{\sigma-1}{2}$.

Hence, an agent deciding whether to manipulate an item i considers the average similarity of i to all elements j in its local neighbourhood of size σ^2. As $f(i)$ is computed as a sum over the entire neighbourhood, empty grid cells are indirectly penalised (as they prevent a positive contribution of this particular cell), such that a tight clustering, rather than just a loose sorting, is induced.

- Lumer and Faieta introduce the use of a *short-term memory* to keep track of the last transported data items and their respective dropping locations. After a picking operation, ants identify the remembered data item, which most closely matches the newly encountered data element, and, subsequently, steer towards its (assumed[25]) location.

- The use of *inhomogeneous populations* of agents, that is, agents with different individual parameter settings, is also an idea first proposed in [51]. This concept was limited to the use of a range of different *step-sizes* (the stepsize describes the number of grid cells an agent can cross in one random move), and an adaptation of the neighbourhood function dependent on this 'speed': through an additional scaling factor slow ants are biased to be more selective than quicker ants.

- In spite of the above modifications, the algorithm suffers from convergence problems. With an increasing number of data items, ant-based clustering and sorting often generates several small clusters of each data type, which are not merged within reasonable computation time.

[23] Using a normalisation to the interval $[0,1]$, the similarity $s(i,j)$ between two data items i and j can easily be transformed to a dissimilarity as $\delta(i,j) = 1.0 - s(i,j)$.

[24] Even though this hasn't been done in Lumer and Faieta's original paper, it is useful to assume that the dissimilarities are normalised to the interval $[0,1]$ in this definition.

[25] In a multi-agent system, the data item may have been removed by another ant in the meantime.

In order to overcome this difficulty, Lumer and Faeita introduced behavioural switches that permit the ants to switch from 'cluster construction' to 'cluster destruction' once the sorting process seems to stagnate.

In [51], results obtained on a simple artificial data set (400 data points sampled with equal probabilities from the four two-dimensional Gaussian normal distributions $(N(0,2), N(0,2))$, $(N(0,2), N(8,2))$, $(N(8,2), N(0,2))$ and $(N(8,2), N(8,2))$ were reported. While the authors indicated that comparable results had been obtained on various synthetic data sets, issues arising with more complex data (such as the determination of a suitable value for the data-dependent parameter α in Equation 3) were not addressed in the paper. Also, evaluation was limited to visual observation and measurements of *entropy* and *average fit* (cf. Section 5.2.4), which provide only very limited insights into the overall sorting quality.

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 neighbourhood relationships as much as possible", that is, as an approximate topographic mapping (as opposed to a pure clustering). However, the results presented in the paper do not suffice in any way to support this claim. The analytical measures used do not capture the preservation of inter-cluster relationships at all, and even intra-cluster sorting is only reflected to a very limited degree.

4.7 Kuntz, Layzell and Snyer's application to graph partitioning

Following Lumer and Faeita's work, a study of the algorithm's applicability to graph partitioning was presented in [47, 48, 49]. While the algorithmic model was adopted mainly unchanged, a dissimilarity measure for pairs of graph nodes had to be derived. For a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges E, the distance between two vertices v_i and v_j was defined as

$$d(v_i, v_j) = \frac{|\rho(v_i) \Delta \rho(v_j)|}{|\rho(v_i)| + |\rho(v_j)|}$$

where Δ indicates the symmetric difference and

$$\rho(v_i) = \{v_j \in V \mid \{v_i, v_j\} \in E\} \cup \{v_i\}$$

Under this measure, vertices that have many shared and few distinct neighbours have a small dissimilarity.

Experiments were performed on a class of fairly small pseudo-random graphs (at most 900 nodes, with most analytical results reported for sizes

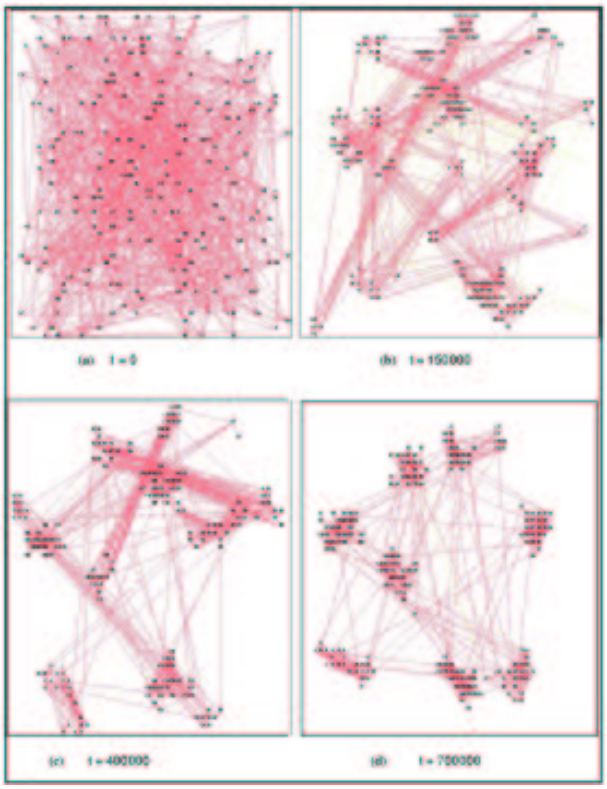


Figure 10: Graph partitioning using ant-based clustering and sorting. Initially, all graph nodes are randomly scattered on the two-dimensional grid. Subsequently, the ants form clusters of closely related vertices.

of 200 and 400 respectively). Apart from the entropy measure, a number of additional performance measures were used in [49]. The clustering capabilities of the algorithm were evaluated using the percentage of misclassified vertices and the percentage of inter-cluster edges (using an upper bound delivered by the classical Fidduccia-Mattheyses procedure for graph partitioning [22]). For this purpose, the 'intuitively' visible clusters had to be made explicit, which was done by running a K-means algorithm on the grid positions of all data elements.

The Pearson correlation (cf. Section 5.2.2) of the dissimilarities between pairs of data items in data space and their respective spatial distances on the grid was computed in order to support the claims made on the ants' capability to generate an isometric embedding. For a graph of 200 nodes a convergence of the Pearson correlation towards 0.65 was reported, and it was concluded that the gap towards a 'perfect' correlation of 1.0 was due to the gathering of "vertices with a small dissimilarity". In Section 7.8 we will see in detail why these claims have to be considered with much care.

Comparisons to alternative embedding algorithms were limited to very few results for a metric multidimensional scaling approach. In [48] plots on the deviations from the 'optimal stress value' (as obtained by multidimensional scaling) were given. The ant-based heuristic consistently performed worse, which the authors explained by the "local optimisation approach" due to which "large values may be modified in the representation". Now, the limited deviation in stress was taken as an affirmation that "a certain distance preserving embedding is guaranteed for small distance values", which was however not further evaluated. In spite of the inferior results in terms of metric stress, the authors saw the algorithm's advantage in its "quasi-linear complexity that allows this ant-based heuristic to handle very large graphs that can hardly be processed by other algorithms". While the claim on "quasi-linearity" was supported by graphs on the heuristic's mean convergence time (again limited to graphs of up to 900 nodes), no results on the relative time performance of alternative approaches were given.

4.8 Applications to the categorisation and visualisation of document collections

Building upon this work on the algorithm's applicability to topographic mapping, several authors have investigated the use of ant-based clustering and sorting for the two-dimensional visualisation of document collections in the form of *topic-maps*.

In [38], the basic algorithm (without Lumer and Faieta's extensions) was adopted unchanged, and results on a small selection of web pages (84 pages) were demonstrated. Analytical results were provided only for one single run of the algorithm: values of 0.63 and 0.56 were obtained for the measures of cluster entropy and purity (cf. Section 5.1.1) respectively. While these values seem to be rather low, it is difficult to judge about the degree of difficulty of the used data set, as no comparative results for other methods were given.

Ramos and Merelo [65] used a modified version of the algorithm to visualise a collection of 931 words taken from Spanish newspapers. However, the main focus of this work was on the algorithmic changes introduced and the demonstration that results comparable to the Lumer and Faeita version of the algorithm could be obtained. In the paper, this was demonstrated using one artificial test set, visual observation and the entropy measure. Results provided for the actual text data were very limited: merely one visual mapping result was shown, which, as the authors readily acknowledged, was far from being optimal.

In [34], Handl and Meyer used a combination of multidimensional scaling and ant-based clustering to dynamically generate topic maps representing the results of Internet-queries. In the course of this work, a number of modifications were introduced in order to meet the time constraints of online-

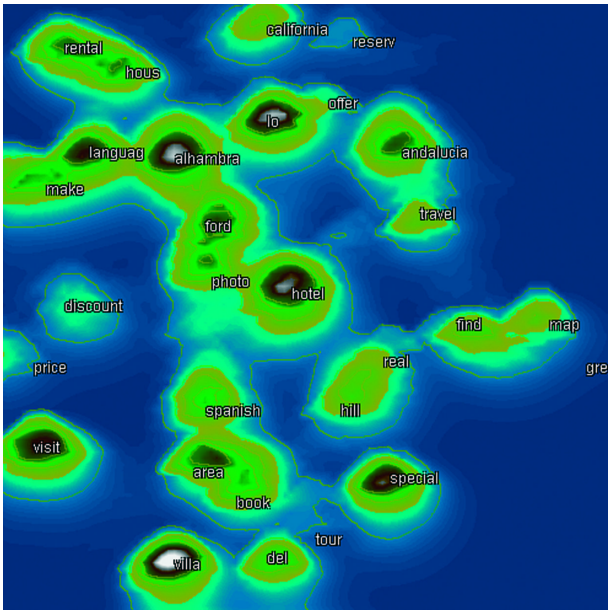


Figure 11: An ant-generated topic map of 1000 Internet documents retrieved for the query "Granada".

queries and to deal with arbitrary data collections. These changes included the following.

- **Eager ants**
  In the original algorithm, ants only contribute to the sorting progress when carrying data elements. These idle phases can easily be avoided by the introduction of ants that directly carry out a picking operation after each dropping operation. An additional index structure was used to keep track of data items on the grid.

- **Jumps**
  Through the use of very large step sizes, the algorithm's runtime can be significantly reduced, and its performance in terms of quality equally improves, as the dissolution of preliminary small clusters is facilitated.

- **Adaptive scaling**
  A first scheme to automatically adapt the parameter α for a given data set was introduced.

- **Stagnation control**
  Failure counters were used to avoid the blocking of ants by outliers in the data (which the ants have difficulties to dispose of). After a certain number of unsuccessful dropping attempts with the same data element, ants drop it deterministically.

In [35], Pearson correlation, inter-cluster correlation and intra-cluster correlation (cf. Section 5.2.2) were used to evaluate the algorithm's performance on artificial and real test data. In particular the results obtained in terms of the inter-cluster correlation raised first doubts on the suitability of pure ant-based clustering and sorting for tasks of topographic mapping, as no clear correlation between the relative distances of clusters in data-space and in map-space could be observed. The algorithm's hybridisation with multidimensional scaling was an attempt to improve the seemingly random positioning of clusters.

4.9 Monmarché's application to numerical data analysis

While the work summarised in the previous sections mostly dealt with the algorithm's assumed capability to combine clustering and topographic mapping, ant-based clustering and sorting has also been applied to pure clustering tasks.

In particular, Monmarché [57] introduced ANTCLASS, a hybridised version of the algorithm that uses one or more alternating sweeps of the ant algorithm and K-means. The combination of these two algorithms is an attempt to counter-balance the advantages and disadvantages of both approaches.

- The ant algorithm provides an estimate of the correct number of clusters K to K-means, thus sparing the automatic or manual determination of an appropriate parameter.
- K-means is initialised with the centres of the clusters generated by the ant-algorithm, which reduces its initialisation problem.
- The generally 'noisy' results of the ant algorithm can be improved by the subsequent application of K-means.

In order to accommodate for the 'pure' clustering task, Monmarché fundamentally modified the algorithm. Instead of manipulating individual data items solely, ants now operate on piles of data items. Entire piles can be transported or, alternatively, one or several data items can be added to a pile or be removed from it.

Naturally, this requires an adaptation of the biases for picking and dropping probabilities, which no longer depend on the neighbourhood function defined in Equation 3, but which take dissimilarities both between piles and within individual piles into account.

Experimental results for ANTCLASS were presented for eight artificial data sets of sizes up to 1100 data items and six real data sets adopted from [8]. The performance was evaluated using the number of clusters detected and the classification error (which is in fact just the negated Rand index,

cf. Section 5.1.1) and compared to that of 10-means and 100-means (i.e., K-means with K set to 10 and 100 respectively for all test sets), and an idealised K-means (initialised with the correct clustering solution). Not surprisingly, ANTCLASS consistently performed better than 10-means and 100-means and worse than the idealised K-means. Our main criticism is this respect is that a comparison to a practically usable K-means may have been more informative as to ANTCLASS's relative performance.

4.10 Parallelisation

Likewise to other ant algorithms, the *implicit parallelism* of ant-based clustering and sorting has often been underlined in the literature. However, so far only one actual parallel implementation has been reported [2], which uses a partitioning of the grid. Unfortunately, no results on the used number of partitions and the obtained speed-up were provided.

Interestingly, no parallelisation scheme based on the assignment of agents to individual processors has so far been reported, even though this is the type of parallelisation essentially suggested by the notion of implicit parallelism. However, in the case of ant-based clustering and sorting such an approach is likely to suffer from an impractically large communication overhead.

In fact, recent results in the field of ant colony optimisation show that, even on a shared memory architecture, the use of individual processors for single ants greatly suffers from the required synchronisation overhead [15]. In ACO, more promising results are obtained by multi-colony approaches, which assign 'ant colonies' to individual processors that cooperate by occasional information exchange only, thus reducing communication and synchronisation costs [56, 74].

What the hell is quality?
(Pirsig)

5 Analytical Evaluation Measures

In Section 2 and 3 we have seen that a variety of different techniques for cluster analysis and topographic mapping exist, many of which are based on quite different considerations as to the conceptual idea of the *quality* of a solution. However, there is a certain agreement on some properties a good clustering and/or mapping solution should possess, which have lead to the introduction of a number of quantitative evaluation measures. The most important of these are revised in the section on hand.

5.1 Measures for cluster analysis

5.1.1 External measures

The first group of analytical evaluation functions available for cluster analysis are those devised for benchmark problems in which the right number of clusters and the correct classification for each data item is known. Evaluation becomes far more straight forward in these cases, as the desired properties of a partitioning (which are, up to a certain degree, a matter of the problem definition) can be neglected, and we can merely focus on the validity of the obtained cluster assignments.

- **Number of clusters**
  Naturally, when the real structure of the data is known, the number of determined clusters is a first indicator of an algorithm's performance. It is particularly important to keep track of this figure, as the identification of too many or too few clusters crucially affects other analytical performance measures.[26]

[26] For example, the performance both in terms of the purity and the intra-cluster variance improves, if too many small clusters are determined.

- **Measures based on the distribution of the known class labels**
  The next three measures we discuss all *directly* apply the knowledge of the class labels, in that they are concerned with the purity and/or the completeness of the generated clusters with respect to the true class memberships.

  The *purity* of a cluster $C_k \in C$ is defined as the percentage of the predominant data type according to the known real class labels $t \in T$, that is,

  $$Pur(C_k) = \max_{t \in T} \frac{N_{tk}}{N_k}$$

  where N_k is the size of the cluster C_k, and N_{tk} is the number of elements of class t within this cluster. The purity $P(C)$ of an entire partitioning is then computed as the mean purity of all clusters. It is limited to the range $]0, 1]$ and it is wanted to be maximised.

  Also, the relative degree of randomness of a partitioning can be evaluated using the notion of *cluster entropy*. It is a more comprehensive measure than the purity, as it takes the distribution of all classes within each cluster into account. The entropy of a cluster is

  $$Entr(C_k) = -\frac{1}{log(N)} \sum_{t \in T} \frac{N_{tk}}{N_k} \; log(\frac{N_{tk}}{N_k})$$

  and, again, the overall entropy $E(C)$ is computed by averaging over the set of all clusters. The cluster entropy is limited to the range $[0, 1]$.[27] It is wanted to be minimised.

  Finally, the *F-measure* [76] adopts the ideas of precision and recall from information retrieval. Each *class* t (inherent to the data) is regarded as a the set of N_t items desired for a query; each *cluster* C_k (generated by the algorithm) is regarded as the set of N_k items retrieved for a query; N_{tk} gives the number of elements of class t within cluster C_k. For each class t and cluster C_k precision and recall are then defined as $Prec(t, C_k) = \frac{N_{tk}}{N_k}$ and $Rec(t, C_k) = \frac{N_{tk}}{N_t}$, and the corresponding value under the F-measure is

  $$Fmeas(t, C_k) = \frac{(b^2 + 1) \cdot Prec(t, C_k) \cdot Rec(t, C_k)}{b^2 \cdot Prec(t, C_k) + Rec(t, C_k)}$$

  where equal weighting for $Prec(t, C_k)$ and $Rec(t, C_k)$ is obtained if $b = 1$. The overall F-value for the partitioning is computed as

  $$F(C) = \sum_{t \in T} \frac{N_t}{N} \max_{C_k \in C} (Fmeas(t, C_k))$$

[27] A value of 1 reflects a uniform mixture of classes within each cluster, a value of 0 indicates the sole presence of pure clusters.

It is limited to the interval $[0, 1]$ and should be maximised.

- **Indices for the comparison between two partitionings**
  Other methods that are commonly applied to compare the obtained clustering result to the known cluster structure are indices based on statistics of the pairwise relative cluster assignments, which can also generally be applied to evaluate the compliance between two partitionings of the same data set.

  Given the partitionings U and V, the quantities a, b, c and d are computed for all possible pairs of data points i and j and their respective cluster assignments $c_U(i), c_U(j), c_V(i)$ and $c_V(j)$, where

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

  Hence, a and d keep track of correspondences between the two partitionings, whereas b and c count clear deviations. The most well-known index is the Rand index [66], which is defined as

$$R(U, V) = \frac{a + d}{a + b + c + d}$$

  Obviously, R is limited to the interval $[0, 1]$. A value of 1 is only obtained for a perfect correspondence between the cluster assignments of the partitionings U and V, and smaller values show an increasing discrepancy between the two solutions.

  A number of variations of this index exist, such as the adjusted Rand index [40], which introduces a statistically induced normalisation in order to yield values close to 0 for random partitionings, or the Jacard Coefficient [41], which is given as $J = \frac{a}{a+b+c}$, thus applying a stricter definition of correspondence.

5.1.2 Internal measures

If data sets are tackled whose real structure is unknown, the evaluation of the clustering results becomes much more involved. Measures that can be applied in these cases try to capture the two objectives of cluster analysis that were informally outlined in Section 1.1: the minimisation of intra-cluster distances (resulting in compact clusters) and the maximisation of inter-cluster distances (resulting in well-separated clusters). Additionally,

the size of the individual clusters might be taken into account, in order to favour well-balanced solutions.[28]

- **Intra-cluster variance [50]**
  The popular *intra-cluster variance* or *sum-of-squared-errors minimum variance criterion*, the measure locally optimised by the K-means algorithm, is based on the first of the above concepts, the minimisation of intra-cluster distances. It is given as

  $$V(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, \mu_k)^2$$

  where C is the set of all clusters, μ_k is the centroid of cluster C_k and $\delta(.,.)$ is the distance function employed to compute the deviation between the data item i and its advised cluster centre.

  The lower bound of the obtainable variance depends on the data and the number of clusters used, but can at best be equal to zero.

- **Combined approaches**
  More enhanced approaches combine measurements on intra-cluster and inter-cluster distances, which is either done linearly or non-linearly.

  An example for a linear combination is the SD-validity index [33], which is given as

  $$SD(C) = \alpha Scat(C) + Dis(C)$$

  Here, α is a weighting term, which trades off the relative importance of the *average scattering for clusters* $Scat(C)$ and the *total separation between clusters* $Dis(C)$, where

  $$Scat(C) = \frac{1}{K} \sum_{C_k \in C} N_k \frac{|\sigma(C_k)|}{|\sigma(C)|}$$

  and

  $$Dis(C) = \frac{D_{max}}{D_{min}} \sum_{C_k \in C} \left(\sum_{C_l \in C} \delta(\mu_k, \mu_l) \right)^{-1}$$

  $\frac{D_{max}}{D_{min}}$ denotes the ratio between the maximum and the minimum inter-cluster distance found in C, $\sigma(C)$ is the standard deviation of data items from their assigned cluster centres, $\sigma(C_k)$ is the standard deviation computed for an individual cluster C_k, and N_k is the number of data items assigned to cluster C_k.

[28] If the size of the clusters is not considered, trivial solutions for the maximisation of inter-cluster distances and the minimisation of intra-cluster distances may exist. This becomes an issue, if the evaluation measure is directly used as an optimisation criterion.

The alternative, a non-linear combination, is, for example, used in *Dunn and Dunn-like indices* [20, 61]. They measure the ratio between cluster distances and cluster diameters, which should be large in a good clustering solution. The Dunn index is defined as

$$DU(C) = \min_{C_k \in C} \left(\min_{C_l \in C} \left(\frac{\delta(\mu_k, \mu_l)}{\max_{C_m \in C} diam(e)} \right) \right)$$

Here, the diameter $diam(C_m)$ of a cluster C_m is simply computed as the maximum intra-cluster distance. *Dunn-like indices* use a more enhanced scheme for the computation of these diameters, which makes the approach more robust to the presence of noise.

Note, that all measures introduced in this section can not only be used for evaluation purposes, but also as explicit optimisation criterion in a clustering algorithm. Additionally, they find application for the semi-automatic determination of the number of clusters that best describe a given data set. For a comprehensive overview of other indices and clustering validation techniques, the reader is referred to the comprehensive survey in [33].

5.2 Measures for topographic mappings

In light of the essential differences between the mapping techniques introduced in Section 3 (not so much in terms of algorithmic implementation, but mainly in terms of their specific understanding of topology-preservation), it becomes clear that the quantitative evaluation of a topographic mapping is a complex undertaking. Before surveying evaluation functions that have been introduced towards this goal in the literature, we will therefore revise the different demands that can be made with respect to topology-preservation.

5.2.1 Categorisation of topographic mappings

The different choices for the definition of a perfectly neighbourhood-preserving mapping can be grouped into three main categories [25, 26, 27].

- **Preservation of similarities**
  This is the strongest demand that can be imposed on a topographic embedding. It requires distances in data-space to be exactly matched by those in map-space. In a plot of the distances in data-space versus those in map-space, the resulting figure would be a straight line at a 45 degree angle.

- **Perfect correlation of similarities**
  This criterion is slightly weaker than the one above. Distances in data-space and map-space are required to coincide only up to a multiplicative and/or additive constant. The corresponding plot would be a line at an arbitrary angle and offset.

- **Preservation of similarity orderings**
  In this category no constraints on the relationships between absolute distances are imposed. Instead, a ranking of pairwise distances in both spaces (according to their size) is established and the obtained ranks are required to match. In a plot the preservation of similarity orderings shows in the distances in data- and map-space being related by any monotonically increasing function.

If the intrinsic dimensionalities of data-space and map-space differ, none of these criteria can usually be perfectly satisfied. Analytical evaluation measures are therefore necessary, which capture the degree of discrepancy from a perfect topographic mapping.

While, at first sight, a natural choice of categorisation for these measures seems to be the type of neighbourhood-preservation obtained by each one of them, we will see in the following that most of them only partially comply with the above definitions. A large number of evaluation functions additionally take considerations related to the relative importance of local neighbourhood structures into account. For a categorisation of the different measures we therefore distinguish between those that account for the preservation of relationships on various scales and those that merely focus on immediate neighbourhood relations.

5.2.2 Measures accounting for topology-preservation on various scales

The measures that deal with topology-preservation both globally and locally, do so with different trade-offs, as they are based on different concepts of topology-preservation.

- **Stress-based evaluation**
  The *stress* [81] function used in metric multidimensional scaling directly captures the discrepancy towards the perfect preservation of absolute distances. Given the mapping Θ that assigns a position in map-space to each data item, the pairwise dissimilarities $\delta(i,j)$ between elements i and j in data-space, and the corresponding pairwise distances $d(i,j)$ in map-space, the traditional stress function is defined as

$$S(\Theta) = \sum_{i=1}^{N} \sum_{j<i} (\delta(i,j) - d(i,j))^2$$

  that is, minimising stress corresponds to minimising the sum of squares of the deviations between the distances in data-space and the distances in map-space. The measure yields 0 for the perfect preservation of the absolute distances.

Variations of this stress measure introduce an additional weighting term that takes the scale of the dissimilarities in either data- or map-space into account. This way, for example, deviations in the mapping of small dissimilarities can be penalised more heavily than deviations in the mapping of large dissimilarities. One example of such a stress function is the *Sammon measure* [68], which is defined as

$$SM(\Theta) = \frac{1}{\sum_{i=1}^{N}\sum_{j<i}\delta(i,j)} \sum_{i=1}^{N}\sum_{j<i} \frac{(\delta(i,j) - d(i,j))^2}{\delta(i,j)}$$

- **Correlation-based evaluation**
  The *Pearson correlation coefficient* [63] 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 and those in map-space. The Pearson correlation is computed as

$$C(\Theta) = \frac{sd\delta - \frac{2 \cdot sd \cdot s\delta}{N(N-1)}}{\sqrt{sdd - \frac{2 \cdot sd^2}{N(N-1)})(s\delta\delta - \frac{2 \cdot s\delta^2}{N(N-1)})}}$$

  where

$$sd = \sum_{i=1}^{N}\sum_{j<i} d(i,j)$$

$$s\delta = \sum_{i=1}^{N}\sum_{j<i} \delta(i,j)$$

$$sdd = \sum_{i=1}^{N}\sum_{j<i} d(i,j)^2$$

$$s\delta\delta = \sum_{i=1}^{N}\sum_{j<i} \delta(i,j)^2$$

$$sd\delta = \sum_{i=1}^{N}\sum_{j<i} d(i,j)\delta(i,j)$$

  $C(\Theta)$ 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 correlation. A useful property of the Pearson correlation is its invariance under the scaling of the dissimilarities by a constant factor. Its drawbacks are, however, that it is strongly affected by outliers and heteroscedastic data.

If we are purely interested in the preservation of similarity orderings, the Pearson correlation can be applied to the ranks of the pairwise dissimilarities instead of the absolute distances. This results in the so-called *Spearman rank correlation* [63].

When the real cluster distribution is known, the correlation of the respective distances between cluster centres 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.*

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.*

- **Evaluation based on nearest-neighbour-lists**
  A number of very recent approaches towards the evaluation of topology-preservation all work with nearest-neighbour-list, that is, they are purely concerned with the violations of similarity orderings.

  The *topographic product* [4] has been explicitly designed to capture neighbourhood violations on a global scale and to be largely insensitive to local distortions of the map. It requires the computation of kth-nearest-neighbour-lists in both data-space and map-space. For each data item i, its lth nearest neighbour in data-space is denoted as $n_i(l)$ and its lth nearest neighbour in map-space is denoted as $\eta_i(l)$. The quantity

$$P_i(\Theta, k) = \left(\prod_{l=1}^{k} \frac{\delta(i, n_i(l))d(i, n_i(l))}{\delta(i, \eta_i(l))d(i, \eta_i(l))}\right)^{\frac{1}{2k}}$$

  then provides information about the preservation of the neighbourhood relations of data item i, while limiting the sensitivity to permutations of similarity orderings to the level k (i.e., for each data item only deviations in the neighbourhood-relation with data items that are NOT within the set of the kth nearest neighbours are taken into account). The deviation from the value of 1 then reflects the discrepancy towards an ideal mapping. Finally, in order to obtain an indication of the map's overall quality, the average value of $log(P_3(i,k))$ for all data items i and all degrees of locality k is computed, resulting in the topographic product

$$TP(\Theta) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{k=1}^{N-1} log(P_i(\Theta, k))$$

which is 0 for perfect preservation of similarity orderings.

All measures of map quality that have been introduced above are purely based on the comparison of distances or distance-orderings in data- and in map-space. None of them takes the shape of the mapped data manifold into account, which makes them prone to fail in the case of non-linear data manifolds. In Villmann et al. [78] the *topographic function* [78] is therefore introduced. It uses a definition of topology-preservation that imposes the preservation of immediate-neighbour relations in both mapping directions from data- to map-space and the inverse. While neighbours in map-space are given by the grid structure, Delaunay triangulation is used to to determine neighbours in data-space. The topographic function of the mapping Θ between data-space and map-space is then given as

$$TF(\Theta, k) = \begin{cases} \frac{1}{N} \sum_{i=1}^{N} V_i(\Theta, k) & \text{if } k \neq 0 \\ TF(\Theta, 1) + TF(\Theta, -1) & \text{if } k = 0 \end{cases}$$

where $V_i(\Theta, k)$ counts the neighbourhood-violations in both mapping directions and is defined as

$$V_i(\Theta, k) = \begin{cases} |\{j \mid d_{\max}(i,j) > k \wedge \delta(i,j) = 1\}| & \text{if } k > 0 \\ |\{j \mid d(i,j) = 1 \wedge \delta(i,j) > |k|\}| & \text{if } k < 0 \end{cases}$$

where the distance computation in map-space is based on the maximum norm $d_{\max}(i,j)$ for $k > 0$, and the Euclidean distance $d(i,j)$ for $k < 0$. The topographic function is constant zero only if the mapping is perfectly topology-preserving with respect to Villmann et al.'s definition. Otherwise its distribution provides information about the level at which violations of neighbourhood-relations occur.

5.2.3 Measures accounting for local topology-preservation

A number of measures also exist that are purely concerned with local topology-preservation. These have primarily been introduced for the evaluation of the performance of different types of self-organising maps.

- **Topological defect [64]**
  The *topological defect* is a restriction of the topographic function, that only takes the preservation of immediate neighbourhood relations in

the mapping direction from data-space to map-space into account (hence, 'false' neighbours are not penalised).

It is defined as

$$TD(\Theta, k) = \sum_{i=1}^{N} \sum_{l \geq k} S_i(\Theta, l)$$

where $S_i(\Theta, l)$ counts the number of data items, which are immediate neighbours of i in data-space and l steps apart in map-space

$$S_i(l) = |\{j \mid d(i,j) = l \wedge \delta(i,j) = 1\}|$$

The largest k for which $TD(\Theta, k) > 0$ corresponds to the largest topological defect in the output space.

- **Topographic error [44]**
  The topographic error also captures topological defects on a local scale. It has been first introduced for the analysis of self-organising maps and it is defined as

$$TE(\Theta) = \frac{1}{N} \sum_{i=1}^{N} U_i(\Theta)$$

  where $U_i(\Theta)$ indicates discontinuities in the mapping from data- to map-space.

$$U_i(\Theta) = \begin{cases} 1 & \text{best and second best matching unit for } i \text{ are not adjacent} \\ 0 & \text{otherwise} \end{cases}$$

- **Minimal distortion [52]**
  The principle of *minimal distortion* reflects the error introduced to the data through a successive mapping from data. to map-space and back to data-space.

  In the context of self-organising maps, this is simply the computation of the average distance between each data vector and its best matching unit, also referred to as *quantisation error*.

5.2.4 Additional measures introduced for ant-based methods

In this section we will finally introduce the measures of *entropy*, *mean fit* and *mean dissimilarity*, which are the ones most commonly used in the literature on ant-based clustering and sorting. All of these three measures are very limited in scope and they have not been used in the general data-mining literature.

Given a spatial distribution of data items, a measure of *entropy* [51] can be defined that reflects the degree of clustering on the grid. For this means,

the grid is divided into a set R of evenly sized square regions and the number of data items N_r within each region $r \in R$ is determined. The entropy of the entire grid is then given by

$$E(\Theta) = -\sum_{r \in R} N_r * log(\frac{N_r}{N})$$

where N is the total number of data items on the grid. Using different resolutions of grid regions, this measure permits to perceive the emergence of clusters (of different scales) on the grid. However, as no information about the similarity of the clustered data items is provided, the measure by itself is not very expressive in the context of either cluster analysis or topographic mapping.

The neighbourhood function $f(i)$ can be used to compute the *average local fit* [51] of the spatial distribution of the data elements on the grid. For each item i, the neighbourhood function $f(i)$ is evaluated resulting in the formula

$$F(\Theta) = \frac{1}{N} \sum_{i=0}^{N} f(i)$$

where N is the total number data items on the grid. Recall that the neighbourhood function is defined as

$$f(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j (1 - \frac{\delta(i,j)}{\alpha}) & \text{if } f(i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

The mean fit takes values in the interval $[0, 1]$ and, during the sorting process, it is to be maximised. While this evaluation function reflects the local sorting on the grid to some degree, it is hard to interpret for two reasons.

- The obtainable range of values depends on the parameter α, which varies for different data distributions. This makes it hard to summarise evaluations across different test sets.
- The neighbourhood function takes both the similarity and the density of the local neighbourhood into account. While this is a necessary feature of the neighbourhood function for the clustering process, it is not entirely useful for evaluation purposes. It makes it hard to infer the actual average similarities observed within the neighbourhood, that is, to distinguish between low mean fit values due to (1) a high density with low similarity values and (2) a low density with high similarity values (cf. Figure 12).

These difficulties can be partly overcome by summing over the modified neighbourhood function

$$k(i) = \frac{1}{N_{occ}} \sum_j \delta(i, j)$$

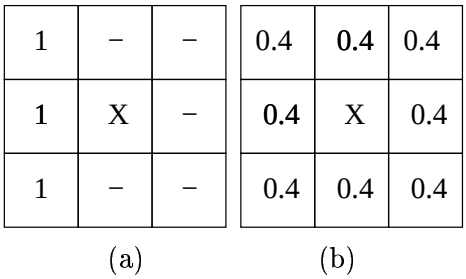


Figure 12: Two distributions of similarities. 'X' signifies the reference data element (i.e., the current ant position), '-' signifies an empty grid cell. Under the measure of mean fit both distributions take the value of 0.4. The measure of dissimilarity, in contrast, gives a dissimilarity of 0 for (a) and 0.6 for (b), which is more appropriate.

where N_{occ} is the number of actually occupied grid cells within the local neighbourhood of data item i on the grid. The resulting measure is that of *mean dissimilarity* [47].

Parvula (nam exemplo est) magni formica
Ore trahit, quodcunque potest, atque addit
Quem struit; hand ignara ac non incauta.
(Horacius)

6 Modifications to ant-based clustering and sorting

In Section 4 an overview of previous research on ant-based clustering and sorting has been given. In sight of the evaluations and comparisons performed for the algorithm so far, one is inclined to ask the following questions.

(1) Why have evaluation measures for cluster analysis found so little application?

Certainly not for a lack of measures, when, as we have seen in the previous section, the data-mining literature provides a large pool of evaluation functions. The main problem is one related to the nature of the algorithm's outcome: it does not generate an explicit partitioning but a spatial distribution of the data elements. While this may contain clusters 'obvious' to the human observer, an evaluation of the clustering performance requires the retrieval of these clusters, and it is not trivial to do this without human interaction (e.g., 'marking' of the clusters) and without distorting the resulting partitioning (e.g., by assuming that the correct number of clusters has been identified).

(2) Why has evaluation been limited to such a small range of benchmark data?

This is, at least partly, due to the difficulty of devising appropriate parameter settings for ant-based clustering and sorting on different types of data, which

makes its application to new data sets rather tedious.

A rigorous evaluation of the performance of ant-based clustering and sorting requires a solution to both of the above problems. In this section, we address these issues in order to prepare the ground for the comparative study presented in Section 7.

We start by investigating a number of modifications to the algorithm and their potential to improve the algorithm's performance. In this context, we pay particular attention to two properties of the spatial distribution generated by the ant algorithm: the compactness of individual clusters and the spatial separation between them. Both of these are crucial for the unambiguous interpretation of the resulting partitioning, and for the working of the scheme of cluster retrieval that we present subsequently. Finally, we describe how suitable parameter settings can generally be derived from the data and present a new scheme of self-adaptation for the scaling parameter α.

6.0.5 Basics

The basic ant algorithm (see Algorithm 1) starts with an initialisation phase, in which (i) all data items are randomly scattered on the toroidal grid; (ii) each agent randomly picks up one data item; and (iii) each agent is placed at a random position on the grid. Subsequently, the sorting phase starts: this is a simple loop, in which (i) one agent is randomly selected; (ii) the agent performs a step of a given *stepsize* on the grid; and (iii) the agent (probabilistically) decides whether to drop its data item. In the case of a 'drop'-decision, the agent drops the data item at its current grid position (if this grid cell is not occupied by another data item), or in the immediate neighbourhood of it (it locates a nearby free grid cell by means of a random search). It then immediately searches for a new data item to pick up. This is done using an index that stores the positions of all 'free' data items on the grid: the agent randomly selects one data item i out of the index, proceeds to its position on the grid, evaluates the neighbourhood function $f^*(i)$, and (probabilistically) decides whether to pick up the data item. It continues this search until a successful picking operation occurs. Only then the loop is repeated with another agent.

For the picking and dropping decisions Deneubourg et al.'s threshold formulae (also see Equation 1 and 2) are used:

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

and

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

Here, we use $k^+ = 0.3$ and $k^- = 0.1$, and $f^*(i)$ is a modified version of

Algorithm 1 basic_ant

```
1:  begin
2:  INITIALISATION PHASE
3:  Randomly scatter data items on the toroidal grid
4:  for each j in 1 to #agents do
5:     i := random_select(remaining_items)
6:     pick_up(agent(j), i)
7:     g := random_select(remaining_empty_grid_locations)
8:     place_agent(agent(j), g)
9:  end for
10: MAIN LOOP
11: for each it_ctr in 1 to #iterations do
12:    j := random_select(all_agents)
13:    step(agent(j), stepsize)
14:    i := carried_item(agent(j))
15:    drop := drop_item?(f*(i)) // see equations 2 and 4
16:    if drop = TRUE then
17:       while pick = FALSE do
18:          i := random_select(free_data_items)
19:          pick := pick_item?(f*(i)) // see equations 1 and 4
20:       end while
21:    end if
22: end for
23: end
```

Lumer and Faieta's [51] neighbourhood function (see Equation 3):

$$f^*(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j (1 - \frac{\delta(i,j)}{\alpha}) & \text{if } (f^*(i) > 0 \wedge \forall j \ (1 - \frac{\delta(i,j)}{\alpha}) > 0) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

This definition of $f^*(i)$ combines two important properties. First, as in the original neighbourhood function $f(i)$, the division by the neighbourhood size σ^2 penalises empty grid cells, thus inducing a tight clustering (rather than just a loose sorting). Secondly, the additional constraint $\forall j \ (1 - \frac{\delta(i,j)}{\alpha}) > 0$ serves the purpose of heavily penalising high dissimilarities, which significantly improves spatial separation between clusters.

If not indicated otherwise, the parameter settings and the adaptive scaling scheme described in Section 6.0.11 are used in the experiments discussed in this section.

6.0.6 Short-term memory with 'look-ahead'

The clustering process on the grid can be significantly accelerated by the use of a 'short-term memory' as introduced in [51]. Recall that in Lumer et al.'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' memorised 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 currently carried data item i. 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 bias, we therefore permit each ant to 'look ahead'. An ant situated at grid cell p, and carrying a data item i, uses its memory to proceed to all remembered positions, one after the other. Each of them is evaluated using the neighbourhood function $f^*(i)$, that is, the suitability of each of them as a dropping site for the currently carried data item i is examined. Subsequently, the ant returns to its starting point p.

Out of all evaluated positions, the one of 'best match' is the grid cell for which the neighbourhood function yields the highest value. For the following step of the ant 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. 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 successfully drops the item.

This modification slightly speeds up the clustering process, which, in Figure 13, is reflected in the quicker rise of the overall and intra-cluster

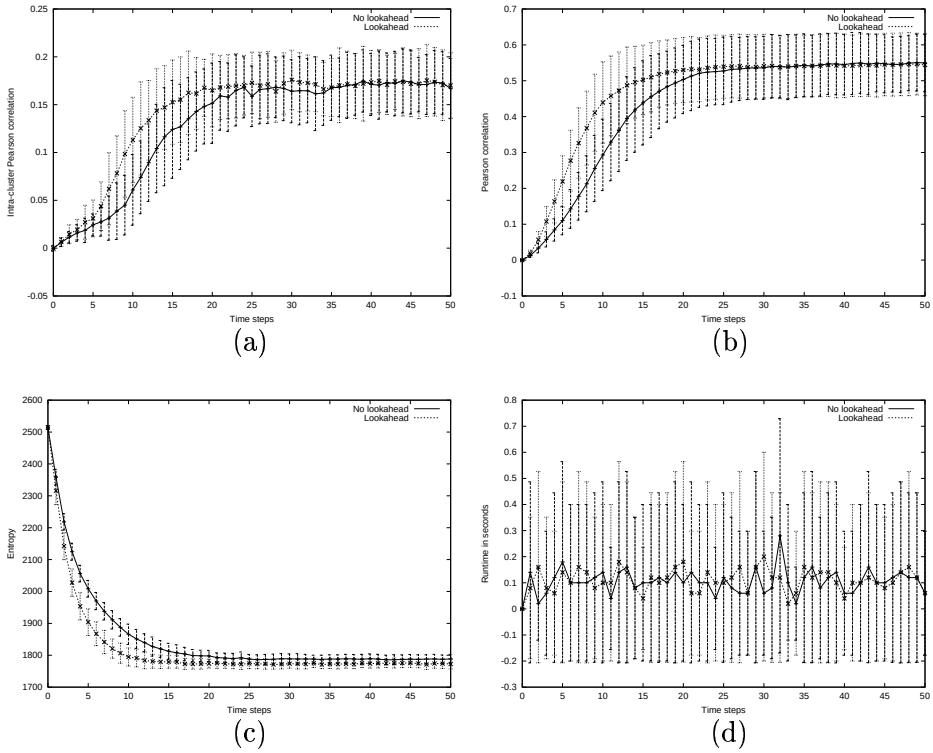


Figure 13: Results for ants with and without look-ahead on the *Square1* test set (cf. Section 7.6.1 for a description of this data set). Mean values and standard deviations for 50 runs. (a) Evolution of the intra-cluster Pearson correlation. (b) Evolution of the overall Pearson correlation. (c) Evolution of the entropy. (d) Average runtime per time step.

Pearson correlation values. It also encourages the compactness of clusters, which can be seen in the lower entropy values. It should also be noted that no significant increase in runtime can be observed.

6.0.7 Increasing radius of perception

The size of the local neighbourhood perceived by the ants limits the information used during the sorting process. It is therefore attractive to employ larger neighbourhoods in order to improve the quality of the clustering and sorting on the grid. However, the use of a larger neighbourhood 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.

We therefore 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

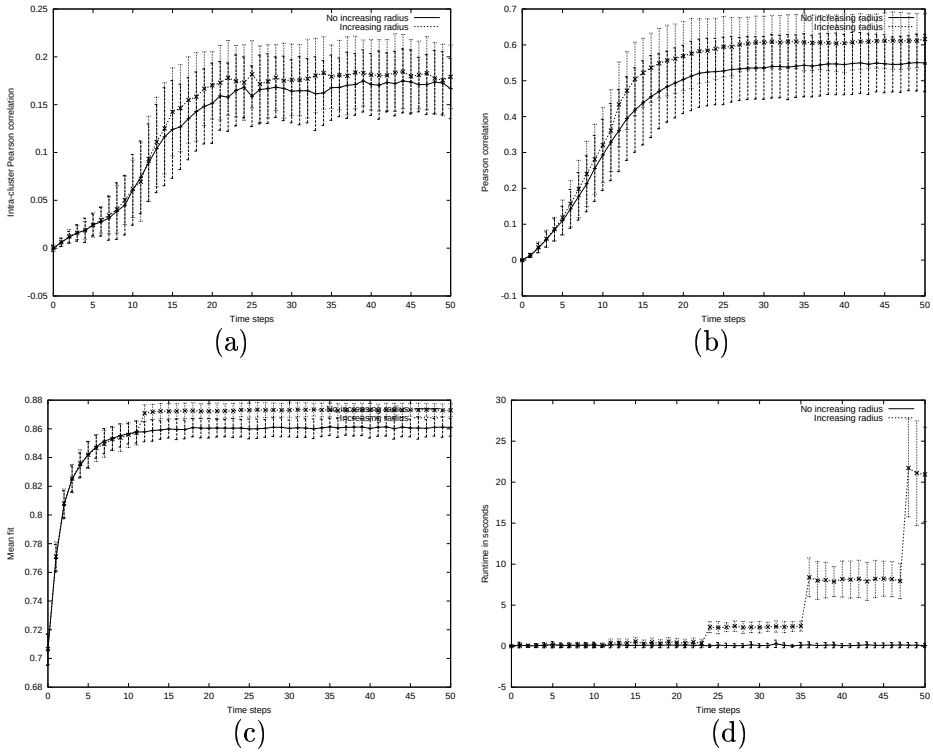
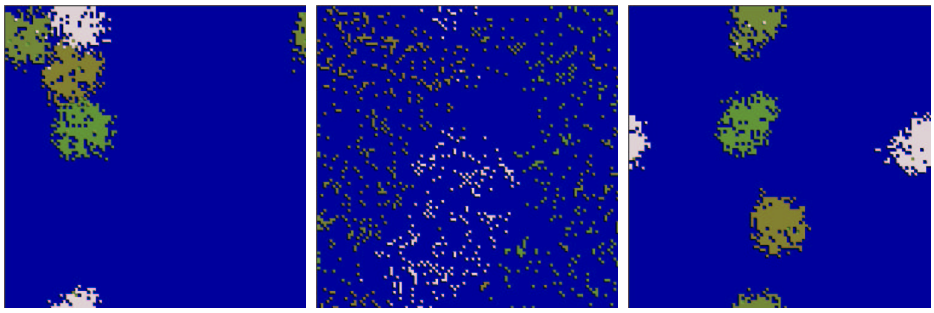


Figure 14: Results for ants with and without increasing radius of perception on the *Square1* test set (cf. Section 7.6.1 for a description of this data set). Mean values and standard deviations for 50 runs. (a) Evolution of the intra-cluster Pearson correlation. (b) Evolution of the overall Pearson correlation. (c) Evolution of the mean fit (for $\sigma^2 = 49$). (d) Average runtime per time step.

it accelerates the dissolution of preliminary small clusters, a problem that has already been addressed in [51, 35]. 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.

In Figure 14 we again show results on a synthetic data set. While the increasing radius improves the sorting quality within individual clusters (which is reflected by the average intra-cluster Pearson correlation and the mean fit), the overall gain is not as large as expected. It is particularly interesting that the first increase of the radius (from 1 to 2) triggers a significant rise of the mean fit; however, this is not the case for any of the further increments. On the other hand, the large radius contributes to a better spatial separation between the clusters, which is the main reason for the significant rise in the value of the overall Pearson correlation (see Section 5.2.2 for a discussion of the effects of spatial separation on the values obtained under the overall Pearson correlation). The plot of the average runtime per time step clearly



(a) (b) (c)

Figure 15: Spatial distribution on the grid at different stages. (a) After the initial clustering phase: the clusters touch. (b) After the interlude with the modified neighbourhood function: the clusters occupy individual grid regions. (c) Final result: the clusters are clearly separated.

shows the quadratical increase of time complexity at each increment of the radius of perception.

6.0.8 Spatial separation

As stated above, the spatial separation of clusters on the grid is crucial in order for individual clusters to be well-defined. Spatial closeness, when it occurs, is, to a large degree, an artefact 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 neighbourhood function, which replaces the scaling parameter $\frac{1}{\sigma^2}$ by $\frac{1}{N_{occ}}$ in Equation 4, where N_{occ} is the actual observed number of occupied grid cells within the local neighbourhood. Hence only similarity, not density, is taken into account, which has the effect of spreading out data items on the grid again, but in a sorted fashion; the data items belonging to different clusters will now occupy individual 'regions' on the grid. Subsequently, we turn back to using the traditional neighbourhood function. Once again, clear clusters are formed, but they now have a high likelihood of being generated along the centres of these 'regions', due to the lower neighbourhood quality at their boundaries.

In order to give a better idea of the functioning of this strategy, Figure 15 provides three snapshots of the spatial distribution on the grid, at different times of the clustering process. Analytically, the improved spatial separation can again be observed in plots of the overall Pearson correlation (not shown here).

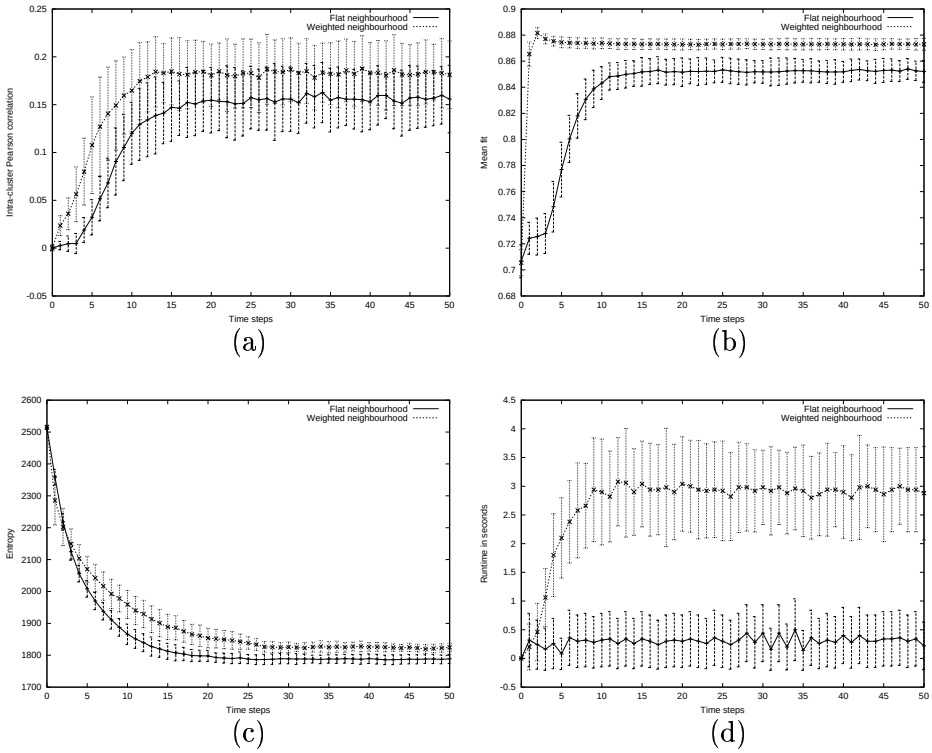


Figure 16: Results for flat and weighted neighbourhoods on the *Square1* test set (cf. Section 7.6.1 for a description of this data set). Mean values and standard deviations for 50 runs. (a) Evolution of the intra-cluster Pearson correlation. (b) Evolution of the mean fit (for $\sigma^2 = 49$). (c) Evolution of the entropy. (d) Average runtime per time step.

6.0.9 Weighted neighbourhoods

In all previous versions of the algorithm *flat neighbourhoods* have been used, that is, in the computation of the neighbourhood function $f^*(i)$ (Equation 4) all grid cells have the same impact. Here, we investigate the effect of *weighted neighbourhoods*, that is, a neighbourhood function where the similarity value for each data item in the local neighbourhood is multiplied by an additional weight term $\omega(i,j)$, to improve the sorting on a local scale.

The neighbourhood function $f^*(i)$ now becomes

$$f^*(i) = \begin{cases} \frac{1}{\sigma^2} \sum_j \omega(i,j) \ (1 - \frac{\delta(i,j)}{\alpha}) & \text{if } (f^*(i) > 0 \wedge \forall j \ (1 - \frac{\delta(i,j)}{\alpha}) > 0) \\ & \text{otherwise} \end{cases}$$

Hence, the conventional neighbourhood function would be the special case of $\omega(i,j) = 1$ for all data items j. We use

$$\omega(i,j) = e^{-\frac{\max(d_x(i,j), d_y(i,j))}{\rho}}$$

where ρ is the radius of perception and $d_x(i,j)$ and $d_y(i,j)$ are the distances (in horizontal and vertical direction) of data item i and j on the grid. Hence, the weights are bounded to the interval $[\frac{1}{e}, 1]$, with higher emphasis being given to the grid cells closest to the ant's position.

In Figure 16 the two different types of neighbourhoods are compared for $\rho = 3$. The improvement of the local sorting quality (see the plots of the intra-cluster Pearson correlation and the mean fit) obtained through the use of a weighted neighbourhood is evident, however, the overall quality is still rather low. Unfortunately, the weighting also results in looser clusters (see the plot of the entropy) and a largely augmented runtime.

6.0.10 Modified threshold functions

We also introduce a set of threshold functions that are different to those proposed by Deneubourg and Lumer and Faeita. The probability threshold for a picking operation is now determined as

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

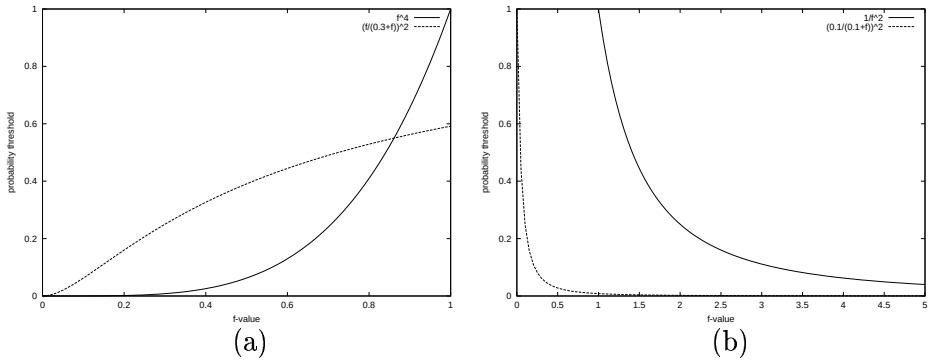
and for a dropping operation we compute

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

These functions have two advantages over the traditional probability thresholds. First, they are computationally cheaper, and, second, they avoid the introduction of the parameters k^+ and k^-.

Equations 5 and 6 have been experimentally derived, but in order to further clarify their characteristics, the graphs in Figure 17 compare them to those used by Lumer and Faieta (with $k^+ = 0.1$ and $k^- = 0.3$). Clearly, the dropping strategy shaped by the new threshold function is far more conservative than that defined by Lumer and Faeita's version. The probabilities in the 'low-similarity' region of the threshold function are lowered; high similarities, in contrast, are additionally rewarded. In fact, the dropping of data elements is completely deterministic for neighbourhood values $f^*(i) \geq 1.0$. Similarly, picking operations are deterministic for the entire range $f^*(i) \in [0,1]$. Only for neighbourhood values $f^*(i) \geq 1$ picking operations become less likely.

Note that the above given threshold formulae are quite different from the ones suggested by Deneubourg et al. and are not applicable to the basic ant algorithm. They have been experimentally derived for the use with our enhanced version (including an increasing radius of perception, a short-term memory with lookahead and an interlude with a modified neighbourhood function), for which they significantly speed up the clustering process. In



(a) (b)

Figure 17: Old and new threshold functions. (a) Dropping operation. (b) Picking operation.

particular, they have to be seen in light of the shift of the range of attainable values $f^*(i)$ resulting from our increase of the radius of perception (see Section 6.0.7).

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 neighbourhood radius, such that, in our implementation, $f^*(i)$ can yield values within the interval $[0, 15]$ after the last increment.[29] Consequently, the picking operation is purely deterministic in the beginning, and, at this stage, it is the dropping operation solely that favours dense and similar neighbourhoods. 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) has 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.

Figure 18 shows the outcome of experiments that compare two enhanced versions of the algorithm: the first making use of Lumer and Faieta's thresholds, and the second using our new probability functions. The plot of the entropy shows that this scheme initially results in a very quick reduction of the entropy, as small clusters are more unstable. In fact, all data items are aggregated in one single cluster with a limited degree of sorting. As there is no spatial separation between clusters at this stage, the overall Pearson correlation is noticeably lower to the case when Lumer and Faieta's threshold functions are used. However, the increment of the radius quickly provokes a separation of the clusters, such that the final overall Pearson correlation values are higher than those obtained with Lumer and Faieta's thresholds.

[29]This can be easily seen: The final radius of perception is 5, hence, the similarities of 1200 grid cells are summed up, which can, at best, yield a value of 120. As we do NOT update the scaling parameter σ when increasing neighbourhood size, it is still set to 8, and we therefore obtain $f = \frac{120}{8} = 15$.

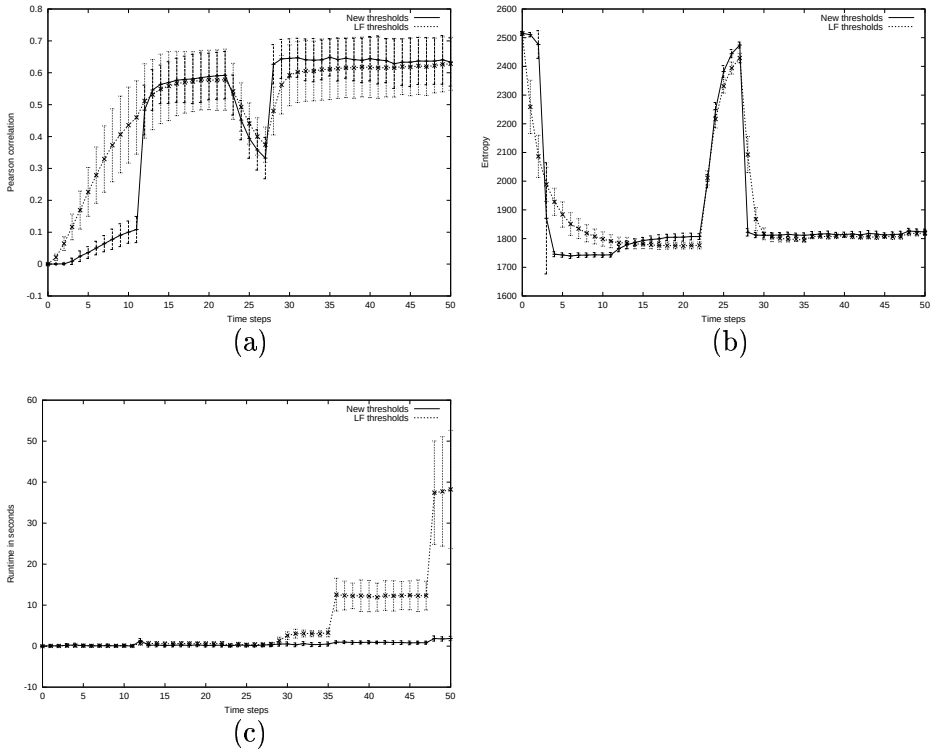


Figure 18: Track of the sorting progress for the new and old threshold functions. Mean values and standard deviation for 50 runs. (a) Evolution of the overall Pearson correlation. (b) Evolution of the entropy. (c) Average runtime per time step.

Also, the clusters are more dense and distinctively separated, which can be seen visually and can also be analytically assessed by the frequency of failure of our scheme of cluster retrieval (see Section 6.0.12). The most interesting property is, however, the significant decrease in runtime afforded by the modified thresholds that comes at no costs in terms of quality.

6.0.11 Parameter settings

Ant-based clustering requires a number of different parameters to be set, some of which have been experimentally observed to be independent 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 t_{start} and t_{end}, which we set to $0.45 \cdot \#iterations$ and $0.55 \cdot \#iterations$.

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

Several other parameters should however be selected in dependence of the size of the data set tackled, as they otherwise impair convergence speed. Given a set of N_{items} items, the grid (comprising a total of N_{cells} 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 $r_{occupied} = \frac{N_{items}}{N_{cells}}$ constant. A good value, found experimentally, is $r_{occupied} = \frac{1}{10}$. We obtain this by using a square grid with a resolution of $\sqrt{10N_{items}} \times \sqrt{10N_{items}}$ grid cells. The stepsize should permit sampling of each possible grid position within one move, which is obtained by setting it to $stepsize = \sqrt{20N_{items}}$. 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, $\#iterations = 2000N_{items}$, with a minimal number of 1 million iterations imposed.

Activity-based α-adaptation

An issue already addressed in [35] is the automatic determination of the parameter α (recall that α is the parameter scaling the dissimilarities in the neighbourhood function $f^*(i)$), which the functioning of the algorithm crucially depends on. 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 individual clusters, and in the limit, all data items would be gathered within one cluster.

Unfortunately, a suitable choice of the parameter α depends on the distribution of pairwise dissimilarities within the collection and, hence, cannot be fixed without regard to the data. However, a mismatch of α is reflected by an excessive or extremely low sorting activity on the grid. Therefore, an automatic adaptation of α can be obtained through the tracking of the amount of activity, which is reflected by the frequency of the agents' successful picking and dropping operations. The scheme for α-adaptation used in our experiments is described below.

A heterogenous population of agents is used, that is, each agent makes use of its own 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 N_{active} moves. During this time, it keeps track of the number of failed dropping operations N_{fail}. The rate of failure is determined as $r_{fail} = \frac{N_{fail}}{N_{active}}$ where N_{active} is fixed to 100. The agent's individual parameter α is then updated using the rule

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

which has been experimentally derived. 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 permits for the specific adaptation of α within different phases of the sorting process.

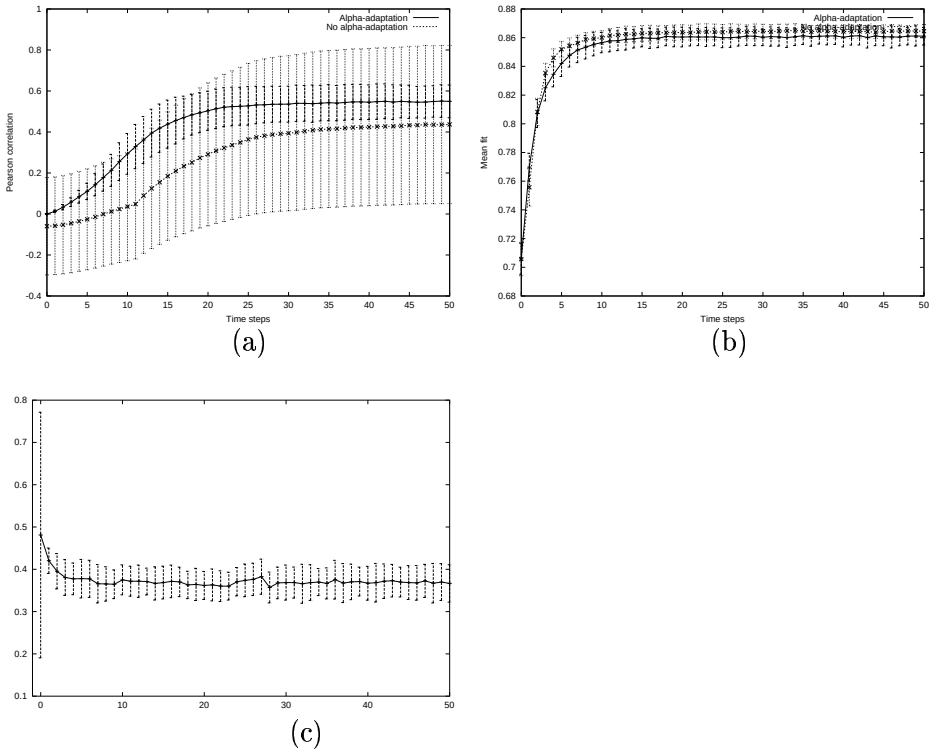


Figure 19: Results for the adaptive and non-adaptive scheme on the *Square1* test set (cf. Section 7.6.1 for a description of this data set). Mean values and standard deviations for 50 runs. (a) Evolution of the entropy. (b) Evolution of the overall Pearson correlation. (c) Evolution of α for the adaptive scheme.

The aim of this adaptive scheme is to obtain results which are comparable to those with a suitable manually determined α-parameter, as this manual determination is extremely tedious even when being familiar with the algorithm. Results on our test sets lead to the conclusion that this aim is met. In general, the non-adaptive algorithm is even slightly outperformed by the adaptive version. Additionally, the adaptive version permits a very quick convergence of the algorithm, as initially large α values favour the quick crystallization of clusters, which are subsequently refined when the values of α decrease. In contrast to this, the non-adaptive version takes a far longer warm-up time before clusters eventually emerge.

6.0.12 Cluster-retrieval

As stated before, the analytical evaluation of the clustering results requires the derivation of an automatic method to retrieve the 'visually' obvious clusters from the grid. In [54] methods from pattern recognition were employed to identify the size of the clusters on the grid and to examine their density.

For our purpose, we chose to to apply a clustering algorithm to the grid positions of all data items. In [49], a K-means algorithm was used towards this goal, this 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 the correct number of clusters is provided).

We therefore apply an agglomerative hierarchical clustering algorithm to the positions of the data items on the grid. The 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 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 an additional 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 $(1, 2]$.

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 the additional weighting term $weight(C_1, C_2)$ that encourages the merging of these elements with the 'core' clusters.

The radius of perception used within the last phase of the ant-based algorithm signifies the minimum distance clusters should have on the grid,[30] 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.

6.0.13 Overall improvement

From the changes introduced in this section, we adopt the following for the ant algorithm used in our comparative study: the data-derived parameter

[30] Provided that the grid is large enough for all clusters to be well separated, of course.

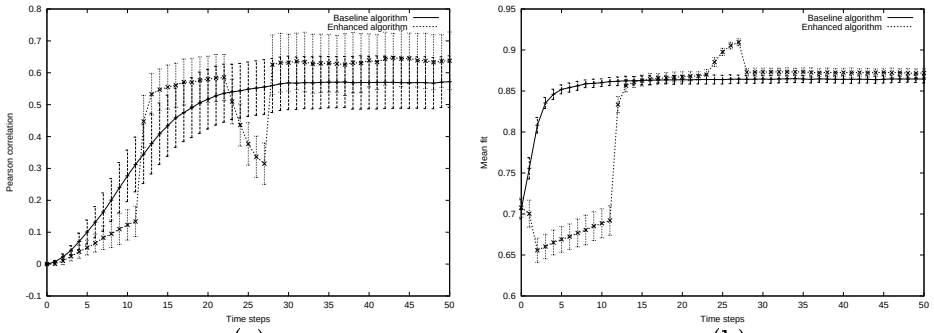


Figure 20: Results for the different versions of the algorithm on the *Square1* test set (cf. Section 7.6.1 for a description of this data set). Mean values and standard deviations for 50 runs. (a) Evolution of the overall Pearson correlation. (b) Evolution of the dissimilarity.

settings and the scheme of α-adaptation, the increasing radius of perception, the interlude with a modified neighbourhood function, the short-time memory with look-ahead and the new probability thresholds. We refrain from using the weighted neighbourhoods due to their significant computational costs.

To conclude this section, we provide graphs on the overall improvement obtained as compared to the baseline algorithm without any extensions. Figure 20 gives the algorithms' performance in terms of the overall Pearson correlation and the mean fit. The plots show that the modifications improve the local sorting quality and the algorithm's performance in terms of the overall Pearson correlation. Most importantly, the generated clusters are compact and spatially well separated. Analytically, this is reflected in the performance of our scheme of cluster retrieval: it works robustly for the results obtained by the enhanced algorithm, whereas it frequently fails for those of the old version.

At the heart of quantitative reasoning
is a single question: Compared to what?
(Tufte)

7 Comparison of ant-based clustering and sorting to alternative methods

In this section, the primary contribution of this thesis is finally presented, which is the experimental investigation of the performance of ant-based clustering and sorting, and its comparison to traditional data-mining techniques. We start by reviewing the main scientific questions that we attempt to answer, and that have shaped the experimental setup. Next, the setup itself is described, including a motivation and description of the algorithms, the evaluation functions and the test data used. Subsequently, the obtained results are presented and discussed.

7.1 Questions of interest

The literature survey in Section 4 has shown that previous research on ant-based clustering and sorting has left several questions related to the algorithm's performance widely 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, are its solutions competitive in terms of quality and runtime?

- **Sorting performance**
  To what degree is the spatial embedding generated by the ant algorithm topology-preserving? Are neighbourhood 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, are its solutions competitive in terms of quality and runtime?

- **Sensitivity to data properties**
  How robust is the ant algorithm's clustering and sorting performance with respect to different data properties? In particular, how strong 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 aspects reflect the main issues investigated in the following comparative study. For their objective and general analysis a number of different issues needed to be resolved, many of which are not trivial.

7.2 Challenges

Both the comparison of the clustering and the sorting performance requires the selection of appropriate contestants from the large variety of existing data-mining methods. As we have seen these methods differ however in the problem definition and the (implicit or explicit) optimisation criterion used, such that the choice of the algorithms must be closely coupled with the selection of suitable evaluation functions.

The evaluation of the quality of a topographic mapping is particularly precarious, as, even for synthetic data sets, the 'ideal' solution is not clearly defined and the applicable measures only capture certain aspects of the generated solution. Also, quality assessment based on individual measures can be heavily misguiding (cf. Section 7.8.2), and a close feedback loop between visual and analytical analysis is therefore indispensable in order to understand and correctly interpret results.

The analytical assessment of the clustering performance poses the additional problem of an unbiased cluster retrieval, an issue that we have addressed in Section 6.0.12.

Most importantly, the algorithms selected for comparison should (i) be general and well-known enough to permit an interpretation of the results by a larger audience while (ii) being sufficiently enhanced to pose a challenge to the investigated ant-algorithm. In this context, it is also crucial to pay attention to introduce as little bias as possible through the selection of evaluation measures and data sets used.

Table 1 and Table 2 summarise the algorithms and evaluation measures we have decided to use in our comparative study. We will shortly motivate these choices in the following and outline the implementation of the individual algorithms.

Table 1: Overview of algorithms.

Cluster Analysis	**Topographic mapping**
K-means	MDS
average link	lower bound
1D-SOM	2D-SOM
ant-based clustering	ant-based sorting

7.3 Algorithms for cluster analysis

In Section 2 the main classes of available clustering methods have been introduced. Out of these, we have selected two of the most commonly used algorithms for the comparison to ant-based clustering: the partitioning method K-means, and a hierarchical agglomerative approach based on the linkage criterion of average link. While K-means has a favourable (linear) runtime behaviour, average link agglomerative clustering is known for its high quality solutions, such that a comparison against both of them provides a good basis for judgements on the performance of the ant-algorithm. Additionally, one-dimensional self-organising maps (1D-SOM) were used, a choice motivated by the fact that SOMs combine the tasks of clustering and topographic mapping (just as claimed for ant-based clustering and sorting).

Both K-means and 1D-SOM usually require the a priori determination of the desired number of clusters K; for the agglomerative clustering algorithm an appropriate stopping criterion must be derived. In order to avoid the introduction of an additional source of error, we 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 the ant-based algorithm is the only clustering method that automatically derives K from the data. The implementations of the four contestants are shortly described below.

7.3.1 K-means

The implementation of the K-means algorithm is based on the batch version of K-means, that is, cluster centres 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 reinitialised. 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 initialisation, and only the best result in terms of intra-cluster variance is returned.

7.3.2 Average link agglomerative clustering

The 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

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

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

7.3.3 One-dimensional self-organising maps

The implementation of 1D-SOM is based on the guidelines given in the description of the SOM Toolbox [77]. 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 initialised. 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 neighbourhood size of $ns_1^{start} = \max(1.0, \frac{1}{4}K)$, which is exponentially decreased to $ns_1^{end} = \max(1.0, \frac{1}{4}ns_1^{start})$. The learning rate during this phase is $lr_1 = 0.5$. The second phase starts with the final neighbourhood size of phase 1, that is, $ns_2^{start} = ns_1^{end}$, and continues to decrease it to $ns_2^{end} = 1.0$. The learning rate in phase 2 is $lr_2 = 0.05$. 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.

7.3.4 Ant-based clustering

Like average link agglomerative clustering, the ant algorithm uses a precomputed matrix of pairwise dissimilarities between data elements. Of the enhancements explained in Section 6 the following ones are used: data-derived parameter settings, the α-adaptation scheme, a short-term memory with look-ahead, an increasing radius of perception, the interlude with a

modified neighbourhood function and the new threshold functions. The method of cluster retrieval introduced in Section 6.0.12 is applied to convert the spatial mapping to an explicit clustering solution. In the following, we refer to this algorithm as *ant-based clustering*.

7.3.5 Gap statistic

For the evaluation of ant-based clustering'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 [75]. This statistic is based on the expectation that the most suitable number of clusters shows in a significant 'knee' when plotting the performance of a clustering algorithm as a function of the number of clusters K. For this purpose, the clustering problem is solved for a range of different values of K and, for each K, the resulting partitioning $C = \{C_1, .., C_K\}$ is evaluated by means of the intra-cluster variance, which is given by

$$V(C) = \sum_{C_k \in C} \sum_{i \in C_k} (\delta(i, \mu_k))^2$$

Here C_k is the kth cluster in the partitioning, μ_k is the corresponding cluster centre, and $d(i, \mu_k)$ gives the dissimilarity between data item i and μ_k. The intra-cluster variance is affected by the number of clusters, such that a plot $Var(K)$ showing the evolution of $V(C)$ as a function of the input parameter K exhibits a decreasing trend that is solely caused by the finer partitioning and not by the actual capturing of structure within the data. The Gap statistic overcomes this effect through a normalisation of the performance curve. B reference curves $R_b(K)$ (with $b \in \{1, ...B\}$) are computed, which are the performance curves obtained with the same clustering algorithm for uniform random reference distributions. Using these, the normalised performance curve ('Gap curve') for $Var(K)$ is then given as

$$Gap(K) = \frac{1}{B} \sum_{b=1}^{B} log(R_b(K)) - log(Var(K))$$

The most suitable number of clusters is determined by finding the first significant local maximum of $Gap(K)$.

For our implementation of the Gap statistic we use the above described K-means algorithm. We compute the performance curves for $K \in \{1, ..., 20\}$, and, for each K, we generate $B = 20$ reference distributions.

7.4 Algorithms for topographic mapping

Section 3 has provided a survey of the types of methods known for topographic mapping. From this pool, we have selected two alternative ap-

proximation methods. These are, first, an iterative approach to non-metric multidimensional scaling, and, second, two-dimensional self-organising maps (2D-SOM), both established visualisation methods. A strong motivation for the use of 2D-SOM was the similarity of its solutions to those of ant-based clustering and sorting: both are grid-based, whereas in MDS data items are assigned continuous positions (which permits additional precision that is not necessarily evident to the human observer but may show in an analytical evaluation). In addition, we use a simple randomised method to obtain a lower bound on map quality.

7.4.1 Non-metric multidimensional scaling

The iterative approximation scheme used for non-metric multidimensional scaling is a gradient descent method that (locally) minimises non-metric stress. The method starts by normalising dissimilarities to cover the interval $[0, 1]$. 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 $lr = 0.05$, $d(.,.)$ is the Euclidean distance between data items in map-space, $d_x(.,.)$ is the deviation in direction x between data items in map-space, and $\delta(,.)$ is the dissimilarity between data items in data-space. The update dy (in direction y) is computed analogously.

7.4.2 Two-dimensional self-organising maps

For 2D-SOM, a square grid was used, with the number of rectangular cells equal to the collection size, that is, using a grid resolution of $\sqrt{N_{docs}} \times \sqrt{N_{docs}}$. All further implementation details are identical to those for 1D-SOM.

7.4.3 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 distance: the Euclidean distance between each pair

Table 2: Overview of evaluation functions.

Cluster Analysis	**Topographic mapping**
F-measure	Pearson correlation
Rand index	Inter-cluster Pearson correlation
Inner variance	Intra-cluster Pearson correlation
Dunn index	Spearman rank correlation
	Topographic error

of cluster positions must be at least $\frac{\sqrt{N_{cells}}}{K}$, otherwise one of the colliding positions is regenerated. Here, N_{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, as it does not provide more information than a pure clustering.

7.4.4 Ant-based sorting

The implementation of the ant algorithm for topographic mapping is identical to the pure clustering version (cf. Section 7.3.4). Merely the post-processing phase for cluster retrieval has been removed. We refer to this algorithm as *ant-based sorting*.

7.5 Evaluation measures

From the evaluation functions introduced in Section 5, we have selected a set of four measures for the analysis of the clustering results and a set of five measures for the analysis of the topographic mappings. Each of these reflects a different aspect of the clustering and sorting quality respectively, so that, combined, they provide an overall picture of the algorithms' performance.

7.5.1 Measures for cluster analysis

As all experiments are run on benchmark data with the correct class labels known, we have the chance to use external evaluation measures. For this purpose we have chosen the Rand index and the F-measure, which both have a tradition of use in the general clustering literature and have previously been applied for the evaluation of ant-based clustering [57, 38]. A weakness of the Rand index is its high sensitivity to the number of clusters identified,

which makes it difficult to compare partitionings with a different number of clusters; this is counterbalanced by the use of the F-measure, which is less affected by deviations in the number of clusters.

While we don't rely on the use of internal evaluation methods, their use in the experiments is interesting for two reasons. First, the application of ant-based clustering in real clustering tasks will require the evaluation of the obtained result in the absence of knowledge on the correct solution. It is therefore interesting to study the performance of the algorithm under internal evaluation functions. Second, these functions provide additional information about the structure of the obtained solutions and can therefore help to understand and analyse results.

We have decided to use the measure of the intra-cluster variance, as this is the function optimised by the K-means algorithm. Its use can therefore provide additional insight in the cases where K-means generates suboptimal solutions, or where the criterion of the intra-cluster variance is misleading. As a second measure we apply the Dunn index, as it captures an aspect of the solution quality that is not reflected by the intra-cluster variance, namely the ratio between inter- and intra-cluster distances.

For each experiment, we also provide the average number of identified clusters[31] (the reader should keep in mind that this number is automatically only determined by the ant algorithm).

7.5.2 Measures for topographic mapping

For the analysis of topographic mappings, we have decided to use the Pearson correlation, which permits us to compare our results with those of previous studies [34, 48]. While the stress function has also been used in previous work on ant-based clustering and sorting, we have decided against its use, as it is not invariant under scaling. Invariance under scaling is an important issue in our comparison, as grid sizes for the ant algorithm are required to be much larger than those for self-organising maps.

Given our knowledge of the cluster structure for all data sets, we can additionally compute the inter-cluster and the intra-cluster Pearson correlation (averaged over all clusters), which provide valuable insights into the composition of the overall Pearson correlation. As a last correlation measure, we additionally use the Spearman rank correlation in order to abstract from absolute distance values.

All of these correlation measures predominantly capture neighbourhood preservation on a global scale. This can put the self-organising maps at a disadvantage, as they optimise topology-preservation on a local scale. We therefore use an extended implementation of the topographic error as an

[31] It is important to note, that none of the used measure is completely insensitive to the number of identified clusters. In fact, the objective comparison of partitionings with a differing number of clusters is a prevailing problem in the data-mining community.

additional measure. It determines the percentage of data items that are nearest neighbours in data-space, and have a distance of more than one grid cell in the (discretised) map-space.[32]

Whilst this measure helps to give an idea of the algorithms' local precision, the reader should note that a comparison of the resulting values between 2D-SOM, MDS and ant-based sorting is quite problematic. This is because, in the mappings generated by 2D-SOM and MDS, grid cells can be occupied by several data items, whereas ant-based sorting permits only one data element per grid cell. Also, the grids used by 2D-SOM are smaller than those of ant-based sorting and MDS (the mapping generated by MDS is scaled and then discretised to yield the same size and resolution as that generated by ant-based sorting). However, a direct comparison is possible for the solutions of ant-based sorting and those of the lower bound method.

7.5.3 Time measurements

In addition to the average performance under these evaluation measures, runtimes of the individual algorithms are also provided. Time and space complexity of the algorithms are summarised in Table 3. The experiments for topographic mapping have been run on a mobile AMD Athlon(tm) XP 2200+ (1.8MHz) with 512 MB main memory. Those for cluster analysis have been run on an AMD Athlon(tm) XP 2400+ (2MHz) with 512 MB main memory.

[32] Note that a use of the original definition of the topographic error is not possible as it is based on the concept of the "best matching unit", which is only defined for 2D-SOM. The reason why we do not simply determine the percentage of data items that are nearest neighbours in both data-space and map-space is the following: in the solutions generated by 2D-SOM data items can have a large number of nearest neighbours in map-space (all those assigned to directly neighbouring neurons), whereas there can be at most four for ant-based sorting (the data items located at any of the four directly neighbouring cells), and there is usually only one for MDS (as long as the grid positions are kept continuous).

Table 3: Time and space complexity of the different algorithms. N is the size of the data set and D is the dimensionality of the data.

Cluster Analysis	Time Complexity	Space Complexity
K-means	$O(DN)$	$O(N)$
average link	$O(N^2)$	$O(N^2)$
1D-SOM	$O(DN)$	$O(D)$
ant-based clustering	$O(N)$	$O(N^2)$
Topographic mapping	Time Complexity	Space Complexity
MDS	$O(N^2)$	$O(N^2)$
2D-SOM	$O(DN^2)$	$O(DN)$
ant-based sorting	$O(N)$	$O(N^2)$

7.6 Experimental data

Altogether, three different types of benchmark data sets are used. First, a range of two-dimensional data sets with fixed cluster properties that permits the modulation of specific data properties. Second, several data sets with randomly determined cluster properties. And, finally, several real data sets taken from the Machine Learning Repository [8].

7.6.1 Fixed cluster properties

In the first class of synthetic data 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.

Table 4 gives the definition of the benchmarks, and Figure 21 shows four sample instances. The benchmarks are variations of the *Square* data set, a data set that has been frequently employed in the literature on ant-based clustering. 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 is 10,

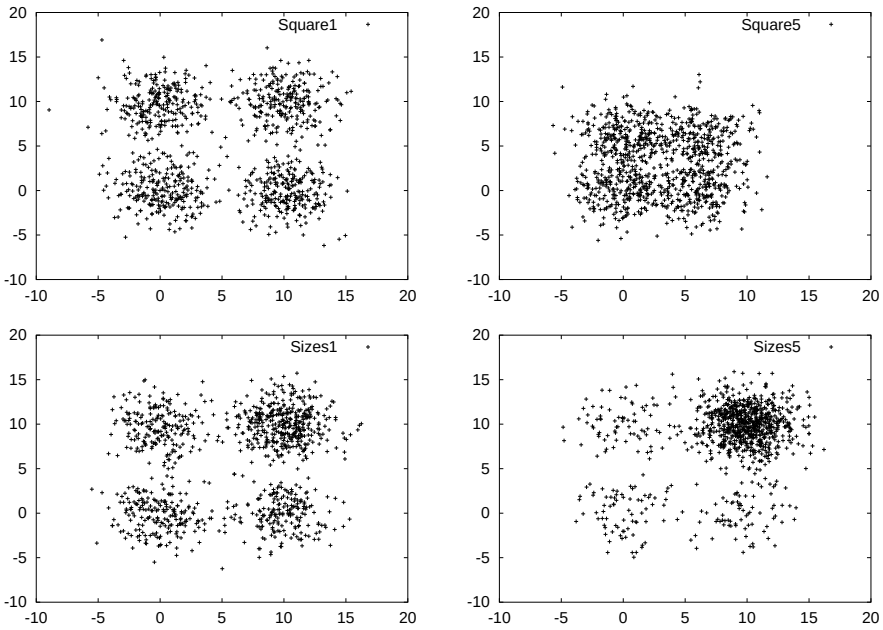


Figure 21: Four sample instances of the *Square1*, the *Square5*, the *Sizes1* and the *Sizes5* benchmark data.

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.

7.6.2 Random cluster properties

The above data sets are useful for the analysis of the algorithms' performance with respect to specific data properties. However, for sake of generality, we have additionally introduced a range of synthetic test sets, which are (almost) completely randomly generated. These test sets are denoted as xD-yC, where x indicates the dimensionality of the data and y gives the number of clusters. Each one of them sets consists of 50 different instances and each individual instance is generated as follows.

We specify a set of y x-dimensional normal distributions $N(\vec{\mu}, \vec{\sigma})$ from which we sample the data items for the y different clusters in the instance. The sample size s of each normal distribution, the mean vector $\vec{\mu}$ and the vector of the standard deviation $\vec{\sigma}$ are themselves randomly determined

Table 4: Summary of the used data sets with fixed cluster properties. D is the dimensionality, C gives the number of clusters, and N_i gives the number of data elements for cluster i. The test sets are generated by multidimensional Normal Distributions $N(\vec{\mu}, \vec{\sigma})$, where $\vec{\mu}$ is the vector of means and $\vec{\sigma}$ is the vector of the standard deviations.

Name	C	N_i	D	Source
Square1	4	4×250	2	$N([0,0],[2,2])$, $N([10,10],[2,2])$ $N([0,10],[2,2])$, $N([10,0],[2,2])$
Square2	4	4×250	2	$N([0,0],[2,2])$, $N([9,9],[2,2])$ $N([0,9],[2,2])$, $N([9,0],[2,2])$
Square3	4	4×250	2	$N([0,0],[2,2])$, $N([8,8],[2,2])$ $N([0,8],[2,2])$, $N([8,0],[2,2])$
Square4	4	4×250	2	$N([0,0],[2,2])$, $N([7,7],[2,2])$ $N([0,7],[2,2])$, $N([7,0],[2,2])$
Square5	4	4×250	2	$N([0,0],[2,2])$, $N([6,6],[2,2])$ $N([0,6],[2,2])$, $N([6,0],[2,2])$
Square6	4	4×250	2	$N([0,0],[2,2])$, $N([5,5],[2,2])$ $N([0,5],[2,2])$, $N([5,0],[2,2])$
Square7	4	4×250	2	$N([0,0],[2,2])$, $N([4,4],[2,2])$ $N([0,4],[2,2])$, $N([4,0],[2,2])$
Sizes1	4	$400, 200, 200, 200$	2	$N([0,0],[2,2])$, $N([10,10],[2,2])$ $N([0,10],[2,2])$, $N([10,0],[2,2])$
Sizes2	4	$571, 143, 143, 143$	2	$N([0,0],[2,2])$, $N([10,10],[2,2])$ $N([0,10],[2,2])$, $N([10,0],[2,2])$
Sizes3	4	$667, 111, 111, 111$	2	$N([0,0],[2,2])$, $N([10,10],[2,2])$ $N([0,10],[2,2])$, $N([10,0],[2,2])$
Sizes4	4	$727, 91, 91, 91$	2	$N([0,0],[2,2])$, $N([10,10],[2,2])$ $N([0,10],[2,2])$, $N([10,0],[2,2])$
Sizes5	4	$769, 77, 77, 77$	2	$N([0,0],[2,2])$, $N([10,10],[2,2])$ $N([0,10],[2,2])$, $N([10,0],[2,2])$

Table 5: Summary of the used real data sets from the Machine Learning Repository. D is the dimensionality, C gives the number of clusters, and N_i gives the number of data elements for cluster i.

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

using uniform distributions over fixed ranges (with $s \in [50, 450]$, $\mu_i \in [0, 100]$ and $\sigma_i \in [0, 5]$).

Consequently, the expected size of instances of xD-$4C$ and xD-$10C$ is 1000 and 2500 data items respectively. During the generation process, cluster centres are rejected if the resulting distributions would have more than 3% overlap. A different instance is used in each individual run of the experiments. Results are presented for six synthetic data sets of this type: these are the sets $2D$-$4C$, $2D$-$10C$, $10D$-$4C$, $10D$-$10C$, $100D$-$4C$ and $100D$-$10C$.

7.6.3 Data sets from the Machine Learning Repository

Table 5 shortly describes the real data sets taken from the Machine Learning Repository. A variety of different benchmarks has been chosen in order to account for different problem sizes and problem structures (such as a differing number of clusters, clusters of varying sizes etc.).

7.6.4 Data Processing

Prior to clustering/topographic mapping, all types of data are subject to the following preprocessing steps: the data vectors are normalised in each dimension. For the ant-based algorithm, average link agglomerative cluster-

ing and multidimensional scaling all pairwise dissimilarities are precomputed and normalised to the interval $[0, 1]$. Both K-means and 1D-SOM require the computation of distances between data items and cluster representatives (which do not necessarily correspond to data items and possibly change in each iteration), such that a precomputation of the distances is not possible for these two algorithms (the same applies to 2D-SOM). This clearly involves an additional computational overhead for these methods, which rises with increasing dimensionality of the tackled data.[33]

The distance functions used are as follows: For the synthetic data sets we use the Euclidean distance, which, for two D-dimensional data items i and j, is defined as

$$d_{euclidean}(i,j) = \sqrt{\sum_{k=0}^{D}(i_k - j_k)^2}$$

The advantage of using the Euclidean distance is the straightforward interpretation and visualisation of the data, which facilitates the derivation of appropriate test sets and the analysis of the results. Also, the Euclidean distance is known to work well if clusters are spatially well separated.

For the real data benchmark set, we use a distance function based on the Cosine measure of similarity. It is given as

$$d_{cosine}(i,j) = 1.0 - 0.5 \cdot (1.0 + \frac{\sum_{k=0}^{D} i_k \cdot j_k}{(\sum_{k=0}^{D} i_k \cdot i_k)(\sum_{k=0}^{D} j_k \cdot j_k)})$$

Hence, we compute the Cosine of the two data vectors, translate and scale the resulting value to lie within the interval $[0, 1]$, and finally convert this similarity value to a dissimilarity value by subtracting it from 1.0. The Cosine measure is the similarity measure generally used with the data sets from the Machine Learning Repository, and, in fact, results obtained with it are, on this data, clearly superior to those obtained with the Euclidean distance.[34]

[33]Note that the use of a precomputed dissimilarity matrix for average link agglomerative clustering and ant-based clustering and sorting is possible only if data collections are small enough for the complete triangular dissimilarity matrix to fit into the main memory. This is no limitation within our study but it would become an issue in VLDB applications.

[34]The reader should note that the distance function used can crucially affect the performance of a clustering algorithm. The differences between the Euclidean and the Cosine measure can be illustrated as follows: The Euclidean distance simply computes the spatial distance between vectors in data space. The Cosine measure, in contrast, first projects all data points on a hypersphere with radius 1 around the origin. The similarity between two data points is then given by the Cosine of the angle between the vectors pointing to the projections of the data points. Obviously, it depends on the data whether such a projection is advantageous or disadvantageous: it can facilitate the clustering problem, but it can also eliminate relevant information (e.g., when the projections of two separate clusters falls together).

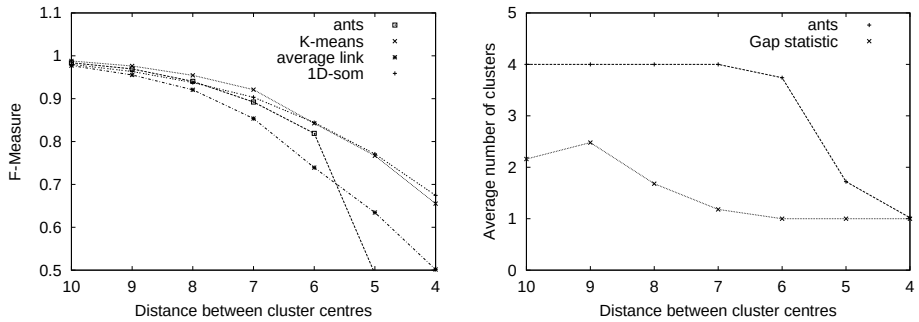


Figure 22: Performance as a function of the distance between the cluster centres on the *Square* data sets. (a) F-measure (b) Number of identified clusters.

7.7 Results for cluster analysis

We now summarise the results obtained in our comparison of the ant-based clustering algorithm with the standard clustering techniques K-means, average link agglomerative hierarchical clustering and 1D-SOM. We first discuss the results obtained on the two types of synthetic data sets and, subsequently, analyse additional aspects of the algorithm's performance revealed by its application to the real data sets. The complete tables accompanying the presented results can be found in the appendix.

7.7.1 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 22a shows a plot of the algorithms' performance (as reflected by the F-measure) versus the distance between neighbouring 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 behaviour, 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 lower inter-cluster distance.

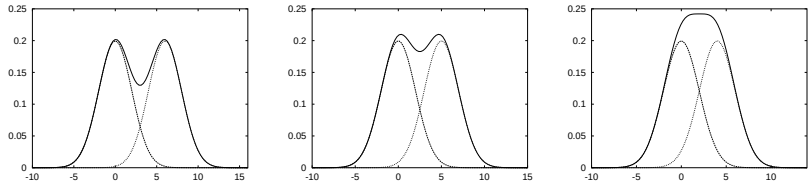


Figure 23: Theoretical density distribution along the connecting line between two cluster centres in the *Square5*, the *Square6* and the *Square7* test set (from left to right). Constructed as the superimposition of two one-dimensional normal distributions with standard deviation 2 and a distance of 6, 5 and 4 respectively.

The results of ant-based clustering 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 better than average link on the first five test sets. Also, in spite of the fact that the cluster 'touch', the ant-algorithm reliably identifies the correct number of clusters on the first three test sets, 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.

For the *Square6* and *Square7* test data, the performance of ant-based clustering 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 1 cluster is identified. However, a plot of the theoretical density distribution along the 'edge' between two neighbouring clusters in this data set puts this failure into perspective: Figure 23 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.

The reader should keep in mind that, different from its competitors, ant-based clustering has not been provided with the correct number of clusters. In order to get a more precise idea of the performance of ant-based clustering, we therefore additionally analyse its success at identifying the correct number of clusters in the data. The comparison in Figure 22b shows that ant-based clustering performs very well, it is much less affected by the lack of spatial separation between the clusters than the Gap statistic.

7.7.2 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 24a).

Ant-based clustering performs very well on all five test sets, in fact it is hardly affected at all by the increasing deviations between cluster sizes. Out

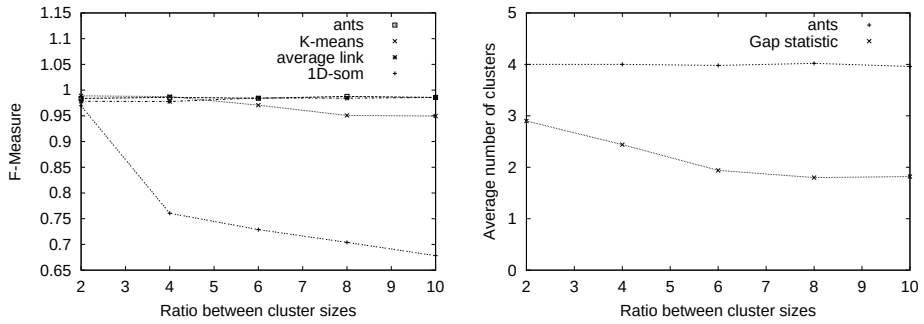


Figure 24: Performance as a function of the ratio between cluster sizes on the *Sizes* data sets. (a) F-measure (b) Number of identified clusters.

of its three contestants, 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. The performance of the Gap statistic is again very weak when compared to ant-based clustering (see Figure 24b).

7.7.3 Summary of the performance on synthetic data

To give an overall impression of the average performance of the four algorithms on more general data, we now provide results on the xD-yC data sets. Here, we focus on the relative quality of the clustering solutions, as assessed by the four different measures. The runtimes of the individual algorithms are then discussed in the subsequent section.

The graphs in Figure 25 show that ant-based clustering performs quite well on all of these data sets. It is only beaten by average link agglomerative clustering that performs marginally better on four out of the six data sets. For the other two data sets (*10D-4C* and the *100D-4C*), both ant-based clustering and average link agglomerative clustering perform the same: they fully reproduce the correct partitioning in each individual run.

By comparison, K-means and 1D-SOM are well beaten. While they also obtain the 'perfect' solution in a few cases, they frequently generate largely suboptimal solutions, in particular on the *xD-10C* data sets.

Running the Gap statistic on the xD-yC sets demonstrates that it is not trivial to identify the correct number of clusters K in this data. Again, ant-based clustering clearly outperforms the Gap statistic, whose results are less accurate and highly deviate (see Table 8.1). This somewhat relativises the merit of average link agglomerative clustering on these benchmarks, as it relies on the specification of K.

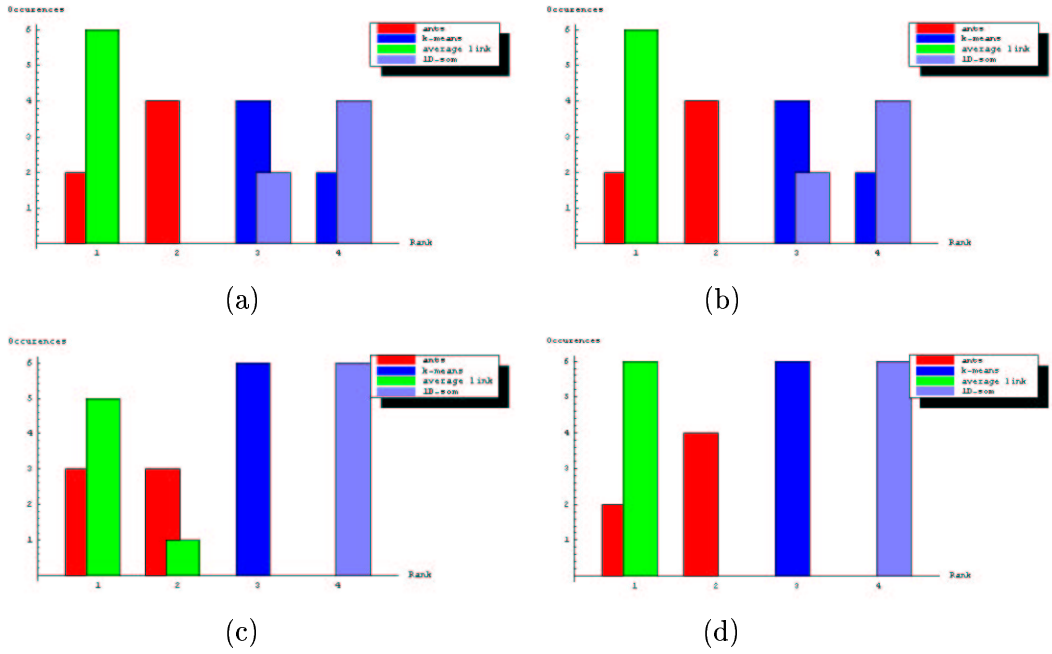


Figure 25: Relative performance of ant-based clustering, K-means, average link agglomerative clustering and one-dimensional self-organising maps on the six xD-yC data sets. For each method, the frequency of the obtained ranks under (a) the F-measure, (b) the Rand index, the (c) Intra-cluster variance and (d) the Dunn index is given.

7.7.4 Summary of the performance on real data

Finally, we provide results on the algorithms' performance on the seven real data sets taken from the Machine Learning Repository. Figure 26 again shows a graph on the frequencies of the obtained ranks for each algorithm and each measure.

The overall picture obtained is far less uniform in this case. No algorithm consistently dominates the others, in particular, a clear discrepancy can be observed between the ranks returned by the individual measures.

Different to the synthetic benchmarks, ant-based clustering repeatedly fails to identify the correct number of clusters in this data. In particular, the number of clusters determined on the *Zoo*, the *Digits*, the *Dermatology* and the *Yeast* data sets is distinctively too low, which obviously affects the values obtained under the external measures (the F-measure and the Rand index). In spite of this the obtained values are worst only on one of these test sets (the *Digits* data) and ant-based clustering even shows a very good performance under one of the internal measures (the Dunn index).

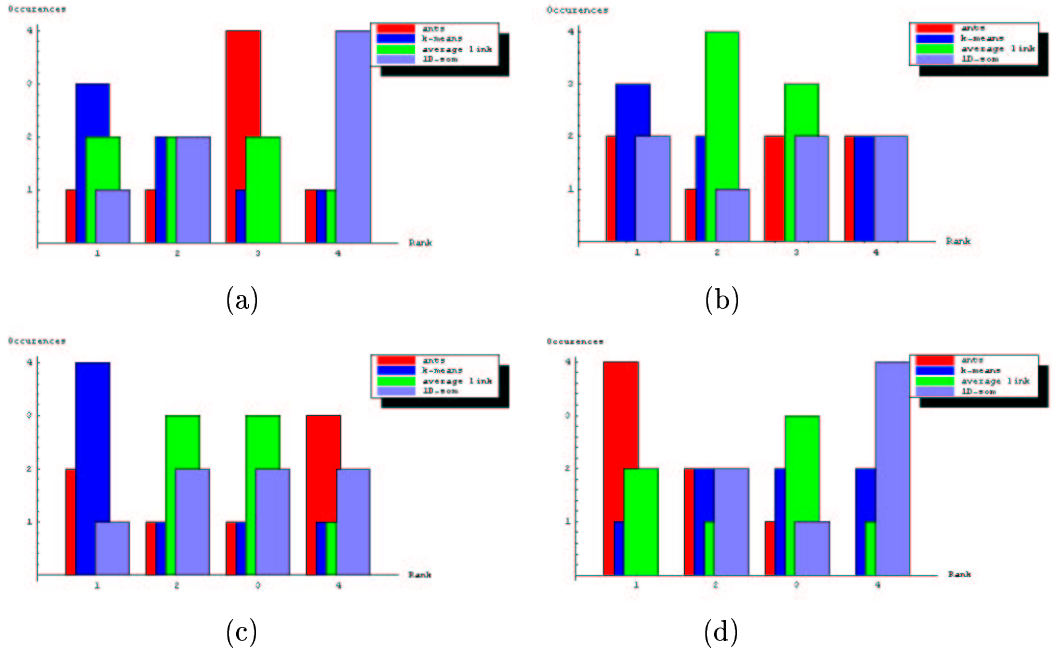


Figure 26: Relative performance of ant-based clustering, K-means, average link agglomerative clustering and one-dimensional self-organising maps on the seven real data sets. For each method, the frequency of the obtained ranks under (a) the F-measure, (b) the Rand index (c) the Intra-cluster variance and (d) the Dunn index is given.

These results indicate that the clusters identified by ant-based clustering 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). Ant-based clustering 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 likewise poor performance of all other clustering methods and the number of clusters predicted by the Gap statistic, which is frequently too low (see Table 8.1).

The results obtained on the *Zoo* data set reveal a second weakness of the ant-based algorithm. Due to the current definition of the neighbourhood function, ant-based clustering requires clusters to have a minimum size in order to construct stable clusters on the grid. The *Zoo* data contains a number of extremely small clusters (the smallest is of size 4) that ant-based

clustering 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 neighbourhood function.

7.7.5 Time performance

For small data sets, the time performance of ant-based clustering is distinctively inferior to that of K-means, average link agglomerative clustering and 1D-SOM. However, it shows a favourable scaling behaviour, such that, with an increase in the dimensionality and the size of the data set, it quickly starts to outperform its contestants (e.g., on the *100D-10C* data set).

Figure 27 gives plots of the runtime for all individual instances of the xD-yC data sets (which differ both in their dimensionality and their size). 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 ant-based clustering all three scale linearly. On the other hand, ant-based clustering, average link agglomerative clustering and 1D-SOM are not affected by the rise in dimensionality, whereas K-means starts to have convergence problems for the higher dimensional data. This causes K-means to exceed the upper limit of iterations (1000 in our implementation) in almost all runs, which results in excessive runtimes.

7.7.6 Discussion

In conclusion, we can say that there are a number of properties that make ant-based clustering 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 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.

On the other hand, the impossibility to specify K can also be considered a disadvantage of ant-based clustering. In specific applications the user might have precise ideas about the number of clusters to be identified, and in ant-based clustering, 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, ant-based clustering'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. K-

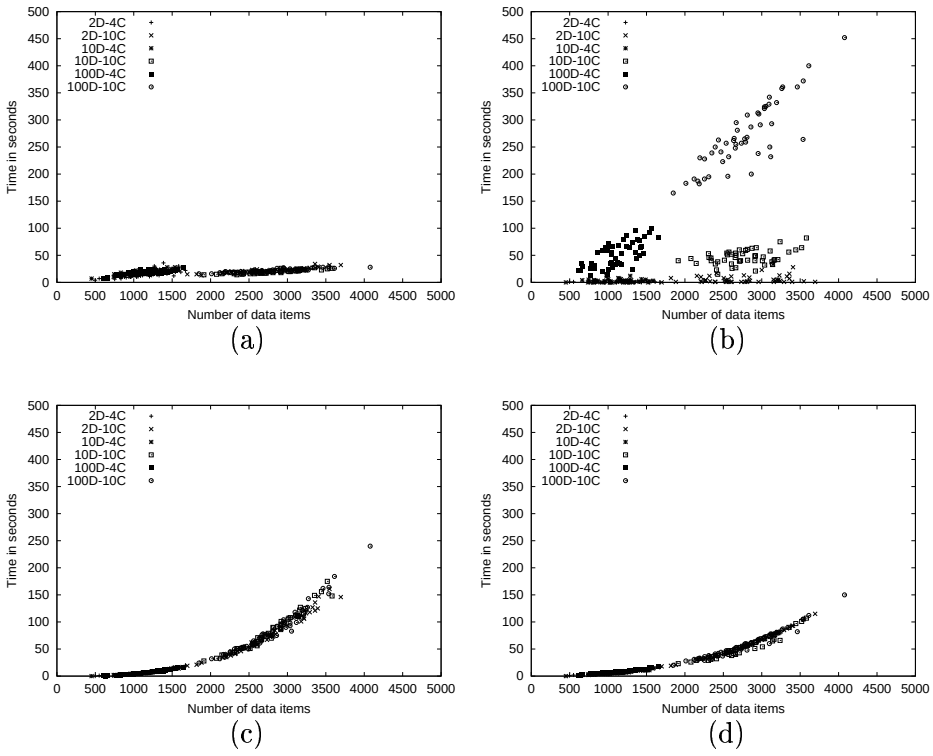


Figure 27: Time performance of ant-based clustering, K-means, average link and 1D-SOM.

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. Yet, the results obtained on real data also show the limitations of ant-based clustering: 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 possible to overcome this problem.

7.8 Results for topographic mapping

In this section, the capacity of ant-based sorting to generate a topology-preserving embedding is analysed using visual inspection and analytical evaluation functions. To put the results into perspective, we compare to those obtained by multidimensional scaling and two-dimensional self-organising maps, and we additionally present results obtained for a random lower bound.

7.8.1 Summary of the performance on synthetic and real data

Extensive experiments on both synthetic and real data permit us to gain an impression of the algorithms' strengths and weaknesses. The data sets used are the same as those employed for our study on cluster analysis, that is, we present results on the *Square*, the *Sizes*, the *xD-yC* and the real data sets from the Machine Learning Repository. The full result tables can again be found in the appendix, and here, we summarise the most interesting results.

Note, however, that we do not discuss the results obtained on the different types of data sets in isolation, as the overall picture is much more uniform for topographic mapping. This is not surprising, if we think about the differences between the two applications, clustering and topographic mapping. A clustering algorithm relies on the correct identification of the structures in the data, and its capacity to identify them very much depends on the properties of the data set tackled (e.g., the degree of spatial separation between clusters). If its fails to discover the principal clusters, this usually results in solutions that are fundamentally different solutions: if a cluster centre is misplaced, a cluster is wrongly split, or two clusters are wrongly merged, this will, in general, cause a change in cluster memberships not only for individual data items but for a large number of elements. Naturally, this is then distinctively reflected by the performance measures. Most methods of topographic mapping, on contrast, work on a much finer level. They do not attempt to identify structures, but their focus is on optimising the map positions of individual data points. For this reasons, changes in a solution are by far more gradual, and the algorithms are therefore less sensitive to particular data properties.

Multidimensional scaling

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, the Spearman Rank correlation and the inter-cluster Pearson correlation.

For the two-dimensional data (e.g., the *Square* data sets, see Figure 28a), where no topological defect needs to occur, all correlation values are close to

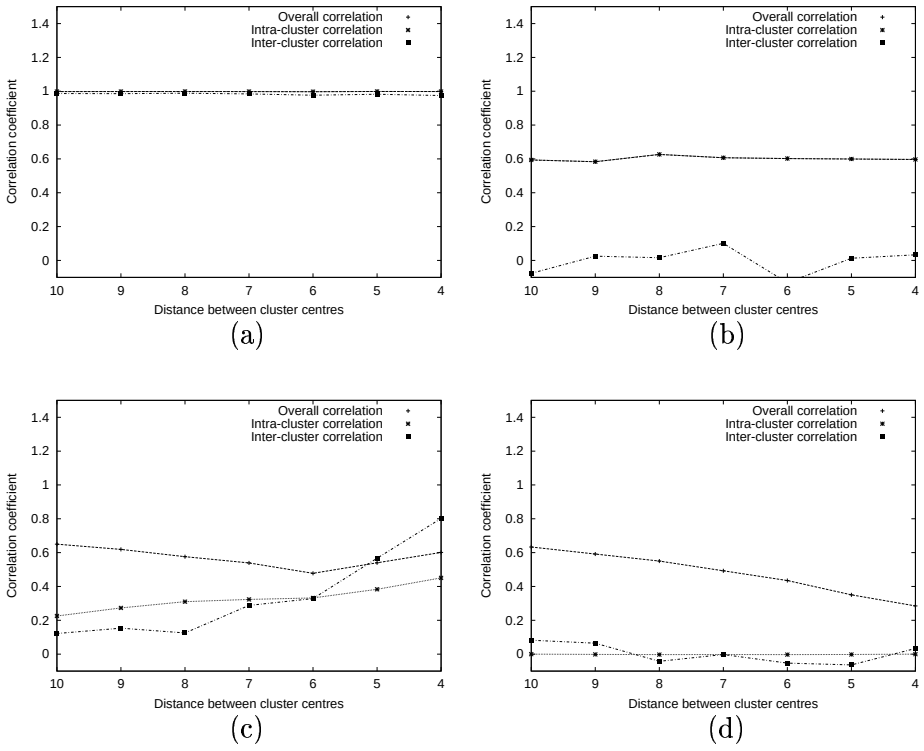


Figure 28: Results under the correlation measures for the *Square* data sets. (a) MDS (b) 2D-SOM (c) ant-based sorting (d) the random lower bound.

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 30a). With an increase in dimensionality both global and local topology-preservation suffer. We show this gradual drop in performance on the *xD-10C* data sets in Figure 29a. An interesting additional aspect of this plot is the increasing gap between the overall Pearson correlation and the intra- and inter-cluster correlation, an issue we will come back to in Section 7.8.2.

Two-dimensional self-organising maps

While MDS clearly preserves local topology to a significant degree (see the plots of the intra-cluster correlation in Figure 28a and Figure 29a), it is, in this respect, repeatedly outperformed by the 2D-SOM, especially for higher-dimensional data.

When looking at the correlation values obtained by 2D-SOM (see Figure 28b and Figure 29b), it is interesting that the overall correlation and the intra-cluster correlation almost perfectly coincide: the overall Pearson correlation obtained seems to be a result of local topology-preservation only. A

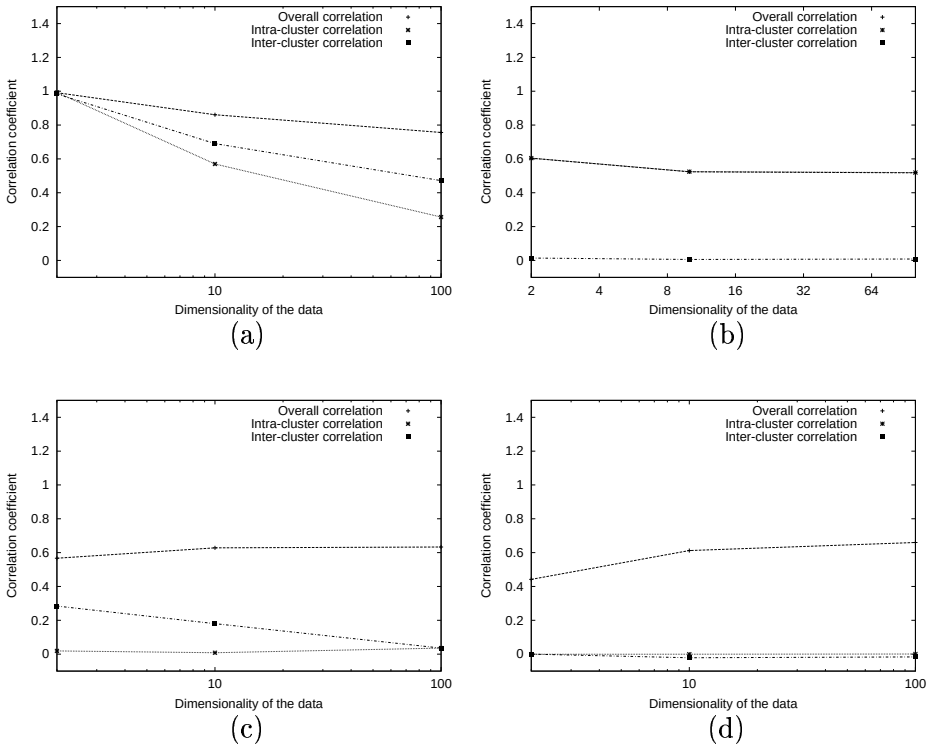


Figure 29: Results under the correlation measures for the xD-10C data sets. (a) MDS (b) 2D-SOM (c) ant-based sorting (d) the random lower bound.

look at 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 mostly randomly determined.

These results are in conformation 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 optimisation 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 a perfect preservation of distance ratios even for two-dimensional data (also see the scatterplot in Figure 30b). Its advantage is, however, that, due to its focus on local topology, it seems to cope much better with a rise in dimensionality.

Ant-based sorting

Ant-based sorting does not perform best under any of the applied evaluation measures. It does, however, obtain reasonably high values under the

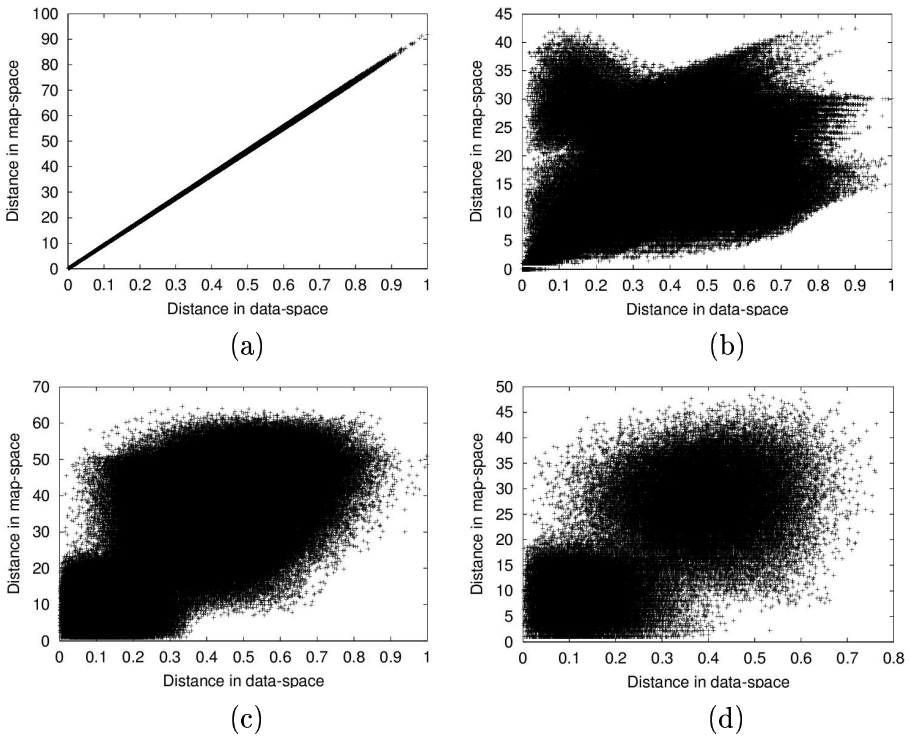


Figure 30: Scatterplots of the distances in data-space versus those in map-space for a solution generated on the *Square1* data set by (a) MDS (b) 2D-SOM (c) ant-based sorting (d) the random lower bound.

measures of the Pearson correlation and the Spearman rank correlation, where it repeatedly beats 2D-SOM. This is in agreement with the results reported by other researchers [48] that have supported the claims on the topology-preserving properties of the algorithm.

However, these results are contradictory to those obtained under all other evaluation measures (the inter-cluster correlation, the intra-cluster correlation and the topographic error): on a large number of data sets, the intra-cluster correlation is very low or even close to zero. Similarly the average inter-cluster correlation is often very low or close to zero, in these cases the standard deviations reveal a wild oscillation, which seems to indicate a random positioning of the clusters (as in the case of 2D-SOM). Scatterplots of the distances in map-space versus those in data-space give an additional idea of how poor topology-preservation is (see Figure 30c). Finally, for the topographic error, the results obtained for ant-based sorting are even weaker: for all data sets the results obtained are very close to the worst obtainable value of 1.0.

7.8.2 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 ant-based sorting seems particularly strange, when one might assume that both inter- and intra cluster correlation comprise the overall correlation value. Our results show that this is not necessarily true.

The high correlation values obtained for ant-based sorting primarily emerge as an artefact 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 correlation values. Note that a similar effect can, to a lesser degree, be observed for MDS on the *xD-yC* data sets. In these data sets, the distance between the individual clusters increases with increasing dimensionality. This is the reason why, in Figure 29a we could observe a growth in the gap between the values obtained for the overall correlation and the inter- and intra-cluster correlations.

7.8.3 Is ant-based 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 ant-based sorting and the lower bound method, clear similarities can be observed (compare Figure 30c and Figure 30d). Analytical evaluation reveals that the Pearson correlation obtained for the lower bound is almost as high as that for ant-based sorting. This is the case for almost all data sets, except for a few of the real data benchmarks: on the *Zoo*, the *Wisoncin*, the *Dermatology*, the *Yeast* and the *Digits* data sets, the correlation values obtained by random cluster mapping are close to zero. This seems to be an indication that at least some of the classes in this 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 ant-based sorting show that those clusters identified by the ant algorithm correspond to distinctive structures within the data (recall that ant-based sorting frequently discovers a too low number of clusters on the real data).

Under the inter-cluster correlation our lower bound method shows an oscillation very similar to that observed in the ant algorithm. However, the

average value is always close to zero, whereas ant-based sorting usually has a positive bias, the degree of which varies for different data sets. As far as local sorting (as reflected by the intra-cluster correlation) is concerned ant-based sorting also outperforms random positioning on several data sets. On the other hand, it is never precise enough to significantly beat the lower bound under the measure of the topographic error.

7.8.4 Data-dependency

It is interesting that, differently to MDS and 2D-SOM, the performance of ant-based clustering varies strongly dependent on the data set tackled. In particular, the results obtained under the intra-cluster and the inter-cluster correlation are far from uniform.

First, the degree of oscillation and the positive bias of the inter-cluster correlation highly differ for different degrees of overlap between the clusters (see Figure 28c). Recalling the clustering results presented in Section 7.7 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 ant-based clustering to identify individual clusters, and the algorithm therefore gathers all four clusters in an individual cluster on the grid. Yet, a 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.

Second, alternating values are obtained for the intra-cluster correlation. Whilst a notable degree of intra-cluster sorting can be observed for the *Square* and some of the real data sets, the intra-cluster correlation is close to zero on the *xD-yC* data sets. This effect is caused by the different distance ratios in these data sets: in the *xD-yC* data, the maximum distance found between elements is very large when compared to the distance between elements belonging to the same cluster. This ratio is by far smaller for the *Square* data, in particular it decreases with an increasing overlap between clusters (see the plot of the intra-cluster correlation in Figure 28c). This permits the ant algorithm to obtain a better intra-cluster sorting, as it succeeds to distinguish more precisely between gradations in distance.[35]

7.8.5 Time performance

For sake of completeness, we also provide graphs on the runtimes of the three algorithms. We show the runtimes for all instances of the *xD-yC* data sets (all of which differ in size). Hence, again, the plots shows both the increase

[35] Note however, that, even then, the algorithm's precision is not sufficient to significantly beat the random lower bound method in terms of the topographic error.

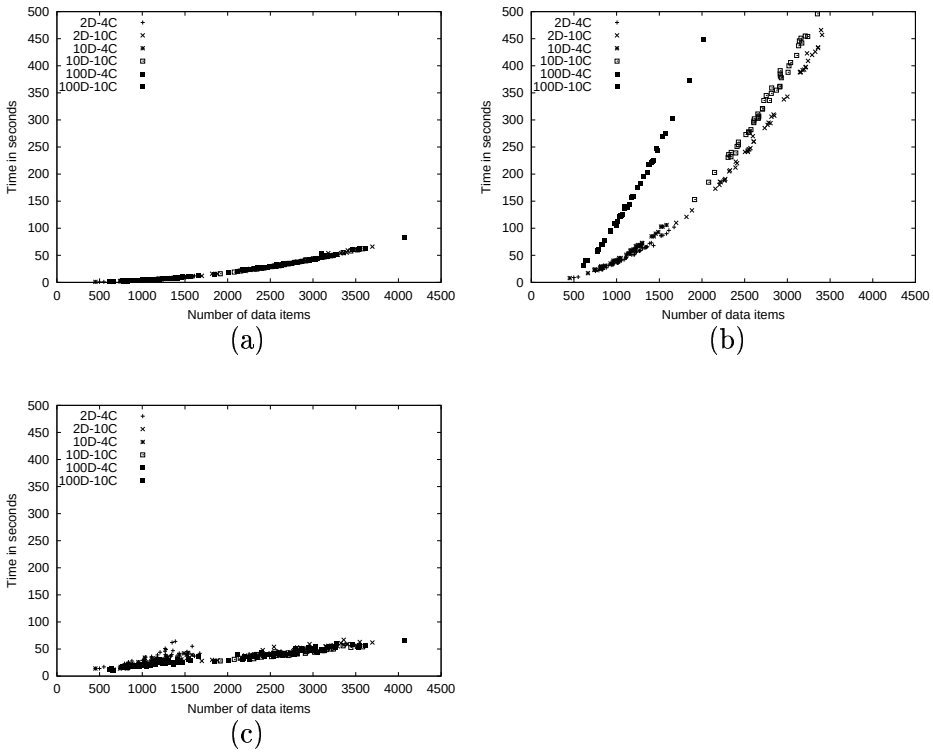


Figure 31: Time performance of (a) MDS (b) 2D-SOM (c) ant-based sorting.

of runtime as a function of the number of data items, and the impact of a rise in dimensionality.

MDS and ant-based sorting both work with precomputed dissimilarity matrices, such that only 2D-SOM is affected by a rise in dimensionality. Additionally, the plots confirm that ant-based sorting is the only one out of the three algorithms that scales linearly with regard to the size of the data set tackled. This is a nice property, but, given the clear inferiority of the produced mapping, it is not very beneficial.

7.8.6 Discussion

In conclusion, we must state that we do not find ant-based sorting to be a satisfactory method for topographic mapping. Both its capacity of 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. In particular, our results indicate that the overall Pearson correlation is unreliable as a means of characterising topology-preservation, unless great care with it is taken. This observation explains why earlier work on ant-based

sorting reported that it exhibited topology-preserving properties, whereas we find little evidence for such behaviour. Using intra- and inter-cluster correlations, and comparison with random mappings, we have been better able to distinguish true topology preservation from 'mere' clustering. We find the latter a more realistic explanation for the sometimes high values of Pearson correlation reported.

While the quantitative performance of MDS is impressive, one disadvantage of the algorithm becomes apparent when visualising data sets that contain a larger number of clusters (e.g., $K = 10$). As the algorithm is primarily concerned with global relationships, local structures can be lost: even if clusters in data-space are well separated, MDS frequently fails to spatially separate them in the two-dimensional representation. This is a limitation, as the clear visibility of cluster structures is definitely an important issue for a good visualisation. 2D-SOM shows similar deficiencies in this respect, as it tends to outstretch clusters on the grid. In the resulting visualisation all clusters touch and cluster boundaries are not clearly revealed.

A two-level approach using a combination of a clustering algorithm and a multidimensional scaling method might therefore be beneficial: after running a clustering algorithm to identify the main clusters, cluster positions could be determined using MDS. The data element of the individual clusters could then again be arranged by MDS. Such a method would have the capacity to preserve both cluster structures and global relationships (up to a certain limit, of course), and simultaneously have a more favourable runtime complexity.

Maintenant le principal est fait. Je tiens quelques évidences dont je ne peux me détacher. Ce que je sais, ce qui est sûr, ce que je ne peux nier, ce que je ne peux rejeter, voilá ce qui compte.
(Camus)

8 Conclusion

In the preceding chapters we have presented the key issues and insights resulting from our work of the last six months. The path towards this point hasn't always been as straight as this thesis might suggest. One property of ant-based clustering and sorting is that the impact of individual algorithmic modifications is very hard to predict. While this is, of course, one of its fascinating properties, it can also make working with the algorithm extremely frustrating when results repeatedly contradict expectation and intuition.

The reader should note that in this thesis we can only describe a fraction of our investigations — it is, however, a summary of those results that we believe are most interesting and meaningful, and we hope that they will serve to answer some of the open questions about ant-based clustering and sorting. In particular, we see the key contributions of this thesis as follows.

- A detailed survey of previous work on ant-based clustering and sorting has been given. We have discussed the results provided by other authors and indicated weaknesses and open questions.
- A new and improved version of ant-based clustering and sorting has been introduced, and we have shown its robust performance across a variety of benchmark data sets.
- Several issues related to the analytical and thorough evaluation of the algorithm have been resolved. These are the derivation of parameter

settings that are generally applicable, a self-adaptation scheme for the data-dependent parameter α, and a method for 'cluster retrieval'.

- A comparison of ant-based clustering to more traditional clustering techniques has been given. The results demonstrated the robust performance of ant-based clustering. The algorithm is largely unaffected by data sets in which the clusters are unequally sized, and it succeeds at reliably separating clusters up to a high degree of overlap. Additionally, an important strength of the algorithm is its capacity to automatically identify the correct number of clusters in the data. However, results on real data also indicated a weakness in this respect: if cluster structures on several levels exist, the algorithm only succeeds to identify the top-level ones.

- A comparison of ant-based sorting to alternative methods for topographic mapping has been given and revealed a very weak performance of the ant algorithm in this respect. The ambiguity of earlier results that has led to their misinterpretation in previous research has been explained and discussed.

The last two points constituted the main goal of this work. We wanted to investigate if (and under what circumstances) the application of ant-based clustering and sorting to real data-mining tasks can make sense, or whether it has to be considered an (interesting) academic toy algorithm. Our results seem to indicate that its use for clustering task can indeed be useful: in cases where the number of clusters in the data is not known, the application of the algorithm may be very interesting. It could either be used on its own, or in combination with another clustering method to provide an initialisation or the input parameter K. As far as topographic mapping is concerned, we would advise against its use, as much better methods exist for this purpose.

8.1 Future work

There are a number of directions in which research on ant-based clustering can be continued. We are convinced that there is still room for improvement of the algorithm, it will however get increasingly harder to obtain more than marginal performance gains. In our eyes, the hybridisation of the algorithm with alternative clustering methods might therefore be a more rewarding and promising line of research.

One aspect of ant-based clustering that we find particularly interesting is its common points with other clustering methods, which we come about when we reflect on the reasons why the algorithm works. One possibility is to contemplate ant-based clustering as a randomised partitioning method with a cluster representation based on representative point sets. Different to other clustering algorithms this point set is not fixed but constantly being

resampled. Ant-based clustering 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 would be a fascinating research topic and might yield a slimmer version of ant-based clustering that incorporates the algorithm's essential mechanisms only.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high-dimensional data for data-mining applications. In *Proceedings of the 1998 International Conference on Management of Data*, pages 94–105. ACM Press, New York, NY, 1998.

[2] P. Albuquerque and A. Dupuis. A parallel cellular ant colony algorithm for clustering and sorting. In *Proceedings of the Fifth International Conference on Cellular Automata for Research and Industry*, pages 220–230. LNCS 2493, Springer-Verlag, Heidelberg, Germany, 2002.

[3] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the 1999 International Conference on Management of Data*, pages 49–60. ACM Press, New York, NY, 1999.

[4] H.-U. Bauer and K. Pawelzik. Quantifiying the neighbourhood preservation of self-organizing feature maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, 1992.

[5] H.-U. Bauer and T. Villman. Growing a hypercubical output space in a self-organizing feature map. *IEEE Transaction on Neural Networks*, 8(2):218–226, 1997.

[6] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, California, 2002. http://citeseer.nj.nec.com/berkhin02survey.html.

[7] J. Bilmes, A. Vahdat, W. Hsu, and E.-J. Im. Empirical observations of probabilistic heuristics for the clustering problem. Technical Report TR-97-018, International Computer Science Institute, University of California, Berkeley, CA, 1997.

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

[9] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence – From Natural to Artificial Systems.* Oxford University Press, New York, NY, 1999.

[10] H. Bourland and Y. Kamp. Autoassociation by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4/5):291–294, 1988.

[11] C. Bregler and S. Omohundro. Surface learning with applications to lip-reading. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 43–50. Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[12] M. Carreira-Perpinan. *Continuous latent variable models for dimensionality reduction and sequential data reconstruction.* PhD thesis, Department of Computer Science, University of Sheffield, UK, February 2001.

[13] D. Corney. *Intelligent Analysis of Small Data Sets for Food Design.* PhD thesis, Department of Computer Science, University College London, UK, September 2002.

[14] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[15] P. Delisle, M. Krajecki, M. Gravel, and C. Gagné. Parallel implementation of an ant colony optimization metaheuristic with openmp. In *International Conference on Parallel Architectures and Compilation Techniques, Proceedings of the Third European Workshop on OpenMP*, 2001.

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

[17] R. Dony and S. Haykin. Optimally adaptive transform coding. *IEEE Transactions on Image Processing*, 4(10):1358–1370, 1995.

[18] M. Dorigo and G. Di Caro. Ant Colony Optimization: A new metaheuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, UK, 1999.

[19] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004. To appear.

[20] J. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974.

[21] M. Ester, H.-P. Kriegel, and J. Sander. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, Menlo Park, CA, 1996.

[22] C. Fiduccia and M. Mattheyses. A linear time heuristic for improving network partitions. In *Proceedings of the Nineteenth IEEE Design Automation Conference*, pages 175–181. IEEE Press, Piscataway, NJ, 1982.

[23] J. Friedman and J. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transaction on Computers*, 23(9):881–889, 1974.

[24] B. Fritzke. Growing cell structures – a self-organizing network in k dimensions. In *Artificial Neural Networks*, volume 2, pages 1051–1056. North-Holland, Amsterdam, Netherlands, 1992.

[25] G. Goodhill, S. Finch, and T. Sejnowski. A unifiying measure for neighbourhood preservation in topogra phic mappings. In *2nd Joint Symposium on Neural Computation*, pages 191–202. Institute for Neural Computation, La Jolla, CA, 1995.

[26] G. Goodhill and T. Sejnowski. Quantifying neighbourhood preservation in topographic mapping s. In *3rd Joint Symposium on Neural Computation*, pages 61–82. Institute for Neural Computation, La Jolla, CA, 1996.

[27] G. Goodhill and T. Sejnowski. Objective functions for topography: a comparison of optimal maps. In *Proceedings of the Fourth Neural Computation and Psychology Workshop: Connectionist Representation.* Springer, Heidelberg, Germany, 1997.

[28] P.-P. Grassé. La reconstruction du nid et les coordinations interindividue lles chez Bellicositermes natalensis et Cubitermes sp. La theorie de la stigmergie: Essai d'interpretation des termites constructeurs. *Insectes Sociaux*, 6:41–80, 1959.

[29] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 73–84. ACM Press, New York, NY, 1998.

[30] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[31] D. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proceedings of the IEEE Conference on Decision and Control*, pages 761–766. IEEE Press, New York, NY, 1979.

[32] H. Gutowitz. Complexity-seeking ants. In *Proceedings of the Third European Conference on Artificial Life*. MIT Press, Cambridge, MA, 1993.

[33] M. Halkidi, M. Vazirgiannis, and I. Batistakis. Quality scheme assessment in the clustering process. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, volume 1910 of *LNCS*, pages 265–267. Springer-Verlag, Heidelberg, Germany, 2000.

[34] J. Handl. Visualising internet-queries using ant-based heuristics. Honours Thesis. Dept. of Computer Science, Monash University, Australia. November 2001.

[35] J. Handl and B. Meyer. Improved ant-based clustering and sorting in a document retrieval interface. In *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature*, volume 2439 of *LNCS*, pages 913–923. Springer-Verlag, Berlin, Germany, 2002.

[36] T. Hastie and W. Stützle. Principal curves. *Journal of the American Statistics Association*, 84(406):502–516, 1989.

[37] G. Hinton, P. Dayan, and M. Revow. Modelling the manifolds of images of handwritten digits. *IEEE Transaction on Neural Networks*, 8(1):64–74, 1997.

[38] K. Hoe, W. Lai, and T. Tai. Homogeneous ants for web document similarity modeling and categorization. In *Proceedings of the Third International Workshop on Ant Algorithms*, volume 2463 of *LNCS*, pages 256–261. Springer-Verlag, Heidelberg, Germany, 2002.

[39] Z. Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *SIGMOD Workshop on Research Issues on Data-Mining and Knowledge Discovery*, 1997.

[40] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

[41] S. Jaccard. Nouvelles recherches sur la distribution florale. In *Bulletin Soc. Vaud. Sci. Nat*, volume 44, pages 223–270, Paris, France, 1908.

[42] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Transactions on Computers*, 32(8):68–75, 1999.

[43] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, New York, NY, 1990.

[44] K. Kiviluoto. Topology preservation in self-organizing maps. In *Proceedings of the International Conference on Neural Networks*, volume 1, pages 294–299. IEEE Press, New York, NY, 1996.

[45] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, Germany, 1995.

[46] P. Koikkalainen. Progress with the tree-structured self-organizing map. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 211–215. John Wiley & Sons, New York, NY, 1994.

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

[48] P. Kuntz and D. Snyers. New results on an ant-based heuristic for highlighting the organization of large graphs. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1451–1458. IEEE Press, Piscataway, NJ, 1999.

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

[50] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):127–138, 1982.

[51] E. Lumer and B. Faieta. 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*, pages 501–508. MIT Press, Cambridge, MA, 1994.

[52] S. Luttrell. A Bayesian analysis of self-organizing maps. *Neural Computation*, 6:767–794, 1994.

[53] L. MacQueen. Some methods for classification and analysis of multivariate observations. In L. LeCam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, Berkeley, 1967.

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

[55] K. Martinetz and K. Schulten. A neural-gas network learns topologies. In *Proceedings of the IEEE International Conference on Artificial Neural Networks*, volume 1, pages 397–402. North-Holland, Amsterdam, The Netherlands, 1991.

[56] M. Middendorf, F. Reischle, and H. Schmeck. Multi colony ant algorithms. *Journal of Heuristics*, 8(3):305–320, 2002.

[57] N. Monmarché. *Algorithmes de fourmis artificielles: applications á la classification et á l'optimisation.* PhD thesis, Laboratoire d'Informatique, Université de Tours, France, December 2000.

[58] H. Nagesh, S. Goil, and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. Technical Report TR-9906-010, Department of Electrical and Computer Engineering, Nort hwestern University, Evanston, IL, 1999.

[59] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 144–155. Morgan Kaufmann Publisher, Los Altos, CA, 1994.

[60] C. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21(8):1313–1325, 1995.

[61] N. Pal and J. Biswas. Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(6):847–857, 1997.

[62] J. Pena, J. Lozano, and P. Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(50):1027–1040, 1999.

[63] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: the art of scientific computing.* Cambridge University Press, Cambridge, UK, 1988.

[64] J. Rahmel. *Topology Preserving Neural Networks – Connectionist Learning of Structured Knowledge.* PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, Germany, October 1997.

[65] V. Ramos and J.J. Merelo. Self-organized stigmergic document maps: Environments as a mechanism for context learning. In *Proceedings of the First Spanish Conference on Evolutionary and Bio-Inspired Algorithms*, pages 284–293. Centro Univ. Mérida, Mérida, Spain, 2002.

[66] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[67] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[68] J. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, 1969.

[69] E. Saund. Dimensionality-reduction using connectionist networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):304–314, 1989.

[70] E. Schikuta. Grid-clustering: A fast hierarchical clustering method for very large data sets. In *Proceedings of the Thirteenth International Conference on Pattern Recognition*, volume 2, pages 101–105. IEEE Press, Los Alamitos, CA, 1996.

[71] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[72] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the Twenty-Fourth Conference on Very Large Databases*, pages 428–439, 1998.

[73] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *SIGKDD Workshop on Text Mining*, 2000. http://citeseer.nj.nec.com/steinbach00comparison.html.

[74] T. Stützle. Parallelization strategies for ant colony optimization. In *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, volume 1498 of *LNCS*, pages 722–731. Springer-Verlag, Heidelberg, Germany, 1998.

[75] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. Technical Report 208, Department of Statistics, Stanford University, 2000. http://citeseer.nj.nec.com/tibshirani00estimating.html.

[76] C. van Rijsbergen. *Information Retrieval, 2nd edition.* Butterworths, London, UK, 1979.

[77] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parkankangas. SOM Toolbox for Matlab 5. Technical Report A57, Neural Networks Research Centre, Helsinki University of Technology, Espoo, Finland, April 2000.

[78] T. Villman, R. Der, M. Hermann, and T. Martinetz. Topology preservation in self-organizing feature maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, 1997.

[79] E. Vorhees. *The effectiveness and efficiency of agglomerative hierarchical clustering in document retrieval.* PhD thesis, Department of Computer Science, Cornell University, 1985.

[80] W. Wang, J. Yang, and R. Mung. Sting: A statistical information grid approach to spatial data-mining. In *Proceedings of the Twenty-Third Conference on Very Large Databases*, pages 186–195. Morgan Kaufmann Publishers, San Francisco, CA, 1997.

[81] F. Young. Multidimensional scaling. In Kotz-Johnson, editor, *Encyclopedia of Statistical Sciences*, volume 5, pages 649–659. Wiley, New York, NY, 1985.

[82] T. Zhang. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the International Conference on Management of Data*, pages 103–114. ACM Press, New York, NY, 1996.

Appendix

Table 6: Average number of clusters identified by ant-based clustering and the Gap statistic on the synthetic and real data sets. For each data set the table shows the correct number of clusters, and the results returned by ant-based clustering and the Gap statistic (means and standard deviations for 50 independent runs). Bold face indicates the better out of the two algorithms.

Data Set	#(Cluster)	ant-based clustering	Gap statistic
Square1	4	**4** (0)	2.16 (0.945727)
Square2	4	**4** (0)	2.48 (1.15308)
Square3	4	**4** (0)	1.68 (1.08517)
Square4	4	**3.74** (0.482079)	1.18 (0.622575)
Square5	4	**1.72** (1.07778)	1.0 (0.0)
Square6	4	**1.02** (0.14)	1.0 (0.0)
Square7	4	**1.02** (0.14)	1.0 (0.0)
Sizes1	4	**4** (0)	2.9 (1.0247)
Sizes2	4	**4** (0)	2.44 (1.08)
Sizes3	4	**3.98** (0.14)	1.94 (0.237487)
Sizes4	4	**4.02** (0.14)	1.8 (0.4)
Sizes5	4	**3.96** (0.195959)	1.82 (0.384187)
2D-4C	4	**4.0** (0.282843)	3.74 (1.1456)
2D-10C	10	**10.4** (1.14891)	2.96 (2.74925)
10D-4C	4	**4.0** (0.0)	4.62 (0.745386)
10D-10C	10	**10.0** (0.0)	9.32 (1.25603)
100D-4C	4	**4.0** (0.0)	4.8 (1.13137)
100D-10C	10	**10.02** (0.14)	9.62 (1.01764)
Iris	3	**3.02** (0.14)	3.1 (0.412311)
Wine	3	**3.0** (0.0)	3.16 (0.366606)
Zoo	7	3.88 (0.430813)	**6.44** (1.45822)
Wisconsin	2	**2.0** (0.0)	4.64 (0.685857)
Yeast	10	**5.36** (1.17915)	2.86 (2.23616)
Dermatology	6	4.36 (0.62482)	**6.28** (1.04)
Digits	10	5.3 (0.806226)	**9.48177** (1.93475)

Table 7: Results for K-means, average link agglomerative clustering, 1D-SOM and ant-based clustering on the *Square1*, *Square2*, *Square3* and *Square4* data sets. The quality of the partitioning is evaluated using the F-measure, the Rand index, the intra-cluster variance and the Dunn index. Cluster numbers (which are automatically determined only for the ant algorithm) and runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best and italic face the third best result out of the four algorithms.

square1	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	**0.988041** (0.00320642)	0.977153 (0.00707426)	*0.979799* (0.00405734)	**0.983819** (0.00435382)
Rand index	**0.988167** (0.00313909)	0.977675 (0.00671268)	*0.980161* (0.00391002)	**0.984049** (0.00423965)
variance	**0.461881** (0.00705653)	0.4669 (0.00863353)	*0.464587* (0.00853489)	**0.463177** (0.00783808)
Dunn index	**3.80823** (0.234739)	*3.50722* (0.261244	3.31952 (0.246082)	**3.67166** (0.257414)
Runtime	**1.76** (1.10562)	*8.84* (0.417612)	**5.86** (0.346987)	16.56 (1.51208)
square2	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	**0.976256** (0.00491627)	0.955406 (0.0217865)	*0.96353* (0.00651575)	**0.969519** (0.00562039)
Rand index	**0.976735** (0.00473009)	0.95777 (0.0168531)	*0.964697* (0.00611194)	**0.970268** (0.00535272)
variance	**0.501385** (0.00755009)	0.515397 (0.0327065)	*0.507721* (0.00845559)	**0.503898** (0.00718242)
Dunn index	**3.62332** (0.204665)	*3.24286* (0.297836)	3.12957 (0.255628)	**3.44473** (0.200323)
Runtime	**1.8** (0.87178)	*8.86* (0.346987)	**5.96** (0.195959)	17.0 (1.44222)
square3	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	**0.954443** (0.00545993)	0.920771 (0.0415269)	*0.937636* (0.00814053)	**0.940436** (0.0088255)
Rand index	**0.956059** (0.00507544)	0.927905 (0.0304267)	*0.94089* (0.00732938)	**0.943159** (0.00800785)
variance	**0.544842** (0.00746298)	0.574133 (0.0600876)	**0.553732** (0.00705513)	*0.556063* (0.0109956)
Dunn index	**3.46342** (0.231103)	*3.06976* (0.313649)	2.90054 (0.246289)	**3.22049** (0.216672)
Runtime	**1.46** (1.02391)	*8.9* (0.412311)	**6.14** (0.4005)	19.82 (2.25113)
square4	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	**0.92108** (0.00881463)	0.853638 (0.0608153)	**0.903298** (0.0112447)	*0.892116* (0.0163831)
Rand index	**0.92583** (0.00776612)	0.872861 (0.0450943)	**0.910911** (0.00938209)	*0.901146* (0.0136203)
variance	**0.592795** (0.00844398)	0.660846 (0.0847222)	**0.602413** (0.00897376)	*0.615184* (0.0137716)
Dunn index	**3.17589** (0.453882)	*2.76794* (0.4123)	2.75991 (0.208915)	**2.9464** (0.214225)
Runtime	**1.12** (1.10707)	*8.84* (0.366606)	**6.0** (0)	19.32 (1.55486)

Table 8: Results for K-means, average link agglomerative clustering, 1D-SOM and ant-based clustering on the *Square5*, *Square7* and *Square7* data sets. The quality of the partitioning is evaluated using the F-measure, the Rand index, the intra-cluster variance and the Dunn index. Cluster numbers (which are automatically determined only for the ant algorithm) and runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best and italic face the third best result out of the four algorithms.

square5	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	3.74 (0.482079)
F-measure	**0.842628** (0.106302)	0.739308 (0.0739826)	**0.844211** (0.0127359)	*0.790195* (0.0608991)
Rand index	**0.855423** (0.106648)	0.780255 (0.0604478)	**0.863385** (0.00937061)	*0.819128* (0.0502641)
variance	**0.670585** (0.121274)	0.797797 (0.116236)	**0.651501** (0.00990396)	*0.725234* (0.0886343)
Dunn index	**2.8305** (0.707372)	2.35972 (0.336774)	**2.54349** (0.235635)	*2.507* (0.364797)
Runtime	**0.68** (0.904212)	*8.94* (0.310483)	**5.98** (0.244131)	20.86 (1.45616)
square6	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	1.72 (1.07778)
F-measure	**0.766903** (0.102714)	*0.634444* (0.0832669)	**0.770903** (0.0143257)	0.491562 (0.131618)
Rand index	**0.792551** (0.12242)	*0.675213* (0.126033)	**0.811144** (0.00912988)	0.409461 (0.220565)
variance	**0.722366** (0.140473)	*0.889196* (0.138232)	**0.698338** (0.00788449)	1.15306 (0.21033)
Dunn index	**2.65029** (0.697334)	*2.27643* (0.266617)	**2.3269** (0.208696)	0.732724 (1.00939)
Runtime	**0.44** (1.20266)	*8.96* (0.28)	**5.56** (0.496387)	23.58 (2.8989)
square7	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	1.02 (0.14)
F-measure	**0.655234** (0.103482)	*0.501443* (0.0761824)	**0.674396** (0.0252717)	0.400392 (0.00274269)
Rand index	**0.718062** (0.115858)	*0.504965* (0.161455)	**0.755204** (0.00996076)	0.252228 (0.0155994)
variance	**0.776582** (0.149036)	*1.0334* (0.149385)	**0.732029** (0.00842417)	1.2799 (0.0165206)
Dunn index	*2.02627* (0.770842)	**2.29646** (0.453341)	**2.09199** (0.17229)	0.036072 (0.252504)
Runtime	**0.3** (0.458258)	*9.06* (0.237487)	**5.68** (0.466476)	31.22 (0.4274)

Table 9: Results for K-means, average link agglomerative clustering, 1D-SOM and ant-based clustering on the $Sizes1$, $Sizes2$, $Sizes3$, $Sizes4$ and $Sizes5$ data sets. The quality of the partitioning is evaluated using the F-measure, the Rand index, the intra-cluster variance and the Dunn index. Cluster numbers (which are automatically determined only for the ant algorithm) and runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best and italic face the third best result out of the four algorithms.

siz^{es1}	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	**0.98885** (0.00314078)	*0.978326* (0.00677288)	0.969586 (0.0138175)	**0.983886** (0.00436117)
Rand index	**0.988728** (0.0032478)	*0.977904* (0.00693731)	0.970237 (0.0138122)	**0.983664** (0.00447799)
variance	**0.468845** (0.00591086)	*0.474835* (0.00789492)	0.478909 (0.0119723)	**0.47158** (0.0070551)
Dunn index	**3.92726** (0.184942)	*3.53855* (0.239409)	3.30924 (0.330923)	**3.67819** (0.20114
Runtime	**0.58** (0.695414)	*8.92* (0.271293)	**5.88** (0.324962)	17.06 (1.33282)
siz^{es2}	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	**0.987388** (0.0036442)	*0.977764* (0.014801)	0.760593 (0.0152671)	**0.985997** (0.00438333)
Rand index	**0.985809** (0.00434821	*0.97629* (0.0109345)	0.791182 (0.00877124)	**0.984** (0.00510024)
variance	**0.501456** (0.00794104)	*0.512786* (0.0258971)	0.623132 (0.00839578)	**0.505443** (0.007825)
Dunn index	**4.12039** (0.218286)	*3.74687* (0.316446)	0.925382 (0.186032)	**3.8295** (0.344227)
Runtime	**0.16** (0.366606)	*9.0* (0)	**5.68** (0.466476)	17.1 (1.13578)
siz^{es3}	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	3.98 (0.14)
F-measure	*0.970663* (0.0778138)	**0.984842** (0.00499006)	0.72883 (0.01171)	**0.984034** (0.0180635)
Rand index	*0.961533* (0.100878)	**0.981967** (0.00683158)	0.73445 (0.00637083)	**0.980536** (0.0201617)
variance	*0.565313* (0.131301)	**0.54156** (0.00776823)	0.62418 (0.0117779)	**0.545097** (0.030049)
Dunn index	**4.14052** (0.791194)	*4.02168* (0.326728)	0.905042 (0.0600104)	**4.03043** (0.379447)
Runtime	**0.42** (1.40129	*9.1* (0.3)	**5.6** (0.489898)	18.02 (1.33402)
siz^{es4}	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4.02 (0.14)
F-measure	*0.95081* (0.111774)	**0.984256** (0.00726937)	0.70401 (0.0113016)	**0.987779** (0.00413597)
Rand index	*0.937989* (0.133983)	**0.979396** (0.0108746)	0.686895 (0.00640328)	**0.984014** (0.00607009)
variance	*0.617169* (0.156553)	**0.579399** (0.0105544)	0.641455 (0.0112817)	**0.577871** (0.00923551)
Dunn index	*4.02909* (1.19551)	**4.17943** (0.349621)	0.943912 (0.0552057)	**4.11362** (0.565701)
Runtime	**0.7** (1.88944)	*9.08* (0.271293)	**5.6** (0.489898)	19.4 (0.916515)
siz^{es5}	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	3.96 (0.195959)
F-measure	*0.949652* (0.109392)	**0.985862** (0.00626907)	0.678249 (0.0160914)	**0.985737** (0.0186955)
Rand index	*0.929428* (0.147586)	**0.981381** (0.00949521)	0.641374 (0.0108568)	**0.981681** (0.0215735)
variance	*0.650721* (0.152824)	**0.609517** (0.0112156)	0.661137 (0.0113542)	**0.611679** (0.0321076)
Dunn index	*4.2761* (1.34627)	**4.28963** (0.362929)	0.979409 (0.0692235)	**4.34447** (0.453915)
Runtime	**0.26** (0.438634)	*9.3* (0.538516)	**6.36** (0.685857)	20.76 (1.42211)

Table 10: Results for K-means, average link agglomerative clustering, 1D-SOM and ant-based clustering on the *2D-4C*, *2D-10C*, *10D-4C* and *10D-10C* data sets. The quality of the partitioning is evaluated using the F-measure, the Rand index, the intra-cluster variance and the Dunn index. Cluster numbers (which are automatically determined only for the ant algorithm) and runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best and italic face the third best result out of the four algorithms.

2D-4C	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0.282843)
F-measure	*0.972734* (0.0740772)	**0.997365** (0.018445)	0.953895 (0.0560186)	**0.990371** (0.0354898)
Rand index	*0.983066* (0.0464064)	**0.998241** (0.012311)	0.971536 (0.036236)	**0.99155** (0.031725)
variance	*0.177598* (0.16193)	**0.127346** (0.0372185)	0.199594 (0.103416)	**0.133784** (0.062165)
Dunn index	*4.45335* (2.09255)	**5.10569** (1.61428)	3.68615 (2.12704)	**5.02245** (1.78885)
Runtime	**0.74** (1.09197)	**6.8** (4.36807)	*7.64* (3.39859)	15.68 (6.60739)
2D-10C	K-means	average link	SOM	ant-based clustering
#(Cluster)	10 (0)	10 (0)	10 (0)	10.4 (1.14891)
F-measure	0.925724 (0.0745782)	**0.99152** (0.0208668)	*0.938804* (0.0390634)	**0.972183** (0.0362037)
Rand index	0.9792 (0.0244765)	**0.997517** 0.00784073)	*0.982783* (0.0145204)	**0.991252** 0.0129368)
variance	*0.129456* (0.0701149)	**0.0926216** (0.013906)	0.139949 (0.031185)	**0.0971026** (0.0187272)
Dunn index	*1.49877* (1.12043)	**2.82852** (0.677426)	1.25126 (0.770043)	**1.87951** (1.55127)
Runtime	**6.08** (7.14658)	77.58 (36.7962)	*57.18* (24.0804)	**21.96** (4.40436)
10D-4C	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	*0.991297* (0.0444016)	**1.0** (0.0)	0.97624 (0.0424158)	**1.0** (0.0)
Rand index	*0.995198* (0.0263031)	**1.0** (0.0)	0.987584 (0.0241659)	**1.0** (0.0)
variance	*0.517712* (0.232325)	**0.479945** (0.0891802)	0.632428 (0.26593)	**0.479945** (0.0891802)
Dunn index	*13.5572* (5.1778)	**13.9513** (4.61918)	10.5125 (5.78808)	**13.9513** (4.61918)
Runtime	**2.74** (2.71153)	**5.9** (3.91024)	*6.22* (3.02185)	17.42 (4.56986)
10D-10C	K-means	average link	SOM	ant-based clustering
#(Cluster)	10 (0)	10 (0)	10 (0)	10 (0)
F-measure	*0.971745* (0.0410038)	**1.0** (0.0)	0.956233 (0.0283843)	**0.999986** (0.0000960888)
Rand index	*0.993207* (0.0118778)	**1.0** (0.0)	0.990942 (0.00765172)	**0.999995** (0.0000318648)
variance	*0.411021* (0.160722)	**0.331714** (0.0305796)	0.517608 (0.119397)	**0.331777** (0.0305041)
Dunn index	*8.22036* (6.67596)	**13.6787** (1.90435)	4.18333 (4.70773)	**13.445** (2.36789)
Runtime	**46.04** (13.1559)	83.26 (33.2955)	*54.64* (19.1141)	**19.54** (3.21378)

Table 11: Results for K-means, average link agglomerative clustering, 1D-SOM and ant-based clustering on the *100D-4C* and *100D-10C* data sets. The quality of the partitioning is evaluated using the F-measure, the Rand index, the intra-cluster variance and the Dunn index. Cluster numbers (which are automatically determined only for the ant algorithm) and runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best and italic face the third best result out of the four algorithms.

100D-4C	K-means	average link	SOM	ant-based clustering
#(Cluster)	4 (0)	4 (0)	4 (0)	4 (0)
F-measure	0.975685 (0.0586472)	**1.0** (0.0)	*0.98469* (0.031972)	**1.0** (0.0)
Rand index	0.98752 (0.0310639)	**1.0** (0.0)	*0.994124* (0.0131139)	**1.0** (0.0)
variance	*1.9766* (0.777986)	**1.65947** (0.157138)	2.01234 (0.763501)	**1.65947** (0.157138)
Dunn index	*05142* (23.4023)	**58.8131** (11.5432)	48.975 (22.7296)	**58.8131** (11.5432)
Runtime	51.72 (23.0834)	**6.34** (3.91208)	**8.26** (3.64313)	*042* (2.96709)
100D-10C	K-means	average link	SOM	ant-based clustering
#(Cluster)	10 (0)	10 (0)	10 (0)	10.02 (0.14)
F-measure	*0.973905* (0.03722)	**1.0** (0.0)	0.956352 (0.0281009)	**0.999174** (0.00578442)
Rand index	*0.992684* (0.0109855)	**1.0** (0.0)	0.992309 (0.00648917)	**0.999728** (0.00190061)
variance	*1.55422* (0.567023)	**1.13759** (0.038661)	1.8793 (0.418234)	**1.13758** (0.0386526)
Dunn index	*54.8165* (39.6492)	**84.9771** (8.84828)	11.6349 (21.2545)	**83.2411** (14.7135)
Runtime	271.48 (61.301)	*85.42* (42.6371)	**59.24** (24.7148)	**21.76** (3.17213)

Table 12: Results for K-means, average link agglomerative clustering, 1D-SOM and ant-based clustering on the *IRIS*, *WINE*, *ZOO* and *WISCONSIN* data sets. The quality of the partitioning is evaluated using the F-measure, the Rand index, the intra-cluster variance and the Dunn index. Cluster numbers (which are automatically determined only for the ant algorithm) and runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best and italic face the third best result out of the four algorithms.

IRIS	K-means	average link	SOM	ant-based clustering
#(Cluster)	3 (0)	3 (0)	3 (0)	3.02 (0.14)
F-measure	**0.824521** (0.0848664)	0.809857 (0.0)	**0.861415** (0.00773914)	*0.816812* (0.0148461)
Rand index	0.816599 (0.101288)	*0.822311* (0.0)	**0.857342** (0.00554881)	**0.825422** (0.00804509)
variance	0.922221 (0.221044)	*0.900175* (0.0)	**0.894906** (0.00147313)	**0.880333** (0.00405812)
Dunn index	**2.65093** (0.4201)	*2.5186* (0.0)	2.07881 (0.253375)	**2.9215** (0.298826)
Runtime	*0.16* (0.366606)	**0.02** (0.14)	**0.08** (0.271293)	3.36 (0.48)
WINE	K-means	average link	SOM	ant-based clustering
#(Cluster)	3 (0)	3 (0)	3 (0)	3 (0)
F-measure	**0.931275** (0.0615274)	**0.925527** (0.0)	0.844292 (0.00492265)	*0.876083* (0.0208369)
Rand index	**0.916761** (0.065321)	**0.904494** (0.0)	0.825494 (0.00444023)	*0.855493* (0.0191702)
variance	*2.58276* (0.089766)	**2.5783** (0.0)	2.58392 (0.00191894)	**2.56946** (0.0125923)
Dunn index	3.7622 (0.354202)	*3.94448* (0.0)	**4.03727** (0.15908)	**4.18856** (0.207914)
Runtime	**0.08** (0.271293)	**0.02** (0.14)	*0.12* (0.324962)	2.82 (0.433128)
ZOO	K-means	average link	SOM	ant-based clustering
#(Cluster)	7 (0)	7 (0)	7 (0)	3.88 (0.430813)
F-measure	0.793335 (0.0683366)	**0.839231** (0.0)	**0.826222** (0.0569534)	*0.81886* (0.0470189)
Rand index	0.886615 (0.0320711)	**0.926674** (0.0)	*0.916267* (0.0291636)	**0.937202** (0.0383113)
variance	**2.15164** (0.0529296)	*2.27059* (0.0)	**2.23617** (0.0585116)	2.5721 (0.146938)
Dunn index	*3.9967* (0.911945)	**6.15496** (0.0)	3.9438 (1.10343)	**6.07854** (0.693924)
Runtime	**0.06** (0.237487)	**0** (0)	*0.08* (0.271293)	2.6 (0.489898)
WISCONSIN	K-means	average link	SOM	ant-based clustering
#(Cluster)	2 (0)	2 (0)	2 (0)	2 (0)
F-measure	*0.965825* (0.0)	**0.965966** (0.0)	0.965766 (0.000861166)	**0.967604** (0.00144665)
Rand index	**0.933688** (0.0)	**0.933688** (0.0)	0.933583 (0.00159676)	**0.93711** (0.00273506)
variance	**1.61493** (0.0)	1.63441 (0.0)	*1.61494* (0.000870785)	**1.61257** (0.000838131)
Dunn index	**5.47121** (0.000178411)	4.91649 (0.000000284355)	**5.45811** (0.015952)	*5.4424* (0.0957218)
Runtime	**0.06** (0.237487)	**1.44** (0.496387)	*2.32* (0.466476)	10.54 (0.498397)

Table 13: Results for K-means, average link agglomerative clustering, 1D-SOM, ant-based sorting on the *YEAST*, *DERMATOLOGY* and *DIGITS* data sets. The quality of the partitioning is evaluated using the F-measure, the Rand index, the intra-cluster variance and the Dunn index. Cluster numbers (which are automatically determined only for the ant algorithm) and runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best and italic face the third best result out of the four algorithms.

YEAST	K-means	average link	SOM	ant-based clustering
#(Cluster)	10 (0)	10 (0)	10 (0)	5.36 (1.17915)
F-measure	*0.431505* (0.00443954)	**0.448316**	0.406728 (0.0174483)	**0.435396** (0.0345797)
Rand index	**0.750657** (0.00124985)	*0.742682*	**0.75227** (0.00267426)	0.678131 (0.0752791)
variance	**1.53798** (0.001611)	**1.60028** (0.00000000652216)	*1.69317* (0.0201931)	1.89537 (0.115468)
Dunn index	**1.69692** (0.109608)	*1.55563* (0.000000120366)	1.22087 (0.154187)	**1.88049** (0.207959)
Runtime	**1.7** (1.0247)	14.04 (0.488262)	*12.16* (0.417612	**9.22** (0.54)
DERMATOLOGY	K-means	average link	SOM	ant-based clustering
#(Cluster)	6 (0)	6 (0)	6 (0)	4.36 (0.62482)
F-measure	**0.948479** (0.0264577)	**0.898637** (0.0)	0.806417 (0.0139962)	*0.845738* (0.0491325)
Rand index	**0.973853** (0.0134017)	**0.95497** (0.0)	0.904861 (0.00932017)	*0.921871* (0.0345612)
variance	**3.79081** (0.00294317)	**3.85732** (0.00000000876454)	4.05837 (0.0202633)	*4.04908* (0.141142)
Dunn index	3.8394 (0.0706464)	**4.70806** (0.00000014988)	*3.84977* (0.239219)	**5.32128** (0.402198)
Runtime	**0.34** (0.514198)	4.70806 (0.00000014988)	**0.58** (0.493559)	*2.84* (0.463033)
DIGITS	K-means	average link	SOM	ant-based clustering
#(Cluster)	10 (0)	10 (0)	10 (0)	5.3 (0.806226)
F-measure	**0.731753** (0.017948)	*0.596675* (0.0)	**0.630484** (0.0225114)	0.503542 (0.0310281)
Rand index	**0.920426** (0.00662569)	*0.865701* (0.0)	**0.897649** (0.00533422)	0.809809 (0.0312331)
variance	**2.13684** (0.00386531)	*2.39675* (0.0000000155022)	**2.24289** (0.0207932)	2.7052 (0.128086)
Dunn index	*3.15963* (0.0960701)	**3.7286** (0.00000047151)	2.5773 (0.239102)	**3.68081** (0.295242)
Runtime	**6.5** (3.25115)	174.12 (3.25355)	*9.9* (2.71477)	**32.34** (0.907965)

Table 14: Results for MDS, 2D-SOM, ant-based sorting and the lower bound on the *Square1*, *Square2*, *Square3* and *Square4* data sets. The quality of the mapping is evaluated using the overall Pearson correlation, the inter- and intra-cluster Pearson correlation, the Spearman rank correlation and the topographic error. Runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best result and italic fact the third best out of the four algorithms.

square1	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998052** (0.00223397)	0.593754 (0.134905)	**0.649971** (0.0895103)	*0.633483* (0.0800831)
Inter	**0.98658** (0.0141674)	-0.0760479 (0.42808)	**0.122131** (0.522214)	*0.0824321* (0.441095)
Intra	**0.998264** (0.00203184)	**0.593944** (0.134624)	*0.225642* (0.0337886)	0.000083418 (0.00990187)
Toperror	**0.33082** (0.0327443)	**0.18664** (0.0233425)	0.98802 (0.00433354)	*0.98686* (0.00432671)
Spearman	**0.985936** (0.00616601)	**0.585377** (0.139087)	*0.572273* (0.105737)	0.558739 (0.0894701)
Runtime	**7.64** (2.33889)	40.44 (0.962497)	*26.36* (8.39466)	**0.02** (0.14)
square2	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.99816** (0.0023534)	0.583481 (0.128066)	**0.619098** (0.0841792)	*0.591872* (0.0978331)
Inter	**0.985898** (0.0157033)	0.0253889 (0.411938)	**0.153803** (0.475741)	*0.0651346* (0.484388)
Intra	**0.998359** (0.00213593)	**0.583655** (0.128151)	*0.273298* (0.0278273)	-0.00145089 (0.00995137)
Toperror	**0.35834** (0.0408938)	**0.1861** (0.0219611)	0.98804 (0.00390364)	*0.98682* (0.00467628)
Spearman	**0.990434** (0.00483227)	**0.578825** (0.132345)	*0.556479* (0.0905213)	0.535054 (0.0954124)
Runtime	**6.62** (0.485386)	41.44 (1.15169)	*20.18* (1.38116)	**0.02** (0.14)
square3	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998461** (0.00228256)	**0.626441** (0.143237)	*0.576335* (0.0899368)	0.550222 (0.0820071)
Inter	**0.988202** (0.01347)	*0.0161074* (0.331508)	**0.125454** (0.485955)	-0.0420097 (0.497772)
Intra	**0.998656** (0.00199927)	**0.626671** (0.143454)	*0.310412* (0.0207632)	-0.00232552 (0.0119472)
Toperror	**0.36572** (0.0329381)	**0.18052** (0.0225524)	0.98842 (0.00540404)	*0.9882* (0.00419047)
Spearman	**0.994399** (0.00309339)	**0.627103** (0.144352)	*0.523253* (0.0947519)	0.495671 (0.0820482)
Runtime	**6.76** (0.471593)	40.94 (1.1386)	*20.96* (1.68476)	**0.06** (0.237487)
square4	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998018** (0.00283233)	**0.606938** (0.135459)	*0.539295* (0.0740858)	0.492626 (0.0933278)
Inter	**0.984031** (0.0160746)	*0.101099* (0.384349)	**0.288078** (0.432039)	-0.00194626 (0.493937)
Intra	**0.998279** (0.00237883)	**0.606973** (0.135395)	*0.32363* (0.0346576)	-0.00226389 (0.0129229)
Toperror	**0.38678** (0.0376336)	**0.18276** (0.024563)	*0.98816* (0.00453149)	0.98852 (0.00426727)
Spearman	**0.995126** (0.00293726)	**0.60921** (0.134109)	*0.500651* (0.0825801)	0.466379 (0.084985)
Runtime	**6.76** (0.51225)	41.26 (0.97591)	*22.88* (1.8071)	**0.04** (0.195959)

Table 15: Results for MDS, 2D-SOM, ant-based sorting and the lower bound on the *Square5*, *Square6* and *Square7* data sets. The quality of the mapping is evaluated using the overall Pearson correlation, the inter- and intra-cluster Pearson correlation, the Spearman rank correlation and the topographic error. Runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best result and italic fact the third best out of the four algorithms.

square5	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.997062** (0.00387012)	**0.602062** (0.15058)	*0.477937* (0.0675938)	0.435375 (0.0682281)
Inter	**0.976511** (0.0235049)	-0.131523 (0.39489)	**0.328466** (0.319278)	*-0.0528227* (0.484992)
Intra	**0.997453** (0.0033401)	**0.602601** (0.150517)	*0.332467* (0.034818)	-0.00250014 (0.00821582)
Toperror	**0.3739** (0.0385721)	**0.18156** (0.0255532)	*0.98976* (0.00450138)	0.9902 (0.00373631)
Spearman	**0.995832** (0.00351596)	**0.60921** (0.150595)	*0.451446* (0.0671786)	0.410945 (0.0699801)
Runtime	**7.18** (1.21145)	53.68 (7.6248)	*27.74* (6.60548)	**0.04** (0.195959)
square6	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998675** (0.00148479)	**0.59929** (0.143526)	*0.539863* (0.124996)	0.35057 (0.0611516)
Inter	**0.982383** (0.0121828)	*0.0132959* (0.302664)	**0.56711** (0.339713)	-0.0633044 (0.443296)
Intra	**0.99879** (0.0013935)	**0.599941** (0.143397)	*0.383291* (0.0709834)	-0.00171855 (0.0115306)
Toperror	**0.36056** (0.0375042)	**0.17914** (0.0289876)	*0.9875* (0.00516624)	0.99112 (0.00339199)
Spearman	**0.998119** (0.00139441)	**0.607054** (0.143968)	*0.54539* (0.122962)	0.334897 (0.0574042)
Runtime	**6.52** (0.4996)	40.02 (0.734575)	*26.1* (7.22011)	**0.04** (0.195959)
square7	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998168** (0.00307702)	*0.596644* (0.132162)	**0.601304** (0.0910511)	0.285073 (0.0451556)
Inter	**0.974787** (0.0198741)	0.0335117 (0.372051)	**0.802072** (0.287686)	*0.0335516* (0.446518)
Intra	**0.9983** (0.00285127)	**0.597087** (0.132306)	*0.45104* (0.0838871)	0.000221359 (0.0106784)
Toperror	**0.33594** (0.0355789)	**0.18158** (0.0292336)	*0.9872* (0.00408901)	0.99204 (0.00334042)
Spearman	**0.998087** (0.00242407)	**0.606822** (0.133915)	*0.595083* (0.0946682)	0.267373 (0.0459184)
Runtime	**6.5** (0.5)	40.16 (0.611882)	*31.96* (8.34736)	**0.0** (0.0)

Table 16: Results for MDS, 2D-SOM, ant-based sorting and the lower bound on the *Sizes1*, *Sizes2*, *Sizes3*, *Sizes4* and *Sizes5* data sets. The quality of the mapping is evaluated using the overall Pearson correlation, the inter- and intra-cluster Pearson correlation, the Spearman rank correlation and the topographic error. Runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best result and italic fact the third best out of the four algorithms.

size1	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.99894** (0.00151814)	0.595063 (0.113708)	**0.664638** (0.0796616)	*0.626958* (0.0784801)
Inter	**0.990254** (0.0107545)	*-0.0182768* (0.385184)	**0.129883** (0.488785)	-0.0290228 (0.38463)
Intra	**0.999004** (0.00145053)	**0.595026** (0.113355)	*0.226773* (0.0357858)	-0.000120192 (0.00992925)
Toperror	**0.31718** (0.0280183)	**0.18378** (0.0212549)	0.98774 (0.00343983)	*0.98626* (0.0043672)
Spearman	**0.988081** (0.00520663)	*0.591721* (0.114082)	**0.603931** (0.0896666)	0.570691 (0.0741016
Runtime	**6.52** (0.4996)	40.12 (0.552811)	*19.62* (1.63573)	**0.04** (0.195959)
size2	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998515** (0.00171947)	0.52041 (0.128352)	**0.705604** (0.0605257)	*0.666442* (0.0842159)
Inter	**0.983255** (0.0174002)	*-0.000170522* (0.332815)	**0.093993** (0.431632)	-0.0153753 (0.514561)
Intra	**0.998254** (0.00196566)	**0.521315** (0.129004)	*0.235049* (0.0373928)	-0.000916508 (0.0110987)
Toperror	**0.30354** (0.0326247)	**0.18524** (0.0240753)	0.98954 (0.0045087)	*0.98838* (0.00516484)
Spearman	**0.991824** (0.00336719)	0.530576 (0.128352)	**0.69805** (0.054497)	*0.661191* (0.0720088)
Runtime	**6.5** (0.5)	39.98 (0.616117)	*19.54* (1.16979)	**0.02** (0.14)
size3	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998603** (0.00163089)	0.498574 (0.153093)	**0.713627** (0.0584983)	*0.689128* (0.0803637)
Inter	**0.987588** (0.00981932)	0.0099596 (0.352388)	**0.0639132** (0.474238)	*0.0516798* (0.524273)
Intra	**0.989054** (0.024263)	**0.499576** (0.15194)	*0.226497* (0.0502727)	-0.000797458 (0.0110066)
Toperror	**0.2779** (0.0260002)	**0.1877** (0.0255814)	*0.98832* (0.00434714)	0.98874 (0.00339299)
Spearman	**0.994412** (0.00270917)	0.520764 (0.155495)	**0.719256** (0.0392552)	*0.691499* (0.0555984)
Runtime	**6.5** (0.5)	40.18 (0.653911)	*0.62*(1.31149)	**0.02** (0.14)
size4	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998017** (0.00244587)	0.482008 (0.120913)	**0.711733** (0.051975)	*0.68275* (0.0737974)
Inter	**0.976541** (0.0231726)	**0.0383082** (0.30288)	*0.0164102* (0.451099)	-0.0580613 (0.443404)
Intra	**0.991305** (0.0201628)	**0.48042** (0.122503)	*0.226325* (0.0483923)	0.000428467 (0.0172332)
Toperror	**0.26594** (0.0214405)	**0.19016** (0.0224752)	0.98864 (0.00416538)	*0.9883* (0.00463573)
Spearman	**0.9951** (0.00205092)	0.51418 (0.11906)	**0.713411** (0.0296426)	*0.670571* (0.0545861)
Runtime	**6.5** (0.5)	40 (0.6)	*21.62* (1.05622)	**0.06** (0.237487)
size5	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.998775** (0.00142566)	0.468981 (0.123809)	**0.714699** (0.0526231)	*0.682098* (0.0715692)
Inter	**0.981139** (0.0155562)	*0.0384139* (0.331701)	**0.0542928** (0.408042)	-0.000318306 (0.437207)
Intra	**0.990384** (0.0236018)	**0.468036** (0.12355)	*0.224297* (0.0612896)	-0.00125337 (0.0153861)
Toperror	**0.25028** (0.0213664)	**0.18938** (0.0275426)	0.98882 (0.00366437)	*0.98856* (0.00406527)
Spearman	**0.996703** (0.00133379)	0.51496 (0.126982)	**0.694167** (0.0296506)	*0.6403* (0.0478808)
Runtime	**4.48** (0.607947)	44.52 (14.1467)	*18.94* (1.74826)	**0.0** (0.0)

Table 17: Results for MDS, 2D-SOM, ant-based sorting and the lower bound on the *2D-4C*, *2D-10C*, *10D-4C* and *10D-10C* data sets. The quality of the mapping is evaluated using the overall Pearson correlation, the inter- and intra-cluster Pearson correlation, the Spearman rank correlation and the topographic error. Runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best result and italic fact the third best out of the four algorithms.

2D-4C	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.980987** (0.0208763)	0.662493 (0.0750375)	**0.711142** (0.117188)	*0.683264* (0.122291)
Inter	**0.866605** (0.187444)	**0.00388271** (0.379593)	*-0.0202273* (0.511556)	-0.0569968 (0.413447)
Intra	**0.967465** (0.0641785)	**0.662079** (0.0760368)	*0.0269852* (0.04267)	-0.00220186 (0.0112236)
Toperror	**0.0404122** (0.0184413)	**0.173258** (0.0264479)	0.989658 (0.00417041)	*0.988517* (0.00452798)
Spearman	**0.801872** (0.102195)	**0.679568** (0.0778839)	*0.609252* (0.131988)	0.593275 (0.117368)
Runtime	**5.98** (2.70917)	49.54 (22.0791)	*3.0* (11.5966)	**0.0** (0.0)
2D-10C	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.991595** (0.0115903)	**0.604537** (0.138094)	*0.567181* (0.0972705)	0.442158 (0.097196)
Inter	**0.983814** (0.0222875)	*0.0142648* (0.180278)	**0.284452** (0.159889)	0.0000611714 (0.141343)
Intra	**0.993106** (0.0173586)	**0.604585** (0.13803)	*0.0190925* (0.0116238)	-0.000602806 (0.00667409)
Toperror	**0.0421871** (0.0138191)	**0.174887** (0.0177502)	0.990285 (0.00291396)	*0.98816* (0.00268826)
Spearman	**0.88141** (0.0482212)	**0.599231** (0.142089)	*0.493767* (0.111133)	0.325025 (0.112398)
Runtime	**37.04** (13.2679)	300.08 (105.614)	*45.06* (9.03418)	**0.04** (0.195959)
10D-4C	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.955704** (0.0302416)	0.594676 (0.0560784)	**0.8037** (0.0505642)	*0.801205* (0.0788317)
Inter	**0.61285** (0.320177)	0.0261163 (0.471431)	**0.103564** (0.422442)	*0.0554841* (0.428656)
Intra	**0.717532** (0.0844408)	**0.594984** (0.0563076)	*0.0232682* (0.0286069)	0.000143623 (0.0106157)
Toperror	**0.611766** (0.109569)	**0.38358** (0.037626)	0.989093 (0.00580587)	*0.986344* (0.00689397)
Spearman	**0.699344** (0.117308)	0.542985 (0.079302)	*0.6305* (0.104271)	**0.632799** (0.12192)
Runtime	**5.3** (2.53969)	52.18 (23.9997)	*24.74* (8.74027)	**0.02** (0.14)
10D-10C	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.86113** (0.0277)	0.524181 (0.0598761)	**0.628333** (0.0562995)	*0.612477* (0.0544629)
Inter	**0.690403** (0.083748)	*0.00606256* (0.151321)	**0.180513** (0.157866)	-0.0211447 (0.101533)
Intra	**0.570103** (0.0413675)	**0.524741** (0.0592961)	*0.0086634* (0.0101216)	-0.000113265 (0.00848473)
Toperror	**0.645234** (0.0751831)	**0.423383** (0.01799)	0.990372 (0.00258523)	*0.987981* (0.00276698)
Spearman	**0.695228** (0.0677271)	*0.407863* (0.109523)	**0.429298** (0.0950472)	0.323902 (0.0888157)
Runtime	**36.88** (10.3935)	342.58 (92.6825)	*4.98* (5.89742)	**0.0** (0.0)

Table 18: Results for MDS, 2D-SOM, ant-based sorting and the lower bound on the *100D-4C* and *100D-10C* data sets. The quality of the mapping is evaluated using the overall Pearson correlation, the inter- and intra-cluster Pearson correlation, the Spearman rank correlation and the topographic error. Runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best result and italic fact the third best out of the four algorithms.

100D-4C	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.941623** (0.0212687)	0.60636 (0.0460589)	**0.832398** (0.0368285)	*0.818357* (0.055765)
Inter	**0.198686** (0.410089)	**0.0675924** (0.530676)	0.00161503 (0.438826)	*0.0264746* (0.475601)
Intra	**0.484044** (0.0754636)	**0.6064** (0.0470262)	*0.0161612* (0.021677)	0.000262384 (0.0131929)
Toperror	**0.666086** (0.0829733)	**0.63131** (0.0397462)	0.988622 (0.0037382)	*0.988263* (0.00444671)
Spearman	**0.652196** (0.154471)	0.500846 (0.0818755)	*0.621364* (0.0916314)	**0.635913** (0.116727)
Runtime	**5.66** (2.58929)	145.88 (67.5626)	*20.16* (5.45293)	**0.02** (0.14)
100D-10C	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.756192** (0.0213985)	0.518027 (0.0220537)	*0.633438* (0.0438062)	**0.659992** (0.0375568)
Inter	**0.471694** (0.112744)	*0.00871748* (0.163685)	**0.0361436** (0.13503)	-0.0160813 (0.150202)
Intra	**0.256738** (0.0249201)	**0.51955** (0.0218987)	*0.00550127* (0.00923007)	0.00100231 (0.00803465)
Toperror	**0.626329** (0.0540881)	**0.729864** (0.017215)	0.989775 (0.00276281)	*0.988229* (0.00254421)
Spearman	**0.52491** (0.078794)	0.305811 (0.0871427)	**0.324658** (0.100384)	*0.308057* (0.109892)
Runtime	**38.16** (13.2384)	902.12 (297.709)	*40.16* (7.61409)	**0.02** (0.14)

Table 19: Results for MDS, 2D-SOM and ant-based clustering, and the lower bound on the *IRIS*, *WINE*, *ZOO* and *WISCONSIN* data sets. The quality of the mapping is evaluated using the overall Pearson correlation, the inter- and intra-cluster Pearson correlation, the Spearman rank correlation and the topographic error. Runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best result and italic fact the third best out of the four algorithms.

IRIS	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.974947** (0.0024008)	*0.612105* (0.0954194)	**0.798703** (0.0693128	0.530764 (0.144448)
Inter	**0.858549** (0.049507)	*0.132489* (0.739822)	**0.856618** (0.102589)	-0.0490503 (0.734462)
Intra	**0.925226** (0.00317158)	**0.617727** (0.0916112)	*0.39357* (0.0674407)	-0.0224151 (0.0204314)
Toperror	**0.340533** (0.0303707)	**0.1828** (0.0393297)	*0.954667* (0.0198214)	0.9604 (0.0199738)
Spearman	**0.737839** (0.0989799)	*0.643817* (0.090443)	**0.757699** (0.0757227)	0.495647 (0.13947)
Runtime	**0.06** (0.237487)	*0.96* (0.397995)	11.44 (0.983056)	**0.0** (0.0)
WINE	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.899512** (0.00385896)	0.53004 (0.0837611)	**0.669822** (0.0300067)	*0.602912* (0.0517487)
Inter	**0.432494** (0.14829)	-0.0324514 (0.731613)	**0.521239** (0.424535)	*0.0282289* (0.775795)
Intra	**0.647269** (0.00752832)	**0.530648** (0.0865037)	*0.159461* (0.035831)	-0.048989 (0.0131521)
Toperror	**0.875393** (0.0181066)	**0.271348** (0.0326665)	0.958652 (0.0189605)	*0.954831* (0.0178351)
Spearman	**0.630209** (0.0280505)	0.529309 (0.08701)	**0.662186** (0.0482374)	*0.560322* (0.0488458)
Runtime	**0.1** (0.3)	*1.7* (0.5)	8.56 (0.897998)	**0.02** (0.14)
ZOO	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.876567** (0.030239)	*0.569825* (0.0608407)	**0.701093** (0.0258439)	0.0132542 (0.00896525)
Inter	**0.123958** (0.120388)	-0.137493 (0.200788)	**0.0614492** (0.231029)	*0.0216274* (0.177167)
Intra	**0.851628** (0.0201897)	*0.566951* (0.0988059)	**0.70855** (0.046781)	-0.109336 (0.0592585)
Toperror	**0.353861** (0.0329458)	**0.101584** (0.0245584)	*0.93802* (0.0234141)	0.98495 (0.0156222)
Spearman	**0.603469** (0.0324946)	*0.503301* (0.0725183)	**0.697777** (0.0209604)	0.00624919 (0.00983746)
Runtime	**0.06** (0.237487)	*0.56* (0.496387)	7.04 (0.564269)	**0.0** (0.0)
WISCONSIN	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.979929** (0.00352834)	**0.600172** (0.0368993)	*0.339405* (0.0526479)	-0.0179488 (0.00723965)
Inter[a]	**1.0** (0.0)	**1.0** (0.0)	**1.0** (0.0)	**1.0** (0.0)
Intra	**0.981203** (0.00345273)	**0.6007** (0.0369377)	*0.342096* (0.0525964)	-0.00400793 (0.00853926)
Toperror	**0.502003** (0.00858965)	**0.262346** (0.0140693)	*0.990186* (0.00376347)	0.993047 (0.00355081)
Spearman	**0.83105** (0.0585367)	**0.648869** (0.0334919)	*0.310401* (0.0447107)	-0.0120124 (0.00650127)
Runtime	**2.12** (0.324962)	*3.78* (1.40414)	31.82 (1.66961)	**0.02** (0.14)

[a]For a two-cluster data set, the inter-cluster correlation is necessarily 1.0.

Table 20: Results for MDS, 2D-SOM and ant-based clustering, and the lower bound on the *YEAST*, *DERMATOLOGY* and *DIGITS* data sets. The quality of the mapping is evaluated using the overall Pearson correlation, the inter- and intra-cluster Pearson correlation, the Spearman rank correlation and the topographic error. Runtimes (in seconds) are additionally provided. The table shows means and standard deviations (in brackets) for 50 independent runs. Underlined bold face indicates the best, bold face the second best result and italic fact the third best out of the four algorithms.

YEAST	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.70202** (0.00626577)	*0.35712* (0.0391684)	**0.378197** (0.0485247)	0.000368857 (0.00551728)
Inter	*0.425819* (0.141841)	**0.520747** (0.242438)	*0.00186549* (0.180457)	-0.0126154 (0.136607)
Intra	**0.72025** (0.00544137)	*0.354477* (0.0597174)	**0.373985** (0.0633439)	-0.00895364 (0.0243345)
Toperror	**0.912399** (0.00490944)	**0.334272** (0.0143734)	*0.987561* (0.00306796)	0.995916 (0.00246454)
Spearman	**0.588083** (0.0366792)	**0.353531** (0.0435066)	*0.329932* (0.0564289)	0.0108658 (0.00784111)
Runtime	**0.14** (0.346987)	100.06 (1.47526)	*16.64* (0.889044)	**0.06** (0.237487)
DERMATOLOGY	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.841184** (0.00932192)	*0.521828* (0.0685706)	**0.694782** (0.040132)	0.0178542 (0.00536327)
Inter	**0.0376024** (0.216176)	**0.297729** (0.417333)	-0.0888879 (0.195882)	*-0.0342341* (0.249856)
Intra	**0.855238** (0.00861104)	*0.523111* (0.0667635)	**0.713158** (0.0395792)	0.00138671 (0.0124873)
Toperror	**0.868361** (0.0202384)	**0.318798** (0.0217445)	0.97235 (0.0114967)	*0.970546* (0.011176)
Spearman	**0.637396** (0.0207703)	*0.464162* (0.0828382)	**0.603801** (0.0584961)	0.0151547 (0.00874899)
Runtime	**0.54** (0.498397)	8.16 (0.611882)	*5.72* (0.491528)	**0.0** (0.0)
DIGITS	MDS	SOM	ant-based sorting	Lower Bound
Pearson	**0.845455** (0.0031926)	**0.395746** (0.0724766)	*0.178435* (0.0286044)	-0.000149016 (0.00240739)
Inter	-0.226339 (0.0629175)	**0.0279154** (0.165654)	*0.00956845* (0.172233)	**0.0150002** (0.161204)
Intra	**0.845208** (0.00315283)	**0.395929** (0.0726598)	*0.178839* (0.0286443)	-0.000833506 (0.00212894)
Toperror	**0.904688** (0.00844458)	**0.294608** (0.0075264)	*0.99745* (0.00109509)	0.999148 (0.000622373)
Spearman	**0.646727** (0.0360438)	**0.365161** (0.0779491)	*0.186743* (0.0264507)	-0.000146374 (0.0025251)
Runtime	**9.04** (5.73048)	696.98 (10.0866)	*321.58* (8.08972)	**0.06** (0.237487)